# PROLES: A Customized Process Applied to Software Engineering Laboratories and Small Business

[1,4]Rodrigo Rocha Silva, [1]Fernanda Yuri Kimura, [2]Joubert de Castro Lima and [3,4]Jorge Bernardino

[1]FATEC Mogi das Cruzes, *São Paulo State Technological College – Brazil*
[2]*Computer Science Department, Federal University of Ouro Preto (UFOP) – Brazil*
[3]*Polytechnic of Coimbra – ISEC, Coimbra - Portugal*
[4]*Centre for Informatics and Systems, University of Coimbra (CISUC) - Portugal*
rodrigo.rsilva@fatec.sp.gov.br, feekimura@hotmail.com, *joubert@iceb.ufop.br, jorge@isec.pt*

***Abstract***

*Most micro and small software development companies do not invest in improving the development-lifecycle processes. Given this, many researchers and software engineers have made proposals for software development processes with lower implementation costs and greater use of human and material resources. In general, it is required that work in academic laboratories be carried out in shorter deadlines, having groups of students practice techniques, such as different methodologies of requirements specification and fast responses to changes, ensuring the assertiveness in the value of deliveries of their projects. In this work, we customized the RUP and adopted Scrum practices, creating a new process called PROLES. PROLES was evaluated in a small company and also in a Software Engineering undergraduate course. PROLES has new particularities that differentiate it from the state of the art, such as a traceability matrix and a reference architecture. Such a matrix is obtained by the reference architecture, serving as a representation of the relationship among requirements and one or more system implementation components. The experimental results prove that PROLES can be adopted as an alternative to a systematic and less-intrusive process.*

**Keywords***: Software Development Process, Scrum, RUP, small companies, Undergraduate course*

## 1. Introduction

According to data from the Brazilian Association of Software Companies - ABES (2013), 49% of the software development companies Brazil are considered micro and small size companies (SMEs) and most of these companies do not invest in improving the development processes of software [8].

Chagas et. al. [1] analyze a small software development company in Maringá city, Brazil, detected that no software development process was adopted in the projects. Pinheiros et. al. [2], analyzing a small company and identified problems related to the established deadlines. These examples illustrate the low investment in software engineering processes of SMEs.

Some practitioners and academics have proposed different process that combine two or more methodologies [9, 10, 11, 12, 13, 14, 15, 19]. The objective of these hybrid approaches was to create a process that maximized the strengths of the involved methodologies while at the same time reducing their weaknesses to improve the software development lifecycle and produce high-quality products [11, 12].

Two methodologies normally used in the previous mentioned hybrid approaches are the Rational Unified Process (RUP) and Scrum, both of which are iterative and incremental approaches and have been widely used with long successful backgrounds [14, 16, 20, 21]. RUP, which is commonly misunderstood as a traditional methodology, because of its many roles and artifacts, can be adapted to a particular project or organization to provide discipline, structure, and guidance throughout the entire software development lifecycle [17].

On the other hand, Scrum is an agile management process that helps steer a project with an iterative and incremental approach during the software development lifecycle [18].

The reality of SMEs in software development is close to that experienced in the Software Engineering Laboratory (SEL) course of Faculty of Technology (FATEC-MC). In both scenarios, the goal is to build software products (or prototypes), where it is desired that students form self-organized teams and rely on the teacher only as a consultant for the development of projects. Thus, we formulated the software development process, named PROLES processes, which is a customization of RUP and Scrum practices.

The main goal of PROLES is to guarantee a constant delivery routine of software artifacts that have business value, i.e. artifacts with quality and high traceability of requirements, being little intrusive. PROLES is not designed to be agile, but rather to enables constant delivery and change control based on the traceability of the implemented requirements.

In order to validate the PROLES process, we present the results of its adoption in a small software development company in Mogi das Cruzes city and also how the performance of students using such process during 9 semesters in SEL discipline in FATEC- MC.

The rest of the article is structured as follows: in section 2 the related works are presented, in section 3 the PROLES process is detailed, in section 4 the obtained results are discussed. In section 5, we present the conclusions and future work

## 2. Related Work

We find in the literature processes for the development of customized software similar to PROLES, i.e. applied to small teams.

**Table 1.** Comparison Between the Related Works

| CHARACTERISTIC | MPA-DS | BOPE | *easY Process* | SSP | AFSP | PROLES |
|---|---|---|---|---|---|---|
| Scrum Basis | X | X | X | X | X | X |
| XP Basis | X | X | X | | X | |
| RUP Basis | X | X | X | X | X | X |
| TDD Basis | | X | | | | |
| BDD Basis | | X | | | | |
| PMBOK Basis | | X | | | | |
| Iteractive and incremental | X | X | X | X | X | X |
| Structure: Vertical | | X | X | X | X | |
| Structure:Horizontal | X | | | | | X |
| DVS | | | X | | | X |
| UML Diagrams | | X | | | | X |
| Cases of Use | | | | | | X |
| Task Chart | X | X | X | | | |
| Prioritization | X | X | X | | | X |
| *Sprints* | X | X | X | X | X | X |
| *Sprint* Retrospective | X | | | | X | X |
| Daily and weekly meetings | X | X | X | X | X | X |
| User Stories | X | X | X | | | X |
| Schedule | X | X | | | X | X |
| Programming in pairs | X | X | X | | X | X |
| Tests | X | X | X | X | X | X |

As an example, we have the agile process model MPA-DS [3], which is based on agile methodologies Scrum and XP, stimulating the communication between the members and supporting the iteration.

Another process developed in academic software engineering laboratories or disciplines, suggesting practices of XP, RUP and Agile Modeling, is *easYProcess* [4]. It is designed with roles as its basis: client, user, tester, developer and manager roles. However, *easYProcess* uses the system view document (DVS) suggested by the RUP to organize information about the scope to be validated by the client. DVS is useful for user stories definition and prioritization.

Another example of a process developed in academy suggests programming in pairs and user stories. The BOPE process [5] adopts Test Driven Development (TDD) and Behavior Driven Development (BDD) concepts to specify user stories when planning each sprint. Without the constant need of a project

manager, this process presents a good alternative for small groups of development, demonstrating flexibility to teams, as it allows teams to organize themselves.

The SSP - Simple Software Process [6] suggests the prototyping of software screens, not for requirements surveying, but to test the architecture defined with users and clients. Thus, the navigation pattern and "raw functionality" of the system are defined in the SSP. It is also a software development process developed in an academic laboratory, being used since 1998 at the Catholic University of Chile.

Some papers suggest the application of a framework, such as the Agile Framework for Small Projects (AFSP) [7]. AFSP guides small companies in adopting agile practices in their projects. It helps the identification of risks based on the agile factors, size and experience of the team, criticality, degree of requirements and the support of the company in the agile development.

The detailed comparison of characteristics of each related work is presented in Table I.

BOPE, *easYProcess* and PROLES, allow different roles to be performed by the same person due to the fact that they are both processes from academy. However, they are more complete in terms of documentation and activities compared to SSP, also from academic laboratory, but without much specification and use of practices during sprints.

## 3. PROLES

PROLES fulfill all the basic premises of software development processes, such as iteractions and verifications with delivery forecast of mandatory artifacts.

The customization proposed within PROLES is strongly focused on design and development, suggesting only the following RUP artifacts:

I. DVS or technical proposal for high-level definition of the scope and purpose of the system. A clear instruction of the problem and the proposed solution, assisting customer's expectations, minimizing risks, and offering a comprehensive view of the Software architecture;

II. Mapping of functional requirements (FRs), non-functional (NFs) and business rules (BRs) for specification of functionalities, presenting the constraints, validations, and exceptions that the system must obey;

III. Description of functionalities: it can be done through the use case document (DUC), describing the interaction between the system and the actors, or user stories to describe the user's goals and how the system will act to achieve them;

IV. Software Domain Mapping: representing the structure and relations of the classes that serve as a model for objects.

In order to enable the constant delivery of functionalities, PROLES adopt the following Scrum practices:

a) Iterative and incremental method (sprint): Each functionality and its requirements serve as input to the sprints;

b) sprint planning: Hold meetings at the beginning of each iteration to define the functionalities to be developed;

c) Periodic meeting: At each sprint, at least once a week, the manager, project leader (or teacher) meets with the team to learn about the progress of the project;

d) Meeting with the client: After each sprint, a meeting with the client shall be held for presenting the result of the sprint and obtaining feedback. Such a meeting represents the Scrum Sprint Review. In the case of groups of students, the teacher also plays the role of client.

PROLES define three roles similar to the *Scrum* roles, for which each description can be found below:

I. *Product owner*: defines the items that make up the product and prioritizes them in *sprints planning*. It also defines the scope of the product and, together with *PROLES Guardian*, works to ensure compliance with project deadlines;

II. *PROLES Guardian*: Ensures that the team respect and follow the values and practices of the project. It also protects the team by ensuring that it does not commit too much to what it is capable of accomplishing during a *sprint;*

III. *PROLES Team:* This is the development team. Unlike the *Scrum Team*, each member has roles such as developer, designer, test analyst, or architect, but everyone works together to complete the activities with which they have committed to the *sprint*. Despite the functional definition, everyone can play different roles, but it is expected to have functional specialists who establish the best practices and design patterns for each area.

PROLES is divided into three phases, just like in the Scrum, although it receives the name of the RUP phases: *Initiation, Elaboration* and *Construction*, that compose the development *Sprints* and the *Transition* phase. Figure 1 illustrates the software development lifecycle in PROLES. Each of these phases is described in the following subsections.

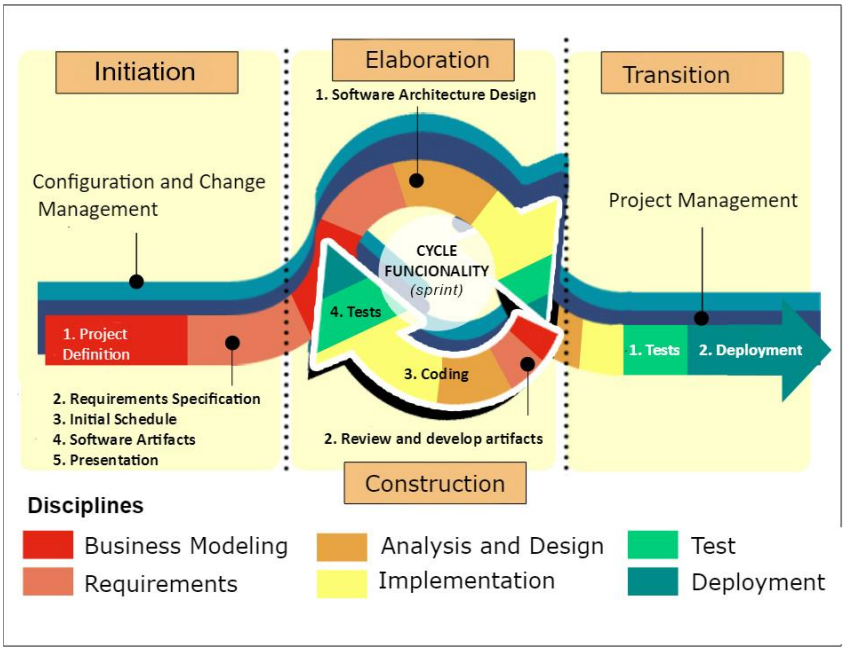


**Figure. 1.** The Elaboration and the Construction phases of the RUP were unified and form the PROLES sprint. Each phase is composed by disciplines.

### 3.1. Initiation

Initiation is the phase where the project scope, the analysis of requirements and the definition of an initial schedule by the Product Owner are set, aiming to obtain the best value for the business and return on investment (ROI). Weekly presentations shall be held where the Product Owner (or teacher) should evaluate what is presented and give feedback so that the team can make the necessary modifications.

In this phase, the DVS artifacts, requirements and weekly reports are also generated and always sent to the *Product Owner* (or teacher). Reports should describe activities performed during the week. This report is used where there is no possibility of practicing the *Daily Scrum Meeting*.

At the end of this phase, the presentation of the developed artifacts occurs and the architectural modeling is started.

### 3.2. Elaboration and Construction

The elaboration and construction phase happens through the modeling, refinement and development of artifacts, such as Software Architecture Document (DAS) and DUCs. In this phase, coding, testing and delivery of the developed artifact, like in the *Scrum's Sprint*, also occurs. Although, each use case is developed during a cycle called the functionality cycle.

PROLES suggest a reference architecture that facilitates the traceability of the implementation of the requirements. Therefore, it is suggested an architecture that has a control layer, guaranteeing the reuse of FRs in any systemic platform. It is desirable that such architecture support high granularity in the implementation of BRs and NFs that require validation, verification, addition and transformation of data.

With the high granularity in the implementation of the requirements, we have the possibility of maintaining a high traceability between software components that make up each developed functionality, increasing the possibility of performing change management in a controlled way.

Figure 2 illustrates a reference architecture for a system that is developed using the Object Orientation paradigm. In this architecture, we observed the use of a control layer for each possible client for the *BackEnd* services, aiming the implementation of the *Front Controller* design patterns.

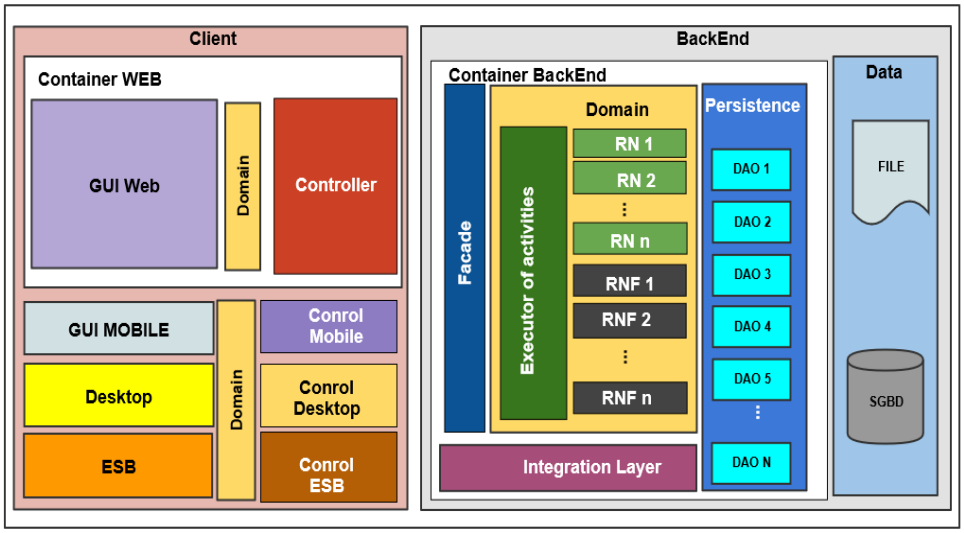


**Figure 2.** Reference architecture proposed by PROLES

The *Front Controller* is responsible for controlling the requests made by the *view* layer and for executing methods that abstract the features developed in a *Facade* (*Facade* pattern). The *Facade* functions as a controller of the entire *BackEnd* of the application, delegating the execution of BRs and NFs to a chain of responsibilities represented by the executor of activities (implementation of the *Chain of Responsibility* pattern).

The execution of the data persistence is done with the implementation of the DAO (*Data Access Object*) design pattern, making the independence of the data model in relation to the mechanism and persistence technology possible.

In the proposed architecture, BRs and NFs meet the Strategy design pattern, so there is low coupling of the code, offering advantages such as the development of BRs, FRs and NFs independently of the control layer, model and vision. This strategy makes it possible to reuse the implementation of BRs in different application execution contexts.

In order to ensure the control of requirement changes, PROLES suggest the creation of a traceability matrix that aims to provide a mapping of which use cases are impacted by the implementation of the requirements that are associated with the strategies. In PROLES, there is a hypothesis that relating the requirements to one or more system components (classes) can facilitate the understanding of how a change will affect the NFs, BRs and classes of the system.

The proposed traceability matrix allows a single mapping to be able to observe the impact from the most macro level of functionality (Use Case or any functional specification) to the most micro level

(code classes that implement each validation, verification, integration, Addition and transformation of data).

Figure 3 indicates the BRs and NFs that are associated to an FR, and it is also possible to identify which classes are impacted in a possible change demand, since they are implemented by a same strategy. Consequently, there is a way to know which use cases will also be impacted by the change.

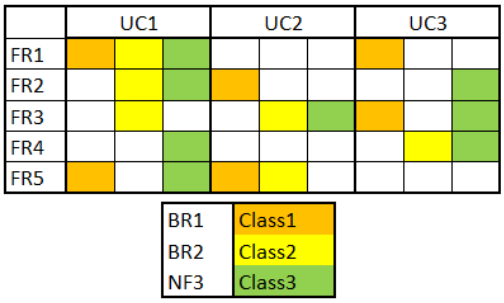


**Figure 3.** Example of matrix traceability.

The traceability matrix illustrated by Figure. 3 shows that FR1 is implemented by UC1 and UC3, however in UC1 it is necessary to implement the BRs: BR1 and BR2 and the nonfunctional requirement NF1. In UC3, only BR1 and the non-functional requirement NF1 are implemented. Finally, we have BR1 implemented by Class1, BR2 by Class2 and NF1 by Class3.

Having defined reference architecture and built a traceability matrix, the DAS artifact can be produced, which aims to present visions and architectural constraints of the system to be developed. With DAS, we demonstrate the composition of the system and how its components communicate, so there are different types of UML diagrams that can be used to represent significant aspects of persistence, control, and vision.

After all the functionality of the *sprints* have been implemented, the project moves on to the transition phase which is the last phase of the project, where the last tests and the delivery of the increment or final product will occur.

### 3.3. Transition

The goal of the PROLES last phase is to carry out the white box and black box tests. Such tests will ensure that all features implemented in the *sprint* are free of problems prior to delivery of the increment or final product.

The implementation of test units with the greatest possible coverage is a key practice in using PROLES. This is so that possible changes are noticeable in development environment, thus enabling the reduction of bugs and greater control of possible impacts.

## 4. Results

In this section, we will present two types of results. The first one is the result of PROLES applicability in a real project for a Small Business. The second result refers to the use of PROLES in the discipline of SLE during 9 semesters.

### 4.1. PROLES in a Small Business

Company X, located in Mogi das Cruzes city, is accounted as a Small Business that develops applications with the purpose of assisting the maintenance management of commercial vehicles. Typically, the projects are developed by a team of 3 to 5 people.

Prior to the PROLES application, the software development process was based on informality, without documentation of requirements and scheduling of the next meetings. All this procedure was justified due to the fact that the team was small and the owner of the company played the role of a project manager. Most of the times, deadlines were not met and there were many changes in the requirements during the development phase because those were not previously foreseen due to the lack of planning during the specification and analysis processes.

In order to help understand the major difficulties faced by the team during software development, we developed a questionnaire which was answered by all the ten employees involved. The biggest related problem they faced was the lack of specification of requirements. In addition, 50% of the people surveyed stated that this problem used to happen in more than 80% of the projects and that it was detected within the development phase, which meant demanding more rework during the implementation.

Regarding the team background, most of the answers have confirmed that they did not hold knowledge in software development processes. It was asked whether any process was applied during software development, however most of the answers stated that the team superficially applies some software process. In this regard, a contradiction was observed, because we asked ourselves how there could be some software process if the team confirms that they have no knowledge on the subject.

Analyzing the result obtained through the questionnaire, it was concluded that the current development process has no systematization. Company X started implementing PROLES by the middle of November 2016, during the development of a mobile app called AppX. The implementation of PROLES was the responsibility of the *Product Owner* together with the *PROLES Guardian*.

During the initiation phase, requirements were examined with the validation of screen prototypes during the first project meeting (*Sprint Planning*). With this, we generated the first version of FRs, NFs, BRs and DVS.

In the Elaboration and Construction phase, the following artifacts were created: Use cases as detailed specification of requirements, reference architecture and database model. Such artifacts served as input to the development of each Sprint.

With the development of the diagram and documents of use cases, a matrix was developed to identify the relationships between the use cases and the requirements. Figure 04 illustrates part of the traceability matrix between requirements and use cases of the project.

Having the project being developed in *Javascript*, the architecture was defined to meet the principles of PROLES, but adapted to the technologies adopted in the project.

Prior to PROLES, the development team met occasionally and informally only at the beginning of the project, with no pre-programmed dates. With the implementation of PROLES, the tasks continued to be discussed and negotiated between the project manager and the team to jointly set the most appropriate deadline.

As proposed by PROLES, the control of the project was carried out by implementing the culture of sending weekly reports, thus maintaining a communication between the team and the project manager, who was often absent from the company. The project manager *feedback* for each report was primordial at establishing the communication process.

During *sprints*, the change and version controls were the main tools of the change management. In order to find out which areas were changed in the software, the team used the version control, documenting what was changed from the previous version to the current one.

In order to validate whether the requirements were developed according to the specification, the developer himself performed the tests on the functionalities developed, which were later validated by the systems analyst. The inconsistencies found were documented by the systems analyst and sent to the developer for correction.

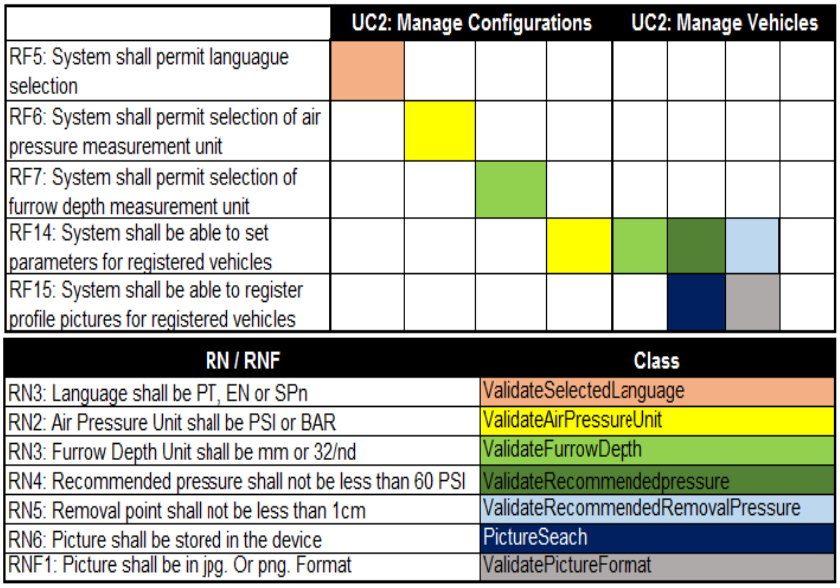


**Figure 4.** Traceability matrix between requirements and use cases of AppX.

In relation to the schedule, it was estimated four months to end the project. Tasks were completed as planned only in the first two *sprints*. The last three weeks were delayed by two weeks each, which represents 30% of the total planned. The low number (5%) of changes in FRs was also evident.

It is worth mentioning that it was the first time that a project had a schedule stipulated from beginning to end, which was an experience for the project manager and the team to get to know better the limitations and performance of each one when facing obstacles.

Some difficulties were observed during the application of PROLES in Company X, many of them due to the fact that the project involves few members and resistance by the team during the first weeks of PROLES implementation. This resistance was even shown by the project manager who was used to delegating tasks and had the power to interrupt the project in order to move others.

When the system analyst in the role of *PROLES Guardian* led the process so that the programmer could get organized, such behavior was interpreted as an imposition of activities, which in some moments, made the systems analyst to be classified as authoritarian, classification previously attributed to the project manager.

It was also observed the difficulty for the project manager to make some decisions to prioritize the use cases to be developed. Prioritization failures were also observed when developing use cases that did not add value to the business. This fact generated great demotivation in the team.

The adoption of the reporting culture was also one of the new habits that faced difficult adaptation. Sometimes, the Jr. developer forgot to send the report or the project manager forgot to read it or did not realize he had received it, which made it difficult to follow the project. In addition, the updates of the tasks in the schedule, often did not happen, which caused the project monitoring to fail.

Resistance to new practices was the greatest difficulty at implementing PROLES. Thus, we understand the importance of knowing the software process before starting the project development, establishing and clarifying the purpose and importance of each member's roles

## 4.2. PROLES in an Academic Laboratory

One of the main challenges of the discipline of SEL at FATEC-MC is to simulate a real-world environment. The topics selected by most of students are e-commerce, academic applications and technical call applications. Common restrictions of students are:

i) they cannot work on the project every day, since they have academic activities;
ii) they do not have useful skills in programming, as well as in software development processes;
iii) they are immature.

Each semester had 6 to 8 *sprints*, ranging from two to three weeks. Table II shows the number of students enrolled, students approved, students who dropped out of the course and students who failed by grade.

**Table 2.** Performance Of Students Working On Projects Using Proles

| Semester | Enrolled | Approved | Gave up | Disapproved |
|---|---|---|---|---|
| 2/2012 | 4 | 2 | 1 | 1 |
| 1/2013 | 49 | 29 | 3 | 6 |
| 2/2013 | 34 | 5 | 23 | 6 |
| 1/2014 | 62 | 17 | 28 | 17 |
| 2/2014 | 72 | 15 | 43 | 14 |
| 1/2015 | 79 | 18 | 51 | 10 |
| 2/2015 | 78 | 14 | 59 | 5 |
| 1/2016 | 99 | 19 | 72 | 8 |
| 2/2016 | 98 | 20 | 43 | 4 |

Table 2 makes it evident that the students who finish the course usually succeed at carrying on the projects and, consequently, are approved. However, there are a large number of dropouts in the discipline. In order to understand the high number of dropouts, we surveyed the data presented during *sprint* deliveries and the weekly reports.

The most detrimental factor for students, as indicated by 61.5% in the execution of the projects, is their time available for dedicating themselves to the project. Students take other disciplines (four other disciplines in average) and the vast majority of them work during the day. The second factor identified as one that generates the greatest difficulty is the familiarity with the chosen technologies, factor which is pointed out by 17.5% of the students.

The defined base architecture is pointed out by 10% of the students as a great impact factor. In the discipline involved in the experiments, the defined architecture works according to Figure 2, discussed in the PROLES *Elaboration and Construction* phase in section 3.

Problems in defining requirements and knowledge of the application domain are pointed out by 6% of the students and only 5% consider the *sprints* time as a critical factor. Among the students who were interviewed, 70.50% stated they have to study during the weeks dedicated to the project. In average, each group dedicated 12 to 16 hours per week on the projects.

The time dedicated to the specification throughout the project was 49% and 51% was dedicated to development. In the course, it is required to present test plans for use cases with higher impact, but 33.5% of students state not to take tests.

Already during the course, it became evident that most students are enthusiastic about problem based methodology with practical activities. All the activities driven by the PROLES customized process.

It became evident that the students who followed the process, respecting the delivery dates, even sometimes not making complete deliveries, were able to complete the project. It was observed that the students who gave up were students who sometimes failed to fulfill a certain stage completely and chose not to submit their work to the teacher, thus not getting feedback, as well as generating an accumulation of activities which led to their giving up the course.

## 5. Conclusions

With the implementation of PROLES in a small-medium company, it was realized that there was a need for the development team to become self-organized, such as groups of students in academic laboratories.

The implementation of PROLES was carried out with a compatible cost to SMEs. This was made possible by reusing the existing resources in the company, restructuring the process of software development, generating self-organization, independence and motivation of the team.

At first, the change of culture generated dissatisfaction among the development team, and it was of great importance their awareness that a defined process is fundamental for the improvement of the final product and also makes the work of all the involved ones, easier. It became evident that in both SBEs

and in academic labs, the PROLES implementation and also any change in a company's software development process, are best accepted when they are less intrusive and easier to manage in small teams.

The proposed reference architecture was fundamental for a better understanding of the division of the system functions into subsystems or modules, as well as the interaction between them. In addition, this reference architecture worked as a facilitator for developers and students with little technical knowledge. Another positive point was the maturity gain of the development team once they gained more knowledge regarding the development and management of software projects, through the application of PROLES and its concepts.

Future works intend to apply PROLES in other companies as well as in different projects, besides inserting alternatives to the practices for projects with differentiated scopes, such as games, big data projects and deep learning projects.

## 6. References


[1] CHAGAS, N. Melhoria do Processo de Desenvolvimento de Software em uma Pequena Empresa Utilizando a Norma NBR ISO/IEC 12207. Maringá: Conclusão do curso de Bacharelado em Informática, 2005. 56p. Cap. 4: Levantamento e Análise dos Dados Obtidos.

[2] PINHEIRO, Diogo Graça. Iniciando o CMMI em uma pequena empresa de engenharia de software. São Paulo: Conclusão do curso de Bacharelado em Engenharia de Produção, 2005.

[3] CORRILO, Mirilian C.A.A, JUBILEU, Andrea P. MPA-DS Modelo de Processo Ágil de Desenvolvimento de Software. 2010

[4] Garcia *et al.* "easYProcess: Um Processo de desenvolvimento para Uso no Ambiente Acadêmico". XII WEI - Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação. 2004

[5] PERERIRA, Igor Muzzetti. Desenvolvendo software inovador em universidades públicas: Adaptando processos ágeis para a realidade de laboratórios de pesquisa e desenvolvimento. 14 de março de 2014

[6] Ochoa et. al. SSP – Simple Software Process. 2010

[7] LEE Seiywan-Seung. AFSP – Agile Framework for Small Projects. 2013

[8] CORILLO, M.; JUBILEU, A. Modelo de processo para micro e pequenas empresas de software com base em metodologias ágeis. 2010

[9] Rahayu, Puji, et al. "Applying usability testing to improving Scrum methodology in develop assistant information system." Information Technology Systems and Innovation (ICITSI), 2016 International Conference on. IEEE, 2016.

[10] Carvalho, W. C. d. S, P. F. Soares, M. d. Soares, M. A. Teixeira da, and L. C. Buiatte, "A Comparative Analysis of the Agile and Traditional Software Development Process Productivity," 2011 30th International Conference of the Chilean Computer Science Society, 2011.

[11] Cho, J., "A Hybrid Software Development Method for Large-Scale Projects: Rational Unified Process with Scrum," Issues in Information Systems 10, 2 (2009).

[12] Bashir, M. S. and M. R. J. Qureshi, "Hybrid Software Development Approach for Small to Medium Scale Projects: RUP, XP & SCRUM," Science International (Lahore) 24, 4 (2012), pp. 381-384.

[13] Nisa, S. U. and M. R. J. Qureshi, "Empirical Estimation of Hybrid Model: A Controlled Case Study," International Journal of Information Technology and Computer Science (IJITCS) 4, 8 (2012), pp. 43-50.

[14] Nuevo, E., M. Piattini, and F. J. Pino, "Scrum-based Methodology for Distributed Software Development," 2011 6th IEEE International Conference on Global Software Engineering (ICGSE), pp. 66-74.

[15] Sarfraz, Muhammad, et al. "Agile Practicing and Outsourcing." IJCSIS International journal of computer science and information security 14.6 (2016): 641-648.

[16] Ionel, N., "Critical Analysis of the Scrum Project Management Methodology," Annals of The University of Oradea, Economic Science Series 17, 4 (2008), pp. 435-441.

[17] Collaris, R. A. and E. Dekker, "Scrum and RUP – A Comparison Doesn't Go on All Fours", Agile Records 1 (January, 2010), pp. 62-65.

[18] Krebs, J., "RUP in the Dialogue with Scrum," http://www.ibm.com/developerworks/rational/library/feb05/krebs/, February 15, 2005, last accessed July 11, 2014.

[19] SILVA, R. R.; KIMURA, F. Y.; LIMA, J. C. ; BERNARDINO, J. Custom Process to Small Business. In: The 29th International Conference on Software Engineering & Knowledge Engineering, 2017, Pittsburgh. 29th International Conference on Software Engineering & Knowledge Engineering (SEKE'17), 2017
[20] Liu, Di, and Zhichao Zhai. "An empirical study of Agile planning critical success factors." (2017).
[21] Tirumala, S. S., Shahid Ali, and Anjan Babu. "A Hybrid Agile model using SCRUM and Feature Driven Development."