Using Extension Projects to Improve Software Engineering Education and Software Quality

The Experience of the "Ricardo de Almeida Falbo" Software Engineering Practices Laboratory

Monalessa P. Barcellos
monalessa@inf.ufes.br
LabES & NEMO, Computer Science Department
Federal University of Espírito Santo
Vitória, ES, Brazil

Vítor E. Silva Souza
vitor.souza@ufes.br
LabES & NEMO, Computer Science Department
Federal University of Espírito Santo
Vitória, ES, Brazil

Patrícia Dockhorn Costa
pdcosta@inf.ufes.br
LabES, Computer Science Department
Federal University of Espírito Santo
Vitória, ES, Brazil

Camila Zacché de Aguiar
camila.z.aguiar@ufes.br
LabES & NEMO, Computer Science Department
Federal University of Espírito Santo
Vitória, ES, Brazil

ABSTRACT

Software Quality (SQ) is a fundamental aspect of Software Engineering (SE) and is approached in different ways and disciplines in Computer Science undergraduate courses. Considering the nature of SQ, its learning should be based on practical experiences. Recently, the curriculum guidelines for undergraduate courses in Brazil were changed to require that 10% of the course workload be dedicated to extension activities. Extension plays a crucial role in students' education, as it aims to provide benefits for society and contribute to a more comprehensive education, including social aspects in addition to technical ones. Considering this scenario, the "Ricardo de Almeida Falbo" Software Engineering Practices Laboratory (LabES) was set up at DI/UFES. It aims to apply SE methods, techniques, and practices to enable students to produce quality software, bringing their education closer to the needs of the various productive sectors and producing results for society. In this paper, we introduce LabES, explain how it works, present some of the produced artifacts and projects, and share lessons learned.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**;

KEYWORDS

Education, Laboratory, Software Engineering, Practice, Extension

ACM Reference Format:
Monalessa P. Barcellos, Vítor E. Silva Souza, Patrícia Dockhorn Costa, and Camila Zacché de Aguiar. 2024. Using Extension Projects to Improve Software Engineering Education and Software Quality: The Experience of the "Ricardo de Almeida Falbo" Software Engineering Practices Laboratory. In *XXIII Brazilian Symposium on Software Quality (SBQS 2024), November 5–8, 2024, Salvador, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3701625.3701680

SBQS 2024, November 5–8, 2024, Salvador, Brazil

ACM ISBN 979-8-4007-1777-2/24/11
https://doi.org/10.1145/3701625.3701680

1 INTRODUCTION

The contemporaneous and digital society has depended more and more on software applications to support people's lives and organizations' activities. The software quality influences the quality of the results obtained when using the software product and is directly related to the practices adopted to engineer it [10]. As the need for quality software grows, the significance of Software Engineering (SE) professionals increases in the industry.

SE is a systematic, disciplined, and quantifiable approach to software development, operation, and maintenance. It emphasizes the need for methods, tools, and best practices to ensure software quality [18]. Hence, to produce quality software, we need proper SE practices, and, to perform such practices, we need good software engineers, who require theoretical and practical knowledge.

Qualified training and the qualification of professionals are increasingly necessary in the society we live in. Specifically in SE education, the quality of professionals is directly related to the quality of education [2]. Additionally, the quality of SE education can contribute significantly to improving the state of the art of software development and help solve problems and crises related to software industry practices [11]. Thus, it is important to teach concepts, processes, techniques, methods, and tools for managing, developing, and maintaining software. Moreover, due to the SE nature, its learning should be based on practical experiences [23]. SE education is a key issue that has implications in the software industry, particularly in emerging countries [24]. Software Quality (SQ) sits at the core of SE as a discipline [6].

In 2014, the Brazilian Education National Plan – PNE 2014-2024 – (law 13.005/2014) established a set of goals aiming at improving the education of Brazilian citizens. As a result of this plan, in December 2018, the Brazilian National Education Board published the Resolution CNE/CES nº 7, which establishes that at least 10% of the workload of undergraduate courses must be dedicated to extension

activities. The undergraduate courses curriculum should adhere to the new extension minimum workload by the end of 2021. The deadline was extended by one year due to the COVID-19 pandemic.

A university is constituted of teaching, research, and extension, and the inseparability of these three pillars characterizes it. However, the latter had been optional for students. With the change in the curricula, extension becomes part of the mandatory training course for all undergraduate students in Brazil. Extension plays a crucial role in students' education. By getting students closer to the external community needs, extension contributes to developing their social and soft skills. In addition, by enabling students to produce solutions to real problems, extension also helps them develop technical (hard) skills. Moreover, extension activities are the main link between the university and society and, thus, allow for providing benefits for the community external to the university.

At the Computer Science Department (DI) of the Federal University of Espírito Santo (UFES), the Computer Science and Computer Engineering undergraduate curricula started to require students to dedicate 10% of the course workload to extension activities in 2022. Extension can be developed through activities in initiatives, such as programs, projects, events, and the provision of services. All the extension initiatives must be recorded in ProEx, the organizational unit devoted to extension at UFES, and the departments are responsible for providing the extension initiatives for the students.

Considering the increase in the demand for extension initiatives, the importance of extension activities to the students' education, the students' interest in roles related to SE,[1] and the relevance of practical experiences to learning SE, we created the "Ricardo de Almeida Falbo" Software Engineering Practices Laboratory (LabES). On one hand, the laboratory aims to enhance the students' education by applying SE methods, techniques, and practices to produce quality software that meets the needs of real users. On the other hand, it seeks to produce software that contributes to society.

As extension activities have only recently turned into a mandatory requirement in many Brazilian undergraduate courses, their use as part of the SQ and SE education has not been much discussed yet. Aiming to take a step towards bridging this gap, in this paper, we report our experience at LabES. We expect to contribute to other education professionals and researchers by sharing practices and materials we have adopted, raising some ideas and concerns, and providing examples of how we have worked in our laboratory.

The paper is organized as follows: Section 2 provides a brief background about the main methods used in LabES; Section 3 discusses related work; Section 4 introduces LabES; Section 5 presents the standard software process we follow in the projects; Section 6 describes how we work at LabES; Section 7 shares lessons learned; and, Section 8 presents our final considerations.

2 BACKGROUND

In this section, we present a brief theoretical introduction to the main SE methods employed at LabES.

PM Canvas. Is a method based on Neuroscience and Project Management foundations to visually represent the main elements of a project plan and the relationships between them. The elements are organized into blocks of fundamental questions (Why, What, Who, How, When, and How much) and represented on a single page. The main purpose of PM Canvas is the agile, collaborative, and clear definition of a project plan. It facilitates understanding of the project, improves communication, increases objectivity, and makes it easier to identify problems [13].

Scrum. Is an agile framework that employs an iterative and incremental approach to optimize predictability and control risks. Scrum involves three roles: *product owner*, played by a person acting on behalf of the client and responsible for maximizing the value of the developed product; *development team*, which is responsible for developing the product; and *Scrum master*, a facilitator who ensures that the development team is provided with an adequate environment to complete the project successfully. The development process is organized in time-box events called *sprints*. Product requirements are recorded and ordered in the *product backlog*, which can constantly change. At the beginning of each sprint, there is a *sprint planning meeting* and the team selects from the backlog the items to be addressed in the sprint and plans the work to be done. The planning result is recorded in the *sprint backlog*. A sprint produces a deliverable product that implements one or more user interactions with the system. During the sprint, the team holds *daily meetings* aiming at optimizing the probability of meeting the sprint goal. Before delivering the increment produced during a sprint, the team performs a *sprint review meeting* to inspect the produced increment and adapt the product backlog if needed. At the end of the sprint, there is a *sprint retrospective meeting*, when the team evaluates and reflects on itself and the project [21] [19].

Continuous Software Engineering (CSE). Consists of a set of practices and tools that support a holistic view of software development to make it faster, iterative, integrated, continuous, and aligned with business [1]. CSE aims at establishing an end-to-end flow between customer demand and the fast delivery of a product or service. The 'big picture' by which this might be achieved goes beyond agile principles and surfaces a more holistic set of continuous activities [9]. In CSE, customers are proactive, and users and other stakeholders are involved in the process, learning from usage data and feedback. Planning is continuous, as is requirements engineering, which focuses on features, modularized architecture and design, and fast realization of changes. Agile practices are employed, including short cycles, continuous integration, continuous delivery, and continuous deployment of releases. There is version control of code, branching strategies, fast commit of code, code coverage, and code reviews. Quality assurance involves automated tests, regular builds, pull requests, audits, and run-time adaption [12].

3 RELATED WORK

Several works (e.g., [3] [17] [8] [7]) have reported the use of projects in SE courses combining theoretical classes with projects addressing real-world problems. However, the dependence on stakeholder participation and the predetermined duration of the curricular component often lead to the projects not being completed. Some works, such as[15] and [16], have reported the experience of using one specific extension project to improve students' knowledge.

[1] A survey performed in March 2023 with Computer Science graduated students of DI/UFES revealed that around 40% of them had played roles related to SE (e.g., developer, project manager).

Among the works we found, only a few have approached extension in the SE context as a continuous initiative, such as a laboratory, program, or extension group. [22] presents the CodeLab-Unifesp extension group, which aims to develop software for social purposes by applying SE techniques and a research and development (R&D) process. The group relies on the active participation of professors, but students assume responsibility for the organization, project management, and technology choice. [4] introduces the Academic Software Factory extension program, which develops software for the university demands, offers consultancy, and supports undergraduate course conclusion work development and internship, where students participate in projects carried out in an environment similar to software organizations.

Comparing our work to [22] and [4], all employ project-based learning (PjBL) [5], use extension projects to teach SE methods and practices, and report continuous initiatives rather than a single project. As for the development process, [4] informs that adopts Scrum and XP, but does not describe the process. [22] adopts an R&D process (problem investigation, solution design, validation, implementation, and evaluation). Only our work describes the process in detail, which, in addition to Scrum, incorporates other methods and practices, such as PM Canvas and CSE. Regarding tools, [4] cites five. Our work is the only one to offer a rich catalog describing in detail the adopted tools and methods – students and the external community can use it. Regarding lessons learned, [22] outlines five (planning flexibility, continuity focus, flexible partnerships, iterative feedback, and use of appropriate technologies and tools). We present 14 new lessons learned from the use of extension projects to improve SE and SQ education.

4 THE LABORATORY

LabES was originally founded in 1999 by Ricardo de Almeida Falbo[2] as the "Software Engineering Research Laboratory". As the name suggests, it focused on research projects. In 2006, the laboratory and its research projects were merged into the Ontology & Conceptual Modeling Research Group (NEMO). In 2021, we decided to "reopen" LabES, now focusing on extension projects. We named the laboratory "Ricardo de Almeida Falbo" Software Engineering Practices Laboratory, to honor and acknowledge the important role Falbo played in the SE community.

Currently, LabES has four senior members (professors) – this paper's authors – and 32 students. Other 42 students were members of LabES in the past three years. At LabES, we employ methods centered on the student and professors act as facilitators. Students are allocated to project teams for project-based learning.

At this moment, five projects are running at LabES: (i) *Favo*: aims to develop an application to foster a culture of innovation in organizations; people can follow innovation actions and take part in ideas programs in a gamified way; (ii) *Hub ES+*: aims to build a virtual creative hub through an online platform that connects the physical spaces of a state innovation structure at Brazilian state *Espírito Santo* with the virtual world, using gamification resources to provide an immersive environment for interaction and training of creatives; (iii) *Marvin*: aims to integrate existing tools at UFES and develop new ones to support the administrative activities of civil servants; (iv) *Seedes*: aims to develop a public web platform for the innovation ecosystem of *Espírito Santo*, collecting data and generating indicators for the local ecosystem; and (v) *SigAMAES*: aims to develop an information system for a nonprofit organization that supports autistic people in *Espírito Santo*; the system is designed to improve the organization of services to the public and to help users manage the institution.

Information about LabES (e.g., its members, projects, contact, localization) is available on its website: https://labes.inf.ufes.br/.

[2]https://nemo.inf.ufes.br/en/equipe/ricardo-de-almeida-falbo/

5 THE STANDARD SOFTWARE PROCESS

We defined a general standard process to use in the projects at LabES. Its purpose is to provide a framework for the main activities to be performed in each project. It was defined considering SE best practices and the context and characteristics of the projects developed at LabES. Usually, at the beginning of the project, there is an agreement about the project scope. Most projects do not have financial support and the students' work is voluntary. Thus, they often work at LabES for some months, until they meet the workload required in the curriculum. Hence, student turnover is high.

The standard process combines incremental [20] and agile approaches, contains activities related to different methods, such as PM Canvas [13], Scrum [21] [19], and CSE [9] [12] [1], and allows for flexibility when applying such methods (i.e., activities can be adapted according to the project's needs). We have defined our own standard process because, after using it in several projects, we could understand its behavior and evaluate and improve it systematically. Moreover, by following the standard process, we would ensure a minimum degree of consistency among the projects. Thus, students would learn a minimum standard set of methods and tools regardless of the project in which they participate. Figure 1 illustrates an overview of the process.

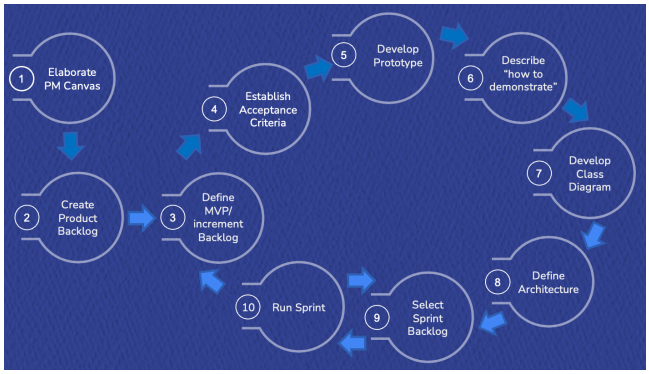

Figure 1: LabES standard process.

The first two activities (*Elaborate PM Canvas* and *Create Product Backlog*) are performed to establish a general view of the project concerning management aspects and the product scope. We use PM Canvas because it provides a simple and fast way to get an overview of management concerns (e.g., risks, team, time, premises). Moreover, it helps introduce the students to project management (the students often start working at LabES before taking a course devoted to this topic). As for the product scope, it is defined in the

product backlog and, at this point of the project, consists mainly of a list of user stories. The user stories are defined at a high granularity level (they can be epics) and are not detailed at this moment. In this activity, the students learn how to elicit and document requirements by following the user story template. To gather requirements, they have the opportunity to talk to the client, the users, or the product owner, which allows them to improve the learning of requirements elicitation by doing it.

Once a general view of the project and product scope is established, an iterative process to produce software increments is performed. In the LabES standard process, an *increment* is a meaningful part of the software, such as an MVP (minimum viable product) or a module. The first activity of the iterative cycle is *Define MVP/Increment Backlog*. In the first iteration, it concerns delimiting the product MVP by selecting the user stories that need to be materialized for producing it and they can be detailed (e.g., epics can be decomposed into atomic user stories). In this activity, the students learn what an MVP is and how to identify the minimum requirements to demonstrate that the software is a viable idea.

The next activity is *Establish Acceptance Criteria* for the selected user stories. In this activity, the user stories are refined by indicating the criteria that must be met when implementing each user story. The criteria will be later used as a basis for tests to evaluate the user story implementation and indicate if it is "done". Then, it is necessary to *Develop Prototypes* that meet the requirements represented in the user stories and respective acceptance criteria and, for each user story, *Describe "how to demonstrate"* it, which consists of describing the expected behaviors of the software when the respective user story is implemented and demonstrated [14].

These activities contribute to students improving their knowledge and skills related to requirements elicitation and documentation. Additionally, they help students explore prototyping practices and supporting tools (e.g., Figma[3]), which are approached in-depth in the last semesters of the curriculum in a non-mandatory course devoted to human-computer interaction. Moreover, by describing "how to demonstrate", the students think about the software behavior. Use case descriptions serve this purpose, but they have not been much used in industry. Hence, we adopted "how to demonstrate" as an alternative to stimulate students to think and document the system behavior (e.g., considering normal, variant, and exception flows). An example of a user story, acceptance criteria, prototype, and "how to demonstrate" produced for a project in LabES is available at https://ufes.es/labes-us-ex.

The next activity is *Develop Class Diagram*, which consists of creating the structural model (UML class diagram) necessary for the MVP. This helps students improve their ability to think in the system at a higher abstraction level, which is key in software development and has been noted as a difficulty for many students. They also experience using tools such as Visual Paradigm and Astah.[4] After developing the class diagram, the classes related to each user story (i.e., the classes that will be used to implement the user story) are identified. Then, it is necessary to *Define Architecture*, which involves defining the technologies to be used and setting the system architecture (e.g., components, patterns). Here, the students have the opportunity to design the system architecture, select frameworks and tools for reuse, and later put them into practice, observing the consequence of each architectural decision taken.

After these activities, the MVP can be produced, which will take several sprints. In *Select Sprint Backlog* the items from the MVP/Increment Backlog that will be addressed in the current sprint are selected and estimated, and in *Run Sprint* they are materialized (i.e., they are implemented, tested, and demonstrated). These two activities involve Scrum practices (e.g., ceremonies) and are repeated several times iteratively until all the MVP/Increment Backlog items are done. At the end of the first iteration, the MVP is finished. These activities allow the students to experience Scrum and software development practices (including coding, testing, continuous integration and delivery, among others), which help improve their technical skills. They can also improve management skills by monitoring project and individual performance using metrics.

Once the MVP is ready, we go back to *Define MVP/Increment Backlog*, select the next software increment to be produced, and perform a new iteration, by repeating the activities for the referred increment. At the end of the last sprint of each increment, the produced increment is delivered (i.e., gets into production).

As it can be noted, the LabES standard process considers two types of iterations. The first one refers to the product increments to be developed (each increment is addressed in one interaction). As explained earlier, the increment represents a meaningful part of the software (e.g., a module). Inside this iteration, there are sprints, which represent iterations performed to produce the increment.

We defined the process in this way aiming to ensure that students would experience several SE activities, instead of focusing only on implementation, for example. We believe that this can better support students learning. Moreover, defining increments helps manage turnover (e.g., we can plan the students' participation by increment). We are aware that this process may not be the most suitable for software organizations in general. However, to better contribute to the students' education through extension projects, we defined a process we considered suitable for our scenario.

6 HOW DOES LABES WORK?

In this section, we provide information about how we work at LabES. In Section 6.1, we explain how students become LabES members. In Section 6.2, we present the LabES Catalog, an important resource used to record and share knowledge. In Section 6.3, we describe how the projects are carried out and show some artifacts used during the projects. In Section 6.4, we refer to the delivery of the projects' results, and, last, in Section 6.5, we provide information about SigAMAES, as an example of a project carried out at LabES.

6.1 Getting into LabES

Once the project is approved by ProEx (the organizational unit devoted to extension at UFES), students can be allocated to it. Students interested in participating in projects at LabES fill in a form available on the LabES website providing information such as their undergraduate course, current semester, and knowledge of SE technologies (e.g., programming languages and frameworks). The professors analyze the students' profiles and the projects' demands

[3]https://www.figma.com/
[4]https://visual-paradigm.com/, https://astah.net/

and, thus, select and allocate students to the projects accordingly. Although it is not a rigid requirement, we have preferred students who have already taken the SE introductory course, which requires some programming courses to be taken first. In this way, the students have a basis of knowledge of SE and can enhance and deepen it through their experience in extension projects. The students usually dedicate from 10 to 20 hours/week to LabES.

When the student becomes a member of LabES, they must go through the *onboarding* process. This process is the first knowledge contribution LabES offers to the students. It consists of studying the main technologies used in the project in which the student will participate. After completing the onboarding tasks, the student is able to work on the project. In general, the onboarding process takes around 20 hours and involves: (i) installing the tools needed to develop in the chosen platform (following instructions from the Catalog – see Section 6.2); (ii) getting the source code and deploying the system locally to test if everything is working properly; (iii) leveling the student's knowledge in the technologies used by the project; (iv) familiarizing the student with other aspects of the project, such as code conventions, testing practices, Git flow, project management, etc. We provide the students with the material they need to perform the onboarding tasks.

6.2 The LabES Catalog

The methods and tools adopted at LabES and the procedures followed to use them are described in the **LabES Catalog** (available at https://labes.inf.ufes.br/catalogo/). For example, a detailed description of the standard process is available there for the students. Other items addressed in the Catalog are methods (e.g., Scrum, PM Canvas), tools (e.g., Git, GitKraken, GitLab, GitLab Pipelines, MAMP, Visual Studio Code), software procedures (e.g., conventional commits, GitLab Flow), configuration and procedures for specific platforms (Jakarta EE, PHP & WordPress), among others.

The Catalog (currently available only in Portuguese) is a knowledge repository and items can be created by professors and students. When a student creates an item, it is verified by a professor and only after approval, it is made available in the Catalog. Figure 2 depicts a fragment of the Catalog item referring to PM Canvas.

The Catalog is an important instrument to enhance students' education. On one hand, it provides an easy way to access useful knowledge and apply it to the projects. By using knowledge recorded in the Catalog items, students increase their knowledge of SE and are better equipped for building quality software. Moreover, the Catalog helps to ensure consistency and standardization in the way the methods and tools are used at LabES. On the other hand, the students are stimulated to share knowledge. To record a new item in the Catalog, the student needs to organize and structure knowledge and make it explicit in such a way that other people can use it. This also contributes to consolidating knowledge and improving students' communication skills.

6.3 Running Projects at LabES

Each project has a professor as its coordinator. Additionally, the project team has the following roles: project manager (or Scrum master), client representative (or product owner), technical leader, and developers. When necessary, the same person plays more than one role (e.g., the professor can be the coordinator and the technical leader). Due to specificities, projects adopt the standard process presented in Section 5 making adaptations. Sometimes, the adaptations occur at the beginning of the project (e.g., an MVP might not be necessary for a particular project). Other times, adaptations may be needed during the project (i.e., "on the fly"). Adapting the process is a learning opportunity for the students. Sadly, complaints that software processes are too rigid and cause bottlenecks in the execution of projects are not uncommon among practitioners. Therefore, by realizing that it is allowed to adjust the process when needed, the students experience flexibility without chaos. They learn that performing a process that meets the project needs is more important than following the process "by the book" and that it is possible to adjust the process by following SE practices.

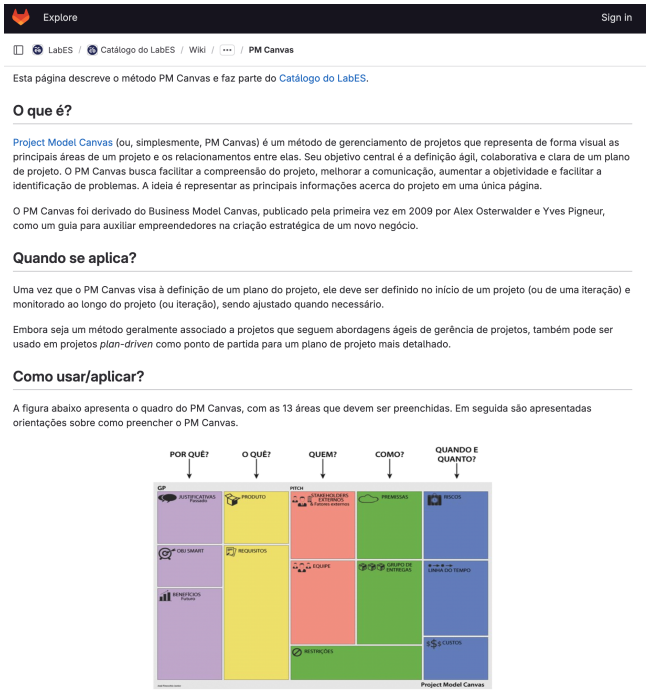

Figure 2: Example of an item of the LabES Catalog.

To manage the project, the coordinator together with the team chooses a management tool to associate with GitLab,[5] which is a standard tool used at LabES. For example, one project has used Trello[6] and connected it to GitLab by using a plugin. As the free version of Trello is limited to a 10-member team, we defined a simple and cost-free management approach combining spreadsheets (based on [14]) with GitLab. Next, we briefly explain two of them.

The "*Sprints*" spreadsheet provides an overall view of the project sprints, by recording information such as each sprint goal, start and end date, available effort (considering the developers' workload), estimated effort (considering the items to be addressed in the sprint and the respective estimate). The information is provided by the team in the sprint planning meeting, which occurs in the

[5]We use a self-managed instance of GitLab, cf. https://about.gitlab.com/install/
[6]https://trello.com

Select Sprint Backlog activity. For each sprint, we create a milestone in GitLab[7] and we link the sprint number to the corresponding milestone. In this way, we easily connect the spreadsheet to GitLab. Figure 3 illustrates an example of the *Sprints* spreadsheet.

The *Backlog* spreadsheet represents the backlog items. It is divided into three sections. In the middle are the items that constitute the *MVP/Increment Backlog* and were recorded in the *Define MVP/Increment Backlog* activity. At the top are the items selected to be addressed in the current sprint (i.e., the *Sprint Backlog*). They are moved from the *MVP/Increment Backlog* in the sprint planning meeting. At the bottom are the items already "done". They are moved from the *Sprint Backlog* at the end of the sprint (in *Run Sprint* activity), when the respective produced results are demonstrated to the product owner and, if approved, are considered done. We use a yellow background to identify under-development items (i.e., work in progress) and a green background to identify concluded items. The number of the items of the *Sprint Backlog* are linked to the corresponding issues in GitLab. In the sprint planning meeting, a student is assigned to each item of the *Sprint Backlog* and the estimates are done. When the student finishes the item, they record the spent effort. The estimated and actual efforts informed in the *Backlog* spreadsheet are used to calculate metrics in the *Sprints* spreadsheet. Figure 4 illustrates an example of the *Backlog* spreadsheet.

Using the spreadsheets to complement GitLab has helped us to deal with management issues. The spreadsheets are used in weekly meetings. We have adopted weekly meetings because daily meetings have not been suitable for our teams. Students work at different times and several of them dedicate 10 hours a week to the projects. Thus, performing meetings once a week has been more appropriate. By using the spreadsheets, the students have a centralized and consolidated view of the project from the management perspective and can observe how they have performed. They fill in the cells and observe how the metrics values change. By analyzing their performance, they can better estimate the next sprints. Performing management activities contributes to broadening the students' understanding of the project and enhancing their competencies.

The spreadsheets are also helpful in calculating the extension workload of each student. The workload value is given by the sum of the actual effort spent by the student on the backlog items done by them, the onboarding tasks, and project meetings. We need to inform to ProEx the number of hours the student dedicated to extension activities so that it will be possible to issue the student's certificate needed to fulfill the curriculum requirement.

For software development, we have adopted version control of code, branching strategies, fast commit of code, code coverage, and code reviews, and also some CSE practices, mainly Continuous Integration (CI) and Continuous Delivery (CD). In a server managed by DI/UFES, a virtual machine was provisioned and registered in GitLab as a *runner*. For each project, a .gitlab-ci.yml file is created to define pipelines, i.e., a sequence of automated tasks to be executed in response to certain events. When developing new features, students create Git branches and each commit sent to GitLab usually triggers unit tests to let developers know if anything was broken. Once the code is ready for review, a merge request is marked as ready and a colleague student is asked to review the code. After approved and merged into the main development branch, the full pipeline with tests and delivery to the *runner* server is performed, using Docker[8] containers, which can be managed using the Portainer[9] management front end. This brings students closer to practices currently used in the software development industry.

6.4 Delivering Results

The results produced in the projects are delivered to the clients. As explained in Section 5, we follow an incremental process, thus, the results are delivered gradually, as increments. The participation of the students in this moment is very important because they can see the software they produced being used in practice and adding value for the client. Moreover, this is the moment in which the purpose of the extension project is materialized by providing benefits to society. After the software is delivered, the project team is also responsible for its operation and, thus, performs maintenance tasks.

Exploring delivering and maintenance activities in SE courses is not very common, even when using PjBL approaches, because it requires the project (or part of it) to be concluded, the software gets into operation, and the client/user reports issues. By participating in extension projects, the students can experience these activities in practice, growing their knowledge and experience.

6.5 SigAMAES: a Project developed in LabES

The SigAMAES project aims at developing an information system for a nonprofit organization that supports autistic people. The initial scope of the system consists of three modules: (i) an autistic registration module, which allows for registering information regarding personal and medical information, as well as information relevant to social services (since the majority of the autistic people supported by the organization belongs to socially vulnerable groups, the system is required to gather a long list of information that may be used to support these persons and/or to generate statistics); (ii) a service/therapy registration module, which allows the registration of the services/therapies offered by the organization (e.g., occupational and speech therapies); the system is capable of managing therapy sessions schedules and subscriptions, as well as waiting lists; since the demand for services/therapies is much larger than the current availability, managing waiting lists and priorities is paramount to the organization; and (iii) a collaborators registration module, which allows the registration of persons working for the organization as well as their responsibilities (for example, it is possible to assign particular therapies to certain collaborators); collaborators are registered according to predefined roles which, in turn, determine their permission levels in the system. Figure 5 depicts the login screen of the system, which allows collaborators to log in and experience the system according to their pre-defined roles. Figure 6 shows the agenda for selected therapies. This allows the visualization and organization of therapy sessions throughout the week. Currently, the system is available only in Portuguese.

About 24 students have worked on the SigAMAES project. The development process has followed the methods described in Section 5 with little adaptation. We have fully embraced the principles of agile approaches and some CSE practices, with the assistance of

[7] Cf. https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/.

[8] https://www.docker.com/

[9] https://www.portainer.io/

# Sprint	Goal	Start	End	Dev workload per week	Number of weeks	Number of devs	Available workload	Estimated effort	Throughput (added value)	% delivered	Actual effort (done items)	Actual - estimated (done items)
1	Develop adm funcionalities - Part I	27/04/2023	11/05/2023	10	2	3	60	60	30	50	31	-1
2	Develop adm funcionalities - Part II	12/05/2023	25/05/2023	10	2	3	60	60	45	75	38	7
3	Develop adm funcionalities - Part III	26/05/2023	15/06/2023	10	3	3	90	90	22	24	26	-4
4	Develop registration funcionalities - Part I	16/06/2023	29/06/2023	10	2	3	60	60	24	40	24	0
5	Develop registration funcionalities - Part II	30/06/2023	20/07/2023	10	3	3	90	68	22	32	24	-2
6	Enable activities plan generation - Teaching workload	23/08/2023	05/09/2023	10	2	3	60	48	13	27	28	-15
...	...	...	....	...	...	...	...	...	...	...	...	...
15	Enable activities plan exportation (*ongoing*)	17/07/2024	30/07/2024	10	3	3	90	61				

Figure 3: Fragment of the *Sprints* spreadsheet.

Sprint	Item	Backlog Item	Imp	Est	Real	How to demonstrate / comments	Dev
Sprint Backlog							
15	#45	TD: Add Code attribute to classes referring to PG programs and undergraduate courses	100	3		Adjust the system Admin's class model to reflect the change made in issue #18	Dev5
15	#41	US-11: As an academic, I want to export my activities plan in Excel format to edit it and sent it to the management center.	100	30		See "how to demonstrate" in the specification requirements document available at https://docs.google.com/document/d/1Yf9YH3P3PhkfZ2	Dev4
...	...	...	...	...	...	...	...
MVP/Increment Backlog							
		US-12: As an academic, I want to generate my activities plan in PDF format to sign it sent it to the management center.	100			See "how to demonstrate" in the specification requirements document available at https://docs.google.com/document/d/1Yf9YH3P3PhkfZ2	
		TD: Improve workload calculation	90			Add fields to the plan screen to show the workload for each category. When changing the items added to the plans, the values must be recalculated.	
		T: Describe the procedure with instructions for using the system MVP.	80			Prepare document for user to use the system MVP.	
		...	...			...	...
Finished Sprints (done items)							
1	#5	US-A5: Record enrollment in undergratuade course	80	8	10	When you click "Administration" and then "Manage Undergraduate Course" in	Dev1
1	#14	US-A14: Record department	100	8	8	By clicking "Register Department" you will be taken to a page where all current	Dev2
1	#10	US-A10: Record department coordinator	70	12	12	By clicking "Register Department" you will be taken to a page where all	Dev1
...	...	...	...	...	...	...	...

Linked to the corresponding milestone in GitLab

Linked to the corresponding issue in GitLab

US: User Story; TD: Technical Debt; T: Task

Figure 4: Fragment of the *Backlog* spreadsheet.

Figure 5: The login screen of the SigAMAES system.

Figure 6: The agenda screen of the SigAMAES system.

tools such as Trello and GitLab. For the implementation technologies, we have chosen a modern stack, which includes the frameworks Next.js[10] for the front end, Java Spring Boot[11] for the back end, and PostgreSQL[12] as the relational database. In addition to being responsible for the entire development process, the team also manages systems deployment issues for both environments,

[10]https://nextjs.org/
[11]https://spring.io/projects/spring-boot
[12]https://www.postgresql.org/

development, and production. We keep the development environment running in the infrastructure of DI/UFES, which facilitates configuration and updates and is cost-free. As for the production environment, to allow robustness and availability, the organization covers the costs of a cloud service, which allows the deployment of both the front-end and the back-end of the system in production (including the database and a bucket service for files).

The development trajectory of the SigAMAES project has been longer than expected, due to (i) student high turnover (as mentioned before), and (ii) the organization extreme growth, which has heavily impacted the system requirements. Fortunately, we have overcome all challenges and the first version of the system was delivered in May 2024. Since then, the organization has been using the system on a trial basis, i.e., in parallel with their old methods (based on paper and low-structured spreadsheets). In order to allow feedback from the clients on possible systems errors, unwanted behaviors, and desired new functionality, we have created a "ticket system" based on Google Forms: the client can open "ticket requests" from the development team by answering the questions in the form, which allows, among other things, the definition of priorities. Figure 7 depicts a picture of the SigAMAES launching event, kindly offered by the organization as a symbol of appreciation for our work.

Figure 7: The SigAMAES launching event.

As for the students learning, this project has offered an invaluable experience on various fronts: (i) the possibility of interacting with real clients and facing the intrinsic challenges involved in eliciting requirements; (ii) the opportunity to learn and work with modern software engineering methods and implementation technologies that are highly desired in the current job market; the learning process is carried out in a relatively safe environment in the sense that students do not risk "losing their jobs" in case of possible mistakes or failures; (iii) the possibility of playing different roles within the project, from junior developer to team leader, depending on the time and/or dedication to the project; and (iv) the chance to see the results of their own work in practice, directly helping an organization that offers such a noble service to society. In fact, several students have informally reported great personal experiences such as: "*... with the knowledge that I have acquired in the project, I was able to succeed in a job recruitment process.*", "*... the leadership experience I have obtained in the project helped me to get a new position in my current job.*", or "*... it is incredible to see the results of my work being directly used to help people*".

As for the future, other modules of the system are under development and should be delivered similarly as mentioned above. As a result of the trust relationship established between LabES and the organization, other projects are being discussed to help the organization on other fronts.

7 LESSONS LEARNED

In this section, we share some lessons we have learned by working at LabES in the last three years. To collect the lessons, the LabES professors hold regular meetings as project coordinators and gather student feedback. Moreover, they observe in practice what goes right, what goes wrong, challenges, etc. These observations are shared and discussed with students and other professors in meetings and chat channels. Once there is an agreement, they are synthesized and recorded. In the text below, we often adopt *should* instead of *must* because the lessons are based on our experiences and perspectives. Further studies may be needed to corroborate them.

The tech lead is key to enhancing technical skills and ensuring software quality: Having a person with knowledge of and experience with the technologies adopted in the project (e.g., programming language, frameworks, database system) is very important. The tech lead contributes to students' learning and to increasing their competencies in the referred technologies. Moreover, the tech lead minimizes risks related to technology issues, which could cause delays in the project or affect the software quality. If the students want to work with new technologies and there is no expert, the new technologies introduction must be carefully evaluated because the lack of a specialist can lead to technical difficulties that hamper project performance and software quality. Although it is important for students to learn new technologies, in extension projects where there is a commitment to external clients, there needs to be a fine balance between learning (e.g., novel technologies) and project goals (e.g., project schedule, product quality).

High student turnover is challenging for learning and project goals: High turnover impacts any software development project and can affect its goals (e.g., cost, time, quality). When it occurs in extension projects that support SE education, it also affects the learning goals. If the student works on a project at LabES for too short a time, they will not experience the different SE activities involved in the project. Although the standard process has been defined as an attempt to minimize this problem, it is not enough to solve it. For example, a student can still join the project during the sprints performed to develop a project's increment and leave the project before that increment has been delivered. As a result, their experience will be more limited and affect their learning. Requiring the students to have a minimum number of hours dedicated to the project would help, but still not solve the problem, because the student can get other opportunities during the project (e.g., a job, an internship) and they should not be prevented from embracing them. Therefore, turnover can be inevitable.

Extension hours is the main currency to "pay" students for their work: A reason for the high turnover is the high demand

for software developers and IT professionals in general, which causes many students to already have internships or even jobs by the time they join a LabES project. It is common for high-demand periods at their jobs or exams in their undergraduate course to cause them to produce less in the project. Considering a fixed number of hours per week and granting it to students might lead them to achieve the hours they need without contributing much to the project. Using a time-tracking system (GitLab comes with one integrated with project tasks) allows students to more accurately register the time dedicated to the project and the coordinator to assess it, also allowing students to go through low availability phases unconcerned, as they are "paid" with the actual hours they dedicate.

A low weekly workload dedicated to the extension project slows down learning: Several students are not on scholarships, are enrolled in courses, and sometimes perform other activities. Hence, they have only a few hours a week dedicated to LabES. This affects the progress of the project (it is slower), as well as the students' learning. In our experience, any meaningful contribution to a project requires at least 10 hours per week of dedication. Hence, we require a minimum of 10 hours a week for students who do not receive scholarships. By doing that, we ensure the smallest workload that enables the student to engage in the project activities and enhance their learning and these students still have time to spend on paid activities if they need to.

Extension projects are unlike other projects used to support learning: Extension project characteristics provide an environment that promotes broader learning opportunities. For example, projects used during specific courses (e.g., a project used to teach requirements elicitation in a Software Requirements course) are usually developed within the time frame of that course, and some adaptations to fit the course needs are made (e.g., some activities, such as delivery and maintenance, could be disregarded). Furthermore, in projects used in specific courses, the project team often remains the same. Extension projects enable the student to participate in all stages of the project (even the ones less explored in specific courses). Additionally, the students have to deal with changes over the project, such as in the project scope and team (although there is a high turnover, some project members stay on longer and need to keep up with the turnover). Hence, extension projects contribute to a complementary and broader experience in SE, enabling students to connect knowledge of different pieces of the software process (e.g., requirements, architecture, coding, testing, management) into a more comprehensive view of how to develop, manage, and maintain software.

Professors act as facilitators, students should take charge of projects: For students to have a deeper experience, it is important that they feel (and are) responsible for the projects. Often, when students work on internships, they focus on specific activities (e.g., coding a functionality), especially if they are in the early stages of the undergraduate course. In LabES projects, the students have the opportunity to participate in all stages of the project and the team is responsible for the project. This sense of responsibility and belonging contributes to learning, as it motivates students to learn more and produce better results.

A catalog is a good learning resource: A catalog of tools, practices, and methods adopted in the projects is a good resource for recording and sharing knowledge. It helps those who use knowledge recorded in it (who learn how to do something – e.g., how to use a particular method or technology) and also those who record knowledge in it (who reinforce their knowledge by making it explicit to transfer to others). However, it is necessary to encourage the students to record knowledge in the catalog, as they generally do not do it proactively. At LabES, we often do that in meetings. For example, the student does something interesting (e.g., setting up an environment using new technologies) and when they tell the team about it in a meeting, the professor encourages them to share that knowledge by registering it in the catalog.

Celebrating achievements together stimulates learning and better results: The achievements (e.g., the delivery of an increment) should be celebrated with the team and other members of the laboratory. The results should be communicated (e.g., project website, social media), the lessons learned should be shared, and the team should feel good about the achievement. This contributes to the sense of belonging and the perception of recognition which, in turn, serve as motivators for students to learn more and do better.

The participation of students in moments with the client is fundamental: It is important for students to see the client receive the product, and realize how important the delivered software is and the value it adds to the client. This also motivates the students and makes them want to learn more and improve the outcomes. In addition, it makes learning more comprehensive, as students may not have contact with the client in other contexts (e.g., in internships, students can focus on implementing functionalities assigned to them and do not deal directly with the client or see how the produced software is used in production).

Considering the student's knowledge level is important for them to follow an appropriate learning path: LabES receives students with different levels of knowledge, from beginners in SE to final-year students who need to complete extension hours. It is necessary to analyze each student's profile and assign them to projects in roles that contribute to their training. For example, less experienced students often start by developing simpler functionalities and being helped by other students and professors. As the student progresses, they perform other activities and participate in the whole software process. More experienced students, in turn, are assigned to roles that require higher expertise and can contribute by sharing knowledge with others. It should be noted that even more experienced students improve their knowledge of SE when working on extension projects, as they experience new challenges when working on new environments, new problem domains, new technologies, new methods, etc.

Associating undergraduate course conclusion work with extension projects is a win-win strategy: Undergraduate course conclusion works can be developed in the context of extension projects. This practice has proved to be positive for both, the project and the student. On the project side, as the student needs a year to do the work (this is the time required in the undergraduate courses at DI/UFES), they will spend at least a year on the project, thus, reducing turnover. Moreover, as the student needs to produce results on which they depend to graduate, they will have an extra motivation to be committed to the project and produce quality results. On the other hand, the student will benefit from a real project and will explore a real problem in their work. In addition,

they can take advantage of the entire structure (e.g., hardware, software, knowledge) of the laboratory to develop their work.

Extension projects are opportunities for research: As a software development environment, extension projects offer opportunities for research. New SE practices, methods, and tools can be experimented within extension projects. This contributes to advances in SE state of the art and also to the extension projects and education/training of students, who will have access to innovative practices, methods, and tools.

Extension projects should be aligned with the curriculum: Although extension projects do not need to be associated with specific courses of the curriculum, they should be aligned with the ones related to SE to avoid contradiction and ensure that extension projects and SE-related courses are complementary. Extension projects contribute to deeper, comprehensive, and practical learning. For that, they should allow the students to go deeper into methods, tools, and practices approached in SE-related courses, connect them to the overall software process, and learn new ones. At LabES, students are allocated to projects based on demand. Thus, we receive students from different undergraduate courses and stages, which would not allow us to associate projects with specific courses. Three professors who teach SE courses at DI/UFES are LabES members and are responsible for ensuring alignment between the courses and LabES projects.

A clear and open dialog with the client is crucial: Although we consider risk issues at the beginning of the projects, it may be necessary to manage the client's expectations concerning the delivery schedule and/or product scope. As discussed earlier, since the projects often have no funding, student turnover is high, which may cause delays beyond planned. In addition, students are still learning and some may take more time than others in the process. Such issues should be stated clearly with clients in advance and, in our experience, due to the nature of such projects, these challenges can be made transparent to the clients during software development. Doing that can help avoid stressful situations for clients, professors, and students. Another issue that should be clear since the beginning of the project refers to the software production environment and further maintenance. Often, clients are nonprofit organizations or small businesses that have little or no computational infrastructure available. Therefore, considerations as to where the software is going to run in production or who is providing maintenance after delivery should be discussed in advance.

8 FINAL CONSIDERATIONS

Extension, together with teaching and research, constitute the triple pillar of universities. By recognizing the important role of extension in education, in 2018, the Brazilian National Education Board established that students must fulfill at least 10% of the workload of their undergraduate courses in extension activities. All undergraduate courses should adhere to this requirement by the end of 2022.

In this paper, we reported our experience at LabES, a laboratory dedicated to SE practices that aims at contributing to the education and training of Computer Science and Computer Engineering students of DI/UFES through extension projects that provide software products for society. By offering opportunities for the students to participate in extension projects, LabES aids in improving the students' knowledge, competencies, and skills and enables them to produce quality software.

Although the use of projects as a means to support teaching and learning is not new, using extension projects to support SE and SQ education and training has not been much explored. Due to the particular characteristics of such projects, it is necessary to grow knowledge on this subject to help education professionals and researchers. By being better educated and trained, students will become better software engineers, able to produce quality software. We expect that by sharing our experiences, perceptions, and some of the challenges we have faced we can help other education professionals interested in using extension projects to enhance SE and SQ education.

The practices and lessons reported in this paper are case-based, i.e., based on our experiences at LabES and perceptions over the last three years. A recognized limitation in this context is the ability to generalize from the case-specific to different cases. Wieringa and Daneva [25] argue that in such cases generalization can be established for similar cases and, although are not universal, they are useful in practice. Hence, although we cannot ensure that the lessons apply to broader contexts, they can be useful in contexts similar to ours. Even so, bias and lack of sound evidence should be considered as limitations of this work. There is still a need to conduct studies to collect the students' perceptions and evaluate how their participation in LabES projects has contributed to improving their education and developing competencies and skills. This will be addressed in future actions. Moreover, we must point out that the standard process still needs a robust evaluation. It has been evaluated in each project based mainly on the rate of deliverables produced as expected and student feedback gathered in regular meetings. So far, the evaluation has been mainly qualitative. To improve the process evaluation, we plan to define a set of metrics for quantitative assessment.

There have been only three years since LabES was created as a laboratory dedicated to extension. We are aware that we have just started our journey at LabES and we have a long path ahead. We are learning from the experiences and improving our practices accordingly. We intend to keep developing extension projects and extend and strengthen the connection with the NEMO Research Group, where SE research is conducted at UFES. Three professors members of LabES are also members of NEMO, which facilitates collaboration. Moreover, we intend to explore new technologies (e.g., the use of Artificial Intelligence-based tools) in projects, to improve students' education and produce software for society.

ACKNOWLEDGMENTS

We thank our dear friend and mentor Ricardo de Almeida Falbo, who inspired us as an example of a great person, Software Engineering professor, and researcher. We also thank DI/UFES for supporting us in creating LabES and providing us with the space we needed, the project partners, and the students. Finally, we thank the Coordination for the Improvement of Higher Education Personnel - Brazil - CAPES (Finance Code 001) and the Espírito Santo Research and Innovation Support Foundation - FAPES (Processes 2023-5L1FC, 2022-NGKM5, 2021-GL60J).

REFERENCES

[1] Monalessa P. Barcellos. 2020. Towards a Framework for Continuous Software Engineering. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering* (Natal, Brazil) *(SBES '20)*. ACM, New York, NY, USA, 626–631. https://doi.org/10.1145/3422392.3422469

[2] K. Beckman, N. Coulter, S. Khajenoori, and N.R. Mead. 1997. Collaborations: Closing the Industry-Academia Gap. *IEEE Software* 14, 6 (1997), 49–57.

[3] Andréa Sabedra Bordin, Lorenzo Mendes Rodrigues, and Tarcisio Casagrande. 2023. Ensino, Pesquisa e Extensão no Ensino de Engenharia de Software: Um Relato de Experiência. In *Anais do XXXI Workshop sobre Educação em Computação*. SBC, 30–40.

[4] Karen Selbach Borges, Tanisi Pereira de Carvalho, and Márcia A Corrêa de Moraes. 2012. Programa de Extensão "Fábrica de Software Acadêmica": Contribuindo para a Formação Profissional na área da Informática. In *Workshop sobre Educação em Computação (WEI)*. SBC, 151–159.

[5] Andreas Breiter, Görschwin Fey, and Rolf Drechsler. 2005. Project-based learning in student teams in Computer Science Education. *Facta universitatis-series: Electronics and Energetics* 18, 2 (2005), 165–180.

[6] Stanislav Chren, Martin Macák, Bruno Rossi, and Barbora Buhnova. 2022. Evaluating Code Improvements in Software Quality Course Projects. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering* (Gothenburg, Sweden) *(EASE '22)*. ACM, New York, NY, USA, 160–169. https://doi.org/10.1145/3530019.3530036

[7] Thelma Colanzi, Leandro Silva, Andressa Medeiros, Paulo Gonçalves, Eniuce Souza, Douglas Farias, and Greicy Amaral. 2023. Practicing the Extension in Software Engineering Education: an Experience Report. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. 514–523.

[8] Simone C Dos Santos, Maria da Conceição Moraes Batista, Ana Paula C Cavalcanti, Jones O Albuquerque, and Silvio RL Meira. 2009. Applying PBL in Software Engineering Education. In *2009 22nd Conference on Software Engineering Education and Training*. IEEE, 182–189.

[9] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous Software Engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189. https://doi.org/10.1016/j.jss.2015.06.063

[10] Alfonso Fuggetta. 2000. Software Process: a Roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (Limerick, Ireland) *(ICSE '00)*. ACM, New York, NY, USA, 25–34. https://doi.org/10.1145/336512.336521

[11] W. Gibbs. 1994. Software's Chronic Crisis. *Scientific American* 271 (1994), 86–95. Issue 3.

[12] Jan Ole Johanssen, Anja Kleebaum, Barbara Paech, and Bernd Bruegge. 2019. Continuous Software Engineering and its Support by Usage and Decision Knowledge: An Interview Study with Practitioners. *Journal of Software: Evolution and Process* 31, 5 (2019). https://doi.org/10.1002/smr.2169

[13] José Finocchio Júnior. 2013. *Project Model Canvas*. Alta Books.

[14] Henrik Kniberg. 2015. *Scrum and XP from the Trenches, 2nd Edition*. Lulu.com.

[15] Bhruno Roan Leifheit, Cassio Ceolin Junior, Daniel Oliveira de Freitas, Bianca Maia Ribeiro, Danielly Cristina do Carmo Neves, Sabrina Rodrigues Fernandes, Williamson Alison Freitas Silva, and Fábio Paulo Basso. 2023. Relato de Experiência no Projeto Engenharia de Software Aplicada à Causas Sociais: AVICO Brasil. In *Anais da VII Escola Regional de Engenharia de Software*. SBC, 338–347.

[16] Ítalo Jonas Lima, José Vitor Silva, Letícia de Oliveira, and André Silva. 2020. Um relato de experiência da Extensão Universitária como prática formativa de estudantes de Sistemas de Informação. In *Anais da XX Escola Regional de Computação Bahia, Alagoas e Sergipe*. SBC, 263–271.

[17] Jonnathan Lopes, Gabriela Medeiros, Dienefer Fialho, and Andréa Bordin. 2017. Resoluçao de problemas no curso de Engenharia de Software: Uma experiência envolvendo Extensão e Ensino. In *Escola Regional de Engenharia de Software (ERES)*. SBC, 97–104.

[18] R.S. Pressman and B.R. Maxim. 2019. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

[19] L. Rising and N.S. Janoff. 2000. The Scrum software development process for small teams. *IEEE Software* 17, 4 (2000), 26–32. https://doi.org/10.1109/52.854065

[20] Nayan B. Ruparelia. 2010. Software Development Lifecycle Models. *SIGSOFT Software Engineering Notes* 35, 3 (may 2010), 8–13. https://doi.org/10.1145/1764810.1764814

[21] Ken Schwaber and Jeff Sutherland. 2020. *The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org.

[22] Denise Stringhini, Daniel AV de Salis, Danilo S Alexandre, Milena de M Siqueira, Felipe B Guerra, and Tiago de Oliveira. 2024. Desenvolvimento de Software com Fins Sociais: Relato de Experiência em Projetos de Extensão Universitária. In *Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES)*. SBC, 82–93.

[23] Kun Tian, Kendra Cooper, and Kang Zhang. 2011. Improving Software Engineering Education through Enhanced Practical Experiences. In *2011 10th IEEE/ACIS International Conference on Computer and Information Science*. 292–297. https://doi.org/10.1109/ICIS.2011.53

[24] L. Vives, K. Melendez, and A. Dávila. 2022. ISO/IEC 29110 and Software Engineering Education: A Systematic Mapping Study. *Programming and Computer Software* 48 (2022), 745–755. Issue 8. https://doi.org/10.1134/S0361768822080229

[25] Roel Wieringa and Maya Daneva. 2015. Six strategies for generalizing software engineering theories. *Science of Computer Programming* 101 (2015), 136–152. https://doi.org/10.1016/j.scico.2014.11.013