Reflective Practice in Software Development studios: findings from an ethnographic study

Tania Mara Dors
PPGIa
PUCPR
Curitiba, Brazil
tania.dors@ppgia.pucpr.br

Frederick M. C. Van Amstel
Design
UTFPR
Curitiba, Brazil
vanamstel@utfpr.edu.br

Fabio Binder
PPGIa
PUCPR
Curitiba, Brazil
fabio.binder@pucpr.br

Sheila Reinehr
PPGIa
PUCPR
Curitiba, Brazil
sheila.reinehr@pucpr.br

Andreia Malucelli
PPGIa
PUCPR
Curitiba, Brazil
malu@ppgia.pucpr.br

***Abstract*— Over the last two decades, software educators have adopted new approaches, techniques, and tools for practical learning. Previous research has found that studio-based learning, which has been in use by some design and architecture courses, is suitable for learning the practical aspects of software engineering. The studies available recognize the presence of reflective practice in software development studios; however, they do not show empirical evidence of its contribution to learning. The goal of this study is to understand the use of reflective practice in software development studios and its contributions to the practical learning of software engineering. The qualitative data were collected using an ethnographic method through participant observation and from written students' self-reflections, which were analyzed using Cycle Coding Method supported by Atlas.ti Qualitative Data Analysis tool. The findings suggest that reflective practice promotes the emergence of new ideas while contributing to the development of skills that are valuable to software engineering professional practice. This research presents a qualitative look at how these are developed in the context of a particular software development studio.**

Keywords— Software Engineering Education, Computer Science Practical Learning, Software Practice, Studio-based Learning, Reflective Practice, Software Engineering Practical Skills Development

I. INTRODUCTION

Corporations and employers have complained publicly about the lack of professional awareness and low levels of communication and teamwork skills from engineering undergraduates [1]. When students finish university, they have the technical knowledge, but not necessarily an adequate professional competence. As a response to these complaints, SE educators have been discussing and experimenting new approaches and tools to make SE education more practical [2], such as laboratory instruction, electronic whiteboards, tablet computers, problem-based learning, active learning, flipped classroom and studio-based learning.

Studio-based learning is a pedagogical approach to practical learning, borrowed from architecture and industrial design, and it has been in use in software disciplines by some universities worldwide since 1990 [3]. Studio-based learning is a response to the mismatch between what employers perceive as necessary abilities and how universities prepare graduates for employment, particularly regarding non-technical skills and the changing expectations and learning styles of students [4]. The studio-based approach emphasizes the development of reflective skills and sensibilities. "The essence of the studio concept is 'reflective practice" [3].

The collaborative learning in the studio helps the students to develop their own skills by practice and groups appeared to be genuinely interested in the work of the other groups. Moreover, the dynamic interconnection of the set of elements in a studio like people, software tools, subject policies and procedures, development methodology, processes, techniques, documents, practices, and products seems to provide a network or web in which software development knowledge and skills are co-created [4].

Despite the spread of this educational approach, the available studies on software development studios mostly describe the implementation experience, focusing on facilities, concepts, and definitions [3, 5, 6, 7, 8,]. Some of them focus on the learning outcomes [9, 10, 11, 12, 13, 14,]. Few of them draw enough attention to the fundamental notion of reflective practice15.

Software studios' experience seems to be efficacy in preparing students to work as software professionals according to the feedback from students and their managers [16]. The exposition to the real-world environment and industry practices in the studio provides the development of professional skills and working in groups, the development of good communication, and the ability to work in interdisciplinary teams. [17] studio also seems to supports the development of a set of skills that cover a range of required employability skills desirable in career profile, according to Career EDGE Employability Development Profile [18].

Software engineering usually accomplishes and supports the adoption of reflective practice [19]. Recognizing the importance

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

of reflection in practice applied to software, IEEE Software devoted an exclusive track in one of the 2014 issues to reflective practice. According to [20]: "Reflection often takes place in cycles of experience followed by the conscious application of learning from that experience, during which a software developer might explore comparisons, ponder alternatives, take diverse perspectives, and draw inferences, especially in new and/or complex situations." In SE education, "reflective practice is now recognized as important for software developers and has become a key part of software studios in universities" [21]. Many papers claim that reflection in the studio is mandatory. However, this is a non-trivial exercise for undergraduate students [21].

The presence of reflective practice is acknowledged in SE education literature; however, few empirical studies verify that. More seriously than that, to the best of our knowledge, there are no studies that can shed light on how reflective practice contributes to the practical learning of software engineering. We know a lot about learning outcomes, yet little about the learning process. This research aims to contribute to bringing some light to this field by providing an in-depth qualitative study to investigate how the reflective practice contributes to the software development and the development of individual competencies needed for professional practice in SE. The next section discusses the concept of reflective practice and software development studios, followed by the research method, and the study results.

II. Background

A. Reflective Practice

Reflective practice is a form of reflective learning by doing identified by Donald Schön [22] in his observations of students' and professors' interactions in architectural studios. Based on this, he proposed a fundamental reorganization of how to think about professional practice and the relationship of theory to practice. He formulated his view of design in terms of "reflective activity" and related concepts, such as "reflective practice", "reflection-in-action", "knowing-in-action", "reflection-on-action" and reflective "conversations with the material" of a design situation.

Reflection-in-action is the reflective form of knowing-in-action, which means, the reflection during the problem-solving process. In the reflection-in-action process, doing and thinking are complementary. "Doing extends thinking in the tests, moves, and probes of experimental action, and reflection feeds on doing and its results. Each feeds the other, and each sets a boundary for the other" [22].

Sometimes, after the problem-solving process is finished, a practitioner may consider what he could have done differently or would do differently next time. This is called reflection-on-action or reflecting on experience. "We may reflect on action, thinking back on what we have done in order to discover how our knowing-in-action may have contributed to an unexpected outcome" [23]. It does not matter if it happens after the fact or during a pause, during the action, in both cases, our reflection has no direct connection to present action.

Knowing-in-action is the knowledge built into and revealed by our performance of everyday routines of action. The knowing-in-action sometimes is labeled as "intuition", "instinct" or "motor skills". In such cases, we continually control and modify our behavior in response to changing conditions [23].

Schön noted that the practice of any profession involves the use of special esoteric "knowledge in action", that according to Polanyi [23] it is tacit knowledge learned not in the abstract but use. He stated that there are three ways of acquiring this kind of knowledge. The first, very unusual, is via self-instruction. The second is via apprenticeship, learning "on line" in "real world" contexts. Because this is both inefficient and can have adverse severe real-world effects, the standard site of learning is the '*practicum'. The practicum is* an educational setting that approximates the world of practice.

According to Schön's observations [22], reflective practice helps students to acquire the kind of artistic talent essential for competence in undetermined areas of practice. Professional artistry refers to the types of competence that practitioners demonstrate in certain practice situations that are unique, uncertain, and conflicting. Nevertheless, he highlighted two points. The first point is that the knowing-in-action characteristic of competent practitioners in a professional field is not the same as the professional knowledge taught in the schools. The second point is that skilled professional practitioners often can generate a new process of Knowing-in-action through reflection-in-action developed in the indeterminate zones of practice [24].

The architectural design was the first professional domain studied by him to develop his epistemology of professional practice based on the concepts of reflection-in-action and knowledge-in-action. He analyzed design education on-site, providing, and studying audiotaped protocols from teaching-learning sessions in the design studio. His first objective was to grasp the central features of education in design.

After that, he extended his analysis to other professions, testing his hypothesis that all professions are "design like' in some relevant aspects. For him, 'design like' professions are those in which there is a pre-conceptualization of design for subsequent execution [25]. He concluded that reflective practice teaching is the key to professional education in general, and he considered that the architecture design studio could be a model for practitioners of other fields [26].

Thus, an analysis was made on the application of the reflective practitioner perspective to the profession of software engineering, which resulted in a framework for adopting this perspective in general and the studio method of teaching in particular into SE education. This analysis also suggested that the adoption of reflective practice methodology as a cognitive tool might help programmers in developing software systems and the students to understand software development methodologies [27].

The design studios observed by Schön, as well as their instructional methods based on practical learning with coaching, is historically related to the tradition of the atelier model, spread by the École des Beaux-Arts and the Bauhaus, which will be explained on the next section. Additionally, the studio education applied for software engineering, and its educational model will be further discussed.

Identify the applicable funding agency here. If none, delete this text box.

B. Studio Concept

Schön referred to studios as a reflective practicum: "A *practicum* is a setting designed for the task of learning practice. In a context that approximates a practice world, students learn by doing, although their doing usually falls short of real-world work." [23]. He argued that the fundamental concepts of designing could be only understood in the context of the doing, through the experience of designing. He believed that reflection-in-action was the basis of any design process. For the new student learning to design, this poses the problem that they are seeking to learn things they cannot grasp ahead of time.

Feedbacks help students understanding their problems, eliminating errors from their proposed solutions, also eventually building their own solutions. Feedbacks are provided in different ways in the studio. Coaching and critique support reflective practice in a studio education. Coaching is about giving instruction or advice. The teaching staff "function as coaches whose main activities are demonstrating, advising, questioning, and criticizing" [23].

Critiques are an essential pedagogical tool in the studio, and these are based on the instructor's expertise and professional experiences [24]. Critique means a systematic and objective examination of an idea, phenomenon, or artifact. Critique is often shortened to "crit", and is used to describe an individual or small group critiques; desk crit is a one-on-one session generally between learner and teacher, and student and critic. Final reviews, or juries, or final critique is used to describe the final, formal, summative critiques [9]. Desk crit is a collaborative activity when the teacher and the student of design work together, discussing and sketching possibilities, and imagining the consequences of design choices. The design teacher works to understand what the student is trying to do with his or her design work and provides feedback on these ideas and works with the student to further develop them. Another form of crit is the design jury. That is when the student presents its drawings and describes their design to three to five local architects, instructors from other studios at the same school, other non-studio faculty members, or representatives of the client if there is one.

Design education researchers refer to several types or settings of critiques, which the instructor uses to interact with students: desk crit, group crit, interim review, final review, and informal interaction [28]. Besides, peer critiques that are comparable cognitive value as well as a significant component of some methods of cognitive apprenticeship [9].

However, not all these types or settings of critique necessarily occur in a software engineering studio, where are usually group projects. Another critical point is that informal critiquing sessions tend to be more constructive and formal critiquing sessions more evaluative.[28].

C. Software Studio

A software studio is an attempt to utilize similar environments to that of a design studio for software-related disciplines. Many names have been given to these studios, such as software studio, software development studio, and software design studio.

The studio-based approach emphasizes the development of reflective skills and sensibilities. "The essence of the studio concept is 'reflective practice" [5]. Software studio helps to reflect on the strengths and weaknesses of current software design practice, beyond it encourages us to consider what we might want to borrow from the culture of architecture to promote the good design of software and systems. Besides, "The studio format supports a relationship with a real client and introduces students to user participation in the design process" [29]. Nevertheless, "Experience from a studio course in software design provokes creative reflection on engineering design education, and on how it may be improved" [6].

A software studio framework was defined to facilitate and guide the creation of a software studio as a learning practice environment [30]. Interpretations of studio education for software disciplines (Computer Science, Software Engineering, Information Technology, etc.) have been explored in the last two decades, with the earliest known software studio implemented at Carnegie Mellon University (CMU) [5]. Worldwide, there are other studio-based approaches in Australia, USA, Poland, and the United Kingdom, implemented in different ways, as a single discipline, as a studio course or as some semesters of studio experience integrated into the curriculum of undergraduate or master's degree [31].

In most of the studies about software studios, the key role of reflective practice is acknowledged. However, few empirical studies verify that. To the best of our knowledge, there is not yet a study on software studios, which is based on careful observation of student and professor interactions, such as the study done by Schön in architectural studios to find the phenomenon [23]. Without such a study, it is difficult for SE education literature to build upon the reflective practice concept and go beyond its mere acknowledgment.

III. Research

A. Objective

The objective of this study is to understand the reflective practice contributions to the practical learning of software engineering in a software development studio. To pursue this objective, instead of relying on protocol studies [32], the method that Schön used for his observations of student-professor interaction [23], this research adopts an ethnographic perspective [33, 34] in the hopes of better grasping longitudinal student-to-student interactions as well, which are essential for software studios. Software studio projects, differently than the architectural projects conducted by a single student that Schön observed, are pursued by a group of students, sometimes from different disciplines.

B. Research Method

The qualitative data were collected through participant observation [35] of mobile application development and from written students' self-reflections made by the end of the project development. The qualitative data analysis [36] followed the Cycle Coding Method, which considers that coding is a cyclical act, so the result is rarely reached in the first cycle of coding data [37]. The Atlas.ti Qualitative Data Analysis tool supported coding.

The research execution stages were: 1) data collection; 2) data preparation; 3) data coding; and 4) data analysis of the individual teams, the ethnography notes, and students' self-reflection. First, the characteristics of the observed studio will be described, and then the results of each stage.

C. Fieldwork Contextualization

The software studio observed is a result of a partnership between university and industry, which offers a two-year undergraduate or graduate complimentary education program centered on the development of mobile applications. This is a software studio program at PUC PR called Apple Developer Academy located in Curitiba-PR. It follows some partnership guidelines and requirements, as the use of a specific method called Challenge Based Learning (CBL) to support the development of the project [38, 39].

Studio staff consists of instructors composed of both programmers and designers, daily available at the studio. For the two-year extension course, 50 graduate students or to be graduated within six months were selected. Among them, designers, developers, and devigners, which are students able to work with both skills, as designers and developers.

In the first year, students learn about concepts and essential practices for software designers, developers, and devigners (a hybrid of both profiles). In the second year, additional practices are provided, besides workshops about usability, monetization, interface design, game design, and others. Students must dedicate fifteen hours a week to the studio. They are encouraged to develop projects not only to meet the course curriculum but also to make their products marketable with the support of a university business department.

The curriculum of the course includes individual and group projects development and proposes to publish the last mobile application developed on the App Store. Each project of the course has different characteristics and time for the conclusion and is called a challenge. Before each challenge begins, the team gives presentations on tools and contents related or required to the challenge, in technical teaching classes or even workshops, including subjects as usability, monetization, interface design, game design.

The interactions between instructor and student occur based on the weekly feedback that students get from instructors concerning their design, at different stages of the development path, in the studio sessions. Instructors evaluate students through a series of presentations and discussions. On the other hand, students continually reflect and revise their projects through the process of working on them and presenting their work, often publicly, and in this case, receiving feedback from the instructor and colleagues, or sometimes external visitors.

The project selected for data collection was one of the projects or challenges developed during the two-year course of this software studio. The first challenge of the course was individual, and for the second and third challenges, groups were formed. The selected project was the third challenge called Mini Challenge [40]. For data collection, the researcher randomly selected two teams of the Mini Challenge for observation.

D. Data Collection

The data for this research is based on the observation of a project that was developed in the software studio, between August and November of 2017, from now on referred to as the Mini Challenge. The staff divided the students into teams, each one composed of 4 to 5 members. The team's composition considered that their members should not have worked together on the previous challenge.

Instructors gave some guidelines for students about the general objectives of the challenge and its goals. The overall objectives for the challenge were to publish an iOS game, tvOs or watchOS in the App Store, to reduce the execution time of a very complex task and perform teamwork with different people (students that not worked together in previous challenges). Also, the challenge goals were to make better use of available production resources to agile parallel development (game design, art, development, business, and marketing) and do what is necessary beyond the assigned roles of each member.

Data were collected daily following the schedule of the software studio mobile application development. The equipment and instruments used to collect were pens, paper, and audio recorders. The participant observation occurred in two teams randomly selected: one developed an adventure game for iPhone (Team A) and another, a musical game for iPad (Team B). At the end of the project, students wrote self-reflections, which were used as documents that contain some evidence of learning outcomes.

E. Data Preparation

After the data collection, it was required a data pre-analysis of collected audio recordings to select them for transcription. So, it was selected audio recordings of all meetings held during all phases of the Mini Challenge project development, from the first project definition meeting until the final presentation of the product, and also all the audio recordings of each studio session. It included interactions made by students on their own team, with students from different teams, with third-party developers, or instructors during project development.

The written self-reflections of the students published into a software studio web page were converted into a word document, and the ethnographic notes were typed in word document by date and time. Finally, each audio recording selected in the pre-analysis step was transcribed into a word document, too.

F. Data Coding

The documents were imported into Atlas.ti software for coding. Besides, the ethnographic notes where analyzed and inserted as comments or added complementary into the memo writings in the corresponding audio recording document during the first coding cycle. In the case of the audio recordings, each quotation of a document from a different speaker was analyzed and codified, individually and required different coding cycles, as represented in Figure 1.

In the first coding cycle, besides the introduction of the ethnographic notes, it was applied to the structural coding to organize the data identifying the speaker (actors of the speech),

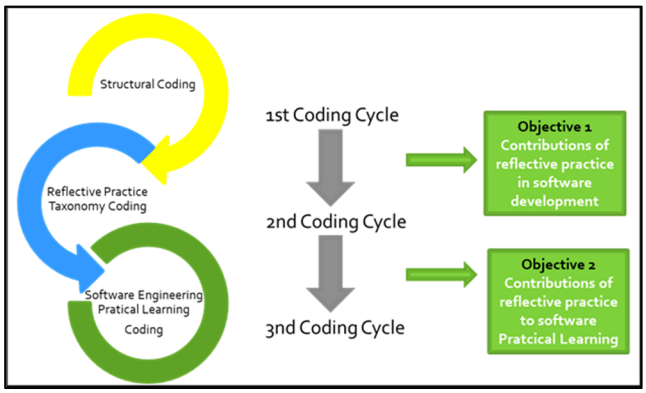

Fig. 1. Research's coding cycles

the project phase, studio session, and type of interaction.

The identified speakers were named one by one. Among them, five instructors, four students of Team A, four students of Team B, and nine students from other teams that made some interaction or reflection in the presentations made by Team A and B or even during the development of their projects. The types of interactions identified refer to the interaction between the student and the instructor, among teammates, with colleagues from another studio team, and with third-parties. The project phases related to SE life cycle were Design, Development, Test, and Project Delivery. Furthermore, the studio sessions were Group Crit, Interim Review Design Idea, Interim Review Design Crit, Final Review, Final Presentation, and also Peer Critique.

The instructor interactions were codified as instructive, guidance, and reflective interaction. It was considered that instructive interaction is when the instructor adds information about something, guidance when the instructor gives guidelines related to software engineering based on the literature or technical knowledge. Finally, instructor reflective interaction is when the instructor tries to provoke a student's reflection about the subject.

For the second coding cycle, it was applied Taxonomic Coding using codes from the reflective practice concept: reflection-in-action, reflection-on-action, and conversation with the material. The objective of these two initial coding cycles was to analyze the reflective practice, its occurrence by type of interaction, frequency, and outcomes rightly related to the individual codes of reflective practice, which results correspond to the first research objective. One more coding cycle with a focus on practical learning was held to analyze the second objective.

IV. Results

In this session, we present the evidence of the studio's characterization, the findings from participant observation, the findings from document analysis, and finally, a cross-analysis that stems from the triangulation of the findings.

A. Studio Characterization

First, the researcher had observed the characteristics of the environment to confirm if it was a software studio according to Bull's framework [30], checking each category of this framework to verify the adherence, once this studio was created before its definition. The studio under evaluation is compliant to software studio definitions, including its structure that fully corresponds to the categories of studio framework developed by Bull [30], which are Physical environment, Facilitation of studio, Modes of education, Awareness, Critique, Culture, Individual's characteristics, Inspiration, Collaboration, and Digital Technology. Also, this studio holds the activities that are commonly associated with reflective practice, such as group crits, interim review, final review, peer critique, and coaching, which can partially confirm the applicability of this concept to software development studios. The studio sessions help and stimulate students to reflect on the critiques, questions, and suggestions from instructors and peers of the studio course, giving support to reflective practice.

B. Findings from the participation observation

As previously explained, the challenge was to develop gameplay in a short period with a free theme. Thus, each project phase was short and required focus and commitment.

Initially, the students had to define the game in terms of objective, customer focus and theme, technical and environmental features, and using the CBL framework to support and guide the project activities. First, the team had to identify the type of game all members liked, and each student researched and brought some related gameplay references to analyze together.

After that, they had to decide among different possibilities for the design, in which situations appear to manage in terms of roles and relationships, planning and acting, information gathering and sharing, problem analysis and understanding, concept generation and adoption, and conflict avoidance and resolution.

Throughout the participant observation, it was noticed that in terms of team interaction, this phase was very rich because team members had to negotiate strategies, and, depending on the level of commitment and alignment with other team members, students adopted appropriate persuasion strategies. They carefully moderated their commitment to their ideas to remain amenable to negotiation. They appealed to common sense, design theories, standard practices, expert practices, user preference, and demonstrations with physical hardware to persuade.

The next step was to present the idea to the instructor in a Group Crit session before presenting it to the other teams and instructors at Design Idea Interim Review. By the end, they reflected on the questions and suggestions from these crit sessions and realigned to begin the development of the gameplay prototype.

A third party under team guidelines was responsible for developing the gameplay soundtrack. So, students had to learn how to integrate with the program and manage outsourcing issues. When they finished the prototype, they had to show it at the Interim Review called Design Crit. One experienced design teacher conducted this session to provide a prototype's critique that will guide students to conclude the project.

At the end of the project, each team presented its final project to all other groups and instructors and external stakeholders

Team A

From reflective practice coding analysis, it was noticed that reflection-in-action resulted in creation (Table IV.1IV), reflection-on-action and material conversation resulted in problem-solving, decision-making, planning, scope management, project management, and time management.

Table IV.1IV - Team A - Reflection-in-action

Code	Quotation
[Instructor Reflective Interaction]	"[...] (Instructor1**):** I consider interesting the idea of a game that stimulates sins. However, I think it will be nice if the game brings some reflection. [...] If you can do something ambiguous enough, it will generate an interesting debate […]
[New Idea]	"[...] (StudentA1): 7 sins were changed to vices. (StudentA2): [...] Not to get too attached to an idea of religion, because the seven sins are closely linked to an idea of religion. [...]
[Reflection-in-action]	(StudentA3): Would the right name be vice? (StudentA2): This is how he describes virtue and vice. Virtue is a positive habit, and vice is negative. (StudentA3): This I wanted to know. Nice! Perfect then. Because as long as people can do what you said, Dante and all. Because the seven deadly sins ended up delimiting to a Catholic church and its definition, right. (StudentA4): Exactly! (StudentA3): So, that is interesting, it opens our minds to see. Because sometimes we find some of the nine sins interesting, more interesting than one of the 7. (StudentA2): Also, have heresy and lies. [...] (StudentA3): But the interesting thing about vices is that we can get into the discussion we started yesterday about current vices like this. What would you consider a vice? [...]"

For instance (Table IV.2IV), when the third-party developer delivered the sound interface for testing, the team realized that the interface had not met the expectations of the team either the submitted requirements. Then, the team did a reflection-on-action to understand if they had correctly communicated the requests to the third-party developer, before preparing the change request document. They concluded that they have successfully communicated the requirements, then did a reflection-in-action on how to do the third-party understand their requirements and decided to submit other video examples and more references with comments.

In another situation, in the middle of the project development, the team realized it would be necessary to reduce the scope to meet the project schedule. The team had to rethink the game's design, to deal with the impacts in the game's narrative, to think in a new distribution of the game stages and to consider the time necessary for development. As a consequence of this, they practiced problem-solving, decision-making, planning, project management, time management, scope management, scope reduction, and more than exercising and enhancing those skills. They had a practical learning experience.

Table IV.2IV - Team A - Conversation with the Material - Reflection-on-action

Code	Quotation
[Conversation with Material] [Reflection-on-action] [Game Stages] [Scope change]	"[…] (StudentA1): So, the point is our initial goal were three stages, […] If we cut to make these two other stages, all three will be poorly made. We cannot handle everything well done. (StudentA2): We prefer to deliver a round phase, good kind of play. (StudentA1): We are changing the phase 1 puzzles not to give the feeling that something is missing. […]".

Therefore, when one encountered a product error, the team first did a reflection-on-action to understand what went wrong, and after a reflection-in-action to problem-solving, as well as taking the actions needed to do so. On the other hand, the conversation with the material, that refers to a conversation with the developed game, resulted in a product dissatisfaction. This led to a reflection-in-action and required a product change, as well as other activities arising from this.

As a result of project observation, it was noticed that students were required to practice skills like communication, collaboration, leadership, teamwork, interpersonal capabilities, and conflict management. Furthermore, throughout the development of the mobile application, students demonstrated engagement, commitment, flexibility, and adaptability, from team organization to technical challenges and new learnings. They also demonstrated the ability to negotiate, to plan, and to document.

The students worked in a team and collaboration. In the design phase, students worked in groups immersively, researching, and studying some game references related to the types of games they would like to develop. They brainstormed ideas collaboratively and, when discussing them, wrote them on stick-notes and placed them on the whiteboard to define the design of the project game. In the development phase, they divided the tasks, with the developers working on the code at the same time, the designer built the design assets, and the developer was responsible for documenting, organizing and making available to the team all documents with project definitions, as well like presentations,

Interpersonal relationship problems occurred with this team during the project development. Depending on the point of view, it could be understood as a conflict of leadership or role, unbalanced distribution of project activities, or lack of empathy. The researcher's goal was not to identify exactly the causes or reasons, but to note that this situation usually happens in companies or real-world jobs and to emphasize that the studio favors this type of experience, too.

Team B

From reflective practice coding analysis, it was noticed that reflection-in-action resulted in the creation, reflection-on-action resulted in problem-solving, and material conversation resulted in problem-solving, decision-making, planning, scope management, project management, and time management.

This team did a peer critique session with a colleague of another studio team, as illustrated in Table IV.3. This student

made a checkpoint of every aspect of the game design. He asked about the overall back story, the theme of the game, the levels, rules, and patterns in the game. Also, about the content design, the characters, items, puzzles, and quests. Regarding the user interface, how the player would receive information and feedback, and how the player would interact with the game. Some of this student's questions helped them to reflect-in-action on some aspects that were not defined yet, mainly related to the game's mechanics.

Technical research is a technical skill practiced in the design and development phase, from distinct ways. At the design phase, students practiced this skill when they were searching for references to the games related to the game they would develop. On the other hand, at the development phase, students exercised it when they need to find a library of codes and an adequate application to work with sound. Technical learning refers to how developers program sound features developed by third-party in the game application.

Table IV.3IV - Team B - Peer Critique

Code	Quotation
[Student Other Team Interaction] [Reflective Interaction] [Game Mechanics] [Soundtrack]	"[…] (StudentO1): Then, the goal is to make contact with the home planet, right? And, the way you are going to do that is by gathering an x score? (StudentB4): That is not right yet, but the idea is a correct sound sequence or a particular score. (StudentO1): And once I get this sound sequence, what happens? Do I step up, or is the game over? (Student B4): Step up to the next phase of the game. (StudentO1): Phase shift. Then will you make different sequences, and will the complexity of the sequence increases with each phase? (Student B4): Exactly! (StudentO1): Okay. And, how will these sequences be? Or, is not set yet? (StudentB4): Hum, we already talked to the third-party developer, and he will help us put together the right sequences, to do kind of harmonic things, and make some sound that makes perfect sense. (StudentO1): And what will the input look like? (StudentB4): You will touch the screen. You will touch the screen at the right time to jump from planet to planet. [...] (StudentO1): But honestly, I cannot imagine the gameplay. I am thinking of how it will be. I will have to hit the right note button. That is the mechanics. Maybe in the right position. Right note in the right position, is it? (StudentB4): Hmm! Hmm! The idea is this is how you get sound feedback. The game is not really for you to squeeze tighten, but to keep the pace so that the music is harmonious to your ears. Because when music has this kind of failure, it is boring, right. [...]".
[Reflection-in-action] [Unforeseen Situation]	"[…] (StudentO1): And where am I going to press? On top of the planet, when it is falling or anywhere? (StudentB4): I had not thought about that yet. [...]"

Learning Experience was a finding observed when students mentioned an experience of a previous challenge to decide on the development. These past experiences were reported by a student from this team, and a student of another studio team, regarding a previous challenge they had worked together, as shown in Table IV.4IV.

Table IV.4IV - Team B - Learning Experience

Code	Quotation
[Student Other Team Reflective Interaction] [Learning Experience]	"[...] (Student B4): Yes. Can I say a fear I have about games that depend on the story? A situation happened to Him (refers to StudentO1). (StudentO1): Ha! There is! There is! (Laughs) (Student B4): Our first challenge, the game *Fiona in the Nebola,* is to have a beautiful story, but we could not develop it. The game was limited because we could not develop the story. We wanted to do things and could not implement it because we had no story. No one could think of a decent story to make the game, and it is over! […]"

Teamwork, collaboration, communication, interpersonal, time management, planning, problem-solving, decision-making, scope management, and project management are skills required for practicing of SE that the students practiced during this project development.

Throughout project development, the researcher noticed that students worked as a team and collaborated. In the design phase, students worked in a group in an immersive manner to define the design. They researched and observed some game references related to the types of games they would like to develop the project. Next, they discussed the ideas collaboratively and wrote them in Post-it® to put the project memory on the whiteboard. In the development phase, they divided the tasks, being that each developer prepared a game feature separately and joined in the end to work together on the code. The designer built the design assets, and the devigner documented, organized, and shared all the documents with project definitions and also a presentation to the team.

They developed oral and written communication, when preparing and presenting their project in the studio sessions, or when making verbal or digital communication within the team. They already had to integrate because they had not worked together on previous challenges.

Same as, the researcher had observed the student's flexibility and adaptability, from team integration to the pursuit of new techniques and software applications for project development. In several situations, students demonstrate commitment and engagement to the project development, as taking pictures of himself with a helmet to get the best angle to create the asset of the game's character or asking for friends to test their game to collect their feedbacks. Also, the ability to negotiate, plan, and to document.

C. Findings from the self-reflections analysis

The students did their individual written self-reflections at the end of the Mini Challenge, which coincided with the end of the first year of the studio course. No rule nor guidelines to write this document were established. Thereby, it was noticed that students analyzed their own experience in the studio as an educational place and/or on the point of view of the proposed challenge.

Students of team A listed some skills used or developed during the project, among them commitment, collaboration,

communication, and teamwork. They reported a particular situation occurred during the development that required skills of communication, interpersonal capabilities, and conflict management, consequently brought them personal learning. A student said the team's commitment to the development of the project. Another student reported collaboration and teamwork. Moreover, one student reported technical learning, two-game development first experience; one of them said design learning, one programming learning, two reported learning by practice, learning experience, and personal learning. From an educational environment perspective, students of Team A said it was challenging, required dedication, and availability, moreover they felt motivated to work on it.

Regarding Team B, the self-reflections mentioned only the studio. They reported they were commitment, had to work with team collaboration to improve communication, interpersonal abilities, and teamwork. From a technical perspective, they said technical learning, design, and programming learning as well as planning, project management, time management, learning by practice, and personal learning.

D. Findings from a cross-analysis

To consolidate the research outcomes, it was made a cross-analysis of each team's participant observation with its respective self-reflections, as well as the outcomes related to the contributions from reflective practice to software development and the development of the individual competences.

So, as represented in Table IV.5, from the cross-analysis of the self-reflection results and the Mini Challenge practical learning contributions, it can be seen that in Team A, regarding the way students relate and interact with each other, the outcomes were Collaboration, Commitment, Communication, Conflict Management, Interpersonal and Teamwork, and regarding technical issues, programming. On the other hand, the team reported design learning and game developing the first experience, and both refer to the full process of software development, encompassing technical issues as well as a learning experience, personal learning, and learning by practice.

As showed in Table IV.5, Team B concerning the way one relates to and interacts with other people, the outcomes were Collaboration, Commitment, Communication, Interpersonal, and Teamwork. Regarding technical issues, programming, and technical research. The team still reported design learning, planning, project management, and time management, which concerns to the full process of software development, including technical learning, apart from the learning experience, personal learning, and learning by practice.

Through this cross-analysis, it was possible to confirm some outcomes of research analysis in the perception of the students, since they matched directly. In the case of the outcomes from reflective practice coding, like decision-making, problem-solving, scope management, and project management, although not explicitly mentioned in self-reflections, they are closely connected with design learning, project business learning, and learning by practice that was reported by teams.

Besides, throughout the project development, the researcher noted in the participant observation that students of both teams had to practice and develop adaptability and flexibility skills for team integration and to search and learn new techniques and software applications needed for project development — also, the ability to negotiate, document, and leadership.

Therefore, the practical learning contributions found in the analysis of both teams from reflective practice or matching of self-reflection and project observations results are the research's outcomes related to contributions to the development of the individual competences.

Besides, by observing the codes in Table IV.5, we were able to identify the technical and non-technical skills required for SE practitioners. Among the non-technical skills, we identified some related to cognitive skills and others of behavioral or attitude skills.

Table IV.5 - Cross analysis of findings from both teams and methods

Analysis Codes	Team A		Team B	
	Participant observation	***Self-reflections***	***Participant observation***	***Self-reflections***
Adaptability	X		X	
Analytical		X		
Availability		X		
Creation	X		X	
Collaboration	X	X	X	X
Commitment	X	X	X	X
Communication	X	X	X	X
Communication		X		
Conflict Management	X	X		
Decision-Making	X		X	
Design Learning		X		X
Interpersonal	X	X	X	X
Interpersonal Problem		X		
Game Development		X		
Leadership	X			
Learning by Practice		X		X
Learning Experience		X	X	
Pair Programming	X		X	
Personal Learning		X		X
Planning	X		X	X
Problem Solving	X		X	
Programming		X		X
Project Business				X
Project Management	X			X
Self-Confidence		X		
Scope Management	X		X	
Teamwork	X	X	X	X
Technical Research	X		X	X
Technical Learning			X	X
Time Management	X		X	X

E. Practical Learning Outcomes

It was noticed that the contributions of reflective practice to software development from reflective practice codes were emerging of new ideas and skills developing of problem-solving, decision-making, planning, project management, scope management, and time management.

On the other hand, by participant observation of Mini Challenge development, it was noticed skills development, like collaboration, verbal or written communication, commitment, interpersonal savvy, communication, adaptability, flexibility, teamwork, negotiation, and managing of outsourcing

developing. Also, the use of learning experience and changing of experience between peers from different teams. Both teams had to learn to manage the outsource development for sound features. On the technical aspect, it was observed: researching technical references, learning how to work, and integrate sound features with the application and self-learning of the new technique, like pair programming.

From the analysis of self-reflections of the teams and Mini Challenge, practical learning contributions of these teams were identified: learning by practice and skills developing of collaboration, commitment, communication, conflict management, interpersonal, personal learning, and teamwork. On technical aspects, design learning, technical research (refers to references of games for designing and programming), technical learning (refers to applicative to lead with sound or audio kits), and programming. Finally, to find the contributions of reflective practice to the development of individual competencies in a software studio, as the data comes from distinct methods of collection, its results were triangulated.

Thereby, the contributions of reflective practice to the development of individual competence and the artistic talent in a software studio are practice and development of skills as collaboration, verbal or written communication, commitment, interpersonal, adaptability, flexibility, teamwork and management of outsourcing developing; learning by practice of mobile software development and pair programming.

Moreover, when students justify decisions or directions based on experience from a previous challenge, it becomes evident that students acquired some learning experience, technical, or personal throughout the studio course.

Thus, one can observe what Schön argued on reflective practice, that it helps students acquiring a kind of artistic talent essential for competence in professional practice, which refers to types of competence that practitioners demonstrate in certain practice situations that are unique, uncertain, and conflicting.

V. Conclusions

This study was performed in one specific software development studio, and it is only covering mobile application development. The research was conducted by one of the researchers and reviewed by others to reduce bias. Also, to minimize bias, a triangulation between the observation and self-reflection was performed.

Given these limitations, it is possible to conclude that the reflective practice in this software development studio contributed to mobile software development as to the development of essential skills recommended by software engineering and computer science curricula. [41, 42].

The reflective practice promotes the process of emerging new ideas and contributes to the practice and development of skills, like collaboration, oral or written communication, commitment, interpersonal, adaptability, flexibility, and teamwork. Besides, it also develops problem-solving, decision-making, planning, project management, time management, scope management, managing of outsourcing development, and new technical skills. In addition to that, the reflective practice emphasizes practical learning, supports the development of technical skills, and seems to be an authentic environment of the relationship between academic disciplines and real-world experiences, where students can practice and learn by practicing, thereby preparing students for the real world.

Reflective practice is not like a content that can learn about, but it is a means for learning practical skills, it is a means for "learning through". The studio' instructors conducted the coach and critique as follow: encouraging students to do the work, while becoming aware of the decisions they were making and the actions they were taking; asking "why" questions and making the "you considered" suggestion, not doing the work for students nor giving the solution. Besides, they reacted to students' work to help them see how they can succeed and how to avoid going the wrong way in the future. Evidence of reflective practice was found in the software studio. The students were motivated to the reflection by coaching, critique, and besides, by the CBL method, which incentives the students to reflect throughout the project development. Thus, it was noticed that CBL contributes to reflective practice, although more research is needed to confirm it. It could be seen that the studio helps to build a culture supportive of critique, emphasizes practical learning, and requires the use and the development of some skills necessary for software engineering practice, which Schön called of development of "artistic talent" required for practitioners.

Finally, the studio is an educational place where students can practice and learn by practicing. Consequently, it better prepares students for real-world practice, thereby better preparing them for practice. Therefore, as Schön argued, reflective practice helps students acquiring the kind of artistic talent essential for professional practice.

Acknowledgment

I would like to thank all the staff, and the students who participated from a software studio program at PUC PR called Apple Developer Academy located in Curitiba-PR Brazil, which admitted to making a participant observation of the development of their projects for this research.

I would like to thank Atlas.ti Scientific Software Development GmbH to provide a software license and support of this software for the development of this qualitative study.

This study was financed in part by the Coordenação de Aperfeiçoamento Pessoal de Nível Superior – Brazil (CAPES) – Finance Code 001.

References

[1] T. B. Hilburn, G. Hislop, D. J. Bagert, M. Lutz, S. Mengel and M. McCracken, Guidance for the development of software engineering education programs, *Journal of Systems and Software*, v. 49, Issue 2, p. 163-169, 1999.

[2] T. C. Lethbridge, J. L. Diaz-Herrera, R. J. LeBlanc and J. B. Thompson, Improving software practice through education: Challenges and future trends, *Future of Software Engineering FOSE 07*, v. 2, p. 12-28, 2007.

[3] J. E. Tomayko, Teaching software development in a studio environment, in *Proceedings of the twenty-second SIGCSE technical symposium on Computer science education (SIGCSE '91*), v. 23, issue 1, 1991, ACM, pp. 300-302, March 1991.

[4] J. Prior, A. Connor and J. Leaney, Things coming together: learning experiences in a software studio, in *Proceedings of the 2014 conference*

on Innovation & technology in Computer Science Education, 2014, IEEE, pp. 129-134.

[5] S. Kuhn, O. Hazzan, J. E. Tomayko and B. Corson, The software studio in software engineering education, in *15th Conference on Software Engineering Education and Training (CSEE&T 2002)*, Proceedings, Kentucky, USA, p. 236-238, 2002.

[6] O. Broadfoot and R. Bennett, Design Studios: Online? Comparing traditional face-to-face design studio education with modern Internet-based design studios, in *paper presented at the Apple University Consortium*, pp. 1–13, 2003.

[7] S. Kopczyńska, J. Nawrocki and M. Ochodek, Software development studio - Bringing industrial environment to a classroom, in *2012 1st International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex 2012)*, 2012, pp. 13-16.

[8] B. Hokanson. The Design Critique as a Model for Distributed Learning, in *The Next Generation of Distance Education. Ed. by Leslie Moller and Jason B. Huett. Springer US. Chap. The design*, v. 9781461417, pp. 71–83, 2012.

[9] A. Carbone, J. Sheard, A studio-based teaching and learning model in IT, in *Proceedings of the 7th annual conference on Innovation and technology in Computer Science education (ITiCSE'02)*, 2002, v. 34, Issue 4, pp. 213-217.

[10] K. Cennamo, S. A. Douglas, M. Vernon, C. Brandt, B. Scott, Y. Reimer and M. McGrath, Promoting creativity in the Computer Science design studio, in *Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE'11)*, 2011, ACM, pp. 649-654.

[11] A. Danielewicz-betz and T. Kawaguchi, Gaining hands-on experience via collaborative learning: Interactive Computer Science Courses, in *2014 International Conference on Interactive Collaborative Learning (ICL)*, 2014, IEEE, December, pp. 403-409.

[12] A. Baker and André Van Der Hoek. An experience report on the design and delivery of two new software design courses, in *Proceedings of the 41th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. Chattanooga, Tennessee, USA: ACM, pp. 519-523, March 2009.

[13] C. D. Hundhausen; Anakrati Agrawal. Pawan Agarwal. Talking about Code: Integrating Pedagogical Code Reviews into Early Computing Courses. ACM Transactions on Computing Education, v.13, Issue 3, article 14, 28 pages, August 2013.

[14] J. Lee, G. Kotonya, J. Whittle and C. N. Bull. Software Design Studio: A practical example, in Proceedings of the *37th IEEE International Conference on Software Engineering (ICSE 2015)*. Florence, Italy, IEEE/ACM, v. 2, pp. 389-397, 2015.

[15] C. N. Bull and J. Whittle, Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach, *IEEE Software,* 2014, IEEE, v.31 Jul/Aug, Issue 4, p. 44-50.

[16] T. Nurkkala and Stefan Brandle. Software Studio: Teaching Professional Software Engineering. In *Proceedings of the 42th ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. Dallas, TX, USA: ACM, pp. 153-158, March 2011.

[17] D. Rosca. Acquiring professional software engineering skills through studio-based learning In *17th International Conference on Information Technology Based Higher Education and Training, (ITHET'18)*, IEEE. Olhao, Portugal, pp. 1-6, Abril 2018.

[18] J. Prior, Laudari Suman, John Leaney. What is the Effect of a Software Studio Experience on a Student's Employability? *In: Proceedings of the Twenty-First Australasian Computing Education Conference (ACE'19),* ACM, Sydney, NSW, Australia, pp. 28-36, January 2019.

[19] O. Hazzan and J. E. Tomayko, The reflective practitioner perspective, in *eXtreme Programming, Proceedings of the XP Agile Universe 2003, New Orleans, Louisiana, USA*, p. 51-61.

[20] T. Dybå, N. Maiden and R. Glass. The Reflective Software Engineer: Reflective Practice, *IEEE Software*, 2014, IEEE.

[21] J. Prior, S. Ferguson and J. Leaney, Reflection is Hard: Teaching and Learning Reflective Practice in a Software Studio, in *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '16). Canberra, Australia*, pp. 7-15, February 2016.

[22] D. A. Schön. Reflective Practitioner: How Professionals Think in action. New York: Basic Books Inc., 1983, 374 p.

[23] D. A. Schön. Teaching artistry through reflection-in-action,in *Educating the Reflective Practitioner: Toward a new design for teaching and learning in the professions*. San Francisco: Jossey-Bass. Re-printed with permission of the publisher via Copyright Clearence Center, Inc. Reproduced with permission at Teachers College, Columbia University, pp. 22-40, 1987.

[24] D. A. Schön. *Educating the Reflective Practitioner: Toward a new design for teaching and learning in the professions*. First Edition, San Francisco, CA, US: Jossey-Bass, 1987.

[25] L. J. Waks, Donald Schon's Philosophy of Design and Design Education, *International Journal of Technology and Design Education*, v.1, p. 37-51, 2001.

[26] A D.A Schön. The Theory of Inquiry: Dewey's Legacy to Education. The Massachusetts Institute of Technology Cambridge, 1992, Massachusetts Published online: 15 Dec 2014. Curriculum Inquiry, v. 22, issue 2, p. 119-139.

[27] O. Hazzan, The reflective practitioner perspective in software engineering education, *The Journal of Systems and Software*, v. 63, Issue 3, p. 161–171, 2002.

[28] O.Yeonjoo, S. Ishizaki, M. D. Gross and E. Yi-Luen Do, A theoretical framework of design critiquing in architecture studios, in *Design Studies 34.3*, pp. 302–325, 2013.

[29] S. Kuhn, The software design studio: An exploration, *IEEE Software*, IEEE, v. 15, Issue 2, p. 65-71, 1998.

[30] C. N. Bull, J. Whittle and L. Cruickshank, Studios in Software Engineering Education: Towards an Evaluable Model, in *International Conference on Software Engineering (ICSE 13)*, 2013, pp. 1063–1072.

[31] C. N. Bull and J. Whittle, Observations of a Software Engineering Studio: Reflecting with the Studio Framework, in *IEEE 27th Conference on Software Engineering Education and Training (CSEE&T 2014)*, 2014, IEEE, pp. 74-83.

[32] A. T. Purcel, J. S. Gero. Drawings and the design process, in *Design Studies vol 19, issue 4*, pp. 389–430, 1998.

[33] D. M. Fetterman. *Ethnography Step by Step*, SAGE Publication Ltd., 2010, 200 p.

[34] C. Nippert-Eng. *Watchig Closely. A Guide to Ethnographic Observation.* Oxford University Press, 2015.

[35] J. P. Spradley. *Participant Observation*. New York: Holt, Rinehart and Winston, 1980.

[36] U. Flick. *An introduction to qualitative research*. Four Edition SAGE Publication Ltd., Printed in Great Britain by Ashford Colour Press Ltd., 2009, 504 p.

[37] J. Saldaña. *The Coding Manual for Qualitative Researchers*. London: SAGE Publication Ltd, 2016, 339 p.

[38] M. Nichols, K. Cator, M. Torres and D Henderson.*Challenge Based Learning User Guide*. Redwood City, CA. Digital Promise, 2016.

[39] F. V. Binder, M, Nichols, S. Reinehr and A. Malucelli, Challenge Based Learning Applied to Mobile Software Development Teaching, in *IEEE 30th Conference on Software Engineering Education and Training (CSEE&T 2017)*, IEEE, 2017, Savannah, Georgia, USA, November, pp.57-64.

[40] F. V. Binder, R. Albuquerque, S. Reinehr and A. Malucelli, Innovation and Active Learning for Training Mobile App Developers, in *International Conference on Software Engineering (ICSE 20)*, 2020, pp. 151-161.

[41] A. Sobe, T. C. Lethbridge, J. L. Diaz-Herrera and J. Barrie Thompson, *IEEE/ACM Joint Task Force on Computing Curricula. Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, IEEE Computer Society Press and ACM Press, August 2004.

[42] Software Engineering 2014. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. IEEE/ACM Joint Task Force on Computing Curricula IEEE Computer Society and Association for Computing Machinery. A Volume of the Computing Curricula Series. February 2015.