

Fostering Diversity in Software Engineering Education: A Challenge-Based Approach to Integrate Non-STEM Students

Afonso Sales^a, Nicolas Nascimento^b, Rafael Chanin^c and Aline de Campos^d

*Pontifical Catholic University of Rio Grande do Sul (PUCRS), School of Technology, Porto Alegre, RS, Brazil
{afonso.sales, nicolas.nascimento, rafael.chanin, aline.campos}@pucrs.br*

Keywords: Software Engineering Education, CBL, Diversity in Tech, Cross-Disciplinary, Mobile Development.

Abstract: This paper proposes an innovative approach to addressing the lack of diversity in software engineering education by integrating non-STEM students into mobile app development programs. Leveraging the Challenge-Based Learning (CBL) methodology, this idea paper explores how targeted instructional strategies and preparatory leveling sessions can empower students from diverse academic backgrounds to succeed in highly technical environments. By breaking traditional barriers to programming education, this approach fosters an inclusive learning environment that enhances technical skills and encourages cross-disciplinary collaboration and innovation. We present the conceptual framework and practical implications of this inclusive model, offering insights into how this approach can be adopted across different educational settings to promote diversity and innovation in software engineering.

1 INTRODUCTION

In the last few years, the growing demand for mobile app developers in the software engineering industry has highlighted the importance of providing accessible and effective educational programs (Dahlander and Wallin, 2018; Thomas and Devi, 2021). However, traditional approaches to software engineering education often present significant barriers for students from non-STEM (Science, Technology, Engineering, and Mathematics) backgrounds, limiting diversity and innovation in the field (Hyrnsalmi, 2023). The lack of inclusion for non-STEM students in programming courses could reinforce these barriers and deprive valuable interdisciplinary perspectives that could contribute to more innovative solutions (McKinsey and Company, 2020).

This paper proposes an inclusive and active learning approach to mobile app development education by integrating non-STEM students into a structured Challenge-Based Learning (CBL) framework (Nichols et al., 2016; Jimarkon et al., 2022). Grounded in constructivist principles, CBL emphasizes real-world problem-solving, collaboration, and


creativity. The insertion of a preparatory leveling addresses technical gaps, enabling non-STEM students to acquire foundational skills and tackle technical challenges alongside STEM peers.


This approach aims to create a replicable model that encourages diversity in software engineering education while providing students with the technical and collaborative skills required for success in the modern tech industry. Therefore, this paper explores the conceptual framework of this model, integrating multiple perspectives to increase creativity and foster collaboration among learners from diverse backgrounds.


The following Section 2 provides the theoretical background and Section 3 details the course structure. In Section 4, we present the program's outcomes and in Section 5, the implications of these results, including the contributions of non-STEM students and the broader impact on their professional development. In Section 6, we summarize the findings and ideas for future research and program improvements.


2 BACKGROUND

The framework for the short programming course described in this study is based on Active Learning and the structure of Challenge-Based Learning methodology. The following section presents the theoretical foundation supporting this experience.

^a  <https://orcid.org/0000-0001-6962-3706>

^b  <https://orcid.org/0000-0002-0080-8822>

^c  <https://orcid.org/0000-0002-6293-7419>

^d  <https://orcid.org/0000-0002-3585-954X>

2.1 Active Learning

Active learning has been widely recognized as a practical approach in programming education, encouraging student participation and promoting collaboration (Doolittle et al., 2023). This pedagogical approach shifts the focus from traditional lecture-based teaching to student-centered learning activities, leading to engaging in discussions, problem-solving, case studies, and hands-on activities that foster deeper understanding and skills development.

Research indicates that Active Learning significantly enhances student performance by encouraging critical thinking and the practical application of knowledge. For example, a study revealed that students attained higher grades and were less likely to fail than those in traditional lecture-based courses (Freeman et al., 2014). These advantages are essential in programming education, where practical skills and problem-solving abilities are crucial.

However, implementing Active Learning in programming education contexts that include non-STEM students presents specific challenges (Theobald et al., 2020). Non-STEM students often have varying levels of technical proficiency and may require support to build foundational knowledge. It demands adapting teaching materials and instructional strategies to accommodate diverse learning needs. It's equally important to lower barriers for novices through user-friendly programming environments and targeted instructional support (Dahlander and Wallin, 2018).

Moreover, this approach enables a collaborative learning environment where students can learn from each other's diverse perspectives (Nascimento et al., 2020). It is especially beneficial for non-STEM students, who can gain insights from their STEM peers while contributing their unique viewpoints (Sandrone et al., 2021). Also, it contributes to preparing students for real-world scenarios where teamwork and communication skills are essential to succeed.

2.2 Challenge Based Learning

Challenge Based Learning (CBL) emerges as a favorable methodology for programming education, encouraging students to solve real-world problems collaboratively while developing necessary technical and social skills (Binder et al., 2017; Chanin et al., 2018). It involves students in an active, hands-on approach to learning that emphasizes critical thinking, creativity, and application of knowledge in critical issues (Jimarkon et al., 2022). Challenge Based Learning is structured into three phases: *Engage*, *Investigate*, and *Act* (Figure 1).



Figure 1: CBL Framework (Nichols et al., 2016).

In the *Engage* phase, students identify a *big idea* and articulate a meaningful and relevant challenge. This fosters intrinsic motivation, as students are more likely to be engaged when they perceive the personal and societal relevance of their work. For instance, students might choose challenges related to environmental sustainability, health, or social justice, which can be addressed through mobile app development.

During the *Investigate* phase, students conduct extensive research to understand the context and implications of their chosen challenge. It involves gathering information from multiple sources, including academic literature, expert interviews, and community input. By integrating diverse perspectives and sources of knowledge, students develop a comprehensive understanding of the problem they are addressing and enhance their research skills in evaluating and synthesizing information critically.

In the *Act* phase, work in teams, students apply their knowledge and skills to implement a solution to the identified challenge. In the context of programming education, this often involves designing, coding, and testing an application. The iterative nature of this phase reflects genuine software development processes, providing students with practical experience.

CBL has been applied in initiatives focused on teaching software development, and research supports its effectiveness in enhancing student learning outcomes. CBL projects helped students better understand the subject matter, improve their problem-solving skills, and increase engagement (Taconis and Bekker, 2023). Also, students reported that working on real-world challenges made their learning experience more impactful (Theobald et al., 2020).

Studies showed that CBL fosters students' motivation to learn by emphasizing the creation of software solutions to address realistic challenges (Binder et al., 2017; Jimarkon et al., 2022). Also, the combination with methodologies such as Design Thinking (DT) has proven effective in guiding students through ideas convergence and encouraging deeper domain research, further enhancing their learning outcomes in software development (Gama et al., 2018).

In summary, CBL is a robust methodology that bridges the gap between theoretical knowledge and

practical application, preparing for the complexities of the modern workforce and capable of promoting diversity, making it a model for innovative education.

2.3 Short Programming Courses

Short programming courses have proven effective in providing coding and software development skills within a condensed timeframe (Lyon and Green, 2021). Often referred to as *bootcamps* or *crash courses*, these immersive experiences allow participants to gain practical expertise in a brief period.

Their primary advantage is a focused, hands-on method. Unlike multi-semester academic programs that include extensive theoretical content, short courses can emphasize practical application. Participants spend most of the time working on projects and solving real-world problems, strengthening soft skills such as teamwork and communication.

However, despite the efficacy in rapidly developing coding skills, these programs often exclude non-STEM students due to demanding prerequisites. It poses notable challenges, underscoring the importance of inclusivity and diversity to drive innovation and growth in the industry.

3 THE COURSE

The course curriculum in this study is a comprehensive and intensive program designed to promote mobile application development, adopting a CBL framework to engage students in realistic problem-solving activities. It has two main components: iOS programming and user experience design (UX).

The classes are held in person Monday through Friday (three hours daily) in an innovative learning environment with modern infrastructure and digital tools to support an engaging and collaborative learning experience. This space includes adjustable seating, brainstorming rooms, and state-of-the-art technology to facilitate creativity and comfort.

Each course edition has a maximum of 25 students, and all participants receive CBL training, which prepares them for this approach. The CBL process begins with defining a big idea and a challenge. Students then develop guiding questions, conduct investigations, and devise and implement solutions collaboratively. Experienced instructors with backgrounds in industry and academia facilitate the learning process, providing guidance, feedback, and support. They ensure that students receive personalized attention and can address any learning issue.

3.1 Leveling

While short-duration courses excel at intensive and focused training, they often assume uniform student skill levels, which disadvantages non-STEM learners lacking foundational mathematics, logic, and computer science knowledge (Handelsman et al., 2022). Consequently, the participation of non-STEM students in these courses is often limited, reducing the diversity of the learning environment and the tech industry (Singer et al., 2020).

To address this gap, our training program employs strategies to increase the inclusivity of short programming courses. One key element is the leveling week at the beginning of the course, which focuses on building basic technical skills and confidence, enabling students from varied backgrounds to engage on more equal terms.

The leveling week covers foundational programming logic, introduction to coding languages, and essential tools and practices used in software development. By providing this foundational knowledge, the program intends to lower the initial learning curve and encourage non-STEM students to progress alongside their STEM peers.

In addition, active and CBL methodologies enhance engagement and outcomes for diverse student populations. By addressing real problems collaboratively, learners leverage their unique perspectives and skill sets, leading to more inclusive solutions (Theobald et al., 2020).

3.2 Course Structure

After the leveling, the course lasts six weeks and is divided into two phases, both of which highly adopt active learning. The first phase focuses on fundamentals, introducing iOS programming and UX Design, while the second phase focuses on applications.

The **Phase 1** introduces coding, storyboards, and resources through lectures, hands-on exercises, and collaborative activities. Students learn essential tools, shortcuts, the Model-View-Controller (MVC) pattern, and UIView, with exercises reinforcing these concepts and UX fundamentals such as personas and paper prototyping. And, the **Phase 2**, provides practical challenges structured around Challenge-Based Learning (CBL), including navigation controllers and increasingly advanced tasks. Students develop and deliver solutions to specific problems and engage in a comprehensive project, from defining a central idea to implementing a final product.

Figure 2 presents an overview of the course structure, describing the main activities.

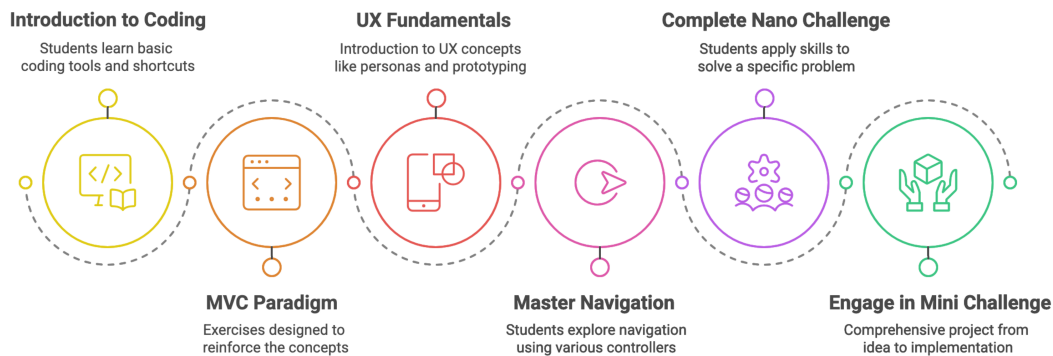


Figure 2: Course structure overview.

3.3 Deliverables

The course employs various types of deliverables to assess and enhance students' learning experiences. These are designed to promote reflection, practice, and documentation of the process, ensuring students develop technical skills and a deeper understanding of their experience. The primary types are:

- Reflections:** encourage students to think critically about their learning experience. These can be in video, audio, or text and it can help them internalize what they have learned, connect new knowledge to existing, and consider how they can apply their skills in the future.
- Exercises:** hands-on tasks that reinforce the concepts taught and are designed to be incremental and build on previous knowledge, allowing students to apply what they have learned in a practical context. Students are encouraged to follow specific development rules or use particular frameworks and can also innovate and expand upon the basic requirements.
- CBL Documentation:** produced throughout the course as students engage in the CBL process, including text, video, audio, and images that capture the learning journey. It serves multiple purposes, such as assessments, portfolios, and storytelling of the challenges faced and solutions developed.
- Nano Challenge:** shorter CBL activities designed to target specific content areas or skills in which the instructor guides and has tight boundaries and a limited time frame. Students engage in a lower-intensity investigation and may not implement their solutions with an external audience.
- Mini Challenge:** more extended activity that allows students to work through the entire CBL framework. This involves starting with a Big Idea, researching, and implementing comprehensive solutions. It provides intense learning experiences that stretch students' abilities and prepare them for more significant, real-world challenges.

3.4 Schedule

A schedule provides a systematic and detailed view of the learning experience with content and deliverables organized by day and week. Table 1 presents the main schedule of the course.

Table 1: Course Schedule and Deliverables.

	Day	Content	Deliverable
Leveling	1	Introduction, Students Presentation and CBL Dynamics	Reflection: Expectations
	2	Variables, Constants, Conditionals and Primitive Types	Exercise
	3	Array, Dictionary and Loop	Exercise
	4	Classes, Structs, functions and optionals	Exercise
	5	Nano Swift	Exercise
Week 1	6	Nano Swift	Exercise
	7	Introduction to Coding, Storyboards, UIImageView, Label, Button	Exercise
	8	TextField, TextView, TextFieldDelegate	Exercise
	9	AutoLayout	Exercise
	10	View, ViewController, MVC	Exercise
	11	Design Guideline	Exercise
Week 2	12	Navigation Controllers; Tab bar controllers; Multiple VCs	Exercise
	13	TableView	Exercise
	14	Nano TableView	Exercise
	15	Nano TableView	Nano Challenge & Keynote
Week 3	16	Design Content	Reflection: <i>How am I doing?</i>
	17	User Defaults, CoreData	Exercise
	18	Rest API	Exercise
	19	Nano API	Nano Challenge & Keynote
	20	SpriteKit	Exercise
Week 4	21	Mini Challenge (Engage)	Reflection: <i>How am I doing?</i>
	22	Mini Challenge (Engage)	
	23	Mini Challenge (Investigate): Git, Pods	Low-fi prototype
	24	Mini Challenge (Act)	
	25	Mini Challenge (Act)	
Week 5	26	Mini Challenge (Act)	
	...		
	30	Mini Challenge (Act)	
	31	Mini Challenge (Act)	
	...		
Week 6	35	Mini Challenge (Final Presentations)	Reflection, Keynote, Prototype

4 RESULTS

The course successfully trained 202 students with different background levels across 12 groups from 2017 to 2024. In 2021, holding regular course sessions was impossible due to the COVID-19 pandemic. In 2022, the course returned with a small class. Table 2 presents the number of groups per year and the number of students in each group.

Table 2: Number of groups and students over the years.

	2017	2018	2019	2020	2022	2023	2024
Groups	1	2	4	1	1	2	1
Students	17	46	78	18	5	26	12

Regarding diversity, the course had 46 non-STEM participants, representing 23% of the total, which enriched the learning environment, promoting cross-disciplinary collaboration and innovative problem-solving. Considering all participants, ages ranged from 16 to 45 years, with a predominant average of 25 years in the groups. Regarding gender, a predominance of males (80%) was observed.

Despite efforts to attract and include women in Information Technology (IT), they are still underrepresented. This imbalance creates a gender structure that affects women's experiences and career trajectories in IT (Kenny and Donnelly, 2019). Organizations have implemented diversity and inclusion initiatives, but retention is still problematic. To address this issue, researchers suggest focusing on the consequences of gender imbalance, developing comprehensive explanations, and evaluating the effectiveness of interventions through comparative and longitudinal studies (Gorbacheva et al., 2018; Aguilon et al., 2020). Figure 3 shows the proportion of men and women among STEM and non-STEM students in our mobile app development program.

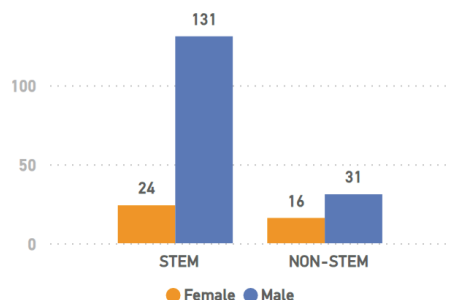


Figure 3: STEM and Non-STEM students by gender.

Analyzing the presence of female and male students in STEM fields, a ratio of 25 to 131 can be observed among STEM students, representing five times as many males as females. In non-STEM students,

the ratio is 15 to 31, which is a significantly smaller difference. It can indicate that including non-STEM students in these courses can encourage new perspectives about gender equality in STEM careers.

The majority of participants are undergraduate students in information technology, with most enrolled in Computer Science, Software Engineering, and Information Systems programs. There were also students from other courses such as Systems Analysis and Development, Internet Systems, Computer Engineering, and Information Technology Management. Some students from related areas, such as Digital Games and Biomedical Informatics, also participated. Other STEM areas also appeared, including Biological Sciences and Mathematics and various engineering programs (Civil Engineering, Chemical Engineering, Control and Automation Engineering, Electrical Engineering, Materials Engineering, Mechanical Engineering, and Production Engineering).

When it comes to Non-STEM courses, there was a significant diversity of study areas among the participants, ranging from Administration, Accounting Sciences, Architecture and Urbanism, and Management Processes through areas with an emphasis on creative processes such as Animation Design, Multimedia Production, Advertising and Marketing, Product Design, Visual Design, and Fashion, up to fields like History, Social Sciences, Law, Physical Education, Psychology, and Music. In the 2023 editions, high school students participated, which allowed for integrating a different perspective into the process, bringing the viewpoints of individuals who were not yet working in specific areas of knowledge.

Most (59%) were enrolled between their undergraduate course's first and fourth semesters or were in high school. The remaining 41% were between the 5th and 10th semesters. These data indicate that the course had a predominance of students beginning their formal higher education studies.

As Figure 4 shows, considering the participants' current occupation at the time of the course, the majority (57%) indicated they were only students, while 23% reported being employees, 18% were interns, and 2% were business owners.

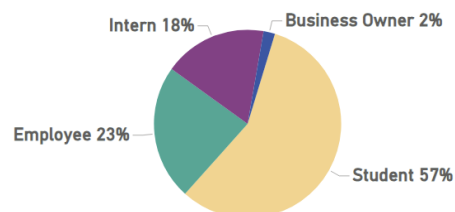


Figure 4: Students occupation.

Regarding prior knowledge of object-oriented programming, 31% indicated that they have no idea what it is, while 28% reported having a reasonable understanding, knowing at least the theory. Another 25% indicated good knowledge, having already developed projects using this model, and finally, 16% indicated advanced knowledge, using it on a regular basis. Figure 5 shows the chart of these data.

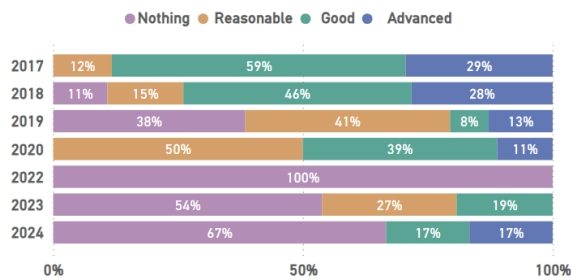


Figure 5: Knowledge in Oriented-Object Programming.

When asked how they would categorize themselves in the IT field, the majority indicated an interest in front-end (33%) and web development (31%). Meanwhile, others mentioned back-end and mobile development, with fewer participants citing game development, software analysis, and testing as their primary interests. (Figure 6).

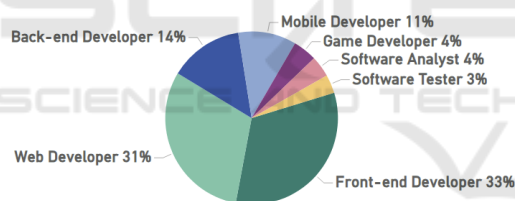


Figure 6: Technical self-description of the students.

Data collection methods include participant interviews and analysis of student projects. Interviews provide insights into students' experiences and perceptions, while project evaluations focus on the technical and creative aspects of the apps developed. Most of the collected data is qualitative and was produced using different media types, allowing the students to express themselves in the way they consider best.

Students developed a variety of mobile applications, ranging from educational tools and health apps to social networking platforms and productivity solutions. These projects demonstrated the technical skills acquired and the participants' creativity and practical problem-solving abilities (Melegati et al., 2020).

Including a leveling week at the beginning of the course helped non-STEM students acquire foundational technical skills, ensuring they could engage in the subsequent phases. Also, CBL creates a dynamic

and multidisciplinary learning environment. For example, students who developed a mobile app to address mental health issues among college students had to investigate mental health resources and effective intervention strategies, integrating knowledge from psychology, design, and technology.

Before the course, none of the students were familiar with CBL. However, all participants highlighted that the methodology was important in maintaining their focus and engagement throughout the course. The instructors' evaluations of the Mini Challenge presented at the end of the course confirmed that students had effectively grasped the content. Although some students suggested that extending the course by one or two weeks would have been beneficial, all participants successfully created a complete mobile application, with some even developing advanced features not covered during the classes.

4.1 Cases

Graduates of the program have pursued various career paths. Some secured positions in tech companies, both locally and internationally, while others continued their education in advanced technology courses. Several non-STEM graduates reported leveraging their new skills in their original fields, enhancing their professional capabilities and opportunities.

After graduating from the program, one of the students continued to learn in a longer course, improving software development and soft skills using the abilities developed in the short course. Further, the student had experience working with 3D programming and even participating in an entrepreneurship program sponsored by a Big Tech company and is currently working for a multinational company.

On a similar note, another student, who also continued to study in a longer course, focused mainly on health challenges and developed many applications that target health problems in society. One featured application developed by this student is an app that helps pediatricians explain complicated medical data to parents and children during consultations. This application is currently available in the App Store and has a rating of 5 (best rate).

Additionally, one other student pursued an iOS software engineer career right after finishing the course. During the course, this student began interviewing for iOS positions, using the course as proof of basic knowledge, and secured a junior role at a local company. Now, the student is a full-time senior iOS engineer for a US-based firm, with previous experience at an investment company and a video-call company.

5 DISCUSSION

Non-STEM students enriched the course with diverse perspectives, fostering innovation and user-centric designs. Feedback showed they felt empowered by their new technical skills and valued the inclusive approach. The course served as a catalyst for the continued professional development of many students in mobile app development. Some expressed a newfound passion for technology, enrolling in further training programs and participating in tech communities. This ongoing engagement underscores the program's effectiveness in fostering long-term interest and growth in the field (Melegati et al., 2019).

One interesting story regarding the program's career-changing potential is about a student who originally studied biological sciences when she joined the short course. This student reported on the final day of the program that at the beginning of the course, in her own words, "[...] *I got really nervous because I was the one who had nothing in common with the course. I was enrolled in biology and had never experienced the world of computing [...]*". These statements reveal the distance computing could have from other areas and that bringing it closer to those fields is challenging for the course staff as much as for the students.

Furthermore, teachers and course designers must foster inclusive environments that bridge knowledge gaps. Strategies such as foundational modules, relatable examples, and continuous mentorship help demystify computing, making it more accessible and engaging for non-STEM students.

This same student ends the report saying "[...] *I learned a lot in the course. Things I was not expecting to be able to do, and today I see my capabilities and I want to learn even more. [...] I intend to stay in the field and also enroll in the longer program because I really enjoyed working with Swift*". This result presents a glance at the potential that a short course could present for non-STEM students in getting to know the software field. This becomes more realistic as these courses are much easier for students to perform in parallel and usually do not require them to quit their course.

The story of this particular student, after finishing the short course, follows a similar path to the stories reported in Section 4. After the short course, the student joined a more prolonged course and had the opportunity to work on several projects, developing both hard and soft skills. The student is currently living abroad and works as a software engineer.

6 CONCLUSION

This paper presents a forward-thinking approach to addressing the lack of diversity in software engineering education by integrating non-STEM students into mobile app development programs. Using Challenge-Based Learning (CBL) and preparatory leveling sessions, the proposed model aims to bridge the technical knowledge gap and enable students from diverse academic backgrounds to collaborate effectively.

This approach fosters a diverse learning environment, provides non-STEM students with the technical skills necessary to succeed in app development, and introduces interdisciplinary perspectives to enhance the overall learning experience.

The implications of this approach extend beyond the immediate educational outcomes. It could operate as a replicable model adapted to various educational settings, promoting greater diversity and innovation within the tech industry. By encouraging cross-disciplinary collaboration, this method aligns with the growing need for software engineering education to be both inclusive and dynamic, preparing students for the multifaceted challenges of the modern workforce.

Future work should explore the long-term impact of such inclusive training programs on participants' careers and the tech industry. Specifically, longitudinal studies could track graduates over several years to assess career progression, skill application in various fields, and continued involvement in technology-related activities. Additionally, research could investigate the effectiveness of similar programs in different contexts and with other non-technical disciplines.

Expanding these initiatives globally could further enhance diversity and innovation in technology. To achieve this, partnerships with international educational institutions and tech companies could be established, facilitating the exchange of knowledge and best practices. Moreover, adapting the curriculum to include emerging technologies, such as artificial intelligence and blockchain, could keep the program relevant and forward-looking.

Incorporating participant feedback, such as extending the course and adding advanced topics, enhances learning. Industry projects and internships further improve practical experience and employability.

In conclusion, this program's inclusive approach not only bridges the gap between STEM and non-STEM students but also enriches the tech industry with diverse talents and perspectives. We can cultivate a more inclusive, innovative, and skilled technological workforce by continuously refining and expanding such educational models.

ACKNOWLEDGMENT

This study was partially supported by the Ministry of Science, Technology, and Innovations from Brazil, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex.

REFERENCES

- Aguillon, S. M., Siegmund, G.-F., Petipas, R. H., Drake, A. G., Cotner, S., and Ballen, C. J. (2020). Gender differences in student participation in an active-learning classroom. *CBE Life Sciences Education*, 19.
- Binder, F. V., Nichols, M., Reinehr, S., and Malucelli, A. (2017). Challenge based learning applied to mobile software development teaching. In *2017 30th CSEE&T*, pages 57–64. IEEE.
- Chanin, R., Sales, A., Santos, A. R., Pompermaier, L. B., and Prikladnicki, R. (2018). A collaborative approach to teaching software startups: findings from a study using challenge based learning. In *Proc. of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pages 9–12.
- Dahlander, L. and Wallin, M. (2018). The barriers to recruiting and employing digital talent. *Harvard Business Review*.
- Doolittle, P., Wojdak, K., and Walters, A. (2023). Defining active learning: A restricted systemic review. *Teaching and Learning Inquiry*, 11.
- Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., and Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23).
- Gama, K., Castor, F., Alessio, P., Neves, A., Araújo, C., Formiga, R., Soares-Neto, F., and Oliveira, H. (2018). Combining challenge-based learning and design thinking to teach mobile app development. In *2018 IEEE Frontiers in Education Conference (FIE)*.
- Gorbacheva, E., Beekhuyzen, J. P., vom Brocke, J., and Becker, J. (2018). Directions for research on gender imbalance in the it profession. *European Journal of Information Systems*, 28:43 – 67.
- Handelsman, J., Elgin, S., Estrada, M., Hays, S., Johnson, T., Miller, S., Mingo, V., Shaffer, C., and Williams, J. (2022). Achieving stem diversity: Fix the classrooms. *Science*, 376:1057 – 1059.
- Hyrnsalmi, S. M. (2023). How diversity and inclusion are approached in software engineering university-level teaching. *2023 IEEE/ACM 4th Workshop on Gender Equity, Diversity, and Inclusion in Software Engineering (GEICSE)*, pages 17–24.
- Jimarkon, P., Shahverdi, M., Dikilitaş, K., and Husebø, D. (2022). Dimensions of multidisciplinary engagement beyond the classroom in challenge-based learning. In *Annual conference of The European Society for Engineering Education*. Uni. Politècnica de Catalunya.
- Kenny, E. J. and Donnelly, R. (2019). Navigating the gender structure in information technology: How does this affect the experiences and behaviours of women? *Human Relations*, 73:326 – 350.
- Lyon, L. A. and Green, E. (2021). Coding boot camps: Enabling women to enter computing professions. *ACM Transactions on Computing Education*, 21(2).
- McKinsey and Company (2020). Diversity wins: How inclusion matters.
- Melegati, J., Chanin, R., Sales, A., Prikladnicki, R., and Wang, X. (2020). MVP and experimentation in software startups: a qualitative survey. In *46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Portoroz, Slovenia, August 26-28, 2020*, pages 322–325. IEEE.
- Melegati, J., Chanin, R., Wang, X., Sales, A., and Prikladnicki, R. (2019). Enablers and inhibitors of experimentation in early-stage software startups. In *Product-Focused Software Process Improvement - 20th International Conference, PROFES 2019, Barcelona, Spain, November 27-29, 2019, Proceedings*, volume 11915 of *LNCS*, pages 554–569. Springer.
- Nascimento, N., Santos, A. R., Sales, A., and Chanin, R. (2020). Behavior-driven development: A case study on its impacts on agile development teams. In *ICSE '20: 42nd International Conference on Software Engineering, Workshops, Seoul, Republic of Korea, 27 June - 19 July, 2020*, pages 109–116. ACM.
- Nichols, M., Cator, K., and Torres, M. (2016). *Challenge Based Learning Guide*. Digital Promise, Redwood City, CA, USA.
- Sandrone, S., Scott, G., Anderson, W. J., and Musunuru, K. (2021). Active learning-based stem education for in-person and online learning. *Cell*, 184:1409 – 1414.
- Singer, A., Montgomery, G. M., and Schmoll, S. (2020). How to foster the formation of stem identity: studying diversity in an authentic learning environment. *International Journal of STEM Education*, 7:1–12.
- Taconis, R. and Bekker, T. (2023). Challenge based learning as authentic learning environment for stem identity construction. *Frontiers in Education*.
- Theobald, E. J., Hill, M. J., Tran, E. T., Agrawal, S., Arroyo, N., Behling, S., Chambwe, N., Cintrón, D. L., Cooper, J. D., Dunster, G. P., Grummer, J., Hennessey, K., hui Hsiao, J. C., Iranon, N., Jones, L. N., Jordt, H., Keller, M., Lacey, M. E., Littlefield, C., Lowe, A., Newman, S., Okolo, V., Olroyd, S., Peacock, B., Pickett, S., Slager, D., Cavedes-Solis, I., Stanchak, K., Sundaravardan, V., Valdebenito, C., Williams, C., Zinsli, K., and Freeman, S. (2020). Active learning narrows achievement gaps for underrepresented students in undergraduate science, technology, engineering, and math. *Proceedings of the National Academy of Sciences of the United States of America*.
- Thomas, C. G. and Devi, A. J. (2021). A study and overview of the mobile app development industry. *International Journal of Applied Engineering and Management Letters (IAEML)*, 5.