LEDS: The Rise of the (Collective) Force — 10 Years of Impact on Innovation and Education in Software Engineering

What we Have Learned by Integrating Teaching, Research, and Extension in Higher Education

Paulo Sérgio dos Santos Júnior
LEDS Research, Department of Informatics, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, Brazil
paulo.junior@ifes.edu.br

Rodrigo Fernandes Calhau
LEDS Research, Department of Informatics, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, Brazil
rodrigocalhau@ifes.edu.br

Moisés Savedra Omena
LEDS Research, Department of Informatics, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, Brazil
omenam@ifes.edu.br

Felipe Frechiani de Oliveira
LEDS Research, Department of Informatics, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, Brazil
felipe.frechiani@ifes.edu.br

Vanessa Battestin
Reference Centre for Training and Distance Education (CEFOR), Federal Institute of Education, Science and Technology of Espírito Santo
Vitória, Brazil
vanessa@ifes.edu.br

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

ABSTRACT

Over the past decade, the Laboratory for Extension in Development of Solutions (LEDS) has evolved from a small local initiative into an academic ecosystem that integrates teaching, research, and extension in Software Engineering. Founded in 2012 at the Federal Institute of Espírito Santo (IFES), LEDS simulates a real-world software development environment where students engage in multidisciplinary projects that serve public and private sector demands. Through project-based learning and interdisciplinary collaboration, LEDS fosters the development of both technical competencies and essential soft skills, enhancing student engagement and preparing them to face real-world challenges. This paper revisits LEDS over ten years after its inception, highlighting its organizational practices and impacts on student education and institutional innovation, through the creation of a research and extension culture grounded in experiential learning. Finally, we reflect on the lessons learned and challenges and offer practical insights for education professionals and researchers seeking to implement academic software factories as drivers of Software Engineering educational transformation and industry collaboration.

KEYWORDS

Software Engineering, Education, Project-Based Learning, Non-formal Learning Environments, Academic Software Factory

1 Introduction

Nowadays, software is part of everybody's lives and a critical asset to software organizations. New technologies and demands constantly arise, increasing the exigency for better software, and better processes and tools to develop it. Educating good software engineers is essential to produce quality software [3].

The market's demand for qualified software engineers and the benefits that working with projects brings to student education have led some universities and institutes to create academic software factories [23] (e.g., LabES [3], Demola [6], Mandacaru.dev [13], and ProAcademy [32]) . This way, academia can approach industry needs both in theoretical and practical knowledge, thus developing soft and hard skills and adding value to the students.

In this context, the Laboratory for Extension [1] in Development of Solutions (LEDS)[2], was created in 2012, as an initiative of teachers from the Department of Informatics of Serra campus at Federal Institute of Education, Science and Technology of Espírito Santo (IFES), with the goal of providing an experimental environment similar to a software factory, bridging the gap between academic teaching and the job market [5].

In this environment, students have the opportunity to engage in extension and research projects and experience technological innovation, contributing to produce solutions for public and private organizations. Moreover, by playing several roles in real-world projects, students and teachers develop hard (e.g., by adopting classic and new software engineering practices and technologies) and soft skills (e.g., by exploring teamwork, communication, time management, and professional responsibility). At LEDS, the projects are developed to meet the demands of real clients and produce solutions that are used in practice. As an academic software factory, LEDS needs to adjust some practices (e.g., align the development cycles with the academic semester) to balance the students' education goals and the clients needs. Therefore, the students are provided with an environment that simulates a software development organization (e.g., teams adopt agile practices, negotiate with the client, plan the project, develop and deliver the product) while leveraging and preserving learning interests (e.g., students attend trainings, participate in seminars, experiment new technologies

[1] By 'Extension' here we mean activities that connect academia and society, such as programs, projects, and events. This expression is also referred as 'University Outreach' or 'Community Engagement'

[2] https://leds.ifes.edu.br/

and help decide on the ones to be adopted). In that way, students have experiences that go beyond a traditional internship, as they experience participation in a project that delivers a product for a real client and, at the same time, have learning experiences that are not very common in most software organizations. The connection between academia and the real world strengthens students' education, making them better prepared for the market and also promoting innovation and regional development through technology.

A significant aspect of LEDS' trajectory has been its role in reinforcing the principle of the indissociability of teaching, research, and extension [15] . By integrating these three pillars in a practical and dynamic environment, LEDS has become a space for academic transformation, promoting a more comprehensive and meaningful learning experience and strengthening the institution's commitment to social impact, innovation, and knowledge production.

Ten years ago, we introduced LEDS in [5], providing an overview of the newly created laboratory and some preliminary results. Now, we report on our experience in the last ten years, considering the evolution of the laboratory over the past decade and its influence on students' education in Software Engineering (SE). We approach the transformations in LEDS' organizational structure, the expansion of research and extension projects, and the impacts on students' development and collaboration with industry. Particular emphasis is placed on the laboratory's contribution to formal higher education, as it has provided students with hands-on experiences that complement classroom instruction, foster interdisciplinary knowledge, and help bridge the gap between academic theory and professional practice in SE. We also discuss lessons learned and challenges we experienced at LEDS. Such discussion is important because to make the most of initiatives like LEDS, some factors should be considered.

With this paper, we expect to contribute to other education professionals and researchers interested in using academic software factories to associate practical and theoretical knowledge to enhance SE education. We believe that by sharing our experience at LEDS, practices and materials we have adopted, raising some ideas and concerns, and providing examples of how we have worked in our laboratory, we can help other people replicate successful practices and face some of the involved challenges.

2 Background

Due to the solution-oriented nature of LEDS and the proximity with real entities as clients, we have relied on theories and practices related to several disciplines, mainly SE (including System Analysis, Software Architecture and Design, Programming), System Thinking and Modeling [28], and Human-Centered Design [8]. Moreover, as a learning-centered environment, LEDS has also adopted teaching and learning strategies (e.g., **Active Learning** [29] and **Project-Based Learning** [21]) that foster deeper understanding and student engagement. Such strategies promote active learning, collaboration, and the development of both technical and transversal skills, reinforcing the laboratory's role as a space for academic and professional growth.

LEDS is characterized as a **Non-Formal Learning Environment (NFLE)** because it has a flexible structure in both content and methodology, while maintaining an intentional and organized approach [27]. In a non-formal learning environment, learning is centered on the needs and interests of the students, often taking place outside traditional institutional frameworks and involving limited direct interaction with instructors. Moreover, non-formal learning emphasizes the development of practical skills and socio-emotional competencies that are frequently overlooked in formal education. These environments have been shown to foster self-awareness and adaptability, enabling learners to respond effectively to the evolving demands of individuals and society [27].

Aligned with the principles of a NFLE, LEDS implements **Student-Centered Learning** (SCL) concepts [34]. SCL is an educational approach that emphasizes the active role of students in the learning process, shifting the focus from the teacher as the sole source of knowledge to the learner as a co-constructor of understanding[20]. This model encourages learners to engage in exploration, problem-solving, collaboration, and reflection, while teachers act as facilitators who guide and support individualized learning paths. It values students' prior experiences, promotes autonomy, and leverages technology and social networks to create dynamic, personalized, and lifelong learning environments that are better aligned with the demands of 21st-century education [25].

To effectively implement SCL principles, LEDS adopts experiential methodologies [3] such as **Learning by Doing** [11] and **Project-Based Learning** (PrB) [21]. *Learning by Doing* is a pedagogical approach where learners are in active roles where they construct knowledge through direct experience and practical engagement [11]. Rather than passively receiving information, learners participate in meaningful tasks that mirror real-world challenges, allowing them to develop understanding through action, reflection, and iteration. Within the LEDS environment, *Learning by Doing* empowers students to explore solutions, make decisions, and collaborate with peers, fostering deeper learning, autonomy, and the development of critical thinking and problem-solving skills. This approach reinforces theoretical content and also bridges the gap between knowledge and application, making learning more relevant and impactful in NFLE. Complementing *Learning by Doing*, PrBL offers students the opportunity to cultivate essential 21st-century skills such as leadership, collaboration, cooperation, ownership of solutions, public speaking, academic writing, and digital literacy. PrBL positions the student at the center of the learning process, with the instructor serving as a facilitator who guides inquiry, problem solving and project execution [7, 21].

Concerning SE education, and to implement the aforementioned SCL principles, LEDS has adopted a set of collaborative and reflective methodologies that promote innovation through group discussion, teamwork, and critical thinking. To operationalize these principles, LEDS integrates several complementary approaches throughout the software development lifecycle. such as **Human-Centered Design** (HCD) [35] and the **Google Design Sprint** (GDS) [19] methodology, both emphasize empathy, problem framing, ideation, and rapid prototyping, ensuring that solutions are co-created with a strong focus on users' real needs and experiences.

HCD and GDS enable the development of soft skills and, as a result, develop teams that understand the client, develop empathy

[3] Experiential learning occurs when learners are actively engaged in an activity or learning experiment.

for the problem, and create solutions that are viable both economically and technically. The feeling of empathy for the problem makes the team want to complete the project, not just because it was requested, but because they have taken on the problem as their own. Additionally, focusing on the technical and financial feasibility of the project encourages the team to create solutions that truly meet the client's needs. This shift in mindset and behavior is crucial for the execution and success of the project.

Complementary to HCD and GDS, LEDS employs principles from **lean manufacturing** [16] and **agile methods** (mainly *Scrum* [26] and *Extreme Programming* (XP) [4]) and **Continuous Software Engineering** (CSE) [2, 14, 18] practices to support iterative development, test-driven design, and collective code ownership. Agile methods employ an iterative and incremental approach to optimize predictability and control risks. CSE goes beyond agility and provides 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 [2].

3 LEDS Overview

Since its inception, LEDS has evolved from a local extension laboratory (2012) to an extension official program at the Serra Campus. In 2016, it gained recognition as an extension network-wide program within IFES [10], eventually reaching a peak of 5 LEDS units across the institution. This article will focus specifically on LEDS Serra (referred to here as LEDS for simplification), the first unit ever created, highlighting its structure, practices, partnerships, and the lessons learned from over a decade of continuous activity. Currently, LEDS involves 9 teachers and 75 students, operating in two rooms (measuring 43,80m² and 111,56m²). It also benefits from the collaboration of industry consultants who support project mentoring and ensure alignment with market needs. These consultants bring valuable market practices in areas such as back-end and front-end development, artificial intelligence, and project management.

LEDS maintains partnerships with other laboratories at IFES and UFES (Federal University of Espírito Santo) to enhance the learning experience in software development: FORMA [4], at UFES, and GAMA [5], at IFES/Serra. The FORMA is a research group that provides expertise in UX and UI, while GAMA contributes with knowledge in Artificial Intelligence and Machine Learning. This exchange of knowledge takes place through joint projects, where students from FORMA and GAMA collaborate directly with LEDS's teams, applying interdisciplinary approaches and advanced technologies to real-world software solutions.

From a SE learning perspective, the partnership among LEDS, FORMA, and GAMA enables students and teachers to share experiences and knowledge on key SE topics (e.g., requirements elicitation and usability) common to the groups involved. This collaborative environment fosters a richer and more integrated educational experience, bridging theoretical foundations with practical applications.

From 2015 to 2025, LEDS executed 32 projects, involving 112 students in collaborative research and innovation initiatives. Over this period, the laboratory mobilized more than R$ 7.9 million in financial resources, showcasing its strong capacity to establish strategic partnerships and attract both institutional and external funding to support students' education . The financial resources are obtained through agreements with private or public organizations and public calls by funding agencies (e.g., FAPES) and are used to provide scholarships to students and teachers and improve the laboratory infrastructure. Most students stay at LEDS until they graduate. The average dropout rate is around 23% (students who leave earlier to enter the job market or for personal reasons).

At this moment, the following six projects are running at LEDS: (i) *Conecta Fapes*, which aims to create a project management system for research and innovation initiatives across the Espírito Santo state, supporting the coordination, monitoring, and governance of R&D efforts. The project is being developed collaboratively with LEDS Colatina and the research groups FORMA and GAMA, combining expertise in SE, UX/UI design, and Artificial intelligence; (iii) *Agent ES*, which focuses on automating business processes through artificial intelligence by implementing an intelligent multi-agent system to support and optimize decision-making workflows within public administration, and is being carried out in partnership with FORMA and GAMA; (iv) *Morango*, which aims to develop a decision support system for strawberry crop management using computer vision techniques, contributing to more efficient and sustainable agricultural practices; (v) *Vepema*, which focuses on automating the management processes of voluntary service providers for the Espírito Santo Court of Justice (TJ-ES), streamlining workflows and improving transparency and control within the judicial system; and (vi) *LEDS Academy*, a collaborative project with LEDS Colatina that seeks to build an online course portal offering educational content based on the practices, tools, and technologies used in LEDS projects, supporting the training and onboarding of new participants. These projects demonstrate the diversity of initiatives that students can be involved in to improve learning.

4 How LEDS Works

At LEDS, the learning process in SE begins as early as the selection of new scholarship students. From the outset, students are immersed in a structured and practical environment that fosters the development of both technical and interpersonal skills. Once selected, they are allocated to multidisciplinary development teams and gradually take part in the entire software development lifecycle. This hands-on experience spans from requirement elicitation and system design to implementation, testing, and deployment, allowing students to apply SE practices in real-world project scenarios while continuously learning and evolving throughout the process.

4.1 Student Selection Process

The main objective of the LEDS student selection process is to identify candidates with high learning potential and foundational skills necessary to engage in real-world software development projects. The process aims assess technical skills and evaluate problem-solving strategies, autonomy, communication skills, and readiness to work in collaborative environments. Over time, this process has evolved to better align with the educational philosophy of LEDS.

Until 2024, the selection process was conducted through a public call announced on LEDS communication channels (Instagram[6],

[4]FORMA: Formalizations of Cognition and Design, https://forma.ufes.br/
[5]GAMA: Machine Learning and Automation Research Group

[6]@ledsifes

LinkedIn[7], and the official website[8]) and was open to all interested students. Candidates were invited to complete a practical challenge, publicly available through the LEDS GitHub repository[9], and subsequently participate in an interview with teachers and experienced LEDS students. The practical challenge evaluated the candidate's capacity to autonomously solve a real-world problem, emphasizing self-sufficiency, critical thinking, and proactive learning. Importantly, candidates were free to choose any technology or framework, allowing assessors to focus on the reasoning and strategy employed rather than tool-specific proficiency. Following the challenge, the interview stage assessed the candidate's ability to articulate and justify their solution, providing insights into both technical understanding and communication skills. This stage also served as an educational opportunity for LEDS students acting as interviewers, offering them experience in this practice.

In 2025, the selection process was improved with the creation of the *LEDS Academy*. Although candidates previously succeeded in the challenge phase, many struggled with fundamental SE concepts and practices (e.g., requirements engineering, object-oriented design, and CI/CD), during their initial months in active projects. These gaps posed challenges to learning and execution within project teams, impacting both students' learning and project quality.

To mitigate these issues, *LEDS Academy* was introduced as a preparatory phase within the selection process. Candidates now enroll in structured learning tracks focused on areas (e.g., back-end or front-end development). The learning materials are developed by current LEDS students, based on practical experience gained through project participation. This approach not only reinforces the students' knowledge but also fosters technical writing and instructional communication skills. All educational content is reviewed by LEDS teachers to ensure technical and pedagogical quality.

After completing the learning track, candidates solve a final practical challenge and submit their solution via a pull request to the LEDS GitHub repository. This is followed by an interview with a faculty member and a senior student. This redesigned process allows for a more thorough and structured evaluation of both technical skills and learning potential.

LEDS Academy is a recent initiative, but has already been successfully applied in the selection process for projects in back-end, front-end, AI, and DevOps. Students reported that the preparatory courses helped them better understand the practical demands of software projects and increased their confidence in applying core concepts. Teachers also noted improved preparedness among candidates and reported that the new process improved the identification of students aligned with the specific needs of each project.

4.2 Team Allocation

At LEDS, teams are organized as follows [30]: *Stream-aligned Teams*, *Platform Teams*, and *Research Team*. A *Stream-aligned Team* is cross-functional and autonomous, responsible for delivering value end-to-end within a specific domain, customer segment, or product stream. This team maintains close communication with stakeholders and promotes continuous delivery and iterative improvement. In *Stream-aligned teams*, students participate in the entire software development lifecycle—from understanding client needs and designing solutions to deploying and validating software.

A *Platform Team* focuses on building and maintaining internal platforms— reusable tools, services, APIs, and infrastructure—treated as internal products aimed at increasing the productivity and autonomy of other teams by abstracting technical complexities and providing scalable, secure, and reliable solutions. In *Platform teams*, the students are responsible for maintaining and evolving internal tools, services, and infrastructure, gaining deeper expertise in areas such as DevOps, cloud computing, security, and deployment.

Complementing these, the *Research Team* is dedicated to investigating and developing methods, tools, and applications that enhance the effectiveness of the Platform and Stream-aligned teams. By applying scientific methodologies, this team explores practical challenges, conducts empirical studies, and proposes solutions that contribute to the continuous improvement of LEDS's processes.

This organizational model fosters a scalable, efficient, and learner-centered environment, enabling students to experience SE practices in a well-defined team structures. Through a structured application process, students select and compete for open positions in these teams based on their interests and skills, allowing for a more personalized and goal-oriented learning experience.

Students are immersed in real-world challenges that stimulate applied research, practical experimentation, and the production of educational content, reinforcing the integration of academic knowledge with societal impact. This dynamic enables the development of technical and soft skills while fostering a culture of innovation and continuous learning. Moreover, students are encouraged to rotate between teams at LEDS, gaining exposure to various roles, technologies, and workflows. This mobility helps them identify their strengths and supports a well-rounded education.

4.3 The LEDS's Standard Software Process

Figure 1 illustrated the standard software process adopted at LEDS, which is composed of the following (sub)processes: *Planning*, *Immersion*, *Development*, , *Project Monitoring and Control*, *Quality Management*, and *Knowledge Management*. In the figure, the arrows indicate that several iterations of the process can be performed during a project. The process serves as a framework and, instead of rigidly following predefined activities, each team uses it as the reference process and discusses and selects the practices and technologies that fit the project.

The ***Planning*** process is responsible for presenting a high-level view of the project's objectives and deliverables to both the team and the client. With this macro perspective, the team and the client can organize testing activities and schedule the delivery of stable versions, for example. The plan is continually reviewed by ***Project Monitoring and Control***, which may adjust the schedule or deliverables as needed. The main outputs of the *Planning* process are: a high-level schedule of the process stages, a high-level Sprint schedule, the team composition, the project's risks and assumptions, the initial project scope, and a general list of the product's features. To support discussions with the client and collaboratively identify key deliverables of this phase, we adopt the *Project Model Canvas*. This visual and structured framework helps align expectations, clarify

[7] https://www.linkedin.com/company/ledsifes/
[8] https://leds.ifes.edu.br/
[9] https://github.com/leds-org

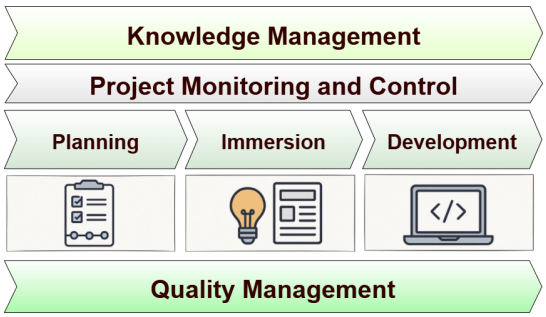

Figure 1: The Standard Software Process

project goals, and promote a shared understanding among stakeholders — serving as a foundation for planning decisions and future adjustments throughout the development process. Additionally, the use of Canvas provides students with a playful and engaging way to understand the various concerns involved in project management, making it an effective pedagogical tool for developing essential skills in a practical context.

Both the *Stream-aligned team* and the *Platform team* participate in the planning phase to gain a shared understanding of what will be developed. This joint participation ensures that the *Stream-aligned team* clearly understands the project goals and what needs to be delivered, while the *Platform team* identifies the type of system that will require infrastructure support. This alignment is essential to synchronize the efforts of both teams, enabling the development team to focus on building the solution and the platform team to proactively prepare the necessary tools, environments, and services. By aligning their objectives early in the process, the teams ensure smoother collaboration and more efficient delivery throughout the project lifecycle.

The ***Immersion*** process aims to understand the problem and propose a feasible solution within the project's timeline, scope, size, and team skills. Thus, the process has two main phases: *Problem Understanding* and *Solution Proposal*. The team must clarify all doubts about the problem and present a viable solution. Insights from this process can trigger revisions to the schedule, cost, scope, and deliverables defined during the Planning process. Its main outputs are: the initial Product Backlog, a prototype of the proposed solution, and the initial software architecture.

During the Problem Understanding phase, students of *Stream-Aligned Team* learn how to apply different requirements elicitation techniques (e.g., Interviews, Document Analysis, and Business Process Modeling) to understand the problem, as well as solution validation techniques. Among these, prototyping stands out. We use non-functional, low-fidelity prototyping (e.g., paper prototype technique [31]) to illustrate our understanding of the requirements and to communicate a possible solution to the problem. This hands-on and accessible approach helps both the development team and the client validate assumptions early, fostering alignment and reducing the risk of rework in later stages.

The students are provided with the opportunity to experience and apply requirement elicitation and specification techniques in a real-world context, enhancing their ability to gather and document meaningful information from stakeholders. Furthermore, they develop the skills necessary to identify and evaluate which requirements are most critical for the success of the project. After this, a Product Backlog is created, and the client is responsible for prioritizing each feature based on perceived value. User stories with higher scores—indicating greater value—are to be addressed first by the development team.

After problem understanding, in the *Solution Proposal* phase, students carry out a knowledge sprint, known as a *spike* [1]. The goal of a spike is to develop a proof of concept (PoC) using the selected technologies for the proposed solution. This allows the development team to explore and learn about the technologies in a practical context, building the necessary confidence to implement the final product. Additionally, the PoC helps the team assess whether the initial deadline presented to the client is feasible or if adjustments are needed. A PoC is built with a focus on learning, not on software quality. As such, it is not concerned with aspects like code quality, testing, or scalability, and the artifact produced is discarded after the *Immersion* process. SE principles and best practices are explored only after the PoC is concluded and the team has validated the proposed approach. Additionally, the PoC helps the team assess whether the initial deadline presented to the client is feasible or if adjustments are needed.

At the end of the spike, students meet with the client once again to prioritize the project requirements listed in the initial backlog. At this point, however, the requirements are still high-level and not fully specified. Typically, they consist of brief descriptions and low-fidelity screen prototypes. The detailed elaboration of these requirements is deferred to the *Development* process, where they will be refined incrementally throughout the project lifecycle.

From an educational perspective, the *Immersion* process enables students to learn about software architecture and technologies within the context of a real project. It also provides them with insight into how early design decisions can directly affect the project timeline. Since the PoC is developed at an early stage, it serves as a practical tool to validate whether the initial project time assumptions are accurate. If discrepancies are identified, the team can revise deadlines or adjust the project scope accordingly — ensuring better planning and risk mitigation from the outset.

Students from the *Platform team* take on a fundamental role by analyzing the technical requirements of the project and determining which tools and services need to be configured in the staging and production environments to support the *Stream-aligned team*. These students are responsible for initiating the setup of continuous integration (CI) and continuous deployment (CD) pipelines, ensuring that the necessary infrastructure is in place to support automated testing and delivery. This experience allows them to deepen their knowledge of DevOps practices while directly contributing to the project's operational readiness.

In parallel, students from the *Research team* also play an important role during this process. Their objective is to investigate practices, tools, or emerging technologies that could support and enhance the work of both the *Stream-aligned* and *Platform* teams. By applying scientific methods to identify innovative or more efficient solutions to address specific project challenges. This collaboration reinforces the connection between research and practice and provides students with experience in applying research skills to solve real-world SE problems.

For example, in the *Conecta* project, a recurring issue was identified regarding the management of user authorization for routes and objects, which required simultaneous modifications to both the front-end and back-end systems. Recognizing this challenge, a student from the *Research team* investigated authorization architectures based on *Google Zanzibar* [22] and *Zero Trust model [24]*, proposed by the National Institute of Standards and Technology[10]. As a result of this research, a robust and scalable access control service was designed and implemented. This standalone service decouples authorization logic from application code and can be reused across multiple projects. It was tested and used in the *Conecta* project. These research-driven initiatives exemplify how students can actively contribute to the improvement of the SE process while deepening their understanding of architectural design, security principles, and automation strategies.

The ***Development*** process focuses on creating software artifacts that realize the project's objectives. The Stream-aligned team may refine the analysis performed during *Immersion* to enhance understanding and produce artifacts. Its key outputs are software deployed to production and software documentation. It is important to highlight that LEDS has established a rule requiring real and valuable deliveries to the client every three months, with intermediate monthly deliveries. To support this, monthly and quarterly milestones are defined and aligned with the client. These milestones help the team maintain focus, manage expectations, and continuously track progress toward the project goals.

This three-month delivery cycle was strategically defined to avoid interfering with students' academic responsibilities throughout the academic semester. Since a typical university semester lasts approximately four months, the final month is often dedicated to exams and coursework deadlines. Therefore, scheduling project deliveries before this final academic push ensures that students can balance their academic commitments without compromising the project's success or their own academic performance.

The development process is based mainly on *Scrum* and XP practices. At the beginning of each sprint, during the planning meeting, the team defines the main deliverable that will bring value to the client based on the prioritized backlog. This deliverable is estimated and broken down into tasks for the sprint. The students participate in all key Scrum ceremonies, including Sprint Planning, Daily meetings, Reviews, and Retrospectives. These practices help maintain alignment, track progress, and foster continuous improvement. The biweekly sprint structure supports frequent feedback and continuous delivery of working software.

During a sprint, the development team has the autonomy to instantiate activities that go beyond pure coding. Depending on the nature and maturity of the requirements, a sprint may include tasks related to requirements specification, software architecture design, testing, and deployment preparation. For instance, since initial requirements, present in the backlog, are often elicited at a high level and accompanied by non-functional prototypes, the team may need to allocate part of the sprint to refining these requirements, designing the system architecture and database schema, and validating these artifacts with the client. In such cases, the primary goal of the sprint is not to produce working code, but rather to ensure a shared and detailed understanding of the system between the team and the client. This includes documenting the specified requirements and architectural decisions. Once this groundwork is completed and validated, the subsequent sprint can focus on implementation, testing, and delivering the increment to production.

Through this process, students progressively learn the meaning and importance of applying SE techniques. Rather than being introduced as abstract theory, these practices emerge organically from real project demands, allowing students to understand their relevance and application in context. Moreover, they gradually learn how to strategically combine techniques — such as requirements specification, architectural design, validation, and deployment planning — to drive project success.

Another important aspect to highlight is the instructional approach adopted during the early stages of the project. In the initial sprints, instructors take responsibility for defining the deliverables and calibrating the difficulty level of the tasks. This strategy stems from the recognition that project teams often begin with limited maturity and experience. To address this, the instructors intentionally design achievable goals that lead to early wins. These small but meaningful victories help students build confidence, maintain motivation, and develop a sense of progress.

As the team evolves and demonstrates increased maturity, the level of challenge is gradually raised. This scaffolding approach keeps students engaged, as they perceive themselves constantly advancing. Eventually, as the instructors observe the team's growing autonomy and technical maturity, responsibility for defining the sprint deliverables is transferred to the students. This transition fosters ownership, reinforces self-efficacy, and maintains a continuous state of challenge and capability [9].

At the end of each three-month cycle, a special ceremony called the "Celebration of Mistakes and Successes" is held, where each team shares both their achievements and the mistakes made during the cycle. The goal is to encourage students to reflect and share their learning with other project members. More importantly, this initiative aims to foster a culture where making mistakes is seen as a natural and necessary part of the learning process—what truly matters is learning from them[12, 17].

At the end of the *Development* process, the final version of the software (or module) is delivered to the production environment. This is followed by a knowledge transfer to the client, when the development team formally hands over the project to the client, and all key aspects of the solution are presented, including the implemented features, architectural decisions, deployment processes, and any supporting documentation. This ensures that the client is fully equipped to use, maintain, and evolve the system beyond the scope of the project.

The ***Project Monitoring and Control*** process is responsible for making decisions that lead to the successful completion of the project by analyzing data produced by the other processes. A project is considered successful only if it delivers software that meets the client's needs. At LEDS, project management is data-driven and integrates key metrics to support SE education. Students use indicators (e.g., Work In Progress, Committed vs Completed, Throughput, and Burnup) to reflect on planning accuracy, delivery capacity, and team focus. These metrics foster critical thinking, encourage sustainable performance, and help students understand how to balance

[10]https://www.nist.gov/

scope, quality, and deadlines in real projects. Guiding questions include: "*Are we focusing on too many tasks at once?*", "Are we delivering what we promised?"", "Can we meet the deadline at our current pace?"", and "Is it possible to handle new requirements without compromising delivery?".

The application of project management metrics within the LEDS learning environment provides students with a data-driven and experiential understanding of project management. By engaging with real projects and monitoring these indicators, students develop critical skills in planning, tracking, and adapting project execution based on data. These metrics enable reflection on delivery capacity, estimation accuracy, scope control, and sustainable work practices, while fostering systemic thinking in multi-team coordination. This approach not only reinforces theoretical knowledge but also cultivates essential competencies such as decision-making, negotiation, and continuous improvement, preparing students to manage complex software projects in professional settings. *Project Monitoring and Control* occurs in parallel with the *Development* process, receiving continuous input from it. As a result, students are constantly reflecting on their planning decisions and revalidating their strategies based on real-time project data. This dynamic fosters a culture of adaptive planning and short feedback loops.

To manage activities and promote collaboration, students use GitHub Projects[11] as the main project tracking tool. Each development team maintains its own kanban board, where tasks are created and updated as GitHub Issues. These issues are directly linked to development efforts, with changes implemented via feature branches and validated through Pull Requests (PRs), following a Gitflow workflow[12]. This integration of planning, metrics, and source control provides students with a realistic and cohesive view of modern SE practices. It also reinforces the importance of using indicators not just for reporting but as active tools for continuous improvement and team alignment.

To ensure software quality, the ***Quality Management*** process is responsible for the definition of quality standards and the assessment of the quality of the software artifacts (e.g., documentation and code) produced throughout the project, determining whether each artifact is ready for delivery. In this process, we focus primarily on the quality of the code being developed. To ensure this, peer code reviews are systematically conducted, promoting knowledge sharing and alignment of coding practices within the team. Additionally, we employ automated quality analysis tools, which provide objective, metric-based insights into code quality. LEDS uses such as SonarQube[13] as a quality analysis tool. This combination of human review and automated feedback enables the team to identify and address maintainability, readability, and reliability issues early, fostering a culture of continuous improvement. From an educational perspective, this approach helps students internalize software quality standards, understand common code smells, and develop the ability to critically analyze and improve their own work and that of their peers — essential skills for professional SE practice. During this process, we introduce students to the fundamentals of interface-level integration testing using Cypress. These tests are integrated into the Continuous Integration (CI) cycle through GitHub Actions, enabling automated verification of system behavior at each code commit. This approach not only demonstrates the practical importance of maintaining a reliable test suite but also reinforces the concept of continuous integration as a key practice in modern software development.

Finally, the ***Knowledge Management*** process is responsible for capturing the knowledge generated during development and making it accessible across the LEDS. The project team produces knowledge artifacts (e.g., tutorials, best practices, other content that benefits the organization long term) so that acquired insights are continuously disseminated. As part of their learning journey, students are encouraged to document their experiences by writing tutorials and technical blog posts (e.g., Dev.to[14]) in both Portuguese and English. They are also motivated to present talks and share lessons learned from their projects at LEDS. In addition, live sessions are held on LEDS' channel on YouTube with students and external guests, allowing the broader community to engage with the knowledge produced throughout the projects. These activities not only reinforce technical understanding but also help students develop public communication and technical writing skills, essential for their academic and professional development. The produced knowledge artifacts are made available for LEDS members as well as the external community. In this way, other students can benefit from the material, and other education professionals can reuse it in their own training and education initiatives. Live recordings and tech talks are available on the LEDS YouTube channel @ledsifes, while written tutorials can be found at https://dev.to/ledsifes. We are currently migrating and organizing these resources—including tutorials and training tracks into a new educational platform, available at academy.leds.dev.br.

4.4 Beyond a Non-Formal Software Engineering Learning Environment

The knowledge acquired by LEDS teachers over more than a decade of experience in non-formal, project-based SE education has enabled the creation of new institutional learning initiatives that go beyond LEDS as a non-formal learning environment. Here, we highlight *interdisciplinary projects* in undergraduate education and *short-term training in SE*.

The first one took the experience of LEDS to the classroom, aiming at an integrated initiative to teach SE disciplines. The implementation of interdisciplinary projects was performed in the Bachelor's degree in Information Systems at the Serra campus (where LEDS is located and its teachers work). It involved the integration of several undergraduate courses into a single interdisciplinary project. In this format, students developed one unified project that fulfilled the requirements of four disciplines – Database, Software Design, Software Engineering, and Web Development – taken in the same semester. The initiative reduced redundancy and provided a more coherent and connected learning experience. Many students appreciated working on a single project across multiple courses, noting that it gave greater meaning to the content and helped consolidate their understanding. Some even chose to delay certain courses to take all four together, maximizing the benefits of the integrated approach. To enable this coordination, teachers aligned their syllabi

[11] https://github.com/
[12] https://docs.github.com/en/get-started/using-github/github-flow
[13] https://www.sonarsource.com/
[14] https://dev.to/

at the beginning of the semester and maintained regular communication, either in person or via messaging groups, throughout the academic term.

The second initiative aimed to share knowledge and experience gathered from LEDS projects with the external community. The short-term training in SE was a project-based program grounded in Agile methodologies and Web Development. Students were assigned a real client project to be developed over the semester. Teachers acted as consultants, introducing content progressively according to students' learning needs. For instance, when students lacked experience with interviews, specific materials were provided, and teachers offered guidance and support. Students worked in two-week delivery cycles (sprints), presenting their progress to the class at the end of each cycle and receiving feedback on improvements and potential directions. Each course enrolled 20 students and was supported by 5 teachers. Students were divided into five groups of four, with each group assigned a dedicated teacher as a tutor, allowing for close monitoring of progress and personalized support. The training was offered for the first time in 2017, with spots for 20 students, and has been a success since. The 2023 edition had a candidate-to-seat ratio of 11.25 applicants per spot. Students reported several positive aspects of the course, highlighting its hands-on approach to learning theory, the realism of the methodology, and the opportunity to apply concepts in actual development projects. They emphasized the value of teamwork, freedom to choose technologies, interaction with real clients, and the dynamic, open environment that fosters creativity and continuous learning. Comments also praised the strong networking opportunities, personalized support from teachers, and exposure to current industry tools and practices.

5 Lesson Learned

This section presents a set of lessons learned over the years working at LEDS. These insights were identified through continuous reflections and discussions involving teachers, students, and clients over the years, based on the successes and challenges we experienced while executing the projects and building the LEDS initiative.

Focusing on soft skills leads to better project results. Our experience has shown that the development of soft skills (e.g., communication, collaboration, empathy, and self-management) produce significant benefits in the medium and long term. However, unlike hard skills, soft skills are more complex to develop, as they involve interpersonal dynamics and internal attitudes. Despite these challenges, we have observed that students who engage in developing these skills tend to improve their ability to communicate clearly and listen actively, which is essential for understanding and specifying software requirements, and for facilitating interaction between developers, clients, and users. Furthermore, they tend to become more resilient, adaptable, and capable of dealing with complex situations and real-world challenges. This kind of individual development takes time and heavily depends on human interactions (e.g., one-on-one meetings) and must be approached according to each person's specific needs. Most LEDS participants are under the age of twenty-five and are still in the process of developing their professional maturity. In this context, the collaborative environment of the laboratory is a privileged space for experiencing and improving these skills, making them better prepared to act professionally, ethically, and responsibly in the most diverse scenarios.

A learning space for teachers is necessary. LEDS is not only a space for student development, is also a safe and collaborative learning environment for teachers. For many educators who have not had prior experience in the software industry, participating in LEDS offers a unique opportunity to engage with real-world challenges, understand market expectations, and reflect on how their academic discipline can be applied in practice. Unlike traditional classroom settings, where teaching is often a solitary activity, LEDS fosters an ecosystem in which teachers can learn from one another. Through daily interaction with colleagues from different disciplines, they exchange experiences, teaching strategies, and technical knowledge—something rarely found in the routine of academic life. This environment encourages experimentation, reflection, and the rethinking of pedagogical approaches based on practical experiences. Here, we share a statement from a teacher: "*Participating in LEDS was a turning point in my career. Working side by side with students on real market needs pushed me out of the academic bubble and into industry challenges. It changed how I teach—I now focus more on practice, clean code, and team communication. LEDS showed me that to connect classes to reality, teachers must experience the market*". This testimony illustrates how LEDS shaped its own culture, which engages and creates opportunities for teachers to rethink their roles as educators, grounded in real-world experiences and shared learning.

Real projects support learning and adaptability. Each project developed at LEDS introduces new knowledge for both students and teachers, making the teaching of SE both challenging and enriching. This continuous change in context requires teachers to remain open to new ways of delivering content, often pushing them beyond their comfort zones. While this may initially cause discomfort, it ultimately contributes to their growth as educators. For students, in turn, every project is a unique learning experience. Although the technical knowledge acquired varies according to the technologies and domains involved, all students undergo a development of soft skills. Across all teams, critical thinking, collaboration, and problem-solving are essential. Thus, despite the diversity of technical learning, students consistently develop core competencies that are vital for their future professional roles.

Physical space significantly influences learning engagement. The design of the learning environment has a profound impact on students' motivation and engagement. Comfortable, welcoming, and flexible spaces — those that include elements such as sofas and soft lighting – encourage students to spend more time in the environment, increasing their sense of belonging and participation. These physical arrangements reinforce the idea that learning is not confined to rigid structures but can flourish in spaces that support creativity, comfort, and collaboration. At LEDS, teams are arranged to sit together at the same table, which fosters faster communication and more fluid cooperation. This physical proximity facilitates constant information exchange, knowledge sharing, and collaborative problem-solving. It creates an environment where peer learning happens naturally and continuously, strengthening teamwork and the collective construction of solutions.

Safe environments accelerate responsibility and learning. When students are placed in real projects within a safe learning environment — where mistakes are not punished but transformed into learning opportunities — they gain the confidence to take on responsibilities early in their careers. At LEDS, this approach allows students to experience all stages of software development, from gathering requirements to presenting the final product, including leadership and multidisciplinary collaboration. This accelerates their professional growth and prepares them for real-world challenges in ways that would typically take years in a traditional work environment. To reinforce this culture, LEDS promotes a regular ritual known as the "Celebration of Mistakes and Successes". In these moments, teams openly share what went right and what went wrong during the project. The goal is to create a psychologically safe environment where students understand that mistakes are part of the learning journey. They are encouraged to reflect, adapt, and keep evolving — turning setbacks into lessons that guide not only their own growth but also that of future team members. For example, one student said: "*LEDS also taught me the value of taking risks in an environment where mistakes are not punished, but transformed into lessons that later serve as guidance for new team members.*" This mindset encourages continuous reflection and growth, turning setbacks into stepping stones for future success.

Cross-team integration reduces communication failures and strengthens collaboration. In large software projects involving teams located in different regions, communication breakdowns often occur when groups — such as design and development — operate in isolation. At LEDS, we experienced this challenge in a project where the design and development teams were working from different locations and perceived themselves as separate units. This separation led to frequent miscommunication, misaligned expectations, and coordination issues. To address the problem, we reorganized the workflow so that all team members — regardless of location — operated as part of a single, integrated team. We synchronized working hours, encouraged joint planning and reviews, and promoted shared responsibilities. As a result, communication improved significantly, and the teams began to function as a cohesive unit. From a SE perspective, this experience highlights the importance of synchronizing distributed teams to ensure effective collaboration. Even when geographically separated, aligning schedules and promoting integration supports agile practices, improves requirement clarity, reduces handoff errors, and enhances overall team performance.

Multidisciplinary and autonomous teams improve accountability and delivery. For software teams to operate effectively, they must be empowered to take responsibility for the entire development lifecycle — from requirements gathering to deployment. At LEDS, we learned that giving teams ownership over delivery dates, technology choices, and all project phases fosters autonomy, accountability, and cohesion. Previously, responsibilities were split among separate teams for requirements, development, and deployment. This fragmented structure led to communication breakdowns, delivery delays, and internal conflicts, as teams did not share a unified vision and often blamed one another when problems arose. By restructuring teams to be multidisciplinary and autonomous, we enabled them to self-manage and collaborate more effectively. This change reduced misunderstandings, improved delivery timelines, and strengthened team identity. Each team became responsible not only for building the solution but also for understanding user needs and ensuring operational delivery — promoting a holistic understanding of the project. From a SE standpoint, this aligns with agile principles, which advocate for cross-functional teams capable of delivering value independently. In educational terms, this structure enhances learning by exposing students to the full development pipeline and teaching them to make decisions collaboratively, develop technical and soft skills, and reflect on the consequences of their choices in real-world conditions.

Real-world extension projects can be an opportunity for applied research. One of the most valuable lessons learned at LEDS is that extension projects, when structured around real client needs, can serve not only as educational and social engagement opportunities but also as a rich source of applied research. By engaging with authentic problems from society, students and teachers are exposed to complex challenges that require innovative and technically grounded solutions — creating fertile ground for academic inquiry. At LEDS, this approach is embodied in the *Research Team*, which works alongside the development and platform teams to investigate methods, tools, and architectures that enhance project execution. Research activities emerge directly from the issues encountered in ongoing extension projects, ensuring that academic efforts are not abstract or disconnected, but instead driven by practical demands and grounded in real use cases.

Integrating Teaching, Research, and Extension is challenging. Our educational practice shows that integrating extension projects into the curriculum is a powerful strategy to promote the integration of teaching, research, and extension. In SE education, this approach allows students to engage with real-world demands, going beyond theory and experiencing the full development cycle. However, this integration presents political, pedagogical, structural, and cultural challenges. It requires institutions to rethink the roles of teachers and students, moving away from the traditional model where the teacher is the central figure. Instead, instructors act as facilitators and mentors, while students take active responsibility for their learning. Working with real clients adds practical value but demands careful planning — such as preparing instructors, managing uncertainties, ensuring infrastructure, and overcoming resistance to new roles and expectations. Integrating multiple courses into a single project is especially complex, requiring coordination across disciplines and technological alignment, for example, between Web Development and Database Systems. Flexibility in both technologies and requirements is essential. Overly rigid specifications can limit contributions from other areas and reduce collaborative learning opportunities. In this model, SE acts as both a thematic and methodological foundation. It unifies diverse knowledge areas through real practices like agile methods, continuous integration, and requirement validation, developing not only technical skills but also leadership, teamwork, autonomy, and critical thinking. This integrated, project-based approach fosters deeper learning and prepares students to deliver meaningful, socially relevant software solutions.

6 Related Work

Several works have reported the integration of teaching and real-world projects to improve SE education. We consider that LabES [3], Demola [6], Mandacaru.dev [13], and ProAdecamy [32] are the closest to the LEDS approach.

LabES fosters the teaching of SE concepts through extension projects, promoting experiential learning and real-world problem-solving. However, LabES does not incorporate SE research directly, relying instead on collaborations with the NEMO group for that purpose. Demola, in turn, focuses on short-term (8–12 weeks) challenge-based learning to co-create prototypes and validate ideas with industry partners. In contrast, LEDS is oriented toward the delivery of final, production level software products, offering a longer and more comprehensive development cycle.

Mandacaru.dev also shares similarities with LEDS, particularly in combining research, innovation, and talent development for regional impact. However, while LEDS conducts this training process through the development of real digital products for internal and external clients, offering students a full immersion in real-world software projects, Mandacaru.dev is more focused on providing structured courses that connect students to practical challenges proposed by partner companies. Finally, ProAcademy emphasizes entrepreneurship and the creation of student-run business teams. LEDS does not focus on entrepreneurship. Instead, students join the LEDS to develop hard and soft skills related to SE.

All these works share similarities (e.g., all employ active learning methods) but also differences that make each one a unique contribution. Some aspects that distinguishes our work from other reported initiatives are: (i) the adopted standard software process - in this paper, we present how SE practices are organized at LEDS and how projects are performed, which increases the knowledge of processes that can be adopted in similar contexts; (ii) the team allocation strategy, which is designed to allow students to immerse in real-world challenges that stimulate applied research and practical experimentation (e.g., students can be allocated on a Research Team, fostering applied research in SE, directly derived from real-world problems addressed in the projects); (iii) LEDS has evolved into a networked laboratory, enabling knowledge generated in one campus to be shared with others and facilitating joint projects with different campuses or institutions; (iv) the knowledge assets offered to the SE community provide knowledge about SE practices and learning (knowledge artifacts produced by LEDS are available in its communication channels – see section 7); and (v) the lessons learned shared in this paper, which enrich the knowledge for other people to conduct initiatives similar to LEDS.

7 Final Considerations

In this paper, we shared our experience at LEDS. Initially introduced as a space for simulating real-world software development projects in an academic environment, LEDS has expanded into an institutionalized program that integrates teaching, research, and extension in a continuous and structured way to enhance students' education in SE.

We described the theoretical foundations and pedagogical approaches that support LEDS, highlighting how they foster technical and human competencies. We also presented the organizational model adopted and explained how the team structure supports autonomy, collaboration, and knowledge transfer. We provided an overview of the standard software process adopted at LEDS, discussed the learning aspects it involves, cited two education initiatives that go beyond LEDS's projects, and shared some lessons learned, which provide actionable insights for educators, institutions, and researchers seeking to implement or improve similar academic software factories. They address aspects such as soft skill development, the role of physical space, autonomy, cross-team integration, the involvement of teachers as learners, and the challenges of curricular integration.

Although the use of academic software factories to support teaching and learning is not new, it is necessary to grow knowledge on this subject to help education professionals and researchers in SE education. By being better educated and trained, students will become better software engineers, able to produce quality software [3]. We expect that by sharing our experiences and materials, we can help other education professionals interested in using academic software factories to enhance SE education.

We must emphasize that the practices and lessons reported in this paper are case-based, i.e., based on our experiences at LEDS. A recognized limitation in this context is the ability to generalize from the case-specific to different cases. Wieringa and Daneva [33] argue that in such cases, generalization can be established for similar cases and, although they are not universal, 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 LEDS projects has contributed to improving their education and the development of competencies and skills. So far, the evaluation has been mainly qualitative.

As future work, we intend to develop a guide for implementing LEDS-inspired academic software factories in other institutions. This guide will consolidate the practices, processes, and structures, offering practical recommendations for adaptation in different educational contexts. Additionally, we plan to carry out studies to obtain the students' feedback in a more structured way and evaluate the effects of participating in LEDS on the students' learning in SE. We also intend to conduct studies to analyze the lessons learned in greater depth, investigating their applicability in other institutional settings. These efforts aim to support the broader adoption of experiential, project-based approaches in SE education.

ARTIFACT AVAILABILITY

Some of the knowledge artifacts produced by LEDS members are available at leds.ifes.edu.br, academy.leds.dev.br, https://dev.to/ledsifes, and https://www.youtube.com/@ledsifes.

ACKNOWLEDGMENTS

This research is supported by Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Finance Code 001, and Espírito Santo Research and Innovation Support Foundation (FAPES) - Processes 2023-5L1FC, 2021-GL60J, 2022-NGKM5, and T.O. 1022/2022.

REFERENCES

[1] Hussein Al Hashimi and Andy Gravell. 2020. Spikes in Agile Software Development: An Empirical Study. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 1715–1721.

[2] Monalessa P. Barcellos. 2020. Towards a Framework for Continuous Software Engineering. In *XXXIV SBES* (Natal, Brazil) *(SBES '20)*. 626–631.

[3] Monalessa P. Barcellos, Vítor E. Silva Souza, Patrícia Dockhorn Costa, and Camila Zacche 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 *Proceedings of the XXIII Brazilian Symposium on Software Quality (SBQS '24)*. Association for Computing Machinery, New York, NY, USA, 552–562. doi:10.1145/3701625.3701680

[4] Kent Beck. 1999. Embracing change with extreme programming. *Computer* 32, 10 (1999), 70–77.

[5] Rodrigo Calhau, Paulo Santos Júnior, Karin Komati, Maxwell Monteiro, Fabiano Ruy, and Vanessa Nunes. 2014. LEDS: Um Ambiente para Impulsionar o Aprendizado em Computação. In *Anais do XXII Workshop sobre Educação em Computação* (Brasília). SBC, Porto Alegre, RS, Brasil, 199–208. https://sol.sbc.org.br/index.php/wei/article/view/10974

[6] Daniel Catalá-Pérez, Mikko Rask, and María de Miguel-Molina. 2020. The Demola model as a public policy tool boosting collaboration in innovation: A comparative study between Finland and Spain. *Technology in Society* 63 (2020), 101358.

[7] Sivachandran Chandrasekaran, Alex Stojcevski, Guy Littlefair, and Matthew Joordens. 2013. Project-oriented design-based learning: aligning students' views with industry needs. (2013).

[8] Mike Cooley. 2000. Human-centered design. *Information design* (2000), 59–81.

[9] Mihaly Csikszentmihalyi. 1990. *Flow: The Psychology of Optimal Experience*. Harper and Row.

[10] Victório Albani de Carvalho and Thiago Chieppe Saquetto. 2024. REDE LEDS: UM PROGRAMA DE PARCERIAS QUE INTEGRA ENSINO, PESQUISA E EXTENSÃO. In *Propriedade intelectual, transferência de tecnologia e inovação: integrando ensino, pesquisa e extensão*, André Romero da Silva (Ed.). Atena Editora, Ponta Grossa, Chapter 5. doi:10.22533/at.ed.8992430085

[11] Richard DuFour and Rebecca DuFour. 2013. *Learning by doing: A handbook for professional learning communities at work TM*. Solution Tree Press.

[12] Amy C Edmondson. 1999. Psychological safety and learning behavior in work teams. *Administrative science quarterly* 44, 2 (1999), 350–383.

[13] Alexandre Feitosa, Artelino Tavares, Henrique Martins, Emanuel Pessôa, Marlon Paiva, Emerson Tomaz, and Allysson Araújo. 2024. Aproximando Indústria e Academia para Lapidação de Talentos em Desenvolvimento de Software no Sertão do Ceará: Um Estudo de Caso sobre o mandacaru.dev. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 433–443. doi:10.5753/sbes.2024.3518

[14] Fitzgerald, Brian and Stol, Klaas-Jan. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123, 176–189.

[15] Nadia Gaiofatto Gonçalves. 2015. Indissociabilidade entre Ensino, Pesquisa e Extensão: um princípio necessário. *Perspectiva* 33, 3 (2015), 1229–1256.

[16] Shaman Gupta and Sanjiv Kumar Jain. 2013. A literature review of lean manufacturing. *International journal of management science and engineering management* 8, 4 (2013), 241–249.

[17] Reuven Hirak, Ann Chunyan Peng, Abraham Carmeli, and John M Schaubroeck. 2012. Linking leader inclusiveness to work unit performance: The importance of psychological safety and learning from failures. *The leadership quarterly* 23, 1 (2012), 107–117.

[18] Paulo S. Santos Júnior, Monalessa P. Barcellos, Fabiano B. Ruy, and Moises S. Omêna. 2022. Flying over Brazilian Organizations with Zeppelin: A Preliminary Panoramic Picture of Continuous Software Engineering. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES '22)*. 279–288.

[19] Jake Knapp, John Zeratsky, and Braden Kowitz. 2017. *Sprint: o método usado no Google para testar e aplicar novas ideias em apenas cinco dias*. Intrínseca, Rio de Janeiro.

[20] Kevin S Krahenbuhl. 2016. Student-centered education and constructivism: Challenges, concerns, and clarity for teachers. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas* 89, 3 (2016), 97–105.

[21] John Larmer, John Mergendoller, and Suzie Boss. 2015. *Setting the standard for project based learning*. Ascd.

[22] Ruoming Pang, Ramon Caceres, Mike Burrows, Zhifeng Chen, Pratik Dave, Nathan Germer, Alexander Golynski, Kevin Graney, Nina Kang, Lea Kissner, Jeffrey L. Korn, Abhishek Parmar, Christina D. Richards, and Mengzhi Wang. 2019. Zanzibar: Google's Consistent, Global Authorization System. In *2019 USENIX Annual Technical Conference (USENIX ATC '19)*. Renton, WA.

[23] Nadja N. Rodrigues. 2013. Praticando Qualidade de Software: Ensinando e Aprendendo seus Valores através de Ambiente Real. In *Anais do IX Simpósio Brasileiro de Sistemas de Informação* (João Pessoa). SBC, Porto Alegre, RS, Brasil, 475–486. doi:10.5753/sbsi.2013.5713

[24] Scott Rose, Oliver Borchert, Stuart Mitchell, and Sean Connelly. 2020. Zero Trust Architecture. doi:10.6028/NIST.SP.800-207 NIST Special Publication 800-207.

[25] Nagayuki Saito. 2015. Development of a Collaborative Skills Training Program Utilizing ICT for 21th-Century Students. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. Association for the Advancement of Computing in Education (AACE), 706–711.

[26] J. Schwaber and Sutherland Ken. 2013. The scrum guide-the definitive guide to scrum: The rules of the game. http://scrum.org

[27] Julian Sefton-Green. 2012. *Learning at not-school: A review of study, theory, and advocacy for education in non-formal settings*. The MIT Press.

[28] Peter M Senge and John D Sterman. 1992. Systems thinking and organizational learning: Acting locally and thinking globally in the organization of the future. *European Journal of Operational Research* 59, 1 (1992), 137–150.

[29] Burr Settles. 2009. Active learning literature survey. (2009).

[30] Matthew Skelton and Manuel Pais. 2019. *Team topologies: organizing business and technology teams for fast flow*. It Revolution.

[31] Carolyn Snyder. 2003. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann.

[32] Paul Tosey, Spinder Dhaliwal, and Jukka Hassinen. 2015. The Finnish Team Academy model: implications for management education. *Management Learning* 46, 2 (2015), 175–194.

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

[34] Gloria Brown Wright. 2011. Student-centered learning in higher education. *International journal of teaching and learning in higher education* 23, 1 (2011), 92–97.

[35] Carla B Zoltowski, William C Oakes, and Monica E Cardella. 2012. Students' ways of experiencing human-centered design. *Journal of Engineering Education* 101, 1 (2012), 28–59.