Implementing ISO/IEC 29110 for Student Software Development: A Structured Approach Through Agile Methodologies and Professional Internships

Abstract: The ISO/IEC 29110 standard provides guidelines and best practices for small entities developing software. This paper presents a structured approach for implementing a software development project based on the ISO/IEC 29110 standard, specifically designed for teams of students in their professional internships, with six-month rotation cycles. This proposed process is the result of eight cycles of experience, each lasting six months, involving a total of 112 students. Its objective is not only to meet quality and efficiency standards but also to provide a comprehensive learning experience for the students. Additionally, project management is described based on agile methodologies. The proposal includes the evaluation of students' performance and the documentation of lessons learned for future projects. This holistic approach aims for students to acquire valuable technical and soft skills while contributing to the development of high-quality software aligned with international standards.

Keywords: ISO/IEC 29110, Software development, Professional internships, Agile methodologies, Development process, Project management, Supervision and mentoring, Quality assurance, Continuous improvement

1 Introduction

The ISO/IEC 29110 standard, "Systems and Software Engineering Lifecycle Profiles for Very Small Entities (VSEs)," provides a structured and standardized framework designed to improve the efficiency and quality of software development in VSEs by defining best practices and processes tailored to their size and characteristics [1]. The ISO/IEC 29110 is organized into profiles of different levels of maturity, Basic, Intermediate, and Advanced profiles [2]. Each profile details a set of essential activities and tasks ranging from project management to software engineering, reinforcing that projects are completed on time, within budget, and with the expected quality, enhancing operational efficiency and market credibility [3].

A fundamental aspect of the standard is the clear definition of roles and responsibilities within the development team, which facilitates more efficient management and better collaboration among team members. Additionally, the standard provides guidelines for the creation and use of documentation and templates, helping to standardize and simplify software lifecycle activities [4].

This paper presents the implementation, based on Agile Methodologies, of a development process based on the ISO/IEC 29110 standard for student teams in their professional internships. This implementation aims to offer a structured and standardized solution to facilitate the acquisition of practical skills and theoretical knowledge, preparing students to face the challenges in the professional world.

The proposal is based on the experience of fourth-year receiving students in the context of eight cycles of practice, each spanning six months, involving a total of 112 students. Its objective is not only to meet quality and efficiency standards but also to provide a comprehensive learning experience for the students. Additionally, project management is described based on agile methodologies. The proposal includes the evaluation of students' performance and the documentation of lessons learned for future projects. This holistic approach aims for students to acquire valuable technical and soft skills while contributing to the development of high-quality software aligned with international standards.

2 Background

To create an effective software development process tailored for student teams in professional internships, various methodologies were employed, combining recognized standards and agile practices. The methodologies and practices included are:

ISO/IEC 29110 focused on improving the efficiency and quality of software development through profiles detailing activities, roles, and responsibilities, it was included, Project Management**,** Software Engineering and Documentation and Templates.

Agile Methodologies (Scrum and Kanban) Agile Methodologies (Scrum and Kanban): These complement the structure of ISO/IEC 29110, providing flexibility suitable for dynamic projects and learning environments:

- *Scrum*
    - Sprints
    - Daily Scrum Meetings
    - Sprint Review and Retrospective
- *Kanban*
    - Work Visualization
    - Work in Progress
    - Continuous Improvement

Training and Development Strategies: Continuous training and development ensure students acquire the necessary skills and integrate effectively into the development process [6]:

- Mentorship Programs
- Technical and Soft Skills Training
- Real Project Simulations [7].

This combination of the ISO/IEC 29110 structure and standards with the flexibility of agile methodologies provides an effective process for student teams in professional internships, aiming to ensure high-quality software projects and equip students with essential competencies for their future careers in the software industry [8].

3 Process for Student Management

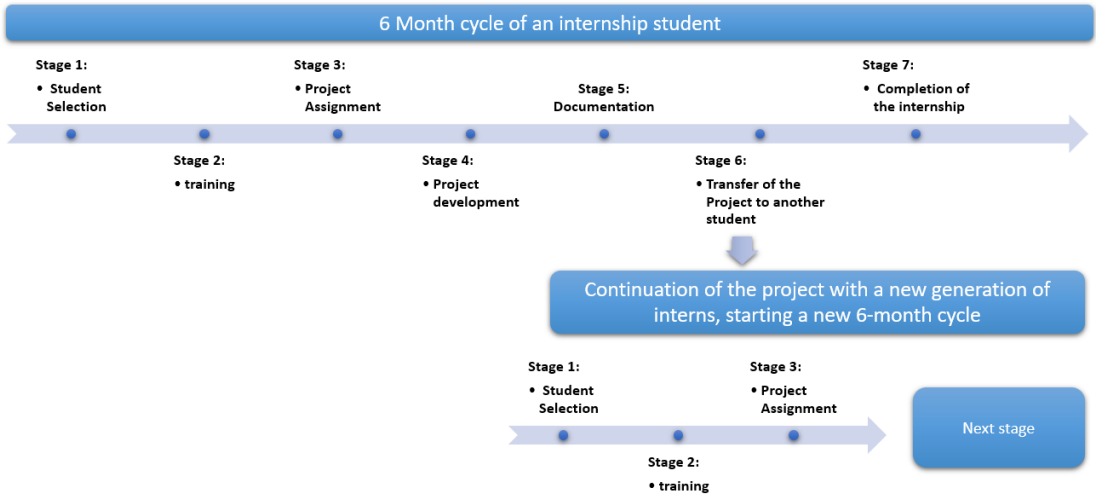

Fig. 1. Internship Student Development Cycle: A Structured 6-Month Program for Professional Growth

The Development Cycle of Intern Students (Fig. 1) shows the design of 6 months of professional internships. This cycle is meticulously divided into seven key stages, each of which plays a crucial role in the professional development and growth of interns.

The process begins with Stage 1: Student Selection, where candidates are chosen based on specific criteria to ensure they are well-suited for the internship. Following the selection, Stage 2: Training involves providing the selected students with initial training to equip them with the necessary skills and knowledge required for their upcoming projects.

In Stage 3: Project Assignment, students are assigned to specific projects that align with their training and capabilities, allowing them to apply what they have learned in a practical setting. Stage 4: Project Development is where students actively engage in the development of their assigned projects, contributing significantly to the project's progress and gaining hands-on experience.

As the project progresses, Stage 5: Documentation ensures that all work is properly documented. This documentation is vital for maintaining continuity, providing a reference for future work, and facilitating knowledge transfer. Toward the end of the internship, Stage 6: Transfer of the Project involves transferring the project to another student. This step is crucial for ensuring continuity and allowing the incoming student to build on the existing work.

Finally, the cycle concludes with Stage 7: Completion of the Internship, where students wrap up their work and prepare to transition into their professional careers. The image also highlights the continuation of the project with a new generation of interns, starting a new 6-month cycle. This ensures a seamless

transition and ongoing development, maintaining the project's momentum and facilitating continuous learning and improvement for the students involved.

3.1 Team members selection

Selecting students for the software development team is a critical step to ensure the project's success and the effectiveness of the educational process [9]. The selection criteria are based on a combination of technical skills, soft skills, and the potential for learning and adaptation. Specific criteria include:

- *Technical Skills:* Students' technical competencies are evaluated, including their knowledge of relevant programming languages, development tools, and software development methodologies. Preference is given to students who have demonstrated strong technical skills through previous projects, academic grades, or work experience.
- *Learning Ability:* It is crucial to select students who show a strong ability to learn and adapt. This includes their willingness to learn new technologies, methodologies, and tools, as well as their ability to apply this knowledge practically.
- *Soft Skills:* Such as effective communication, teamwork, problem-solving, and time management are considered. These competencies are essential for collaborative work and project success.
- *Motivation and Commitment:* Students who show a high level of motivation and commitment to the project are sought. This includes their interest in software development, willingness to contribute to the team, and dedication to completing assigned tasks.
- *Diversity of Experience:* Diversity within the team is valued, selecting students with different backgrounds and levels of experience. This enriches the learning environment and fosters a variety of perspectives and approaches to problem-solving.

3.2 Initial training plan

An initial training plan will prepare the students for their roles in the software development project. It encompasses both technical training and the development of soft skills necessary for success in a collaborative and dynamic work environment [10]. Key components of the initial training plan include:

- *Project Induction:*
    - *Project Overview:* Provide an overview of the project, including its objectives, scope, schedule, and expected deliverables.
    - *Introduction to ISO/IEC 29110:* Conduct an informative session highlighting its importance, structure, and application in the project.
- *Technical Training:*
    - *Tools and Technologies:* Training on the tools and technologies that will be used in the project, such as integrated development

environments (IDEs), version control systems (e.g., GitHub), and project management platforms (e.g., Kanban).
    - *Agile Methodologies:* Instruction on agile methodologies, especially Scrum and Kanban, including their principles, practices, and key roles.
    - *Coding Practices:* Workshops on coding best practices, code quality standards, and software testing techniques.
- *Soft Skills Development:*
    - *Effective Communication:* Training in effective communication techniques, including verbal and written communication, and the use of collaboration tools (e.g., Slack, Microsoft Teams).
    - *Teamwork:* Activities designed to foster collaboration and team cohesion, such as group dynamics and team-based problem-solving exercises.
    - *Time Management:* Sessions on time management techniques, including task prioritization, planning, and progress tracking.
- *Mentoring and Continuous Support:*
    - *Mentor Assignment:* Each student is assigned to an experienced mentor who provides guidance, support, and continuous feedback throughout the project.
    - *Follow-up Sessions:* Regular follow-up meetings to assess student progress, address challenges, and adjust the training plan as needed.
- *Initial Assessment:*
    - *Competency Evaluation:* Initial evaluation of students' technical and soft skills to identify strengths and areas for improvement.
    - *Personalized Development Plan:* Development of a personalized training plan for each student, based on the results of the initial assessment, with clear goals and specific actions for continuous development.

Careful student selection and a robust initial training plan are essential components for the success of a software development project with student teams in professional internships. By combining rigorous selection criteria with comprehensive training that encompasses both technical and soft skills, students are well-prepared to contribute effectively to the project and develop their professional competencies [11]. This approach not only improves the quality of the software produced but also provides a valuable educational experience that prepares students for their future careers in the software industry.

3.3 Role and Responsibility Assignment

Clear role and responsibility definition are fundamental to the success of a software development project with students' teams in professional internships [12]. This approach not only facilitates meeting project objectives but also provides valuable educational experiences that prepare students for future careers in the software industry [13]. The suggested team roles are based on agile

methodologies, adapted for the educational and internship environment [14]. The main roles and their responsibilities are described with a suggested student profile for each one:

1. **Scrum Master** is who acts as the team's facilitator, ensuring that agile practices are followed and that the team can work effectively without impediments.
    - **Student Profile:** A student with leadership skills, organization, problem-solving ability, and a good understanding of agile methodologies.
2. **Product Owner** responsible for managing the product backlog, defining requirements and priorities, and acting as a liaison between the development team and stakeholders.
    - **Student Profile:** A student with communication skills, the ability to understand and prioritize requirements, and a clear vision of the final product.
3. **Development Team** consists of students responsible for implementing software functionalities. This team is cross-functional and may include developers, designers, and testers.
    - **Student Profile:** Students with technical skills in programming, user interface design, and software testing. A diversity of technical skills within the team is valued.
4. **Tester** responsible for ensuring software quality through continuous testing, identifying and reporting defects, and verifying that requirements are met.
    - **Student Profile:** A student with attention to detail, analytical skills, and knowledge of software testing techniques.

3.4 Initial planning

Initial planning is a critical component for the success of any software development project [15], especially when working with student teams in professional internships. This stage involves defining project objectives, developing a detailed work plan, and allocating resources.

Establishing Milestones and Key Dates. Establishing milestones and key dates is essential for monitoring project progress and ensuring it stays on track. Milestones are important reference points that mark the completion of critical phases or deliverables within the project [16]. Key dates are set in the project schedule for each major milestone and task.

Detailed initial planning and establishing milestones and key dates are fundamental for the success of a software development project [17], especially in educational settings with student teams in professional internships. By clearly defining objectives, allocating resources adequately, and setting a realistic schedule with specific milestones, the project can progress efficiently and be completed successfully [18]. These elements facilitate project management and control and provide a clear structure that benefits the learning and development of the students involved.

3.5 Project Execution

Daily Scrum Meetings in the context of a software development project using agile methodologies, daily scrum meetings help to ensure effective communication and coordination among team members [19]. These are brief meetings, usually 15 minutes long, and are held at the same time every day. The main objectives of daily scrum meetings are to review progress, identify impediments, and plan daily activities. For team of students they are fundamental, here are their main benefits:

- **Improved Communication:** Daily meetings ensure all team members are aware of progress and issues, improving coordination and collaboration.
- **Early Problem Detection:** Identifying and addressing impediments quickly prevents minor issues from becoming major obstacles.
- **Adaptability and Flexibility:** Daily planning allows the team to adapt quickly to changes and adjust priorities as needed.

Iterative and Incremental Development is a fundamental practice in agile methodologies, allowing the development team to deliver software in small functional parts, known as increments, over short and repetitive iterations [20]. This approach ensures continuous feedback, constant improvement, and adaptability to changes in client or user requirements.

Effective execution of a software development project with student teams in professional internships is based on rigorous implementation of daily scrum meetings and an iterative and incremental development approach. Daily scrum meetings ensure constant communication and rapid impediment resolution, while iterative and incremental development allows for continuous delivery of functional software, facilitating feedback and constant improvement [21]. This approach not only improves the quality of the final product but also provides valuable practical learning experience for students, preparing them for future challenges in the software industry [22].

3.6 Continuous Supervision

Continuous supervision is an essential component in software development projects, especially when working with student teams in professional internships. This practice facilitates that the team remains aligned with project objectives, stays motivated, and develops professionally. Key strategies for continuous supervision include:

- *Regular Follow-up Meetings:*
  - *Frequency and Format:* Regular follow-up meetings, generally weekly, are established to review team progress, discuss issues, and plan future activities.
  - *Follow-up Agenda:* Follow-up meetings include a review of achieved milestones, identification of current challenges, and planning corrective or support actions as needed.

    - *Progress Documentation:* The results of each follow-up meeting are documented to maintain a clear record of team progress and decisions made [23].
- *Performance Evaluations:*
    - *Key Performance Indicators (KPIs):* KPIs are used to measure team and individual performance, ensuring quality, time, and cost objectives are met [24].
    - *Constructive Feedback:* Constructive feedback is provided based on performance evaluations, highlighting both strengths and improvement areas [25].
    - *Code and Quality Reviews:* Regular reviews of the code developed by students ensure adherence to quality standards and coding best practices.
- *Risk Management and Problem Resolution:*
    - *Early Problem Identification:* Supervisors have to be vigilant for any signs of potential problems and act proactively to mitigate them [26].
    - *Conflict Resolution:* Strategies implemented to handle disagreements or interpersonal issues within the team effectively and constructively.
    - *Support and Resources:* Necessary resources and support are provided to overcome technical or logistical challenges the team may face.
- *Promoting Open Communication:*
    - *Communication Channels:* Clear and accessible communication channels, such as team meetings, instant messaging platforms, and emails, are established to facilitate open and transparent communication.
    - *Satisfaction Surveys:* Periodic satisfaction surveys are used to gather team opinions and suggestions, adjusting supervision strategies based on received feedback.

Mentoring Programs and Training Sessions provide guidance, support, and continuous learning opportunities, helping students acquire new skills and improve their performance [27]. Key components of these programs include:

- *Mentoring Programs:*
    - *Mentor Assignment:* Each student is assigned an experienced mentor, who may be a senior professional from the team or a subject matter expert. Mentors provide personalized guidance and continuous support.
    - *Individual Mentoring Sessions:* Regularly scheduled meetings between mentors and students to discuss progress, address challenges, and provide constructive feedback.
    - *Personalized Development Plan:* Mentors work with students to develop personalized development plans, setting specific goals and steps to achieve those goals.
    - *Emotional and Motivational Support:* Mentors also provide emotional and motivational support, helping students maintain

high morale and overcome difficult moments during the project.

- *Technical Training Sessions:*
    - *Initial Training:* Initial training covering the tools, technologies, and methodologies that will be used in the project. This training includes practical workshops and theoretical sessions.
    - *Continuous Training:* Throughout the project, continuous training sessions are organized to introduce new technologies, update knowledge, and reinforce technical skills. These sessions may include seminars, workshops, and online courses.
    - *Coding and Testing Practices:* Specific workshops on coding practices, software testing, and quality assurance, providing students with the skills needed to develop robust, high-quality software.
- *Soft Skills Development:*
    - *Effective Communication:* Workshops on effective communication techniques, both verbal and written, to improve collaboration and interaction with stakeholders.
    - *Teamwork and Leadership:* Sessions on team dynamics, conflict resolution, and leadership skills, preparing students to work effectively in teams and assume leadership roles when necessary.
    - *Time Management and Productivity Strategies:* Training on time management, task prioritization, and productivity strategies, helping students manage their tasks and meet deadlines.
- *Continuous Evaluation and Feedback:*
    - *Periodic Evaluations:* Periodic evaluations of student progress and performance, providing detailed and constructive feedback.
    - *Goal Review:* Reviewing and adjusting goals in the personalized development plan as needed, ensuring students are on track to achieve their professional goals.
    - **Progress Documentation:** Maintaining a detailed record of each student's progress and development, facilitating evaluation and future planning.

Continuous supervision and mentoring programs and training sessions are fundamental for the success of software development projects with student teams in professional internships [28]. These strategies not only ensure the project stays on track and meets quality and time objectives but also provide students with an enriching and valuable learning experience [29]. Through continuous supervision, personalized mentoring, and ongoing training, students develop the technical and soft skills necessary for their future careers in the software industry.

3.7 Testing and Quality Assurance

Implementation of Continuous Testing essential to ensure the quality and reliability of developed software [30]. In the context of a software development project with student teams in professional internships, continuous testing allows for early detection of errors and ensures that each product increment meets specified requirements.

Code Review and Quality Assurance are critical processes to maintain high quality standards in software development [31]. These processes help identify and correct errors, improve code maintainability, and ensure adherence to best development practices [32].

The implementation of continuous testing and code review are essential elements of quality assurance in software development projects, especially in student teams in professional internships. These practices not only ensure that the developed software is robust and high-quality but also provide students with valuable experience using advanced development tools and techniques. By integrating continuous testing, code review, and quality assurance into the development cycle, a culture of excellence and continuous improvement is fostered, preparing students to face challenges in the software industry [33].

3.8 Review and Retrospective

Procedures for Sprint Reviews a fundamental part of the agile Scrum framework, designed to evaluate the work completed during the sprint and gather feedback from stakeholders. These reviews ensure that product development aligns with client expectations and project objectives.

Methodology for Sprint Retrospectives are regular meetings held at the end of each sprint to reflect on the team's work process and seek ways to improve [34]. The methodology for sprint retrospectives includes a clear structure that fosters reflection, open discussion, and continuous improvement implementation [35]. Key steps in the methodology for sprint retrospectives are:

- *Identification of Improvement Actions:*
    - Based on the discussion, the team identifies specific actions to improve the work process in future sprints. These actions are clear, achievable, and assigned to specific individuals.
- *Documentation and Follow-up:*
    - *Retrospective Documentation:* Key points discussed during the retrospective, including agreed improvement actions, are documented.
    - Action Plan: A detailed action plan is developed to implement identified improvements, including responsible individuals and deadlines.
    - Monitoring Implementation: The Scrum Master monitors the implementation of improvement actions and reviews progress in future retrospectives, ensuring necessary adjustments are made and desired improvements achieved.

Sprint reviews and retrospectives are essential components for the success of software development projects using agile methodologies. Sprint reviews ensure the developed product aligns with client expectations and allow for continuous feedback, while retrospectives foster reflection and continuous improvement of the team's work process [36]. By following structured procedures for both activities, student teams in professional internships can significantly improve their performance, the quality of the software produced, and their learning experience.

3.9 Documentation and Delivery

Project Documentation Requirements a crucial component in software development, especially in educational and professional internship projects. Clear and complete documentation ensures all aspects of the project are well-defined, facilitates communication among team members and stakeholders, and provides a detailed record of product development [37]. Project documentation requirements include:

Formal Project Delivery Process the formal project delivery process is a critical phase marking the transition from development to software operation. This process ensures the final product meets client requirements and expectations and is ready for deployment and use.

Production Deployment: Implement the software in the production environment, ensuring all configurations and dependencies are correctly established [38].

Complete documentation and the formal project delivery process are essential for ensuring the quality and success of a software development project, especially in the context of student teams in professional internships. Detailed documentation facilitates understanding, maintenance, and evolution of the software, while a well-structured delivery process ensures the final product meets client expectations and is ready for effective use [39]. These practices not only improve the quality of delivered software but also provide students with valuable experience in managing and executing software development projects.

3.10 Evaluation and Learning

Methods for Evaluating Student Performance evaluating student performance in a software development project is fundamental to ensuring they acquire the necessary competencies and to continuously improve the educational process. Evaluation methods should be systematic, fair, and oriented towards the professional development of students. The main methods for evaluating student performance include:

- *Formative Evaluation:*

    - *Continuous Feedback:* Students receive continuous feedback throughout the project, based on daily observations and periodic reviews of their work. This feedback is specific, constructive, and improvement - oriented.
    - *Code Reviews:* Peer and mentor code reviews provide direct evaluation of programming skills and adherence to coding best practices.
    - *Follow-up Meetings:* Regular follow-up meetings between students and their mentors to discuss progress, address challenges, and plan improvement actions.
- *Summative Evaluation:*
    - *Performance Evaluations:* At the end of each sprint and the project, performance evaluations include analyzing objective achievement, work quality, and team contribution.
    - *Test Results:* Evaluation based on the results of software tests, including unit, integration, and user acceptance tests.
    - *Project Deliverables:* Review and evaluation of project deliverables, such as documentation, implemented functionalities, and software demonstrations.
- *Self and Peer Evaluation:*
    - *Self-Evaluation:* Students are encouraged to conduct periodic self-evaluations, reflecting on their own performance, identifying strengths, and areas for improvement.
    - *Peer Evaluation***:** Students participate in peer evaluations, providing and receiving feedback from their peers, fostering a culture of collaboration and mutual learning.
- *Evaluation of Soft Skills:*
    - *Communication and Collaboration:* Evaluation of communication and collaboration skills, based on meeting participation, team interaction, and contributions to problem-solving.
    - *Time Management and Responsibility:* Evaluation of students' ability to manage their time effectively, meet deadlines, and take responsibility in the project.
- *Technical Competency Evaluation:*
    - *Programming Skills:* Evaluation of technical competencies in programming, including the ability to write clean, efficient, and maintainable code.
    - *Agile Methodology Knowledge:* Evaluation of understanding and application of agile methodologies, such as Scrum and Kanban, and participation in agile ceremonies.

Documentation of Lessons Learned a critical process for capturing and analyzing experiences gained during the project, identifying improvement areas, and applying that knowledge in future projects [40]. This process contributes to continuous improvement and students' professional development. Key steps for documenting lessons learned include:

- *Information Collection:*

    - *Retrospective Meetings:* Conducting retrospective meetings at the end of each sprint and the project, where students and mentors discuss what worked well, what didn't, and why.
    - *Surveys and Questionnaires:* Using surveys and questionnaires to gather feedback from students, mentors, and stakeholders on different aspects of the project.
    - *Interviews and Discussions:* Conducting interviews and informal discussions with team members to delve into challenges faced and solutions implemented.
- *Analysis of Lessons Learned:*
    - *Pattern Identification:* Analyzing collected information to identify patterns and trends, both positive and negative, that impacted the project.
    - *Impact Evaluation:* Evaluating the impact of key practices, decisions, and events on the success or failure of different project aspects.
- *Formal Documentation:*
    - *Lessons Learned Report:* Creating a detailed lesson learned report that includes descriptions of key experiences, root cause analysis, and recommendations for future improvement.
    - Knowledge Repository: Storing the lessons learned report in an accessible knowledge repository for future teams and projects, ensuring the information is used and applied.
- *Application of Lessons Learned:*
    - *Continuous Improvement Plan:* Developing a continuous improvement plan based on lessons learned, including specific actions, responsible individuals, and timelines for implementation.
    - *Training and Dissemination:* Providing training sessions and workshops to share lessons learned with other teams and students, fostering a culture of learning and continuous improvement.
    - *Monitoring and Review:* Monitoring the implementation of improvement actions and periodically reviewing progress to ensure lessons learned translate into improved practices.

Evaluating student performance and documenting lessons learned are essential components for the success of software development projects in educational settings. Evaluation methods provide continuous and constructive feedback, helping students develop their technical and soft skills. Documenting lessons learned ensures that the knowledge gained during the project is captured, analyzed, and used to improve future projects [41]. These practices not only improve the quality of the work done but also prepare students to face challenges in the professional world and contribute to their comprehensive professional development.

4 Conclusions and future work

This article presents a comprehensive approach to implementing a software development process based on the ISO/IEC 29110 standard, specifically adapted for student teams in professional internships. Combining the ISO/IEC 29110 standard with agile methodologies provides a robust framework that ensures development quality and efficiency while fostering the flexibility and adaptability needed in educational settings. Key practices such as daily meetings, iterative and incremental development, and sprint reviews and retrospectives facilitate communication, collaboration, and continuous improvement.

Supervision, mentoring, and evaluation programs ensure students receive the necessary support to develop their technical and soft skills. Performance evaluation methods provide constant and constructive feedback, helping students improve and prepare for their future careers. Documenting lessons learned captures acquired knowledge and promotes continuous improvement.

Implementing continuous testing and code reviews ensures the developed software is of high quality and meets specified requirements. Complete and detailed documentation facilitates understanding, maintenance, and software evolution, ensuring the final product meets client expectations and is ready for effective use.

The proposal is based on the experience of fourth-year students participating in eight cycles of practice, each spanning six months and involving a total of 112 students. Derived from this process, future work involves continuing to refine and improve the ISO/IEC 29110-based software development process through data collection and analysis from multiple project cycles. This includes identifying best practices and eliminating inefficiencies to optimize the process.

Scope Expansion: Exploring the application of this process to different types of projects and contexts, including larger-scale projects and collaboration with external entities, is necessary. This will allow evaluating how students adapt to these changes and adjusting the process accordingly.

Integration of New Technologies: Incorporating new tools and emerging technologies into the development and testing process is suggested. This includes adopting DevOps practices, advanced test automation tools, and continuous integration platforms, thereby improving development efficiency and quality.

Research on Educational Effectiveness: Conducting studies to evaluate the educational impact of the process on students is crucial. This includes measuring the development of technical and soft skills, student satisfaction, and their preparation for the job market. Results from these studies can inform improvements in the educational process and the integration of new teaching methodologies.

These future work lines aim not only to maintain the relevance and effectiveness of the ISO/IEC 29110-based software development process but also to expand its applicability and continuously improve its educational and professional impact.

References

1. Ribaud, V., Saliou, P., O'Connor, R.V., Laporte, C.Y.: Software Engineering Support Activities for Very Small Entities. In: Riel, A., O'Connor, R., Tichkiewitch, S., and Messnarz, R. (eds.) Systems, Software and Services Process Improvement. pp. 165–176. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15666-3_15.

2. Ribaud, V., Saliou, P.: Process assessment issues of the ISO/IEC 29110 emerging standard. In: Proceedings of the 11th International Conference on Product Focused Software. pp. 24–27. ACM, Limerick Ireland (2010). https://doi.org/10.1145/1961258.1961264.

3. Galvan, S., Mora, M., O'Connor, R.V., Acosta, F., Alvarez, F.: A Compliance Analysis of Agile Methodologies with the ISO/IEC 29110 Project Management Process. Procedia Computer Science. 64, 188–195 (2015). https://doi.org/10.1016/j.procs.2015.08.480.

4. O'Connor, R.V.: The ISO/IEC 29110 Software Lifecycle Standard for Very Small Companies: In: Management Association, I.R. (ed.) Research Anthology on Agile Software, Software Development, and Testing. pp. 1884–1901. IGI Global (2022). https://doi.org/10.4018/978-1-6684-3702-5.ch090.

5. Negrete, M., Infante, U., Muñoz, M.: A Case Study of Improving a Very Small Entity with an Agile Software Development Based on the Basic Profile of the ISO/IEC 29110. In: Mejia, J., Muñoz, M., Rocha, Á., and Quiñonez, Y. (eds.) New Perspectives in Software Engineering. pp. 3–19. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-63329-5_1.

6. Niazi, B.A.S.: Training and Development Strategy and Its Role in Organizational Performance. jpag. 1, 42 (2011). https://doi.org/10.5296/jpag.v1i2.862.

7. Gonzalez-Morales, D., Moreno De Antonio, L.M., Roda Garcia, J.L.: Teaching "soft" skills in Software Engineering. In: 2011 IEEE Global Engineering Education Conference (EDUCON). pp. 630–637. IEEE, Amman, Jordan (2011). https://doi.org/10.1109/EDUCON.2011.5773204.

8. Włodarski, R., Poniszewska-Marańda, A., Falleri, J.-R.: Impact of software development processes on the outcomes of student computing projects: A tale of two universities. Information and Software Technology. 144, 106787 (2022). https://doi.org/10.1016/j.infsof.2021.106787.

9. Løvold, H.H., Lindsjørn, Y., Stray, V.: Forming and Assessing Student Teams in Software Engineering Courses. In: Paasivaara, M. and Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming – Workshops. pp.

298–306. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58858-8_31.

10. Dubinsky, Y., Hazzan, O.: Roles in Agile Software Development Teams. In: Eckstein, J. and Baumeister, H. (eds.) Extreme Programming and Agile Processes in Software Engineering. pp. 157–165. Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24853-8_18.

11. Cimatti, B.: DEFINITION, DEVELOPMENT, ASSESSMENT OF SOFT SKILLS AND THEIR ROLE FOR THE QUALITY OF ORGANIZATIONS AND ENTERPRISES .pdf, http://ijqr.net/journal/v10-n1/5.pdf, last accessed 2024/07/11.

12. Tam, C., Moura, E.J.D.C., Oliveira, T., Varajão, J.: The factors influencing the success of on-going agile software development projects. International Journal of Project Management. 38, 165–176 (2020). https://doi.org/10.1016/j.ijproman.2020.02.001.

13. Karunasekera, S., Bedse, K.: Preparing Software Engineering Graduates for an Industry Career. In: 20th Conference on Software Engineering Education & Training (CSEET'07). pp. 97–106. IEEE, Dublin, Ireland (2007). https://doi.org/10.1109/CSEET.2007.39.

14. Fernanda, S., Manuel, S., Germania, R., Samanta, C., Danilo, J., Patricio, A.: Agile methodologies applied in teaching-learning process in engineering: A case of study. In: 2018 IEEE Global Engineering Education Conference (EDUCON). pp. 1201–1207. IEEE, Tenerife (2018). https://doi.org/10.1109/EDUCON.2018.8363366.

15. Chow, T., Cao, D.-B.: A survey study of critical success factors in agile software projects. Journal of Systems and Software. 81, 961–971 (2008). https://doi.org/10.1016/j.jss.2007.08.020.

16. Sunmola, H.O.: Evaluation of Motivating and Requiring Factors for Milestones in IT Projects. Procedia Manufacturing. 51, 1469–1477 (2020). https://doi.org/10.1016/j.promfg.2020.10.204.

17. Paasivaara, M., Lassenius, C.: Collaboration practices in global inter-organizational software development projects. Softw Process Imprv Pract. 8, 183–199 (2003). https://doi.org/10.1002/spip.187.

18. Eckert, C.M., Clarkson, P.J.: Planning development processes for complex products. Res Eng Design. 21, 153–171 (2010). https://doi.org/10.1007/s00163-009-0079-0.

19. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Using scrum in a globally distributed project: a case study. Softw Process Imprv Pract. 13, 527–544 (2008). https://doi.org/10.1002/spip.402.

20. Flora, H.K., Chande, S.V.: A Systematic Study on Agile Software Development Methodologies and Practices. 5, (2014).

21. Shafiq, S., Hafeez, Y., Ali, S., Iqbal, N., Jamal, M.: Towards Scrum Based Agile Framework for Global Software Development Teams. Mehran Univ. res. j. eng. technol. 38, 979–998 (2019). https://doi.org/10.22581/muet1982.1904.11.

22. Mundra, A., Misra, S., Dhawale, C.A.: Practical Scrum-Scrum Team: Way to Produce Successful and Quality Software. In: 2013 13th International Conference on Computational Science and Its Applications. pp. 119–123. IEEE, Ho Chi Minh City, Vietnam (2013). https://doi.org/10.1109/ICCSA.2013.25.

23. Stray, V., Sjøberg, D.I.K., Dybå, T.: The daily stand-up meeting: A grounded theory study. Journal of Systems and Software. 114, 101–124 (2016). https://doi.org/10.1016/j.jss.2016.01.004.

24. Beisheim, N., Stotz, F.: Key Performance Indicators for Design and Engineering. In: Stjepandić, J., Rock, G., and Bil, C. (eds.) Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment. pp. 341–351. Springer London, London (2013). https://doi.org/10.1007/978-1-4471-4426-7_30.

25. Glassey, R., Balter, O.: Put the Students to Work: Generating Questions with Constructive Feedback. In: 2020 IEEE Frontiers in Education Conference (FIE). pp. 1–8. IEEE, Uppsala, Sweden (2020). https://doi.org/10.1109/FIE44824.2020.9274110.

26. Leveson, N.: A systems approach to risk management through leading safety indicators. Reliability Engineering & System Safety. 136, 17–34 (2015). https://doi.org/10.1016/j.ress.2014.10.008.

27. Abdool, A., Pooransingh, A.: An industry-mentored undergraduate software engineering project. In: 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. pp. 1–4. IEEE, Madrid, Spain (2014). https://doi.org/10.1109/FIE.2014.7044180.

28.Vaughan, K.: The role of apprenticeship in the cultivation of soft skills and dispositions. Journal of Vocational Education & Training. 69, 540–557 (2017). https://doi.org/10.1080/13636820.2017.1326516.

29. Al-Meshari, A., Kokal, S.: A Mentor/Mentee Training Program: A Success Story. In: All Days. p. SPE-108854-MS. SPE, Anaheim, California, U.S.A. (2007). https://doi.org/10.2118/108854-MS.

30. Hamdan, S., Alramouni, S.: A Quality Framework for Software Continuous Integration. Procedia Manufacturing. 3, 2019–2025 (2015). https://doi.org/10.1016/j.promfg.2015.07.249.

31. McIntosh, S., Kamei, Y., Adams, B., Hassan, A.E.: The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects. In: Proceedings of the 11th Working Conference on Mining Software Repositories. pp. 192–201. ACM, Hyderabad India (2014). https://doi.org/10.1145/2597073.2597076.

32. Wang, Y., Zhang, X., Yu, L., Huang, H.: Quality Assurance of Peer Code Review Process: A Web-Based MIS. In: 2008 International Conference on Computer Science and Software Engineering. pp. 631–634. IEEE, Wuhan, China (2008). https://doi.org/10.1109/CSSE.2008.1141.

33. Holck, J., J  rgensen, N.: Continuous Integration and Quality Assurance: a case study of two open source projects. AJIS. 11, (2003). https://doi.org/10.3127/ajis.v11i1.145.

34. Erdoğan, O., Pekkaya, M.E., Gök, H.: More effective sprint retrospective with statistical analysis. J Software Evolu Process. 30, e1933 (2018). https://doi.org/10.1002/smr.1933.

35. Hundhausen, C., Conrad, P., Tariq, A., Pugal, S., Flores, B.Z.: An Empirical Study of the Content and Quality of Sprint Retrospectives in Undergraduate Team Software Projects. In: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training. pp. 104–114. ACM, Lisbon Portugal (2024). https://doi.org/10.1145/3639474.3640074.

36. Wood, R.: Agile Project Management. (2024). https://open.ocolearnok.org/informationsystems/chapter/chapter-13-agile-project-management/

37. Taulavuori, A., Niemelä, E., Kallio, P.: Component documentation—a key issue in software product lines. Information and Software Technology. 46, 535–546 (2004). https://doi.org/10.1016/j.infsof.2003.10.004.

38. Rodríguez, P., Haghighatkhah, A., Lwakatare, L.E., Teppola, S., Suomalainen, T., Eskeli, J., Karvonen, T., Kuvaja, P., Verner, J.M., Oivo, M.: Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software. 123, 263–291 (2017). https://doi.org/10.1016/j.jss.2015.12.015.

39. Kajko-Mattsson, M.: A Survey of Documentation Practice within Corrective Maintenance. Empirical Software Engineering. 10, 31–55 (2005). https://doi.org/10.1023/B:LIDA.0000048322.42751.ca.

40. Chirumalla, K.: Development of a methodology for lessons learned practice: from post-project learning to continuous process-based learning. Luleå University of Technology, Luleå (2013).

41. Chelimsky, E., Shadish, W.R. eds: Evaluation for the 21st century: a handbook. Sage Publications, Thousand Oaks, Calif (1997).