A Case Study of a Software Development Process Model for SIS-ASTROS

Camila Hübner Brondani, Otávio da Cruz Mello, Lisandra Manzoni Fontoura
Departamento de Computação Aplicada – DCOM
Universidade Federal de Santa Maria – UFSM
Santa Maria, Brazil
{chbrondani, odmello, lisandra}@inf.ufsm.br

***Abstract* — Context: technological innovation projects, developed in universities in partnership with companies and/or the government need processes that can handle the characteristics of the institutions involved. The academic environment is often used to dynamic methods, but contracts require plan-driven processes. Goal: the goal of this research is to understand the needs of the parties involved (university and government/enterprise) and provide an adapted software process to satisfy those necessities. Method: a case study considering a project between the Federal University of Santa Maria (UFSM) and the Brazilian Army (BA) for the development of an Integrated Simulation System was conducted. Initially, problems in the development were detected and a process was defined. It was then evaluated and improved over the iterations, through team meetings. Results: the experience acquired in the project was consolidated as lessons that could be used to assist the process definition of projects with similar characteristics. Conclusion: innovation projects involving the collaboration of universities, government and/or companies are successful if an adequate process is established to treat specificities of the academy, not only in relation to characteristics of the work but also the team.**

Keywords—process, triple helix, ASTROS, case study.

I. INTRODUCTION

The university today, besides the academic activities and the pure research, promotes the development of applied research aiming to generate innovation solutions from issues presented by governmental institutions and enterprises. The triple helix thesis states that universities are distancing themselves from having a secondary social role, although important, of providing higher education and research, and is taking a primary role equivalent to the industry and the government, generating new industries and companies [1].

In this context, the Federal University of Santa Maria (UFSM) started a project to develop an Integrated Simulation System for the Brazilian Army (BA) in 2015. One of the initial challenges of the project was defining an adequate process model. Since there was an agreement between the BA and the UFSM with predetermined goals and deadlines, a plan-driven process model would be more satisfactory. On the other hand, the research needed for the system development could not be predictable, requiring investigation, prototype development and evaluation. The set of requirements was vague; the team was formed of high-skilled workers with autonomy. Based on these aspects, agile methodologies could be considered more proper.

Understanding the peculiarities of a project involving the university and government is crucial to choose an adequate process model that can satisfy the needs of both parties.

In this article, we will describe the lessons learned during four years of a research project between the UFSM and the BA and the process model that was used in this development, adapted over the iterations. Our goal is that the lessons learned and the process model can assist the definition and/or adaptation of models that are used in projects involving universities and governmental institutions or private companies.

The project in question proposes the development of a Tactical Virtual Simulator aiming the military training in tactical operations related to the use of an ASTROS battery (Artillery Saturation Rocket System).

Some important characteristics of the project are: need of meeting semi-annual goals pre-defined at the start of the project, difficulty of defining requirements due to the system's complexity, the unfamiliarity of the development team with the area of application, high-skilled workers, and constant need of innovative solutions research.

Those attributes led the definition of the software process. An evolutionary and iterative life cycle was defined, where intermediary versions of the software are generated and evaluated constantly by the client, easing the definition of new requirements for the next iteration. Besides that, milestones were defined with the purpose to satisfy contractual obligations. The process was evaluated and improved over the course of the project and the experience acquired was consolidated as learned lessons to be used in the development of other similar software projects.

This article is organized as follows: in Section II, important concepts to the comprehension of this work are introduced. In Section III, related works are discussed. In Section IV the context of the case study is described. In Section V, we define the proposed process. In Section VI, discussion and analysis are presented. At last, in Section VII we discuss our final considerations and comment on future works.

II. BACKGROUND

Modeling of software process has been a very challenging problem and constantly debated in the software development community in the past 30+ years, largely due to the complex nature of the software development process that involves not only the technical knowledge and skills but also

DOI reference number: 10.18293/SEKE2019-098

many other factors, such as human, management, quality assessment, and cost [2]. The modeling of business processes aids in the comprehension and optimization of existing business processes, and also in the conception of new business processes to make organizations more competitive and efficient [3].

Software development strategies have gradually shifted from the traditional waterfall model to more dynamic and responsive iterative, multi-cycle strategies. The reason usually cited is the need to minimize risk in the process [4].

Traditional iterative software development efforts such as spiral development or iterative enhancement can be considered adaptations of the waterfall software life cycle [5]. This is because these methods generally assume that the entire documentation required by the waterfall method will still be produced, but will be rewritten and updated during each cycle rather than once for the entire software process [5].

Agile processes are different from traditional software processes in that the time per cycle is very short and many fewer formal methods are employed. They focused on repeated lightweight practices for rapid and continuous delivery of software in small chunks with close collaboration from the customer as well as among members of the development teams.

No rigid plan or requirement is determined in advance, as these can change during the development process. Being flexible and adaptive to changes are in the DNA of agile methods while still achieve the ultimate goal of producing customer satisfied software within the time and cost framework [5]. Extreme Programming and Scrum are two software development processes that fit this description [5].

Many software development methodologies fall in between plan-driven development and agile development, and exhibit several of the characteristics of agile development. Examples include incremental development, prototyping, and DSDM (Dynamic Systems Development Method) [6].

To mitigate the impacts of abrupt paradigm changes and support organizations that don't want to stop following all traditional practices some proposals were developed for hybrid processes that incorporate principles of agile and traditional paradigms [7].

III. Related Work

The study from Cotugno and Messina [8] presents an overview of the development process, focusing on the Scrum methodology adopted by the Italian Army for the development of software systems using open code technologies.

Benedicenti et al. [9] relate the experience of an agile application in the defense sector They describe the experience of creating a control and command system for the Italian Army. The delivery of the project happened after 13 sprints of five weeks, meeting all the needs of the users and satisfying the regulatory requirements of the army. Acquiring this positive result demanded collective effort to change the development culture, since there was natural resistance to change, and the need of highest possible support level to guarantee the continuity of the selected process. The article presents the positive results quantified.

As well as this article, the work from Cotugno and Messina [8] and Benedicenti et al. [9] describe methodologies and techniques used in the software development in an military environment. The main difference is that both are only focused on agile methods.

The work from Jenkis [10] describes the implementation experience of PRO-SOFTWARE, a software quality project involving the government, industry and academy (the triple helix). The goal was strengthening the software industry in Costa Rica, assisting organizations in improving their software processes. Therefore, Jenkis [10] proposes a methodology based on the quality improvement using the Capability Maturity Model (CMM) as base.

IV. Case Study Design

In order to address the research objective, we designed an exploratory case study, which involved a real-world software project. We define a software process based on identified process and analyze this process over several iterations. This section describes the design of the case study.

A. Project Context

The SIS-ASTROS project started in 2015, and is predicted to end in 2020. The main goal of the project is the development of an integrated simulation system to support the teaching of doctrines related to the use of a rocket artillery battery. The development team is formed of 7 doctor professors, 3 researchers, 4 developers, 7 master's degree students and 13 undergraduate students.

In addition, the requirement of the projects were described in high level of abstraction, the UFSM's team did not have the knowledge of the domain and the project required some innovative solutions, mainly related to the simulator's integrity, 3D scenarios generation and autonomous navigation. It is predicted to transfer the technology to the BA at the end of the project.

On the other side, the professors and researchers have long experience in the research field, developing researches to provide innovative solutions in different areas of computer science.

B. Methodology

At the beginning of the development, there was not a process model clearly defined in the project, so the artifacts were not standardized and the flow of activities did not follow a pre-defined roadmap. This scenario brought difficulties in the project management and fomented the definition and elaboration of a software process for the project. Therefore, from this necessity, this research project was initiated. The methodology used by the team to conduct the case study was composed of the four phases described below.

Diagnosis: identification of the problems happening on the project and possible solutions. In this stage, many problems related to the inexistence of a defined software process were found. The discovery of the problems occurred through meetings with the partied involved in the project.

Planning: from the problems identified in the last stage, a process model to be used in the project was proposed, aiming to solve these problems and satisfy the characteristics and necessities of the project and the team at the same time.

Implementation and Evaluation: during the three following years, the process was implemented and improvements were incorporated to it, intending to adapt the project to current needs.

Analysis: the results obtained over the course of the project are presented and the acquired experience is described as lessons learned.

C. Problems Diagnosis

The issues found during the Diagnosis phase can be summarized as follows.

Unfamiliarity with the application domain

The UFSM team did not have knowledge about military doctrines neither terminologies of the field. The manuals were rich in details and very extensive, making it difficult for the team to understand and learn.

Difficulties related to requirements definition

Being an innovative software, the set of requirements was not defined. There were a lot of concerns and doubts about how the simulation system would work and which features would be necessary.

Complexity of solutions

Complex and innovative computational solutions were required to solve the technical issues found during the development.

Rework

The team project was composed of workers with different skill levels, the professors and the researches were high skilled, master's degree students possessed an average level of skill and undergraduate students were low skilled. Since there was a large number of trainees, many problems in the source code were found, like defects, lines that were hard to comprehend and maintain, and issues related to class structuring.

Communication difficulties

Due to the hierarchic communication structure with the client, the information goes through several levels until the decision taking. This communication flow causes problems like developmental delay, when for example, the team needs to wait for an answer to a doubt.

High team turnover

The students remain in the project while they are taking their graduation or master's degree course, on average two years. Therefore, we have high turnover.

Requirements instability/Changing Requirements

Constant changes in requirements, mainly during the test phases. Many changes occur because of divergent opinions, often due to lack of vision of the whole.

V. PROPOSED PROCESS OVERVIEW

In the planning phase, a software process was developed with intent of proposing solutions to the issues identified during the diagnosis phase while meeting the needs and peculiarities of the government and the academy. On one side, we have a stakeholder that gives priority to software documentation, rigid definition of iterations and deadlines, while on the other side, we have a self-managing team that is focused on development and coding.

The process initially defined was constantly evaluated through the phases and iterations (Implementation and Evaluation phase). The evaluations were performed during meetings, when the parties involved would discuss which practices gave positive results and which should be reviewed, and with this feedback, the process was improved.

The current process is described in Figure 1 (life cycle vision), Figure 2 (iteration activities) and Figure 3 (change management sub-process activities).

Some considerations about the process are described in the following section.

A. Process Life Cycle

Aiming to include the formal deliveries, foreseen in the contract, the life cycle was organized in phases and iterations, as depicted in Figure 1. Two phases are planned: initiation and construction, finished with a major milestone.

The initiation phase only happens once and is responsible for defining an overview (abstract) of the system in development and giving a clear comprehension of the business domain that is related to this system.

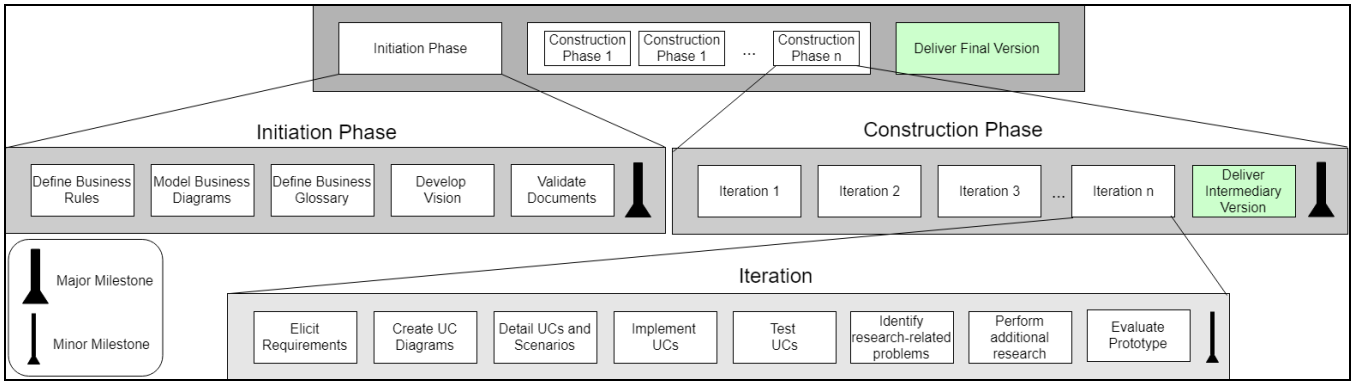

Fig. 1. Life Cycle Vision

The construction phase, on the other hand, is responsible for the execution of the technical activities that will generate a new version of the software. A project can have as many construction phases as needed, and each one can have multiple iterations. Both phases must respect contractual obligations, for this reason, they are finished with a major milestone that indicates a formal delivery to the client.

Since the phases usually refer to bigger time spans (semesters, years), it was chosen to break them in many iterations with the purpose of speeding up the process. Each iteration has its complete development cycle, from requirements definition to version evaluation (Figure 2). The software versions developed in the iteration are always delivered when the phase ends (major milestone). The number of phases and the amount of iterations in each of these phases depend on the project and can be adjusted.

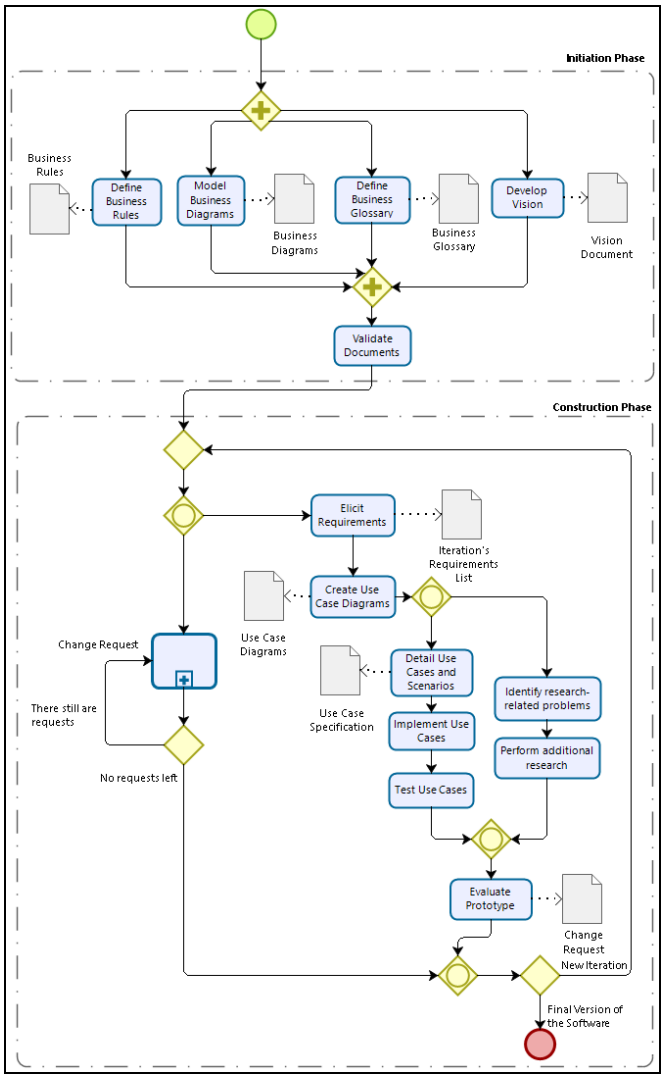

Fig. 2. Iteration Activities

At the end of the iteration, meetings with the client are held to present the intermediary version of the software, in which possible improvements, changes and evolutions are discussed. These meetings are important to track the current progress of the development team.

B. Activities and Artifacts

The initiation phase is composed of four main tasks that happen at the same time: define business rules, model business diagrams, define business glossary and develop vision. These tasks generate the artifacts business rules, business diagrams, business glossary and vision document, which are formally evaluated by the client. Before the construction phase starts, it is extremely important that the artifacts generated during the initiation phase have been approved by all the stakeholders (task validate documents). When the respective documents are finished and approved, successive iterations start in each phase. Each one has a set of tasks that generate an intermediary version of the software. In the first task, the stakeholders meet to define the requirements that must be implemented in the cycle.

After the requirements of that iteration are defined and prioritized, the specification and detailing tasks start. Diagrams and requirements specification documents are created to assist the team members during the development and the technology transfer process. All the artifacts created in this phase are managed in a requirements management tool.

The tasks identify research-related problems and perform additional research are executed simultaneously, due to constant innovative solutions research. These are incorporated in the simulator in the next iteration.

Once the modeling ends, the team can finally start implementing and testing. If there are issues with a feature that cannot be fixed during the defined cycle, or the programmers are late in the development, an artifact is generated reporting the features that could not be finished, so they can be implemented in the next cycle. As soon as the iteration finishes, the client validates the intermediary version of the software, defining additions or changes that should take place. These are documented and serve as input for the next iteration's requirements definition.

Change requests can be submitted at any time, either to include or modify a requirement that was previously defined. In the main process, the procedure of submitting a change request is seen as a sub-process (Figure 3).

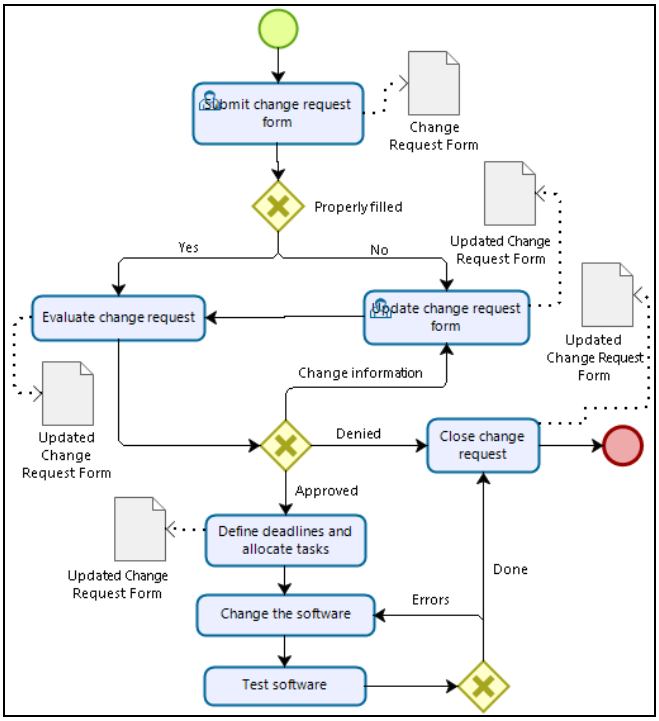

Fig. 3. Change Management Sub-process Activities

This sub-process is basically a flow of activities to manage the changes in the project. First, a stakeholder submits a change request, which is reviewed by a committee and, if the request is relevant, the change is incorporated in the version of the software. However, there are times when no requests are submitted in an iteration, so the change management process will not necessarily occur in the flow, therefore, being optional.

C. Roles

The project team was organized in levels: researcher professors (part time), professionals and researchers (full time), master's degree students and trainees (part time). The researcher professors guide the students in solving research problems and developing their academic works.

Professors are also responsible for the project management. Professionals and researchers are in charge of planning the tasks to be assigned to each member of the team, communicating obstacles to the management and organizing the daily routine of the team. Master's degree students are responsible for guiding the undergraduate students in their activities, helping solving issues regarding tasks assigned to them. The product owner is responsible for the communication between the development team and the client; all the requests from the team are centered on this person, which will track them until they are complete.

The team is collaborative, all the workers are assigned to close workrooms and there is constant exchange of knowledge between the team members.

VI. Discussion and Analysis

The plan-driven approach served as foundation to the process definition. Using the basic principles: analysis, design, construction and verification, we have the basis for the flow of activities, supporting development of specific documents related to each phase of the project. The contractual aspect of the project, that demands deliveries on a timeframe, is contemplated with milestones at the end of each phase. The documents submitted are important for the requirement of technologic transference at the end of the project.

Allied to traditional models, we've decided to apply some characteristics of agile methodologies to the process as well, in order to emphasize the collaborative and communicative principles of the team and the final user, allowing incremental deliveries and also supporting the constant change requests without affecting or causing time and/or financial damage to the project [11].

Therefore, the method used in the creation of the process was defined as a hybrid between plan-driven and agile models, using the most advantageous characteristics, aligned with the goals of the project. In addition, for each issue found, actions were taken in the software process, aiming to solve or minimize them. They are described below.

Unfamiliarity with the application domain

The solution found was including some tasks at the start of the process with the purpose of comprehending the application domain. Diagrams that represented the domain were elaborated in collaboration with the stakeholders. Besides that, glossaries were also created, that are being maintained through the course of the project. The BA team has been formally validating these documents.

Difficulties related to requirements definition

It was decided to work on intermediary versions of the software that were evaluated periodically by the BA team. When all the parties approved the prototype, a new set of requirements for the next iteration would be defined.

Complexity of solutions

It required applied research and development of master's essays and final papers exploring necessary solutions for the development of the simulator guided by a researcher in the field.

Rework

The solution was the constant refactoring of the source code, especially at the beginning of the project. Before the formal deliveries, there were periods intended for the code refactoring, with the purpose of improving legibility and documentation, as well as removing unnecessary code lines. We now focus on continuous source-code reviewing. Additionally, there is an internal hierarchy where experienced members assist new ones, helping them developing high-quality artifacts.

Communication difficulties

A formal communication flow was defined so that the parties involved track the information requests.

High team turnover

Some experienced professionals (researchers and programmers) were hired full time. Teamwork is encouraged and constant experience exchange between trainees and experienced members happen, thus, the team shares the knowledge of the system.

Requirements instability/Changing Requirements

Usage of incremental and iterative development, focused on periodic presentations of the intermediary version of the software. A formal change request process was also created.

Based on the results obtained by the execution of the process and the continuous monitoring of the team and the client since the beginning of the project up until now, it was possible to define some of the best practices and adopted decisions that reflected positively on the quality and progress of the project. It is believed that these practices can be applied in software development projects that involve academy and government and/or industry.

Client's periodic homologation

Iterative development allows the team to deliver a functional product to the client at the end of each iteration or cycle. The client can use this prototype over a period of time and provide feedback for the developers in terms of definition of new requirements, change requests and issue reporting. Usually, changes are incorporated into the requirements baseline to be implemented in the next iteration.

Use of diagrams to represent the business domain

Business diagrams helped the team to comprehend the business domain, making future communications more fluid. These diagrams were also used by the client to communicate

with other parties involved in the project. It was possible to represent the BA doctrines fully and clearly, preventing the team from reading manuals that are complex and difficult to understand.

Cooperative work

Team members can learn from each other. The more experienced guide the less experienced. Additionally, each team member knows what the others are developing, and can exchange information. The development of a particular activity becomes priority of the group as a whole, and not property of a certain team member only. The master's degree students mentor undergraduate students in research, that way, team members develop a common sense of responsibility that brings them closer.

Effective communication

The agile processes support the idea of face-to-face communication as the most effective and efficient method of transmitting information in the development team. The UFSM team is allocated in a sole environment. However, since the client team is located in another state, face-to-face communication is not possible. Therefore, to build an efficient communication method, it was necessary to center the communication on the Product Owner. This person is responsible for bringing the military vision to the project and evaluating, along with the team, the enhancements or changes that should take place to ensure that the software fulfill the needs of the BA. Bimestrial face-to-face meetings are scheduled.

Definition of a change management process

The change management process helped to monitor change requests and limited the number of unnecessary requests without the global comprehension of the system.

In the SIS-ASTROS project, we have defined one initiation phase and five construction phases, with duration of six months each. In each construction phase, three bimestrial iterations were established, since there are many part-time workers in the project that need to conciliate their work in the project with their academic obligations, teaching, in the case of professors, and classes and university assignments, in the case of students.

During this time, we managed to meet the goals defined in the project within the time and budget. The BA is satisfied with the results obtained and future projects are being discussed. The formal change request process reduced rework, and the amount of defects in the software has been dropping over time at the same pace performance (response time) has been increasing.

VII. Concluding Remarks

Projects involving the collaboration between universities and government and/or industry are successful if suitable procedures to handle the needs and peculiarities of the parties involved are established. In the described case study, hybrid process types proved to be satisfactory because they explore plan-driven characteristics – based in contracts, at the same time agile methods are suitable for innovative projects, which involve high-skilled professionals.

It was possible to experience practices from both investigated process in this project, reflecting positively in the developmental quality and progress, solving issues previously detected and establishing a set of learned lessons that can be used in other similar software development projects.

As future work, the main idea is to review the process periodically along with the team, continuously verifying the relevance of the activities and artifacts. As the process is thoroughly used, it may be possible to optimize some activities, thereby making the process less bureaucratic.

The fact that this approach was only applied in one project, even if in successive iterations during three years, was a limitation associated with this article.

Acknowledgment

We thank the Brazilian Army for the financial support through the SIS-ASTROS Project (813782/2014), developed in the context of the PEE-ASTROS 2020.

References

[1] H. Etzkowitz and C. Zhou, "Hélice Tríplice: inovação e empreendedorismo universidade-indústria-governo," *Estudos Avançados*, vol. 31, no. 90, pp. 23–48, 2017.

[2] R. A. Haraty and G. Hu, "Software Process Models: A Review and Analysis," *International Journal of Engineering & Technology*, vol. 7, pp. 325–331, 2018.

[3] K. C. Laudon and C. G. Traver, *Management Information Systems*, 12th ed. Sao Paulo: Prentice Hall, 2011.

[4] H. M. Olague, L. H. Etzkorn, W. Li, and G. Cox, "Assessing Design Instability in Iterative (agile) Object-Oriented Projects," *Journal of Software Maintenance and Evolution: Research and Practice*, pp. 237–266, 2006.

[5] B. Ramesh, L. Cao, and R. Baskerville, "Agile Requirements Engineering Practices and Challenges: An Empirical Study," *Information Systems Journal*, vol. 20, no. 5, pp. 449–480, 2010.

[6] G. Van Waardenburg and H. Van Vliet, "When agile meets the enterprise," *Information and Software Technology*, vol. 55, no. 12, pp. 2154–2171, 2013.

[7] W. Chaves, D. S. Carvalho, P. F. Rosa, S. Soares, M. Antonio, and L. C. Buiatte, "A comparative Analysis of the Agile and Traditional Software Development Processes Productivity," *30th International Conference of the Chilean Computer Science Society*, IEEE, pp. 74–82, 2012.

[8] F. Cotugno and A. Messina, "Adapting SCRUM to the Italian Army : Methods and (Open) Tools," *IFIP International Federation for Information Processing*, 2016.

[9] L. Benedicenti, A. Messina, and A. Sillitti, "iAgile: Mission Critical Military Software Development," *International Conference on High Performance Computing & Simulation iAgile:*, pp. 545–552, 2017.

[10] M. Jenkins, "PRO-SOFTWARE: A Government-Industry-Academia Partnership that Worked," *17th Conference on Software Engineering Education and Training (CSEET'04)*, 2004.

[11] I. Sommerville, *Engenharia de Software*, 9th ed. São Paulo: Pearson Prentice Hall, 2011.