A Quasi-Experimental Evaluation of Teaching Software Testing in Software Quality Assurance Subject during a Post-Graduate Computer Science Course

https://doi.org/10.3991/ijet.v17i05.25673

Isaac Souza Elgrably(✉), Sandro Ronaldo Bezerra Oliveira
Graduate Program in Computer Science (PPGCC), Federal University of Pará (UFPA), Pará, Brazil
isaacelgrably@gmail.com

Abstract—Software testing is regarded as a key activity in the software development cycle, as it helps information technology professionals to design good quality software. Thus, this is an essential activity for the software industry, although with all its nuances high priority is still not being given to learning about it at an academic level. The purpose of this work is to investigate a teaching strategy for software testing which involves acquiring academic skills within a curriculum based on active teaching methodologies. A teaching model was designed for this to coordinate the different areas of a subject, and then a controlled quasi-experiment was carried out in a post-graduate course to evaluate the application of this model. The results obtained demonstrate that there was a considerable learning gain in the experimental group that adopted the teaching approach, when compared with the control group that relied on a traditional approach. The student t test was employed to determine the learning efficiency.

Keywords—software testing, software engineering, software engineering education, active teaching methodologies

1 Introduction

Software testing plays a critical role in both building software quality and determining whether or not the desired quality has been achieved. An improvement in quality is essential and a focus on best practices and emerging technologies can help to enhance performance [1].

Recent literature has shown that there is a need to teach topics related to software testing in institutions in a more active and practical way. In addition to technical knowledge, social skills should also be taught to students [2], [3]. Vam Damn [4] states that, as well as having technical skills, software testers, must also have social skills and Veenendaal [5] stresses that testers must have personal skills so that they can influence people and communicate in a way that makes them feel they are an essential, part of a software project.

For this reason, in this study, a teaching approach was adopted for software testing that was based on active teaching methodologies and aimed at making improvements.

This is an ongoing study that is being carried out by the authors, and is underpinned by a number of factors that emerged from previous studies. These include the following: (i) a systematic mapping that helped to locate topics and academic content related to software tests [6], (ii) an academic curriculum (syllabus) was drawn up that was designed for teaching Software Testing [7], and also served as a knowledge management instrument that could be assessed by the control and experimental groups; (iii) a teaching plan based on an active methodology [8] together with strategies adopted to optimize the students' learning experience. These were employed as a means of teaching the topics in the experimental group, which had to be coordinated by the teaching model designed for this work, (iv) there was a diagnosis of some aspects of software testing/ teaching in the main Brazilian universities [9], and, finally, (v) after the experimental group was formed, there were several findings and good practices for remote teaching based on active methods [10]. When taken together, these factors served as the basis for conducting the quasi-experiment which is outlined in this article.

The main Research Question (RQ) of this work is designed to address the challenge of assessing the academic skills required for the education of students. It can be stated as follows: If professors in Software Engineering courses employ a set of active teaching methodologies and collaborative teaching practices instead of adopting traditional approaches, will students be able to acquire more academic competences related to Software Testing?

This question will be broken down into 4 other research questions, (shown in Section 5.2), to analyze each teaching unit adopted in the syllabus on an individual basis. A quasi-experiment was conducted with postgraduate students enrolled in a graduate computer science program to learn software quality with a view to comparing the learning effectiveness of the newly designed teaching approach with that of the results obtained from a more traditional pedagogical approach, (similar to [11] but without the use of pre-tests). The results of the quasi-experiment showed that there was a considerable difference between the degrees of learning effectiveness, since the new teaching approach obtained better results than those obtained from traditional teaching.

The objective of this work is to show the learning gain of students when a teaching approach is adopted that is based on the use of active methodologies, while taking into account that the experimental and control classes used the same syllabus [7] and both focused on teaching content related to software testing.

2 Teaching tests for computer students

The Brazilian Computer Society (SBC) states in its curriculum guidelines for training that academic subjects in Brazilian institutions should give priority to the learning of skills aimed at personal development rather than the assimilation of traditional content, and ensure that these skills are fully acquired [12].

Computing Curricula 2020 (CC2020) is an initiative that was launched jointly by several professionals in computing, to compile and summarize the current curricular

guidelines for academic programs that run Bachelor Degree courses in Computing, as well as planning future curricular guidelines [13].

The range of academic competences taken from the national curricular guidelines of the Ministry of Education of Brazil - MEC [14] was selected for this work, with t the asset mapping compiled in [6] being used as a benchmark. The selected competences are outlined below:

1. Specify, design, implement, maintain and evaluate computer systems, by employing appropriate theories, practices and tools,
2. Employ methodologies based on criteria that can ensure and sustain data quality throughout all the developmental stages of a computational solution,
3. Plan, specify, design, implement, test, check and validate computer systems,
4. Understand and apply processes, techniques and procedures for the software construction, evolution and evaluation,
5. Evaluate the quality of Software Systems, and
6. Design, apply and validate principles, standards and best practices in Software development.

2.1 Objectives, syllabus and teaching plan

The ACM / IEEE – Association for Computing Machinery / Institute of Electrical and Electronics Engineers guide [15] states that there is no single formula for devising a perfect syllabus for subjects in Computing, although there are some specific recommendations and strategic suggestions in the report that are useful for a wide range of institutions.

Academic curricula in computing must be constantly updated, so they can keep abreast with the constantly evolving needs of the software industry. The methods of learning and academic content must also be updated so that students can acquire market-oriented job skills [9], [16].

Once the required skills had been determined, it was possible to form a syllabus that could include all the necessary subjects [9]. It was suggested that knowledge could be disseminated in a progressive way more effectively, if the content referring to software tests was divided into 4 teaching units, as can be seen in Table 1.

Table 1. Syllabus units

Teaching Units	Objective	Related Competences
Software Engineering	To provide a basic theoretical knowledge of testing and quality and conceptualize a number of testing terms, models and practices.	(1), (4)
Software Construction	This teaching topic is designed to formulate concepts for identifying common errors, establishing good coding practices, defining peer code review principles, and refactoring.	(2), (3), (6)
Quality	This teaching topic is designed to formulate concepts for building testable projects, analyzing evolutionary and agile requirements, and preparing test cases.	(1), (2), (4), (6)
Software	These will consolidate the knowledge acquired in the previous units; when	(1), (2), (3),

Teaching Units	Objective	Related Competences
Tests	put into practice, this teaching unit must be used with the maximum number of practical teaching strategies.	(4), (5), (6)

A teaching plan was drawn up so that the syllabus could be put into effect, and the academic content linked to the teaching methodologies and levels of learning. Each teaching methodology has its own learning objectives, based on Bloom's revised taxonomy [17].

Table 2 outlines the teaching topics and the methodologies used, as well as the predicted results and the expected learning levels for each topic. The next section explains how these parameters were derived from the methodology employed.

Table 2. Teaching topics, expected results and learning levels in units

Teaching Units	Topics	Expected Results	Learning Levels
1 Software Engineering	1.1 Introduction to Testing and Quality	The students must know the basic concepts of software engineering when related to testing and quality control.	Remember / Factual
		The students should be able to correlate the relationship between test content and computational problems.	Understand / Procedural
	1.2 Introduction to software creation and its developmental methods	The students must understand the software creation process.	Understand/ Procedural
		The students should be able to analyze computational problems and make decisions which can assist in solving them.	Analyze / Conceptual
		The students should be able to evaluate solutions and make decisions based on their knowledge.	Evaluate / Factual
		The students must be able to plan, build and correct computational solutions.	Create / Procedural
2 Software Construction	2.1 Software Creation Concepts	The student must know, the concepts of software construction and be able to differentiate between them.	Remember / Factual and Conceptual
		The students must know the basic concepts and work products related to software creation and how they apply to testing.	Understand / Factual
		Students should be able to evaluate the different concepts that they have learned and their relationships, dependencies and complementary features.	Evaluate / Conceptual
	2.2 Definition and implementation of defensive programming techniques and software maintainability	The students must recognize situations that are suitable for the application of good coding practices.	Remember / Conceptual
		The students must be able to know and handle strategies, as well as understanding best practices for software evolution and maintenance.	Analyze / Factual and Procedural
		The students must be able to make adjustments and improvements to achieve computational solutions.	Create / Metacognitive
3 Quality	3.1 Software quality concepts	Students should know the basic concepts of software quality and how to relate them to what was learned previously.	Remember / Factual

Teaching Units	Topics	Expected Results	Learning Levels
	3.2 Software quality assurance strategies aligned with tests	Students should understand the use of design patterns in software construction.	Remember / Conceptual
		Students must understand the relationship between requirements and tests so that they can create testable requirements.	Understand / Conceptual
		Students must be able to find solutions by taking into account the concepts and practices of testing, construction and quality assurance.	Analyze / Factual
		Students should be able to apply the knowledge they have acquired in the subject and find practical solutions to computational problems.	Apply / Procedural and Metacognitive
4 Software Test	4.1 Initial concepts for adopting test approaches	Students should know the different practices and ways of creating work products in testing.	Remember / Factual
		Students must understand how work products in testing can progressively support a software project.	Understand / Procedural
		Students should be able to understand the relationship between test concepts and their role in software evaluation and problem discovery.	Understand / Factual
		Students must know how to apply learning techniques to solve computational problems.	Apply / Conceptual
		Students must know how to build work products in testing in a structured way.	Create / Procedural
	4.2 Construction of work products for validation and the verification of software tests	Students should be able to recognize and solve problems about non-functional work products.	Remember / Factual
		Students must interpret and verify work products of testing in situations requiring a connection.	Understand / Conceptual
		Students must remember the parameters for testing practices and testable requirements.	Remember / Factual
		Students must evaluate the solutions and uses of learning concepts.	Evaluate / Conceptual
	4.3 Test-oriented development project	Students should be able to understand decision-making factors based on testing and quality parameters, to assist them in their development.	Understand / Factual
		Students must know how to handle strategies and apply best practices for the evolution of code maintenance and work products in testing.	Apply / Procedural
		Students should be able to evaluate the quality, operational features, and issues of coding and work products in testing.	Evaluate / Conceptual
		Students should find solutions based on good coding practices and aligned with tests.	Create / Metacognitive
	4.4 Test-oriented project evaluation	Students must acquire criticality about software products to enable them to evaluate the importance of testing techniques and practices.	Evaluate / Conceptual
		Students must be able to define strategies and offer solutions to computational problems.	Apply / Conceptual and Metacognitive
		Students should be able to evaluate work products and correlate them with solutions to create a derived part of work products.	Evaluate / Conceptual

On the basis of the academic content obtained from the mapping work carried out in [6], the syllabus defined the 4 teaching units shown in Table 2, where each of these teaching units has the items that can be seen. Each of the teaching units is designed to produce different kinds of knowledge in a progressive way. This means a part of the content that appears in one unit can be revisited and displayed in a different way in another teaching unit with different expected results and levels of learning, (as explained in Computing Curriculum 2020 (CC2020) [13]. This is intended to consolidate the student's knowledge of the academic content in a conceptual and practical way.

The different learning levels selected for each teaching unit were chosen by the researchers, on the basis of a literary analysis which can be found in many computer knowledge guides, such as: ACM/IEEE [18] and SBC curricula [19, 5]. Thus, the Syllabus was compiled and peer-reviewed with researchers who were specialists in Software Engineering. More details about this stage can be found in Syllabus [7].

The expected results of each teaching unit were obtained to achieve the teaching objectives outlined in the syllabus, and active teaching methodologies were employed, in conjunction with teaching tools, practices and agile techniques. This teaching plan is focused on the students, with the professor as a supportive figure and the knowledge produced should correspond to what is required in the labor market. One should try to break the traditional teaching paradigm which is based on an educational theory that knowledge should only be spread through traditional lecturing techniques.

The new teaching environment seeks to provide a better way of teaching a subject to t students, by encouraging them to engage in critical thinking and thus obtain the highest possible standard of learning. More details about the teaching plan can be found in [10].

2.2 Scope of the definition

Within the scope of this work, can be found the construction of knowledge based on academic competences. The SBC 2017 Curriculum Guide [12] recommends that syllabuses should be created on the basis of competency-based learning, as advocated by the MEC [14], but that researchers and professors should be allowed to define the competencies and teaching strategies that will be used.

The ongoing plan for the Computing Curricula 2020 (CC2020) is rooted in learning theory, which strongly advocates the adoption of a competency-based approach when issuing computing curricula [20].

A syllabus specifically designed with content related to software tests was used within the scope of the definition outlined above [7]. This syllabus can be followed either by adopting a traditional classroom approach or using a teaching plan with active teaching methodologies, group work and practical strategies [8], with the aim of building core academic competences related to software testing for students.

3 The teaching approach

The Brazilian Ministry of Education [14] lays down that the teaching methodology for computer courses should be student-centered and supported by the instructor as a facilitator of the teaching-learning process. Moreover, the professor should act as a mediator by a) also showing the applications of the theoretical content, b) stimulating competition, c) encouraging teamwork, d) motivating students to study, and e) developing communication and negotiation skills.

The SBC 2017 Curriculum Guide [12] recommends that, whenever possible, active methodologies should be used in computer science courses so that the students can spend more time on activities in which they are the leading figures in the teach and learning process.

Active learning must be encouraged for software testing students so that there can be progressive learning and an effort should also be made to employ pedagogical models that can create scenarios that provide opportunities for active learning [21].

The planned model is designed to follow a set of timed phases based on the model by Portela et al. [22], in addition to having definite phases for the planning of the subject, (as in the model designed by Benitti [23].)

This model gives priority to the construction of knowledge based on academic competences and content that is derived from many sources, practical teaching activities and collaborative activities. In addition, it provides this content by employing several active teaching methodologies, which are starting- points that can lead to more advanced processes of reflection, cognitive integration, generalization, and the preparation of new practices [24]. Figure 1 shows the phases that make up the teaching model that is displayed and then discusses each phase.

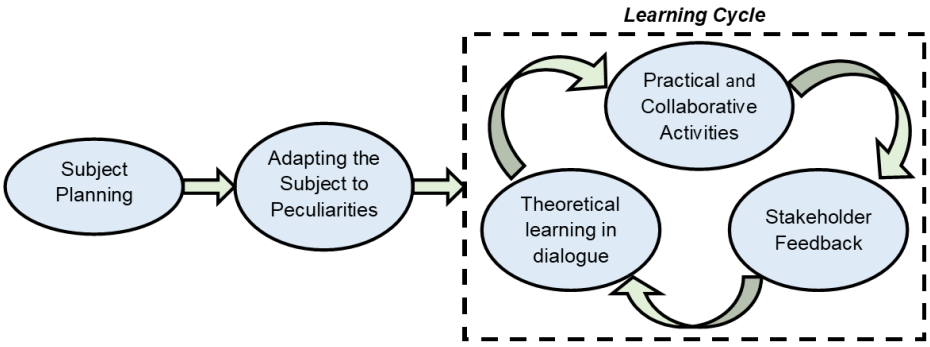

Fig. 1. Learning model of the methodology

In the following subsections. each phase of the model is described. As a means of contextualizing this quasi-experiment, the employment of the software testing technique will be used as an example.

3.1 Phase I: Subject planning

At first, the professor or person responsible for carrying out the subject must build it. Thus, the first stage necessary is to select the competences that must be acquired for teaching the students. For the purposes of this study, the competences of the MEC [14] were used, (as shown in the previous section).

This is followed by, an important task which is the selection of academic content for the subject, since the [25] study found that the community sees the need for better education and training in software testing. This is because they are professionals that come from universities with a limited knowledge of this area of teaching. In view of this, this model recommends that the content should originate from several different sources, such as:

1. Academic curricula,
2. Curriculum Guides,
3. ISO Models, Certification Guides and Standards,
4. Techniques and activities employed in industry.

The topics found in the SWEBOK curriculum guide [26] and related to software testing were used to create the teaching units for this study. Authors who want to use this model are advised to make a strategic alignment with their disciplines and refer to notes from published guides of knowledge, systematic reviews and systematic mappings.

The content topics used in this work were selected from different sources of knowledge, with the aim of seeking academic topics taken from the ACM/IEEE [18], SBC curricula [19, 12], and knowledge of quality improvement models related to software testing, such as TMMI [27] and TMMI Agile [28]. Authors who want to compile a list of content topics are advised to combine knowledge of the software industry with academic subjects.

There are knowledge guides at basic levels, such as the ISTQB Foundational Level Certification Guide [29] and ISO 29119-5 [30], which give useful advice on the content that will be taught in each topic and which can be consulted in [7], since software testing has a wide range of different approaches and views, depending on the authors. It is advisable to select some global and broad reference-points which have some application to the labor market; therefore, the agile testing practices of Laing and Graves [31] and Crispin and Gregory [32] have been used in this research because they match the content of agile methodologies that has been widely used in the labor market. Authors are ad-vised to take note of these tips when creating their academic content.

Bloom's revised taxonomy [17] was used to predict the expected results and the level of learning that could be attained in each topic, as is also recommended by the current Computing Curricula 2020 (CC2020) [13]. Finally, while compiling the syllabus, academic content and teaching topics, the authors are advised to provide illustrations for their national and international curriculum guides, and also to analyze systematic reviews, systematic mappings and computer-based diagnostic assessment.

This is because a lot of content has been created and updated, and a quick update of academic content is always essential to prepare students for the labor market.

3.2 Phase II: Adapting the subject to peculiar features

After the contents and expected results have been selected, they must be aligned with teaching methodologies and backup materials that can help achieve the desired learning level. For this reason, the different contexts that the professors want to incorporate in their subject must be taken into account.

The peculiar features and teaching conditions are determining factors for the creation of a subject. They range from teaching in a hybrid way, remote teaching or even stimulated learning that relies on some central methodology, such as PBL – Problem-Based Learning. In the model displayed, these factors are included after the creation of the disciplinary content, so that there are no constraints with regard to content as a result of these adjustments.

In the case of this study, the subject was adapted to learning via practical teaching, collaborative work and remote teaching. In a teaching context that involves active methodologies, it is expected that dialogues and an exchange of knowledge between students that is mediated by the professor, will take place in a general way. In light of this, even in a remote teaching situation, the authors point out that classes must remain synchronous, while also being recorded and made available from a teaching platform for students, in accordance with the regulations of each country.

In the case of practical activities, the authors advise that more time should be devoted to tutorials (and manuals on the tools and practices that will be used) so that students can absorb the knowledge more easily.

In light of the practical challenges facing collaborative work, similar activities should be carried out asynchronously for the students and examples drawn on that are based on everyday situations in the labor market. These and other situations arose during the experimental work carried out by this work, and further details can be found in [10].

3.3 Phase III: Theoretical learning in dialogues

This model advises that at the beginning of the classes of each teaching unit, a theoretical teaching approach should be adopted for disseminating the knowledge. It is useful if the professor in the area has some technical knowledge or if a professional can be invited to give a presentation about the academic content. If there is more than one professor or mediator for the subject, the model advises that at the end of each class. a meeting should be held to discuss what happened, as a means of improving the next classes.

The methodologies that were employed in the quasi-experiment for the theoretical learning cycle included expositional dialogue classes, discussion of practical cases and lists of exercises for the consolidation of learning, as well as backup materials such as books, academic articles / papers, certification guides and tutorials, and these tools are available in the class. Details of how they operate can be found in [10].

3.4 Phase IV: Practical and collaborative activities

This phase enables students to improve their learning and skills through practical activities, and offers the participants a chance to test the knowledge they acquired in the previous phase. The ACM / IEEE guide [15] recommends that students doing information technology courses should be practice-oriented and that in some subjects, the practical side of applying information technology is essential for the success of the graduate's future career.

This model advises that practical activities should be carried out in groups in a collaborative way so that there can be an exchange of knowledge and experience between students, and it is possible to replicate the situations that may occur in industry. These practices might arise from employing different methodologies, and include: practical projects [22], PBL [33], [34], serious games [35] or programming dojos [10]. It is advisable that a professor or other professional should supervise these practical activities so that they can coach and mentor the groups, sort out any technical problems, assist in the way the practices are carried out and act as a mediator if any possible conflicts arise in the groups. The professors who will use this model must have the necessary skills to teach practical knowledge, since when they are running a graduate program, this practical knowledge must be transferred to other professors as recommended in the guidelines [36]. As in Phase III, if there is more than one professor or mediator, there should be an ¨after-class¨ meeting to discuss what has taken place.

The practices carried out in this phase of the study were programming Dojo (with the Scratch tool, similar at [37]) and the Java language; the practical project, as indicated [38], divided into two parts. Further details will be provided later, in the Execution section.

3.5 Phase V: Stakeholder feedback

At the end of each iteration of a teaching unit from the model, students must present the results of their practical and collaborative activities to the professor. Everyone involved must provide feedback on the teaching unit, including an analysis of the following: the contents shown, the teaching methodologies used, any adaptations that must be made and the difficulties encountered. This feedback must be stored and made available by the professor, and may include a spreadsheet, cell phone application, gamification or classroom recording.

The model does not suggest which technique is required for sending feedback and, leaves this to the professors to decide. If there is still more than one teaching unit, the model cycle must be repeated from Phase III onwards.

Two different forms of feedback collection methods were employed for this study: (i) the recording of practical collaborative activities – this was because at the end of the presentation, the students were encouraged to comment on the task and the learning they had acquired in each teaching unit, (ii) two questionnaires, through which students could assess aspects of the subject and teaching approaches anonymously.

4 Related works

The work carried out by Benitti [23] sets out a methodology based on learning objects that are used to teach different computing subjects, although the case study of the work was conducted in a software testing subject. It made use of case studies to evaluate the possible learning gains that could be obtained from the methodology.

The author divided her methodology into 2 phases: in the first phase the contents to be covered must be defined, with priority being given to material acquired from reference curricula and other sources, such as certifications, standards and maturity models. In the second phase of the methodology, the mapping of the learning level of each content is only based on the first cognitive process dimension of Bloom's revised taxonomy [17].

The methodology of this work is mainly differentiated through its use of the knowledge dimension of Bloom's revised taxonomy [17], which correlates the content with active teaching methodologies. The purpose of this is to achieve learning levels and encourage teamwork so that students are confronted with challenges that are closer to real situations in the software industry, as recommended by ACM / IEEE [15] and SBC [12].

Another work that employs a teaching methodology for the creation of a software testing subject is that of Liu [39], which combines different online and offline teaching methods and is divided into 3 phases. In Phase 1, there is pre-class learning, through articles and training manuals provided by the professor; in Phase 2 there is a period to ¨internalize¨ and absorb the knowledge acquired in the previous phase, with the aid of different active teaching methodologies, (such as a flipped classroom method with a focus on industrial software testing problems. Finally, in Phase 3 students watch video summaries to extend their knowledge-building capacity and do online tests and homework to broaden their knowledge of software testing through an online platform with feedback for each task performed.

The main differential of the work shown is in the way the academic content was built, since there is a phase designed for defining what will be learn and setting a threshold for the learning to be achieved, based on Bloom's revised taxonomy [17]. This, involves employing active teaching methodologies for knowledge construction.

Another approach to the subject of software testing is by Enoiu [40], and consists of a project-based testing course which employs models in remote teaching within a context of software industry standards, this entails students watching online lectures before classes and later on, holding group discussions in virtual classrooms; in addition, it includes pedagogical techniques such as group work, student presentations and discussions. It is also recommended that an online learning platform is created to store teaching material.

The aim of the study by Furtado et al. [33] is to help teach the topic of Statistical Process Control (SPC), by separating the content into four teaching units aligned with a set of academic competences and its expected results with the aid of Bloom's taxonomy [41]. The authors achieved this by developing a competency model [42]. The content is disseminated through student-focused learning and a set of active teaching methodologies.

The differentials of this study are based on Bloom's revised taxonomy [17], which is a recent form of evaluation in the learning process and, employs its own methodology with the aim of being able to help researchers form new subjects in the future.

The analysis of each of these related works illustrated the use of different methodologies and teaching-learning strategies for teaching software testing or other topics in Software Engineering. The recent prominence given to teaching the subject of software testing was also noted, (as mentioned in the work by [25].)

One of strengths of this project can be attributed to the fact that its designers have created their own syllabus for teaching tests [7]. These comprise several items derived from different sources, which can be better viewed in the Objective, Syllabus and Teaching Plan section, based on Bloom's revised taxonomy. Its results can be predicted, as recommended by the Computing Curricula 2020 (CC2020).

Some weaknesses of this work are as follows: i) the design of the quasi-experiment, since there was a time interval of one year between the study undertaken by the control group and the experimental group; ii) the difference between a group that attended in-person classes and another conducted through remote classes (due to the COVID-19 pandemic); and iii) the failure to use pre-testing to assess learning efficiency this meant that the results could be biased because the experimental group was taught remotely, without any control of what was being accessed by students during the pre-test.

5 Evaluation

The aim of this quasi-experiment was mainly to evaluate the possible effectiveness of learning software tests at different levels of learning by adopting the teaching approach outlined here and comparing it with the results achieved in traditional classes in a post-graduate program in Computer Science.

5.1 Research and evaluation strategy

A quasi-experiment was conducted with two different graduate classes in Computer Science to evaluate the teaching approach and make a comparative assessment of the effectiveness of each of the Teaching Units that make up the syllabus [7].

The method employed for this was a quasi-experiment [43], which is an empirical interventional study used to assess the effect of making an intervention in a targeted population without random assignment. This is because it is not possible to control and select the students who will take part in the classes, as well as the fact that this study is longitudinal and annual, and has both in-person and remote teaching t environments.

This form of quasi-experiment allows a statistical comparison to be made between the behavior of the experimental group and a control group. Thus, the quasi-experiment was carried out as follows: the quasi-experiment was conducted in 2 different semesters. The participants were students enrolled in a Software Quality course of the Graduate Program in Computer Science at the Federal University of Pará in

Brazil. There was no screening or selection of students since they were master's, doctoral or specialized students who enrolled in the subject without knowing about the quasi-experiment. This system served as a means of ensuring the participants were divided into a control group or experimental group at random so that the statistical results would have greater validity.

Interventions were made with traditional classes and with the teaching approach adopted. The experimental group took part in the learning activities carried out for the teaching approach described in this article, while the control group participated in classes that relied on traditional teaching methods. At the end of each teaching unit, the results of the evaluative activities were collected. While the experimental group carried out different activities, (outlined in the teaching approach), the control group had to answer a list of questions about the contents learned, to achieve this result. At the end of the quasi-experiment carried out, the students answered a perception questionnaire and the experimental group also answered one about their learning experience.

Table 3 summarizes the information about the quasi-experimental design.

Table 3. Summary of the case study

Groups	Preparation	Interventions		Conclusion
Control	Allocation of groups in each semester, according to which subjects the students were enrolled in, and with regard to the class capacity (15 students)	Traditional classes.	Application of exams.	Content perception questionnaire.
Experimental		Classes based on the methodology employed.	Application of lists of exercises, Dojos and case studies.	Content and learning perception questionnaire.

5.2 Research questions and hypotheses

When seeking to answer the RQ outlined in the Introduction, a set of other research questions was defined, one for each teaching unit in the current syllabus and these were evaluated in the control and experimental classes.

Grades were awarded from 0 to 10, but in each unit, it was necessary to analyze the level of intervention on the basis of Bloom's revised taxonomy [17], as specified in the syllabus. Thus, the instruments used to evaluate the students' activities were aligned with the expected results of the topics in each unit.

Table 4 shows the study objectives of each of the teaching units designed for the construction of knowledge of software tests, their research questions and instruments, together with their null hypotheses. The variables are directly related to the instruments used to evaluate students from both groups.

Table 4. Details of study objectives

Study objective 1
Research question 1 (RQ1): How effective is the learning of the Software Engineering unit when the Software Testing approach is adopted instead of r the traditional approach at the ¨Create level¨?
Hypothesis H01: There will be no difference between the scores obtained by the Experimental and Control groups at the Create level.
Variables
TI1 - List of Exercises 1 TI2 - List of Exercises 2 TI3 - List of Exercises 3 TG1 - Dojo 1 P1 - Exam 1
Formulation: Ma > Mb, where: a = Experimental Group b = Control Group Experimental Group scores: $Na_i = \frac{((TI1*1)+(TI2*1)+(TI3*1)+(TG1*1{,}5))}{4{,}5}$, where i is a student of Group a Control Group scores: Nb_i = P1, where i is a student from Group b Average of the scores of students in group a: $Ma_i = \frac{\sum_{i=1}^{m} Nai}{m}$, where m is the number of students in Group a Average of the scores of students in group b: $Mb_i = \frac{\sum_{i=1}^{m} Nbi}{m}$, where m is the number of students in Group b
Instruments: Lists of exercises, Dojo and Exam.
Study objective 2
Research question 2 (RQ2): How effective is the learning of the Software Construction unit when the Software Testing approach is adopted instead of r the traditional approach at the Create level?
Hypothesis H02: There will be no difference between the scores obtained by the Experimental and Control groups at the Create level.
Variables
TG2 - Dojo 2 P2 - Exam 2
Formulation: Ma > Mb, Where: a = Experimental Group b = Control Group Experimental Group scores: Na_i = TG2, where i is a student from Group a Control Group scores: Nb_i = P2, where i is a student from Group b Average of the scores of students in group a: $Ma_i = \frac{\sum_{i=1}^{m} Nai}{m}$, where m is the number of students in Group a Average of the scores of students in group b: $Mb_i = \frac{\sum_{i=1}^{m} Nbi}{m}$, where m is the number of students in Group b
Instruments: Dojo and Exam.
Study objective 3
Research question 3 (RQ3): How effective is the learning of the Quality unit when the Software Testing approach is adopted instead of the traditional approach at the Analyze level?
Hypothesis H03: There will be no difference between the scores obtained by the Experimental and Control groups at the Analyze level.

Variables
EC1 - Case Study 1 P3 - Exam 3
Formulation: Ma > Mb, where: a = Experimental Group b = Control Group Experimental Group scores: Na_i = EC1, where i is a student from Group a Control Group scores: Nb_i = P3, where i is a student from Group b Average of the scores of students in group a: $Ma_i = \frac{\Sigma_{i=1}^{m} Nai}{m}$, where m is the number of students in Group a Average of the scores of students in group b: $Mb_i = \frac{\Sigma_{i=1}^{m} Nbi}{m}$, where m is the number of students in Group b
Instruments: Case study and Exam.
Study objective 4
Research Question 4 (RQ4): How effective is the learning of the Software Testing unit when the Software Testing approach is adopted instead of the traditional approach at the Create level?
Hypothesis H04: There will be no difference between the scores obtained by the Experimental and Control groups at the Create level.
Variables
TG3 - Dojo 3 EC2 - Case Study 2 P4 - Exam 4
Formulation: Ma > Mb, where: a = Experimental Group b = Control Group Experimental Group scores: $Na_i = \frac{(TG3*1,5)+(EC2*2,5))}{4}$, where i is a student of Group a Control Group scores: Nb_i = P4, where i is a student of Group b Average of the scores of students in group a: $Ma_i = \frac{\Sigma_{i=1}^{m} Nai}{m}$, where m is the number of students in Group a Average of the scores of students in group b: $Mb_i = \frac{\Sigma_{i=1}^{m} Nbi}{m}$, where m is the number of students in Group b
Instruments: Dojo, Case study and Exam.

5.3 Instrumentation

The existing variables in each unit were used to collect the data needed to answer research questions 1, 2, 3 and 4. The tasks were adapted to each different level of learning, to extract this knowledge from the students, using a pre-established schedule for each group. For example, it was determined that to achieve the expected result "The students must be able to make adjustments and improvements to computational solutions". In the control group, the students had a subjective question about how to refactor a pseudocode, while in the experimental group there was a practical class that employed the Dojo methodology, in which students had to achieve the same result.

The rest of the agenda will be shown later in Table 5. Thus, regardless of the type of instrument used, the activities were designed with the aim of enabling students to face real-world situations that can occur in the software industry.

The activities were corrected by 3 specialists in the area, who were not involved in teaching the content. If there was a disparity in the grades given, there was a meeting to decide how to reach a consensus. The scores for each teaching unit were calculated in accordance with the variables of each object of study in Table 4. The level of learning that had to be reached for each research question, was assessed in terms of the highest expected result that could be obtained from the content of that teaching unit, following Bloom's revised taxonomy.

Even when different instruments were employed, the comparison between the experimental and control groups proved to be reliable, because it was ensured by the fact that in each Teaching Unit a certain level of learning could be reached through Bloom's revised taxonomy. This meant that whether in a Dojo or in a conventional multiple-choice test, the content of the questions had the same level of taxonomy learning, so that the comparison was not distorted.

Possible outliers in the samples were handled (or deleted), with the aim of mitigating problems that could arise from different instruments in each group, (control and experimental), involving random errors. Additionally, a third quasi-experiment is also being carried out to increase the sample size.

Regarding systematic errors caused by measuring instruments, as there were team activities in the experimental group, there was a general awareness that these errors cannot be completely avoided, but only reduced, and bonus grades were given to the more attentive students, when possible. However, since the main object of this study is to enhance the effectiveness of learning through active methodologies which prioritize teamwork, the authors gave priority this design feature of the quasi-experiment.

The grades obtained in each activity were only made available to students at the end of the quasi-experiment. Finally, the collection of feedback from students was also carried out in the way recommended by the model.

5.4 Execution

The first part of the quasi-experiment with the control group was carried out in person in August 2019 and the second part with the experimental group was carried out in August 2020. This had to be adapted to remote learning strategies, owing to the pandemic of COVID-19, in an elective course of Software Quality in the Post-graduate Program in Computer Science (PPGCC) at the Federal University of Pará (UFPA) in Brazil.

All quasi-experimental team members were officially enrolled in the course and those in the first class were informed about the quasi-experiment. Altogether, there were thirty students enrolled, fifteen in each of the classes. All the participants in the quasi-experiment were volunteers and signed an Informed Consent Form – TCLE at the standard that the Post-graduate Program in Computer Science (PPGCC) requires and the research followed the guidelines outlined in the work by Petousi and Sifaki [44]. The subject consisted of a total of 60 hours, and 32 classes during the semester,

each lasting up to 2 hours. In the case of the experimental group, the classes were given remotely with adaptations to the way the methodologies were implemented [10], but keeping all the necessary methodological procedures. Students from both groups had access to backup materials for studying the contents taught. The quasi-experiment schedule is displayed in Table 5.

Table 5. Case study schedule

Days	Control Group	Experimental Group
Inaugural Class	Presentation of the Syllabus [7], inclusion of the subjects and teaching plan. Availability of backup material.	Presentation of the Syllabus [7], inclusion of the subjects and teaching plan [8]. Availability of backup material.
Classes 2 to 7	**Expository classes of a classical form:** About Teaching Unit 1 – Software Engineering: With topics 1.1 and 1.2, shown in Table 2. **Exam:** Evaluative task with multiple choice and discursive questions about the content taught in the teaching unit.	**Expositional dialogue classes:** about Teaching Unit 1 - Software Engineering: With topics 1.1 and 1.2, shown in Table 2. **List of Exercises:** List of multiple choice exercises with subjects included in the content of the teaching unit, based on ISTQB and TMMI certification exams. The activity was carried out on an individual basis. **Dojo Randori:** Challenge for code building and unit testing for a banking service using the Scratch online platform. All the students had to play the roles of a pilot, co-pilot and the audience. A grade was given to all the students who took part, which took into account the number of complete and correct challenges. This activity was largely collaborative, with all the students in the remote classroom working together, but the students were assessed individually, which meant that some of them had lower grades than others.
Classes 9 to 13	**Expository classes of a classical form:** About Teaching Unit 2 – Software Construction: With topics 2.1 and 2.2, shown in Table 2. **Exam:** Evaluative task with multiple choice and discursive questions about the content taught in the teaching unit.	**Expositional dialogue classes:** about Teaching Unit 2 - Software Construction: With topics 2.1 and 2.2, shown in Table 2. **Dojo Kake:** In this practice, multiple teams carry out the activity by working in parallel. Each team compiled a list of challenges in Java and the professors helped by mentoring the concepts learned in the teaching unit. Each team is awarded a grade based on the degree of correctness and quality used in the code. The activity was divided into groups of three or four students, and the grades were given to each group in an equitable manner.
Classes 14 to 19	**Expository classes of a classical form:** About Teaching Unit 3 – Quality: with topics 3.1 and 3.2, shown in Table 2. **Exam:** Evaluative task with multiple choice and discursive questions about the content taught in the teaching unit.	**Expositional dialogue classes:** About Teaching Unit 3 – Quality: with topics 3.1 and 3.2, shown in Table 2. **Practical Project:** The practical project was a progressive activity, and at first the students analyzed the implementation of a software interface After this, they conducted an Ad-hoc test analysis to locate errors and failures and determine new requirements for the system. Each team is awarded a grade that matches the degree of success and

		quality employed in the activity. The activity was divided into groups of three or four students, and the grades were given to each group on an equitable basis.
Classes 20 to 32	**Expository classes in a classical form:** About Teaching Unit 4 – Software Tests: with topics 4.1, 4.2, 4.3 and 4.4, shown in Table 2. **Exam:** Evaluative task with multiple choice and discursive questions about the content taught in the teaching unit.	**Expositional dialogue classes:** About Teaching Unit 4 – Software Tests: with topics 4.1, 4.2, 4.3 and 4.4, shown in Table 2. **Discussion of Practical Cases:** Presentation of experience reports and existing tools in the software development industry or in project laboratories at universities so that students can be confronted with problems that arise in real-world environments. **Dojo Kake:** In this practice, multiple teams work in parallel to carry out the activity. Each team compiled a list of Java code refactoring, (code improvement) and had to rely on bug tracking. The professors helped with mentoring the concepts learned in the teaching unit. Each team is awarded a grade that reflects the degree of success and quality shown in the activity. The activity was divided into groups of three or four students, and the grades were given to each group on an equitable basis. **Practical Project:** The teams built a prototype and formed a test plan on the basis of what had been collected in the Ad-hoc test analysis in the previous teaching unit, and presented the results to the professors. Each team is awarded a grade that reflects the degree of success and quality shown in the activity. The activity was divided into groups of three or four students, and the grades were given to each group on an equitable basis.
Feedback	Content Perception Questionnaire.	Content Perception Questionnaire. Questionnaire on teaching approaches.

Each group had its own professor while the quasi-experiment was being carried out and was assisted by three other professors. Hence, only one professor taught the control group, while another one taught the experimental group, with the exception of a practical case-study and discussion classes where a specialist was invited to teach the class. For further details of the findings and the quasi-experiments of the experimental group, see [10].

6 Data analysis

In this section, there is an examination of the data obtained from the case study explained in this work. An analysis will be carried out on how each of the research questions refers to the teaching units, and finally, there will be an analysis of the main Research Question.

6.1 Analysis of research question 1

In response to RQ1, (which can be seen in Section 5.2), the two-tailed student t test was chosen for independent samples so a comparison could be made between the experimental and control groups in teaching unit 1. This took account of the normality of the data and its objective was to evaluate the difference between two populations with varying treatment conditions and two samples (treatments).

A standard alternative string (Two Sided) with a confidence interval of 95% was used to calculate the degree of variance. The p-value of the F test = 0.08053, which is greater than the 0.05 significance level. In conclusion, there is no significant difference between the two mean variances, so the hypothesis test used for this research question was the Student-t hypothesis for two means with equal and unknown variances. It was noted that with a 5% significance level, H01 could be rejected; that is, there is statistical evidence to show that the means are different, when these indicators are con- confirmed by the p-value = 0.00151 < 0.05. Table 6 summarizes the results obtained for RQ1.

Table 6. Comparison of the learning effectiveness of the participating groups (Student-t) in teaching unit 1

Variables	**Experimental Group**	**Control Group**
	Evaluation	***Evaluation***
Sample size	15	15
Minimum	4.9	2
Maximum	9.3	8
Sum total of Scores	116.1	86.5
Median	8	6
First quartile	7.6	4,5
Third quartile	8	7
Average	7.74	5.76
Standard Deviation	1.14005	1.850354

6.2 Analysis of research question 2

In response to RQ2, (which can be seen in Section 5.2), a comparison was made between the experimental and control groups in teaching unit 2, and the choice resulting from the comparison made for RQ1 was followed.

When the two variances are compared, the p-value of the F test is p-value = 0.3431, which is greater than the 0.05 significance level. In conclusion, there is no significant difference between the two mean variances, so the hypothesis test used for this research question was the Student-t hypothesis for two means with equal and unknown variances. It was noted that with a 5% significance level H02 could be rejected; that is, there is statistical evidence to show that the means are different, when these indicators are confirmed by the p-value = 5.87e-05 (0.0000587) < 0.05. Table 7 summarizes the results obtained for RQ2.

Table 7. Comparison of the degree of learning effectiveness between the participating groups (Student-t) in teaching unit 2

Variables	Experimental Group	Control Group
	Evaluation	***Evaluation***
Sample size	15	15
Minimum	7.5	3
Maximum	10	8.5
Sum Total of Scores	129	86.5
Median	8.5	6
First quartile	8	4.5
Third quartile	9	7
Average	8.5	5.43
Standard Deviation	1.535299	1.989855

6.3 Analysis of research question 3

In response to RQ3, (which can be seen in Section 5.2), a comparison was made between the experimental and control groups in teaching unit 3, followed by the choice resulting from the comparison made for RQ1.

When the two variances of RQ3 are compared, the p-value of the F test = 0.02669, which is lower than the 0.05 significance level, so the hypothesis test used for this research question was the Student-t hypothesis for two means with equal and unknown variances. In conclusion, there is a significant difference between the two mean variances. It was noted that with a 5% significance level, we can reject H03; that is, there is statistical evidence to show that the means are different, when these indicators are confirmed by the p-value = 7.915e-06 (0.000007915) < 0.05. Table 8 summarizes the results obtained for RQ3.

Table 8. Comparison of the degree of learning effectiveness between participating groups (Student-t) in teaching unit 3

Variables	Experimental Group	Control Group
	Evaluation	***Evaluation***
Sample size	15	15
Minimum	7.5	3
Maximum	10	8.5
Sum Total of Scores	129	86.5
Median	8.5	6
First quartile	8	4,5
Third quartile	9	7
Average	8.6	5.76
Standard Deviation	0.8904253	1.656876

6.4 Analysis of research question 4

In response to RQ4, (which can be seen in Section 5.2), a comparison was made between the experimental and control groups in teaching Unit 4 and the comparison made for RQ1 was followed.

In a similar way to RQ3 in the variance test, the p-value of the F test = 0.3431, which is greater than the 0.05 significance level, so the test used for this research question was the Student-t hypothesis for two means with equal and unknown variances. In conclusion, there is a significant difference between the two mean variances. Finally, it was noted that with a 5% significance level, we can reject H04; that is, there is statistical evidence to show that the means are different, when these indicators are confirmed by the p-value = 0.0004864 < 0.05. Table 9 summarizes the results obtained for RQ4.

Table 9. Comparison of the degree of learning effectiveness between participating groups (Student-t) in teaching unit 4

Variables	**Experimental Group**	**Control Group**
	Evaluation	***Evaluation***
Sample size	15	15
Minimum	8.8	4
Maximum	9.8	8
Sum Total of Scores	138.6	89.5
Median	9	6
First quartile	8.8	4.75
Third quartile	9.8	6.6
Average	9.24	5.96
Standard Deviation	0.4792852	1.32916

6.5 Analysis of main research question

As defined in the Introduction, the main Research Question (RQ) can be stated as follows: "If professors of Software Engineering courses adopt a set of active teaching methodologies and collaborative teaching practices instead of employing traditional strategies, will students be able to develop more academic skills related to Software Testing?".

In an attempt to answer the RQ, a quasi-experiment was carried out with a control class where a traditional teaching method was employed and an experimental class with the constructed approach. From the results obtained in the research questions when they were broken down into Q 1, 2, 3 and 4, there was a gain of learning which reflected that more of the academic competences listed in all the teaching units that employed active teaching methodologies and collaborative practices were acquired when compared with what could be attained through the traditional approach.

Thus the authors consider the results to be significant, although they appreciate that they should not be generalized to a great extent, Later in Section 7, the results will be

discussed and in Section 8 the authors will address the question of threats to the validity of this research.

7 Discussion of results

The results obtained from the rejection of hypotheses H01, H02, H03 and H04 suggest that this approach has a higher learning effectiveness than what can be achieved by a traditional classroom methodology, since the average grades obtained by the experimental group during the evaluations were significantly higher than those of the control group.

These results can be attributed to the fact that the active teaching methodologies used in the experimental group are particularly focused on students, and carrying out practical and collaborative activities, as well as on the teaching approaches adopted. In view of this, the results obtained are similar to those of several authors who adopted more student-centered approaches and strategies [33], [22], or relied on active teaching methodologies [45] and competence-based learning [23], [46].

The fact that the results of H03 and H04 in the experimental group are much higher may be due to the fact that in teaching units 3 and 4, respectively, all the activities were carried out in a team and were of a practical nature. It can also be explained by the way, the quasi-experiment was carried out in a graduate program of Software Quality Assurance and the fact that the students might have had professional experience in the labor market or in activities related to the program's research groups. However, this same factor also applies to the control group students. In the case of this work, no analyses were carried out to assess the performance of students in other subjects, nor was it determined if they had any previous experience or knowledge of software testing topics.

When the qualitative feedback provided by the students was analyzed and account taken of what drove them to learn software testing, it was noted that most students believe that learning the subject is important and that the experience gained from testing would open up a wide range of opportunities in the job market, since there is very little knowledge of this among working professionals. Students from both groups found the contents of the current syllabus [7] to be extremely useful and sufficient for learning about software testing. Students in the experimental group evaluated the practical activities carried out by means of active teaching methodologies as being beneficial, even though they took place through remote teaching.

The control group had a large workload consisting of theoretical classes and an evaluative test at the end of each teaching unit. Thus, students may have had difficulties in learning certain subjects solely from traditional lectures with little chance of collaboration and an exchange of knowledge between students. This may have resulted in a score that was considerably lower than what was achieved by the experimental group.

However, some weaknesses in our approach were noted during the quasi-experiment carried out with the experimental group. Although there were a considerable number of practical tasks, this approach still included expositional dialogue clas-

ses to teach the academic content required by the syllabus [7]. In fact, the students believed that there were too many expository classes and stated that there should have been more Dojo activities and practical projects linked to the subject. Perhaps, this result can be accounted for by the profile of the students who were involved in the quasi-experiment conducted by the control group. The authors of this work intend to determine more precisely the nature of the learning profiles and preferences of the students who are most suited to the new teaching approach so that they can be adapted to it.

Another weakness of the approach concerns the time constraints for carrying out practical activities, especially with remote learning. Thus, a more suitable adaptation and review of the learning activities could lead to a better experience for the students, and enable them to carry out any of the activities within the stipulated period. The amount of academic content in the syllabus [7] was also a possible weakness, as a large number of classes was necessary to cover all of it, and it required students to carry out some activities outside of class time.

It should be noted that the quasi-experiment was carried out over a period of 32 classes in both groups, so that all the teaching units in the syllabus could be covered. The model used for the teaching approach recommends the professor should break down the content into teaching units and use them separately in different subjects throughout a course, as a means of overcoming the problem of the excessive amount of content.

The activities that led to the best results were group activities, although the Dojo Randori activity carried out with all the students in the experimental group was their most difficult task, because a large number of students were unfamiliar with the content, and their different backgrounds ended up having an adverse effect on the score.

A final factor that may have prevented the experimental group from making a real improvement is linked to the question of good practices since these are essential for professors involved in remote classes, according to the regulations laid down by the MEC - Brazilian Ministry of Education and Culture. These state that even in a synchronous class a video of the class activities should be made available to students later, to ensure that they have access to classroom materials and other external aids, before and possibly during the activities. However, the experiment with the control group was conducted before the pandemic of COVID-19 in asynchronous class, then they didn't have this.

8 Threats to validity

Any result obtained in academic research should be treated with some caution, especially with regard to the possible generalizability of the results. Moreover, in the case of the study undertaken here, there are some threats to validity that can influence the results. These are outlined below, together with some actions that can be taken to mitigate them.

8.1 Internal validity

According to Travassos, Gurov and Amaral [47], internal validity helps to define whether the observed relationship between treatment and outcome is causal, and not the result of the influence of some other uncontrolled or measurable factor. This means that, there are unforeseen events that can lead to distortions in the results.

The experimental and control groups were formed by enrolling students in the Software Quality course in a graduate program for two consecutive semesters. It should be noted that this course was not mandatory and none of the students were invited to do it. This decision was made so as not to influence the formation of the classes and reduce the risk of confounding factors, and the possible threat of statistical regression, and thus make the control group and the experimental group as similar as possible and statistically equivalent.

The existence of a threat of internal validity related to maturation is possible because researchers cannot restrict the search for external knowledge to that of the subject being studied by the students. As a means of trying to reduce outside influences, the backup material for students from both groups related to content, was the same and professors were always available to answer questions and help any student from either group outside of class time.

Three specialists corrected the evaluations to mitigate the internal threat of bias by instrumentation, and it was not the professors who taught the subject to the groups. Moreover, a statistical researcher who was not involved in the design and implementation of the subject, had the sole function of conducting the data analysis. To ensure impartiality in the evaluation, the specialists were not given any information about the students who had carried out the activity.

Another possible threat of bias from instrumentation may be caused by the approach adopted in the subject offered to the experimental group and the different forms of evaluation found between the two groups.

As stated earlier, the subjects were taught by two different professors, one for each group. This may have resulted in one group learning less than the other, not only because of the effects of the approach, but also because of the depth of knowledge of the group's professors and their capacity to disseminate content. The qualifications of the professors were being a Master in Computer Science and a Doctor in Computer Science. One way to mitigate the risk of bias was for the professors to compile the syllabus and draw up a teaching plan together.

8.2 External validity

The threat to external validity is a condition that limits the ability to generalize the research results. Thus, in the case of a quasi-experiment carried out in an academic context with graduate students in computer science, these results should only be generalized within that academic world.

The quasi-experiment was carried out with a very small sample of students, and has not yet been replicated with another type of population, or at different academic levels and in other universities. Thus, the ability to generalize the results obtained is

limited, because most of the students have graduated in computer courses and are already post-graduate students.

With regard to the teaching approach adopted in the experimental group which involves carrying out practical teaching activities with challenges similar to those facing the software industry, there is no way to guarantee that the skills and knowledge acquired by students can be replicated in real-world situations that occur in the job market.

8.3 Construct validity

Construct validity refers to the relationship between the instruments and participants in the case study and, in this work, the results of the research questions. The main construct validity concerns the effectiveness of learning between the experimental and control groups at cognition levels from Bloom's revised taxonomy [25], where these results may not be sufficient to measure the learning achieved by students at each recommended learning level.

In light of this, some statements cannot be made that are based on the results of the case study; for example, it cannot be asserted that students who were able to answer the lists of exercises with certified content are able to take the tests and be approved. The practical knowledge acquired may also not be applicable to different scenarios.

8.4 Conclusion validity

These threats address the validity of some inferences about the correlation between treatment and effect, which can undermine the statistical results. Until the time that these are released, the data collected from the populations of the participating groups is very small. In some cases, different statistical tests were used for different research questions in this quasi-experiment, while taking into account the variance of data from the sample in that teaching unit. The objective of the most robust statistical tests was to try to circumvent the problem of the low statistical power of the distribution of data obtained. This problem may have arisen because the experimental group had team activities, which led to data homogeneity.

9 Conclusion

This article investigated the first experimental results of the application of a teaching approach for software testing based on the contents of the syllabus [7], and also an experimental approach (that followed a teaching plan [8] and was supported by active methodologies), which obtained a better grade from learning in all the teaching units when compared with the control group, which relied on traditional expository classes. As the main objective of the work was to analyze the cognitive levels of learning, it was decided to make a comparison between very extreme approaches in each of the groups examined in this work. The article sought to answer the following RQ: "If professors of Software Engineering courses adopt a set of active teaching methodolo-

gies and collaborative teaching practices instead of traditional strategies, will students develop more academic skills related to Software Testing?" The new approach was more efficient in all the teaching units, according to the Student-t test, with a P value lower than 0.05 in all the comparisons that were made.

A secondary research contribution made by this study was the use of a teaching model that was applied in the experimental group to help the teaching course? This model was used because of the COVID-19 pandemic, which meant that an experiment that had been designed for in-person classes had to be adapted to a remote form of teaching. It should be made clear that the objective of this work is not to evaluate the model, although we could not fail to show how it operates and its importance for the progress of the activities.

The results of this work can be considered to be significant, since the students participating in the experimental group obtained a good level of learning from the software testing content. However, owing to threats to validity, a broad generalization of the results is not possible at this time.

There are some weaknesses in this study that stem from the fact that there is only a small source of results to validate its statistical effectiveness. Another factor is that there was a need to adapt the form of on-site teaching to remote learning, which may have led to some improvement in the results. This might be due to the lists of exercises and distant learning activities carried out by the experimental group, since we could not control what they accessed online while doing the activities, unlike the tests undertaken by the control group, which were in person.

In future work, we intend to replicate the quasi-experiment in other Software Quality classes of the Postgraduate Program in Computer Science (PPGCC) of the Federal University of Pará (UFPA) to evaluate the effectiveness of teaching software tests with active teaching methodologies. Another goal is to evaluate a possible gain in motivation and learning which can be achieved by using the constructed teaching model in different academic subjects of the Postgraduate Program in Computer Science (PPGCC) at the Federal University of Pará (UFPA).

Finally, in future studies, the authors seek to determine how far the evaluations of different levels of knowledge are within the domain of active methodologies for teaching software tests. This can make it possible to determine which teaching techniques and practices among the active methodologies are more effective and motivational.

10 Acknowledgments

The authors would like to thank CAPES (Coordination for the Improvement of Higher Education Personnel) for granting an institutional doctoral scholarship (linked to the PPGCC/UFPA) to enable the student-researcher of this article to conduct the research that appears in this article. Additionally, the authors would like to thank the participants, students and monitors, who carried out the quasi-experiment described in this article. This article is part of the results of the SPIDER Project - Software Process Improvement: DEvelopment and Research (http://spider.ufpa.br) at UFPA.

11 References

[1] O'Regan, G. (2019). Concise Guide to Software Testing. Springer International Publishing. https://doi.org/10.1007/978-3-030-28494-7

[2] Dolezal, D., Posekany, A., Vittori, L., Koppensteiner, G. & Motschnig, R. (2019). Fostering 21st Century Skills in Student-Centered Engineering Education at the Secondary School Level: Second Evaluation of The Learning Office. 2019 IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE43999.2019.9028646

[3] Sánchez-Gordón, M., Rijal, L. & Colomo-Palacios, R. (2020). Beyond Technical Skills in Software Testing. Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops. https://doi.org/10.1145/3387940.3392238

[4] Van Dam, K. (2019) The Future of Testing. The Future of Software Quality Assurance. Springer International Publishing. 197–205. https://doi.org/10.1007/978-3-030-29509-7_15

[5] Van Veenendaal, E. (2019). Next-Generation Software Testers: Broaden or Specialize! The Future of Software Quality Assurance. Springer International Publishing. 229–243. https://doi.org/10.1007/978-3-030-29509-7_18

[6] Elgrably, I. & Oliveira, S. (2019). A Proposal for Teaching or Applying Tests with a Focus on Agile Methods made through an Asset Mapping. 16th International Conference on Information Systemas & Technology Management.

[7] Elgrably, I. & Oliveira, S. (2020a). Construction of a curriculum adhering to the Teaching of Software Tests Usingelements of Agile Context. IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE44824.2020.9274266

[8] Elgrably, I. & Oliveira, S. (2020b). Model for Teaching and Training Software Testing in an Agile Context. IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE44824.2020.9274117

[9] Elgrably, I. & Oliveira, S. (2021a). A diagnosis on software testing education in the Brazilian Universities. IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE49875.2021.9637305

[10] Elgrably, I. & Oliveira, S. (2021b). Remote teaching and learning of software testing using active methodologies in the COVID-19 pandemic context. IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE49875.2021.9637426

[11] Papadakis S, Kalogiannakis M (2020) Learning Computational Thinking Development in Young Children With Bee-Bot Educational Robotics. In Advances in Early Childhood and K-12 Education (pp. 289–309). https://doi.org/10.4018/978-1-7998-4576-8.ch011

[12] Zorzo, F., Nunes, D., Matos, E., Steinmacher, I., Leite, J., Araujo, R., Correia, R. & Martins, S. (2017). Reference Documentation of Training for Undergraduate Computer Courses. Sociedade Brasileira de Computação (SBC).

[13] CC2020 Task Force, "Computing Curricula 2020." ACM, Nov. 15, 2020. https://doi.org/10.1145/3467967

[14] MEC (2016). National Curriculum Guidelines for Undergraduate Computer Courses (Dcn16). Brazil.

[15] ACM/IEEE (2017); Information Technology Curricula: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology a Report in the Computing Curricula Series Task Group on Information Technology Curricula.

[16] Hong, Q., Lu, W., Feng, P., Wei, H., & Cheng, Z. (2015). Occupational Ability Oriented Graduate Education in Software Engineering. International Journal of Emerging Technologies in Learning (IJET), 10(8), 25. https://doi.org/10.3991/ijet.v10i8.5214

[17] Anderson LW, Krathwohl DR (2001). A Taxonomy for Learning Teaching and Assesing: A Revision of Bloom's Taxonomy of Educational Objectives. Longman.
[18] ACM/IEEE (2013). Computer science curricula 2013. Curriculum guidelines for undergraduate degree programs in Computer Science.
[19] SBC- Sociedade Brasileira de Computação (2005). SBC Reference Curriculum for Undergraduate Degree Courses in Computer Science and Computer Engineering. Grupo de trabalho responsável – CR2005.
[20] Frezza, S., Clear, T. & Clear, A. (2020). Unpacking Dispositions in the CC2020 Computing Curriculum Overview Report. IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE44824.2020.9273973
[21] Lauvås, P. & Arcuri, A. (2018). Recent Trends in Software Testing Education: A Systematic Literature Review. in The Norwegian Conference on Didactics in IT education.
[22] Portela, C., Vasconcelos, A., Oliveira, S. & Souza, M. (2021). An Empirical Study on the Use of Student-Focused Approaches in the Software Engineering Teaching. Informatics in Education. https://doi.org/10.15388/infedu.2021.13
[23] Benitti, F. (2018). A Methodology to Define Learning Objects Granularity: A Case Study in Software Testing. Informatics in Education, V. 17, No. 1, pp. 1–20. https://doi.org/10.15388/infedu.2018.01
[24] Morán, J. (2015). Changing education with active methodologies. In: Media Convergences, Education and Citizenship: young approaches. Coleção Mídias Contemporâneas, v. 2, n. 1, p. 15-33.
[25] Garousi V, Rainer A, Lauvås P, Arcuri, A (2020) Software-testing education: A systematic literature mapping. In Journal of Systems and Software (Vol. 165, p. 110570). Elsevier BV. https://doi.org/10.1016/j.jss.2020.110570
[26] Bourque, P. & Fairley, R. (2014). SWEBOK Guide V3.0. Available: www.swebok.org.
[27] TMMI Foundation (2018). Test Maturity Model Integration – TMMI Release 1.0.
[28] TMMI Foundation (2019). TMMi in the Agile world – TMMI Release 1.3.
[29] ISTQB (2018). International software testing qualifications board. available at: https://www.istqb.org
[30] ISO/IEC/IEEE (2016). International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing," in ISO/IEC/IEEE 29119-5 First edition 2016-11-15, pp.1-69.
[31] Laing, S. & Greaves, K. (2015). The Testing Manifesto. Available: http://www.growingagile.co.za/2015/04/the-testing-manifesto
[32] Crispin, L. & Gregory, J. (2014). Testing and Devops: In more Agile Testing: Learning Journeys for the Whole Team. Addison-Wesley Professional.
[33] Furtado, J., Oliveira, S., Chaves, R., Telles, A. & Colares, A. (2021). An Experimental Evaluation of a Teaching Approach for Statistical Process Control in Computer Courses. International Journal of Information and Communication Technology Education. 17, 1, 154–171. https://doi.org/10.4018/IJICTE.2021010110
[34] Al-Abdullatif, A. M., & Gameil, A. A. (2021). The Effect of Digital Technology Integration on Students' Academic Performance through Project-Based Learning in an E-learning Environment. International Journal of Emerging Technologies in Learning (IJET), 16(11), 189. https://doi.org/10.3991/ijet.v16i11.19421
[35] Paschoal, L., Oliveira, M., Melo, S., Barbosa, E. & Souza, S. (2020). Evaluating the impact of Software Testing Education through the Flipped Classroom Model in deriving Test Requirements. Proceedings of the 34th Brazilian. https://doi.org/10.1145/3422392.3422489

[36] Papadakis, S., Vaiopoulou, J., Sifaki, E., Stamovlasis, D., Kalogiannakis, M. & Vassilakis, K. Factors That Hinder in-Service Teachers from Incorporating Educational Robotics into Their Daily or Future Teaching Practice. In Proceedings of the 13th International Conference on Computer Supported Education (CSEDU 2021) - Volume 2, pages 55-63 ISBN: 978-989-758-502-9 ISSN: 2184-5026.

[37] Amnouychokanant, V., Boonlue, S., Chuathong, S., & Thamwipat, K. (2021). Online Learning Using Block-based Programming to Foster Computational Thinking Abilities during the COVID-19 Pandemic. International Journal of Emerging Technologies in Learning (IJET), 16(13), 227. https://doi.org/10.3991/ijet.v16i13.22591

[38] Levanova, E. A., Galustyan, O. V., Seryakova, S. B., Pushkareva, T. V., Serykh, A. B., & Yezhov, A. V. (2020). Students' Project Competency within the Framework of STEM Education. International Journal of Emerging Technologies in Learning (IJET), 15(21), 268. https://doi.org/10.3991/ijet.v15i21.15933.

[39] Liu, A. (2020). Design of Blending Teaching Mode for Software Testing Course. 2020 15th International Conference on Computer Science & Education (ICCSE). https://doi.org/10.1109/ICCSE49874.2020.9201740

[40] Enoiu, E. (2020). Teaching Software Testing to Industrial Practitioners Using Distance and Web-Based Learning." Frontiers in Software Engineering Education. Springer International Publishing. pp. 73–87. https://doi.org/10.1007/978-3-030-57663-9_6

[41] Bloom (1956). Taxonomy of Educational Objectives: The Classification of Educational Goals.

[42] Portela, C. (2018). An Iterative Model for Teaching Software Engineering Based on Student-Focused Approaches and Industry Training Practices, Tese de Doutorado, Universidade Federal de Pernambuco.

[43] Gribbons, B., & Herman, J. (1996). True and quasi-experimental designs. Practical Assessment, Research & Evaluation, 5(1), 14.

[44] Petousi, V. and Sifaki, E. (2020) 'Contextualising harm in the framework of research misconduct. Findings from discourse analysis of scientific publications', Int. J. Sustainable Development, Vol. 23, Nos. 3/4, pp.149–174. https://doi.org/10.1504/IJSD.2020.115206

[45] Fonseca, V. & Gomez, J. (2017). Applying Active Methodologies for Teaching Software Engineering in Computer Engineering. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, V. 12, No. 4, pp. 182–90. https://doi.org/10.1109/RITA.2017.2778358

[46] Yoshioka, S. & Ishitani, L. (2018). An Adaptive Test Analysis Based on Students' Motivation, Informatics in Education, V. 17, No. 2, pp. 381–404. https://doi.org/10.15388/infedu.2018.20

[47] Travassos, G., Gurov, D. & Amaral, E. (2002). Introduction to Experimental Software Engineering. Coppe/UFRJ, Rio de Janeiro, Relatório Técnico: RT-ES590/02.

12 Authors

Isaac Souza Elgrably, Doctoral student in Computer Science with emphasis in Software Engineering from Graduate Program in Computer Science (PPGCC) at Federal University of Pará (UFPA). His research areas are: Software Testing, Software Engineering and Software Quality.

Sandro Ronaldo Bezerra 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 (FACOMP) and Graduate Program in Computer Science (PPGCC) at Federal University of Pará (UFPA). 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 (email: srbo@ufpa.br).

Article submitted 2021-07-22. Resubmitted 2021-11-29. Final acceptance 2021-12-13. Final version published as submitted by the authors.