Informatics in Education, 2021, Vol. 20, No. 2, 277–296

DOI: 10.15388/infedu.2021.13

An Empirical Study on the Use of Student-Focused Approaches in the Software Engineering Teaching

Carlos dos Santos PORTELA[1], Alexandre Marcos Lins de VASCONCELOS[2], Sandro Ronaldo Bezerra OLIVEIRA[3], Mauricio Ronny de Almeida SOUZA[4]

[1]*Faculty of Information Systems, Federal University of Pará (UFPA), Cametá, Brazil*
[2]*Informatics Center, Federal University of Pernambuco (UFPE), Recife, Brazil*
[3]*Graduate Program in Computer Science, Federal University of Pará (UFPA), Belém, Brazil*
[4]*Computer Science Department, Federal University of Lavras (UFLA), Lavras, Brazil*
e-mail: csp@ufpa.br, amlv@cin.ufpe.br, srbo@ufpa.br, mauricio.ronny@ufla.br

Received: March 2020

Abstract. The software industry is not satisfied with the preparation level of newly graduated professionals in Computing undergraduate courses. There is a predominance of traditional approaches to the Software Engineering (SE) teaching which proved to be inefficient, because they focus on the content from the professor´s viewpoint. This research aims to investigate if the use of student-focused approaches in the SE teaching can develop more technical competencies to apply in industry than when traditional approaches are applied. For this, an iterative model has been defined to integrate the main student-focused approaches and a controlled experiment was carried out in four undergraduate courses. The data were collected from structured interviews with students and analyzed using ANOVA. The results showed no significant statistical difference between student-focused and traditional teaching approaches in the development of SE competencies. However, these results were impacted by the motivation and commitment of the experiment students.

Keywords: student-focused approaches, software engineering teaching, competencies development.

1. Introduction

Software Engineering (SE) can be defined as "the application of a systematic, disciplined, and quantifiable approach to the development, operation and maintenance of software" (IEEE Computer Society, 2014a). In this context, it is of paramount importance to train professionals of excellence in SE, so that they can develop high quality software. So, the Computing undergraduate courses (e.g. Information Systems and Computer Science) try to prepare their students to work in the software industry (Subrahmanyam, 2009). However, the industry has complained that such undergraduate courses do not teach students all the competencies they need to get their jobs done efficiently (Ng and Huang, 2013).

In this way, software companies have to complement the knowledge of the new graduates with extra training in order to provide the other competencies related to the software engineering (Dagnino, 2014). This lack of training of the professionals may be the result of inadequate teaching in the University courses (Fox, 2015).

According to Mirian-Hosseinabadi *et al*. (2010), many professors adopt traditional approaches to teaching SE, such as expository classes, which turn out to be inefficient because they are content-focused rather than competencies-focused. Ng and Huang (2013) identified many gaps among the practices used by different companies versus how students learn at their undergraduation courses.

1.1. *Research Question and Goal*

In order to address these problems, it is necessary to increase the amount of technical competencies developed during the students' formation. So, the main research question is:

Research Question (RQ): *If the professors of the Software Engineering (SE) courses adopt a set of student-focused teaching approaches and industry practices rather than adopting traditional approaches, will students develop more technical competencies to apply in software industry?*

To answer this research question, it is necessary to understand the current student-focused teaching approaches described in the literature and the training strategies adopted by the software industry. In general, the software industry focuses on short-term training for new employees (Dagnino, 2014). This can favor the development of general competencies focused in specific technologies. Thus, if these training strategies were adapted to the academic environment, they could favor the development of technical competencies in a gradual way (Ng and Huang, 2013). Thus, the research goal presented in this article can be defined as:

Research Goal (RG): *Define an iterative model based on the main student-focused approaches that are applied in the Software Engineering teaching, incorporating the training practices adopted by the industry, so that students develop technical competencies at the application level.*

The objective of this research is not to create a new teaching approach, but to integrate the practical approaches identified in the literature into a single iterative model in order to establish a learning cycle (Kolb, 1984) and to contemplate the different learning styles of students (Campbell and Johnstone, 2010).

As a differential, this model incorporates training practices adopted by the software industry, such as coaching and mentoring (Gomes *et al*., 2015). In this way, this model intends that students can develop technical competencies at the level of knowledge application, in a way similar to the level required by a new professional in the Software Engineering area (IEEE Computer Society, 2014b).

1.2. *Software Engineering Teaching*

There is a consensus among researchers that practical approaches are most suiTable for teaching SE (Malik and Zafar, 2012) (Marques, Quispe and Ochoa, 2014) (Mirian-Hosseinabadi *et al*., 2010). Some of the authors emphasize that this teaching should be more student-focused in order to increase the students' motivation, participation, relationship with classmates and, consequently, their learning (Malik and Zafar, 2012).

On the other hand, there is no consensus on "how" to teach SE, since each teaching institution adopts its own methods or approaches according to the experience of its professors (Marques, Quispe and Ochoa, 2014). Initially, the approach based on practical projects was widely adopted. However, there is not a currently predominant approach to conduct these practical experiences (Malik and Zafar, 2012). During the last years, alternative approaches have been proposed and adopted for SE education (Mirian-Hosseinabadi *et al*., 2010). Depending on the context of teaching, the approaches are more or less effective.

Most of these studies highlight the need to provide the student with the opportunity to undertake practical activities, but do not specifically mention any pedagogical approach to doing so. Although there are several proposals reported in the literature, there is no clear solution to deal with the challenges encountered in SE teaching (Marques, Quispe and Ochoa, 2014). In the proposed model, it was decided to adopt the practical approaches reported in the literature, since they allow a more student-focused teaching.

1.3. *Adaptation of Industry Practices*

Since software industry adopts its own training strategies to compensate the complain that undergraduate courses do not properly train professionals, a possible solution would be to anticipate the use of these practices in the training of undergraduate students (Dagnino, 2014). To this end, Portela *et al*. (2017) identified which training strategies are used by Software Process Improvement (SPI) consultants to develop competencies and abilities in professionals. Given the various limitations of the academic environment, the authors adapted the industry practices selected according to the resources available in the academy.

Initially, Portela *et al*. (2017) mapped SE topics of the ACM/IEEE curriculum guidelines to the specific practices of the CMMI-DEV model (SEI, 2010). Then, they carried out a survey of training strategies with ten SPI consultants, who also act as SE professors. Finally, these strategies were adapted and incorporated in a SE Course and their impacts in the student learning was analyzed. The results showed an increase of approximately 20 percent in student perception of learning about SE topics with the use of industry training strategies.

The main contribution of this work (Portela *et al*., 2017) was to provide a set of industry practices that can be adapted to the academic context in SE teaching: dynamics, use of games, coaching and mentoring. These practices were incorporated in the teaching model presented in this article.

1.4. *Article Structure*

In addition to this introductory section, Section 2 presents the research methods, highlighting the design and the participants of a controlled experiment; Section 3 describes the proposed teaching model in order to integrate the student-focused approaches identified in the literature with the industry practices identified in Portela *et al*. (2017); Section 4 reports the execution of the controlled experiment, through the application of the model in four SE courses; Section 5 presents a quantitative and qualitative analysis of the results obtained in the experiment; Finally, Section 6 presents the contributions, conclusions and future work of this research.

2. Research Methods

2.1. *Experiment Design*

Seeking to answer the RQ presented in Subsection 1.1, the following null hypothesis was defined:

> **H0:** *Student-focused teaching approaches allow the development of less, or equal, Software Engineering technical competencies, at the application level, than the traditional teaching approach.*

In "student-focused teaching approaches", the focus is on the subject and on the relationship among the people involved in the teaching and learning process (Richardson, 2018). In these approaches, the teaching content is seen as external and plays a secondary role. Thus, the professor should be a "facilitator of learning", providing the conditions for students to learn. On the other hand, the "traditional teaching approach" is characterized by the transmission of knowledge accumulated by the professor (Kropp and Meier, 2014). In this approach, the professor can act independently of the interests of the students in relation to the contents of the course.

The term "competencies" is defined by the ACM/IEEE Curriculum Guidelines (ACM/IEEE, 2013) as the ability to remember previously taught content (to know), understand the information and the significance of the content presented (to understand) and use the learned content in new and concrete situations (to apply).

The dependent variable "technical competencies" in Software Engineering was measured using the SWECOM evaluation process (IEEE Computer Society, 2014b) to determine if the independent variable "student-focused teaching approaches" has effect on the final result.

In the H0 hypothesis, the treatment is "student-focused teaching approaches" and the experiment's control is the "traditional teaching approach". The alternative hypothesis is:

> **H1:** *Student-focused teaching approaches allow the development of more Software Engineering technical competencies, at the application level, than the traditional teaching approach.*

The "teaching approaches" variable was measured from the systematic reviews of Malik & Zafar (2012) and Marques, Quispe and Ochoa (2014). The independent variable "traditional teaching approach" (Kropp and Meier, 2014) was manipulated in the experiment to evaluate its influence on the dependent variable.

The research protocol defined for this experiment followed the guidelines of Jedlitschka, Ciolkowski and Pfahl (2007). Initially, in Stage I – Theoretical Background, the problems, the goals and the research context were identified in order to find the necessary information for the next steps of the controlled experiment. Next, the studies related to the hypotheses and variables involved in the experiment were identified. In Stage II – Experiment Planning, the objectives of the experiment, the study sample, the variables operationalization, the design and the procedures of data analysis were defined. Those data were analyzed through ANOVA analysis of variance (Easterbrook *et al*., 2007).

In Stage III – Experiment Execution, the professors were trained in the model (see Section 3) and a course material was developed. This stage was performed in April 2018 (Course A-I and Course B-I) and in September 2018 (Course A-II and Course B-II). The training lasted 2 hours, where the model documentation and an application example were presented. It is important to note that professors already had experience in using student-focused teaching approaches. All of them had already done practical projects and used games or dynamics in classroom. This facilitated their understanding of the proposed model. Then, the professors developed the course material (around 2 weeks) following the training instructions. This material was reviewed and adjusted by the model's authors. Next, data collect instruments were applied in order to ascertaining the prior knowledge of the students and forming the groups of the experiment in a leveled way.

The data collecting was performed through structured interviews based on the SWECOM competency assessment procedure (IEEE Computer Society, 2014b). This instrument consisted of a spreadsheet to assess a list of technical skills of SE professionals. For each SE skill, the interviewer marked an option ("have", "need" or "gap") (IEEE Computer Society, 2014b) according to the students' discursive response. Each students' statement that clearly showed a skill was marked with "have". If the student said that he/she did not have the skill or answered incorrectly, the option "need" was marked. If the response were incomplete, it was marked as "gap". At the end of the interview, each "have" was scored as 1 point and each "gap" was scored as 0.5 point. The "need" option was not scored. So, all points were summed and the percentage of skills for the knowledge unit evaluated was obtained. This process of data collecting was aided by MS Excel spreadsheets.

At the end of the course, the same instruments were applied to collect data aiming to identify the competencies developed during the teaching approaches application. In the post-course interview, it was considered the professors' observation and the students'

performance in the practical project of the course. So, in Stage IV – Data Analysis, the students' competency level in both approaches (traditional and student-focused approaches) were identified, according to the difference between the pre and post-course number of skills identified in the interviews.

2.2. *Population Sample*

The target audience for the controlled experiment was: professors who are teaching a course in the SE area; students who are attending the course of the SE area. A consent term was presented to the target audience of the experiment in order to obtain the commitment to the research. The experiment was executed in the undergraduate degree in Information Systems at the Federal University of Pará (UFPA), Cametá Campus, in the first semester of 2018 and in the second semester of 2019.

In Course A-I of the experiment, 12 students from the "Management of Software Projects" subject were involved and in Course A-II, 19 students from the same subject. In Course B-I, 18 students from the "Analysis and Design of Systems I" subject participated and in Course B-II, 17 students.

In each one of the four courses, students were divided into three groups in order to isolate the evaluated approaches:

i) Traditional approach.
ii) Student-focused approach.
iii) The proposed teaching model (student focused approach plus industry practices).

Students from Course A-I and Course A-II were in the 6th semester of the undergraduate and most of them had links with university research projects. This meant that they had a certain maturity to handle responsibility and teamwork. Students from Course B-II and Course B-II were in the 5th semester and most of them had no extra-class experience. These characteristics impacted on the results obtained (see Section V-B).

3. The Proposed SE Teaching Model

The proposed model establishes well defined phases, arranged in an iterative cycle in order to meet different learning profiles, which are based on the classification proposed by Fleming and Mills (1992). The iterative cycle is based on the learning theory of Kolb (1984) and on the iterative teaching methodology proposed by Gary *et al*. (2013).

Fig. 1 presents the six steps that make up this teaching model and the iteration flow between them.

This model focuses on the reading of technical articles that describe experience reports and on video lessons with topic fundamentals, combining problem-based learning (PBL), discussion of practical cases, use of games, simulators or dynamics, practical project realization and reflection.

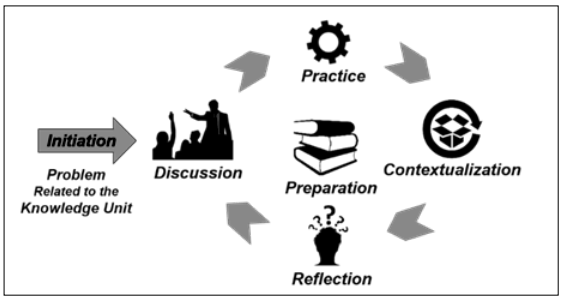

Fig. 1. Interactive Model for Software Engineering Teaching.

Thus, its instantiation allows to present, practice and apply the topics of SE units in order to provide a more contextualized learning for the students. Consequently, it allows to develop the technical competencies necessary for the students to enter in the job market.

In the following subsections, each phase of this model is described. To contextualize this description, the Requirements Engineering (RE) knowledge unit, used in the Course II of the experiment, was selected as an example.

3.1. *Initiation Phase*

Initially, the professor should select a knowledge unit that will be the object of study. In addition, it is necessary to identify which competencies are intended to be developed in the students. Taking as example the Requirements Engineering unit, the competencies to be developed are:

- Software Requirements Elicitation.
- Software Requirements Analysis.
- Software Requirements Specification.
- Software Requirements Verification and Validation.
- Software Requirements Process.

Each of these competencies is divided into a set of activities (IEEE Computer Society, 2014b). For example, to get the Software Requirements Analysis competency the student should:

- Uses appropriate domain analysis techniques.
- Performs analysis of requirements for feasibility and emergent properties.

For a new professional in the SE area, the SWECOM (IEEE Computer Society, 2014b) suggests that the level required is to be able to "assists in domain analysis". To select competencies for other units, the professors should consult the SWECOM model (IEEE Computer Society, 2014b). The selected competencies will determine the technical activities to be developed in the Practice phase (see Subsection 3.4). Subsequently, the study of each knowledge unit is started from the identification of a problem related to the project to be developed.

This phase is strongly based on the PBL approach and on the coaching process. The main intervention of the professor when acting as a coach is the elaboration of meaningful questions and the establishment of challenging goals.

3.2. *Preparation Phase*

This phase occurs in parallel to all phases in the iterative cycle presented in Fig. 1. It consists in watching videos related to the units and in the reading of articles with reports of practical cases of the industry in order to base, exemplify and create an understanding on how these topics are applied. Thus, textual (articles) and visual (video) stimulation occur. This preparation can be done in the classroom, assisted by the professor, or at home (extra-class), at the initiative of the students.

For the Requirements Engineering unit, the professor can search for articles with experience reports in the Proc. of RE conferences, as well as videos in repositories. The professor can also create his/her own material (article or video).

The main objective is to provide resources so that the student can build his/her theoretical background through the study of concepts and definitions related to the knowledge unit studied.

Besides, videos and articles with experience reports may be selected as the focus of the Discussion phase.

3.3. *Discussion Phase*

The students should conduct a Discussion with a specialist in the knowledge unit. If is not possible the visit of a professional to the classroom, the discussion can take place based on a video, article reading, videoconference or even through a technical visit to a software company. This discussion allows students a closer look on how certain topics are applied in the professional practice of a software engineer, as well as a critical analysis and the possibility of reuse the proposed solutions by these professionals.

The discussion between students, professor and guest professional should happen around the problem raised in the Initiation phase or identified when reading articles or watching videos. Other possibility would be the professional to describe problems faced in the day-to-day of his/her company and the students question about these problems.

Because this phase involves counseling and motivation, it is recommended that such a professional has at least a basic knowledge of the mentoring process in order to guide other people in the methods, tools and techniques that he/she uses while performing his/her activities. After the discussion is done, students are expected to have most of their questions answered and obtain suggestions on methods, tools and techniques to be applicable in the Contextualization phase.

Before applying their knowledge, it is necessary for the students to understand it by performing playful activities in the Practice phase.

3.4. *Practice Phase*

This phase allows students to internalize certain competencies based on practical activities, like the use of games, that allow living learning experiences that are not possible through lectures and theoretical discussions. Additionally, it allows professors to motivate and stimulate students in learning. To the students, it allows the learning of concepts related to the content of games, besides developing aspects of interaction and communication with their peers.

At this phase, the professor can also adapt and apply training practices of the industry. Thus, a recommendation is to perform dynamics (interactive activities) related to the knowledge unit or a training workshop on the use of a support tool. Both practices allow interaction between the students and the facilitator (the professor or a guest professional).

3.5. *Contextualization Phase*

In this phase, students will finally integrate the acquired competencies through the realization of a practical project of software development. This project seeks to contextualize the application of the topics through the adoption of immersive teaching/learning techniques, such as the use of real clients/problems, definition of deadlines (schedule), and the division of roles and responsibilities by students.

The main objective is to provide an experience similar to the industry projects. Thus, students will be able to learn techniques related to project definition, management, programming, validation, analysis, user interviews, documentation, among others.

In addition to the technical part, this experience should allow developing skills to evaluate potential solutions, negotiation with clients, group work, communication and interpersonal relationships. However, it is important to highlight that the professor must consider the limitations of the academic environment in the definition of the theme, scope and quality expected for the project. Thus, it is necessary to select which competencies will be prioritized over others during the project.

In this phase, the professor can perform many stages of the coaching process. For example: making inquiries to assist students in choosing methods and techniques; clarify goals of the practical project; assist in the definition of the roles and responsibilities of the team as well as in the definition of the tasks and schedule; request and give feedback, among others.

3.6. *Reflection Phase*

At the end of the Contextualization, the students present the results of the practical project to the professor. After this presentation, they should reflect on the learning experience and engage in discussion groups on the lessons learned.

The students should basically answer 4 questions, based on Scrum's Sprint Retrospective (Kniberg, 2008):

- What methods and techniques applied helped in the project development?
- What methods and techniques not applied by the team could have helped?
- What were the main difficulties of the team?
- What the team can change to the next iteration?

These reflections should be stored by the professor in a database to serve as a repository of experiences that may assist future students who will study the same knowledge unit. If the professor chooses to administer more than one unit, the iterative cycle in the Fig. 1 must begin again from the Initiation phase.

4. Controlled Experiment

Initially, the professors defined the knowledge units to be addressed in the experiment and selected the technical competencies to be developed by the students. At this stage, they selected the instructional material, such as articles, videos, dynamics and games, as well as planned the practical project.

In Course I-A of the experiment, the professor selected as the practical project the development of a Web Portal for the City of Oeiras at Pará state, Brazil. Thus, in the first class (Initiation phase), he presented the problem: "How to manage a software project so that it can succeed?". In Course I-B, the other professor selected as the goal of the practical project the development of a Scientific Event Management System.

In this class, the students discussed with the professors about ideas on how to solve these problems. Thus, the professors exercised the coach role through suggestion and explanation of specific techniques, for example, interviews and brainstorm. Subsequently, in the second class (Preparation phase), the professors indicated videos about the main concepts of Project Management and the reading of technical articles with case studies on the application of Scrum in software companies.

In the third class (Discussion phase), the guest professionals (a project manager) gave a workshop on the agile framework Scrum, answering questions and discussing with the students. In the fourth class (Practice phase), the professors performed the dynamic called "Airplane Factory", which consists of making paper airplanes following an agile development cycle. It allows students to experience a development process based on Scrum.

Finally, in the fifth class (Contextualization phase), the practical project was started. The students interviewed the Product Owners. They had 3 weeks (between the fifth and tenth classes) to carry out this phase. Thus, by applying the Scrum, they estimated effort and time through the technique Planning Poker. In the last two classes, eleventh and twelfth classes, the students delivered the Product Vision, Product Backlog, Communication Plan and Project Schedule to the clients.

In Course II-A of the experiment, the professor selected as the focus of the course the requirements elicitation of a mobile application for Pre-university Test. In Course II-B,

the professor defined as the target the requirements elicitation of a mobile application for support the attendant of auto mechanics. Thus, in the lesson related to the Initiation phase, the following problems were presented:

1. How to identify project stakeholders?
2. How to collect customer needs?
3. How to record project requirements?

The students discussed their solution ideas with the professors, who acted as coaches suggesting specific techniques such as storyboards and prototyping. In the second class (Preparation phase), the professors indicated videos about Requirements Elicitation and the reading of technical articles on the difference between Functional and Non-Functional Requirements.

In the third class (Discussion phase), a professional (requirements analyst) gave a workshop on How to Elicit Requirements in Industry, generating discussions in the class. In the Practice phase, during the fourth class, the professors performed the dynamics "Drawing Diagrams", which consists of drawing a diagram structure (similar to a web page) from the description of a student-client. The other students try, through different techniques, such as interview and prototyping, to represent the diagram described by the client.

In the fifth class (Contextualization phase), the students interviewed the client who requested the mobile app. In total, the students had 4 classes (2 weeks) to carry out the practical project. In this sense, they were able to apply requirements elicitation techniques such as use case modelling, class diagrams and entity-relationship model. Additionally, based on the suggestion of the guest professional, they elaborated storyboards and screen prototypes, which aimed to help the client to better understand the scenarios and functionalities necessary for the application. Finally, in the ninth and tenth class, the students delivered the client a Requirements List document.

4.1. *Course I – Project Management Unit*

The first knowledge unit of the experiment was the Software Project Management, taught in two courses of "Software Engineering" (first semester 2018 – Course I-A and second semester 2019 – Course I-B), in the Informatics Laboratory of UFPA Cametá Campus, with classes from 2:00 p.m. to 6:00 p.m. Firstly, the students responded the interview based on SWECOM competency assessment about the Project Management unit based on their prior knowledge. Table 1 present an interview spreadsheet answered by a student at the beginning of the course (before applying the teaching approaches).

In the example of Table 1, of the 11 skills (points) that the student could have, the professor identified 5 points, being 3 Have (1 point each) and 4 Need (0.5 point each). Thus, this student had 45% of the technical competencies required for the Project Management knowledge unit.

So, the class was divided into three groups according to the results obtained in the data collecting pre-course. Thus, a ranking was generated, where the 1st place was allocated to Group A, the 2nd for Group B and the 3rd for Group C and so on, aiming at

Table 1
Example of Interview Spreadsheet (Pre-Course I-B)

SKILL	HAVE	NEED	GAP
Discuss common behaviors that contribute to the effective functioning of a team.		X	
Create and follow an agenda for a team meeting.	X		
Identify and justify necessary roles in a software development team.	X		
Understand the sources, hazards, and potential benefits of team conflict.		X	
Use an ad hoc method to estimate software development effort (e.g., time) and compare to actual effort required.		X	
Using a particular software process, describe the aspects of a project that need to be plan-ned and monitored.	X		
Track the progress of some stage in a project using appropriate project metrics.		X	
Compare simple software size and cost estimation techniques.			X
Identify risks and describe approaches to managing risk (avoidance, acceptance, trans-ference, mitigation), and characterize the strengths and shortcomings of each.			X
Explain how risk affects decisions in the software development process.			X
Identify security risks for a software system.			X

balancing the teams. Group A followed the traditional approach (control group), taking classes about the Project Management unit from 2:00 p.m. to 4:00 p.m. This traditional approach consisted of the classes that the professor already ministered without the mod-el. Group B followed the student-focused approach (treatment 1) through the teaching model. However, this group did not adopt industry practices. Group C followed the model plus industry practices (treatment 2), as coaching and mentoring. This difference between these two groups (B and C) was established to analyze the impact of industry practices on the competencies development.

Table 2
Example of Interview Spreadsheet (Post-Course I-B)

SKILL	HAVE	NEED	GAP
Discuss common behaviors that contribute to the effective functioning of a team.	X		
Create and follow an agenda for a team meeting.	X		
Identify and justify necessary roles in a software development team.	X		
Understand the sources, hazards, and potential benefits of team conflict.	X		
Use an ad hoc method to estimate software development effort (e.g., time) and compare to actual effort required.	X		
Using a particular software process, describe the aspects of a project that need to be planned and monitored.	X		
Track the progress of some stage in a project using appropriate project metrics.	X		
Compare simple software size and cost estimation techniques.			X
Identify risks and describe approaches to managing risk (avoidance, acceptance, transference, mitigation), and characterize the strengths and shortcomings of each.		X	
Explain how risk affects decisions in the software development process.		X	
Identify security risks for a software system.			X

Both groups B and C had classes from 4:00 p.m. to 6:00 p.m. Thus, the time of each experiment section did not exceed 2 hours per day. The students of Group C were assisted by veteran students who performed mentoring and by the professor who performed coaching. At the end of the teaching of the topics of this knowledge unit, the students answered the interview questions about the competencies acquired. Table 2 present an example of an interview spreadsheet of the Project Management unit developed by the same student who responded the instrument showed in Table 1.

In Table 2, the professor/interviewer identified 8 points, from 7 responses checked as Have and 2 as Need. In the end of the course, this student had 73% of the technical competencies required for the Project Management unit, showing a growth of 28% compared to the beginning of the course.

4.2. *Course II – Requirements Engineering Unit*

A second knowledge unit, which followed the protocol reported in Section 4.1, was taught in two courses of "Analysis and Design of Systems I" (first semester 2018 – Course II-A and second semester 2019 – Course II-B), in the Informatics Laboratory of UFPA Cametá Campus, with classes from 08:00 a.m. to 12:00 a.m. Group A (control) took classes on the Requirements Engineering unit from 08:00 a.m. to 10:00 a.m. Group B (treatment 1) and Group C (treatment 2) took classes from 10h00 to 12h00.

This second knowledge unit was carried out aiming to reinforce the results obtained in the first unit and to increase the population of the study. Thus, summing up the 2 units, applied in 4 courses, in the evaluation of the model participated 4 professors with master's degree and 66 undergraduate students. This population sample represents 94.27% of the sample interviewed in the survey (70 participants) conducted during the PhD research (Portela *et al.*, 2017). The survey questions were related to the learning of SE knowledge units (ACM/IEEE, 2013). The results obtained reinforce the claim that students are not developing SE technical competencies in an adequate way.

5. Results and Discussion

5.1. *Quantitative Analysis*

The pre and post-experiment data were analyzed using statistical analysis. Thus, firstly it was used the T Student Test, which is recommended when the data test follows a normal distribution, but the population variance is unknown. The T-Test analyzes the statistic, distribution and degrees of freedom to determine a P value (probability) that can be used to determine whether the population means differ.

In Course I, as shown Table 3, the P value for the difference of the pre and post-experiment results in all groups was less than 0.05. Thus, it can be said that there was development of competencies in Project Management unit, regardless of the approach (Control, Treatment 1 and Treatment 2).

Table 3
Results Analysis of Pre and Post-Course (T-Test)

Course	Group A (Control)	Group B (Treatment 1)	Group C (Treatment 2)
Course I-A	P = 0.01384	P = 0.00581	P = 0.00567
Course I-B	P = 0.00084	P = 0.00001	P = 0.00085
Course II-A	P = 0.07775	P = 0.27217	P = 0.00601
Course II-B	P = 0.03085	P = 0.00585	P = 0.00697

In Course II, as shown Table 2, the P value was less than 0.05, except in the Group B of the Course II-A. Thus, it can be affirmed that there was a significant difference in competencies development related to Requirements Engineering unit in the three approaches.

In order to verify if there was a significant difference in the development of SE technical competencies between the teaching approaches, it was used a second statistical test, the Variance Analysis (ANOVA) (Easterbrook *et al.*, 2007). ANOVA determines which of the individual level comparisons are significant. So, this type of test is useful in comparing three or more means (groups or variables) for statistical significance.

Its value is in looking at variation between and within the means. So, it not only compares the means (like the T-Test) but examines the variation in the means calculation. So, the ANOVA test is robust to the assumption of normality.

When ANOVA results lead to rejection of the null hypothesis (students-focused approaches $\leq$ traditional approach), there is evidence that the averages between levels differ significantly. This result represents that all averages are equal or that the treatment is less than the control.

From the ANOVA analysis in Table 4, it was observed that there was a significant difference between the 3 groups only in Course I-A, being P = 0.00153. Additionally, for the Course I-A, the difference between the groups was analyzed using ANOVA.

It was observed a significant difference (P = 0.00153) between Treatment 1 (student-focused approaches) and Control (traditional approach). This difference was also observed between Treatment 2 (student-focused with industry practices) and Control, where the value of P = 0.00606.

For the other 3 courses (I-B, I-A and II-C), no statistically significant differences were identified between the results obtained in the experiment as shown in Table 4.

Table 4
Results Analysis Between Approaches (ANOVA)

Course	Knowledge Unit	No. Students	ANOVA
Course I-A	Project Management	12	P = 0.00153
Course I-B	Project Management	19	P = 0.76018
Course II-A	Requirements Engineering	18	P = 0.63130
Course II-B	Requirements Engineering	17	P = 0.66789

From the results of these four courses, it can be inferred that no have significant statistical difference between student-focused teaching and traditional teaching approaches in the development of SE technical competencies.

5.2. *Qualitative Analysis*

According to the professors who adopted the model, the SE area is "very theoretical" and the model allows the students to "experience a practice of subjects", which makes classes "more interesting and motivating". In addition, they pointed out that the possibility of differentiated classes with videos, articles and workshops allows a "greater interaction among students" and makes the discussion and presentation of doubts "more pleasant and interesting".

Although the four professors agreed that the model is easy to use, they pointed out that the documentation might be a little more concise to be more objective. In this sense, a revision was made in the text, together with the professors participating in the experiment, in order to synthesize it.

Finally, the professors emphasized that the users of the model need to be willing to innovate in didactics to identify and select articles, videos, guest professionals and dynamics. In addition, they need to know and be aware of the coaching and mentoring practices to make correct use of the model. As for the willingness to innovate in didactics, the model defines a specific profile as the model's target audience, that is, professors who act as facilitators of the teaching-learning process. Related to the selection of lessons material, the model's documentation provides a base of articles, video and dynamics that can be instantiated in the classroom. The coaching and mentoring practices were adapted to the academic context and their main steps are described in the model documentation.

In the Course I (Project Management unit), the students emphasized that the practical project allowed them applying the agile framework Scrum for project management and planning the effort and time estimation through Planning Poker technique. In addition, students reported that they have been able to understand and elaborate the Product Vision, Product Backlog, Communication Plan, and Project Schedule documents.

The students participating in the Course II (Requirements Engineering unit) emphasized that practical activities, such as interview dynamics and the creation of screen prototypes, allowed a better understanding of the concepts covered in the course.

It was observed from the realization of the controlled experiment that the lack of motivation and commitment of the students had a direct impact on the results of the instantiation of the model. According to reports from students of Course I, the main difficulties were "managing the time available for the development of activities" and "non-compliance with deadlines". These problems with time and deadlines management are recurrent in SE projects (Pressman and Maxim, 2014).

However, in Course II of the experiment, the main difficulty related by 67% of the students was the "lack of commitment from part of the team", such as "absence in work

meetings" and "no commitment to the tasks to be performed". These lack of commitment with the experiment activities is directly related with the low motivation of these students to learn SE.

In addition to knowledge level, students' performance is positively correlated with their effort, attitude and motivation to learn (Martínez-Rodríguez *et al*., 2019). The impact of these factors on the experiment's results are. reported in the Subsection VI-A.

6. Concluding Remarks

6.1. *Conclusions*

This article presented a controlled experiment that sought to answer the following question of causality-comparison: "Does the student-focused teaching approaches allow the development of more SE technical competencies than the traditional approaches?". In the Course I-A of the experiment, where the model was instantiated for the Project Management knowledge unit, the student-focused approach was more efficient, according with ANOVA. However, these results were not repeated in the other courses of the experiment, with the P value of ANOVA greater than 0.05.

According with the analysis of a participant professor, the students' lack of motivation and commitment directly impacted the results of the instantiation of the model. At the beginning of the experiment, the students of all groups answered the following question: "How useful do you consider the Software Engineering area for your professional preparation?" The answer options were:

- Completely useless.
- Almost never useful.
- Occasionally useful.
- Moderately useful.
- Very useful.
- Essential.

In the courses I-B, II-A and II-B, some students in Treatment group responded that they considered the SE area "moderately useful", while others responded "completely useless".

Most students in the control group responded that the SE area was "very useful" or "essential" for their training. Additionally, the main difficulty reported by most students was the "lack of commitment from part of the team".

According to Martínez-Rodríguez *et al*. (2019), the students' level of satisfaction with the area of study is directly related to their motivation. To ensure satisfactory academic achievement, the level of academic motivation and interaction between students needs to be increased. In the proposed model, there is no practice directly related to these two factors.

Souza *et al*. (2018) highlights the use of Gamification that refers to the use of game elements (i.e. Points, Leaderboards and Badges) to engage and motivate students in

performing learning activities. Gamification can be applied to motivate students in conforming to desired behaviours, such as acquiring the habit of applying specific techniques, or being more participative in the classroom. In the Software Engineering, is used as a strategy to induce learners to use specific SE abilities or practices, by promoting competition or systematically rewarding learners as they perform expected actions or show expected behaviours.

If applied to the proposed teaching model, these practices can, for example, address "absence in work meetings" giving individual points by the attendance at out-of-class meetings and address "no commitment to the tasks to be performed" through badges for each task completed and a leaderboard by student.

Although the results of the experiment do not allow to validate the alternative hypothesis H1 (Section 2.1), the relevance of the student-focused teaching approaches to the SE education area is highlighted as it proposes to collaborate with the development of technical competencies at application level during the undergraduation. Consequently, the relevance of this work can be extended to the industry, since it seeks to meet the demand for professional training in the SE area.

From the experience of using the proposed model in the controlled experiment, it was observed that the model allows teaching a knowledge unit in a short iterative cycle (on average, 9 lessons of 50 minutes). This may have a direct impact on undergraduate courses in the area, mainly Computer Science, Information Systems and Computer Engineering, which, unlike the Bachelor in Software Engineering, generally have 3 courses that deal with SE. In this way, the professor will be able to manage the available workload in a more adequate way and to manage the topics of the knowledge units in an integrated way.

It is expected that the use of the industry practices used in this research can help to reduce the industry's dissatisfaction with the level of training of the newly graduated professionals who intend to work in the SE area. Their integration in the model allows students to perform practical activities that explore sensory stimuli (visual, auricular, textual and kinesthetic) in an iterative way through a learning cycle. Thus, students with different learning profiles can know, understand and apply the SE contents and develop the technical competencies of the knowledge units.

6.2. *Limitations*

The fact that the Course I-B, Course II-A and Course II-B do not point out that a student-focused approach can be more effective than a traditional approach does not mean that the null hypothesis is true, only that there is insufficient evidence to support it.

In this evaluation of the use of the model, the efficiency of the treatments (student-focused approaches and industry training practices) was observed only in Course I-A of the Project Management unit. However, some observations on its application in the other courses are required.

According the design of the experiment, all groups carried out practical projects. Thus, all teams had the opportunity to participate in practical activities in order to reduce

the problems of comparison. In addition, the professors' prior experience in student-focused approaches and the students' motivation to study SE are relevant outside the environment of a controlled experiment. This reduction of the variables involved is a characteristic of the positivist posture, adopted by the researchers of this work and associated to the controlled experiment method (Easterbrook *et al*., 2007).

Another limitation was the low sample of the participating population of the experiment: 4 professors and 66 students. Each group (A, B and C) was composed between 5 and 7 students. This small sample decreases the statistical power of the experiment. For this reason, the experiment needs to be replicated and/or supported by the execution of other research methods, such as case study and action research.

Additionally, the authors emphasize their positivist positioning, because the characterization of a large population through sampling techniques requires a belief in the reductionism (Creswell, 2002). However, future works will be carried out to reinforce the conclusions obtained because there is not enough evidence from this experiment to justify the changing of a traditional teaching approach to a student-focused approach.

6.3. *Future Works*

As a future work, in order to deal with the variable "student motivation", which directly impacted the results of the controlled experiment, the model should be updated to introduce gamification strategies in all its phases. Gamification has as its main benefits, recognized by researchers in the SE area, the engagement and motivation of students in teaching-learning activities (Souza *et al*., 2018).

Subsequently, it should be carried out a case study to observe how the use of the new version of the model by the professors occurs and if such use has direct relation with the development of SE technical competencies in the students. A case study was chosen because, according Easterbrook *et al*. (2007), it provides in-depth understanding on how and why certain phenomena occur, and may reveal the mechanisms by which cause-effect relationships occur (i.e. the use of teaching approaches and the development of technical competencies).

References

ACM/IEEE (2013). *Computer Science Curricula 2013 – Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. https://dl.acm.org/citation.cfm?id=2534860

Campbell, V., Johnstone, M. (2010). The Significance of Learning Style with Respect to Achievement in First Year Programming Students. In: *Proc. of the 21st Australian Software Engineering Conference*, pp. 165–170.

Creswell, J. (2002). *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*. Sage Publications.

Dagnino, A. (2014). Increasing the effectiveness of teaching software engineering: A University and industry partnership. In: *Proc. of the 27th Conference on Software Engineering Education and Training (CSEE&T)*, pp. 49–54.

Easterbrook, S., Singer, J., Storey, M-A., Damian, D. (2007). *Selecting Empirical Methods for Software Engineering Research*. Guide to Advanced Empirical Software Engineering, Springer.

Fleming, N., Mills, C. (1992). *Not Another Inventory, Rather a Catalyst for Reflection*. To Improve the Academy.

Fox, A. (2015). Teaching Practical Software Engineering in the 21st Century. *Invited Talk at Brazilian Conference on Software: Theory and Practice*, Belo Horizonte.

Gary, K., Lindquist, T., Bansal, S., Ghazarian, A. (2013). A Project Spine for Software Engineering Curricular Design. In: *Proc. of the 26th Conference on Software Engineering Education and Training (CSEE&T)*, pp. 299–303.

Gomes, A., Barcaui, A., Scofano, A., Gomes, D. (2015). *Coaching e Mentoring*. Publisher FGV, Rio de Janeiro.

IEEE Computer Society (2014a). *SWEBOK – Guide to the Software Engineering Body of Knowledge*. `https://www.computer.org/education/bodies-of-knowledge/software-engineering`

IEEE Computer Society (2014b). *SWECOM – Software Engineering Competency Model*. `https://www.computer.org/volunteering/boards-and-committees/professional-educational-activities/software-engineering-competency-model`

Jedlitschka, A., Ciolkowski, M., Pfahl, D. (2007). *Reporting Experiments in Software Engineering*. Guide to Advanced Empirical Software Engineering, Springer.

Kniberg, H. (2008). *Scrum and XP from the Trenches*. `http://infoq.com/br/minibooks/scrum-xp-from-the-trenches`

Kolb, D. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall.

Kropp, M., Meier, A. (2014). New sustainable teaching approaches in software engineering education. In: *Proc. of IEEE Global Engineering Education Conference (EDUCON)*, pp. 1019–1022.

Malik, B., Zafar, S. (2012). A Systematic Mapping Study on Software Engineering Education. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 6(11), 3343–3353.

Marques, M., Quispe, A., Ochoa, S. (2014). A Systematic Mapping Study on Practical Approaches to Teaching Software Engineering. In: *Proc. of Frontiers in Education Conference*, pp. 1–8.

Martínez-Rodríguez, R., Alvarez-Xochihua, O., Victoria, O., Arámburo, A., Fraga, J. Á. (2019). Use of Machine Learning to Measure the Influence of Behavioral and Personality Factors on Academic Performance of Higher Education Students. *IEEE Latin America Transactions*, v. 17, n. 4, pp. 633–641.

Mirian-Hosseinabadi, S., Aghakasiri, Z., Sadeghi, A., Delfani, P., Ghandehari, M. (2010). Emphasizing experiences in teaching software engineering courses. In: *Proc. of the 2nd International Conference on Education Technology and Computer (ICETC)*, pp. 149–153.

Ng, P., Huang, S. (2013). Essence: A framework to help bridge the gap between software engineering education and industry needs. In: *Proc. of the 26th Conference on Software Engineering Education and Training (CSEE&T)*, pp. 304–308.

Portela, C., Vasconcelos, A., Oliveira, S. Souza, M. (2017). The Use of Industry Training Strategies in a Software Engineering Course: An Experience Report. In: *Proc. of the 30th Conference on Software Engineering Education and Training (CSEE&T)*, pp. 29–36.

Pressman, R., Maxim, B. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

Richardson, D. (2018). ACM retention committee: Student-focused initiatives for retaining students in computing programs. *ACM Inroads* 9(2), 13–18.

SEI (2010). *Capability Maturity Model Integration for Development – V 1.3*. `http://www.sei.cmu.edu/reports/10tr033.pdf`

Souza, M., Veado, L., Moreira, R., Figueiredo, E., Costa, H. (2018). A systematic mapping study on game-related methods for software engineering education. *Information and Software Technology*, 95, 201–218.

Subrahmanyam, G. (2009). A Dynamic Framework for Software Engineering Education Curriculum to Reduce the Gap between the Software Organizations and Software Educational Institutions. In: *Proc. of the 22th Conference on Software Engineering Education and Training (CSEE&T)*, pp. 248–254.

C. Portela, PhD in Computer Science with emphasis in Software Engineering by the Informatics Center, Federal University of Pernambuco. Currently he is director and professor at the Faculty of Information Systems, Federal University of Pará, Campus Cametá. His research areas are: Informatics in Education, Software Engineering and Human-Computer Interaction.

A. Vasconcelos, PhD in Computer Science at the University of York, GB (1993); Post-doctoral Fellow in Software Engineering at Universidad Politécnica de Valencia, Spain (2011); Associate Professor at the Informatics Center, Federal University of Pernambuco, since 1995. Director of Information Technology Center from march 2013 to october 2015. He is currently an instructor, implementer and assessor of MPS.BR and has experience in Software Engineering, acting on the following research topics: Software Quality, Methodologies and Software Processes, Environments and CASE Tools, Software Testing and Requirements Engineering.

S. Oliveira, PhD in Computer Science with emphasis in Software Engineering and did his postdoctoral internship at the Informatics Center, Federal University of Pernambuco. Currently he is professor and researcher at the Faculty of Computing and Graduate Program in Computer Science, Federal University of Pará. He is the Lead Coordinator of the SPIDER research project, which has won many scientific awards and has already graduated many doctoral, master, graduate and scientific initiation students in Computer Science. He is consultant, appraiser and instructor of the MPS.BR and CMMI software and service quality models. His research areas are: Informatics in Education, Software Engineering and Software Process Improvement.

M. Souza, PhD degree in Computer Science with emphasis in Software Engineering, from the Federal University of Minas Gerais in 2019, and also holds MSc degree in Computer Science from the Federal University of Pará (UFPA). He is a professor at the Federal University of Lavras since 2019. His research interests include Software Engineering Education, Software Quality and Software Process Improvement.