Software Testing Education: dreams and challenges when bringing academia and industry closer together

Stevão Alves de Andrade
Universidade de São Paulo
São Carlos, São Paulo
stevao@icmc.usp.br

Vânia de Oliveira Neves
Universidade Federal Fluminense
Niterói, Rio de Janeiro
vania@ic.uff.br

Márcio Eduardo Delamaro
Universidade de São Paulo
São Carlos, São Paulo
delamaro@icmc.usp.br

ABSTRACT

Software systems are present in people's lives and they are increasing to the same extent as their complexity and their criticality. Therefore, we must ensure that these systems maintain high quality in order to behave as expected. To develop high quality software, it is essential to have qualified people who are knowledgeable about Validation and Verification (V&V) techniques, especially software testing. This paper reports on the teaching process in two undergraduate courses in two different contexts: Computer Science students, who can dedicate more time during the day to studying, and Information Systems students, who can only study during the evenings. To engage and motivate the students in the context of software testing learning, we studied ways to bring real industry problems to the classroom in order to adopt the Problem-based Learning (PBL) approach. We chose two real open source projects which, considering the feedback from students and professors, was a good decision. However, the approach requires students to take extra classes rather than teacher-centered approaches. Extra classes may hinder the approach when the class consists of students who work during the day, thus developing a balance between student-centered and teacher-centered can be a good solution in such contexts.

KEYWORDS

Software testing, education, industry, problem-based learning, PBL

ACM Reference Format:
Stevão Alves de Andrade, Vânia de Oliveira Neves, and Márcio Eduardo Delamaro. 2019. Software Testing Education: dreams and challenges when bringing academia and industry closer together. In *XXXIII Brazilian Symposium on Software Engineering (SBES 2019), September 23–27, 2019, Salvador, Brazil.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3350768.3353903

1 INTRODUCTION

Software systems are ever present in people's lives, and with the advent of new technologies such as the Internet of Things (IoT), smart cities, and industry 4.0, these systems are likely to increase [13]. As these systems grow, their complexity and criticality also become more challenging, and therefore there is a concern to ensure that these systems should maintain high quality to behave as expected. On the other hand, to develop quality software, Verification and Validation (V&V) techniques need to be adopted, especially software testing activities [19]. In this context, it is natural that the demand for professionals who dominate these competencies also increases. We can prove this trend using platforms that offer job positions. In a quick search conducted on 06/2/2019 on *LinkedIn*[1], using the word *"Software"* as filters for "industry", and "all Brazilian territory" for "location", 2,757 positions were found when we used the keyword *"Programmer"*. When we performed the search using the keyword *"Test Analyst"*, the platform returned 5,033 positions. That is, although we have not searched systematically, it can be observed that there is a significant lack of qualified professionals in the software testing area.

This lack of skilled labor may be related to the way software testing is taught nowadays. Besides that, proper software testing education enables the creation of a quality software development culture. According to Krutz et al. [11], students are not motivated to test software; they are interested in building software without worrying about ensuring its quality. Several factors may impact this, such as the lack of specific software testing courses offered to students, and the lack of more practical approaches that better prepare students to tackle problems in the industry.

To increase student engagement and performance in the classroom, several **active learning** approaches and methods have been proposed. According to Bishop and Verleger [4], active learning style is a broad category that encompasses many other styles. A consensus is that students become responsible for their learning, unlike traditional approaches in which this role is delegated to the teacher [6]. Problem-Based Learning (PBL) has gained prominence in this scenario since, according to Hmelo-Silver [10], its application can help students develop five skills: (i) flexible knowledge; (ii) practical problem solving; (iii) autonomous learning; (iv) effective collaboration; and (v) intrinsic collaboration. Liang [12] conducted a study on job opportunities for software testers and concluded that these are essential skills and abilities required by professionals in this field. Thus, this approach is interesting to tackle problems that are very common for newcomers in the market, such as a lack of practical skills, a lack of patience to solve problems, low ability to cooperate and a difficulty in working in groups.

Thus, this paper reports on including the PBL approach in the software testing teaching process in two university classes and in two different contexts: a full-time course comprising Computer Science students, and an Information Systems course consisting of students who attend evening classes. There are many variations of PBL, however Barrows [1] describes six key characteristics: (i) learning

[1] https://www.linkedin.com

SBES 2019, September 23–27, 2019, Salvador, Brazil

ACM ISBN 978-1-4503-7651-8/19/09…$15.00
https://doi.org/10.1145/3350768.3353903

is student-centered; (ii) learning occurs in small groups; (iii) a tutor is present as a guide or facilitator; (iv) real problems are presented at the beginning of the learning sequence, before any preparation or study takes place; (v) we use the problems encountered as tools to achieve the required knowledge and problem-solving skills; and (vi) new information is gained through self-directed learning. To satisfy item 4 and to try to approximate real industry problems in software testing teaching, we planned to use real-world case studies from the industry. By combining practice to teaching software testing concepts, we expected to train the students more carefully concerning the development and understanding of the concepts presented during the course.

Thus, as the **main objectives** of this work, we highlight: (i) the investigation of problems and challenges in choosing a real industry problem which is suitable for students; (ii) the study of mechanisms capable of minimizing the problems associated with students' lack of motivation; and (iii) an exploration of the results using empirical evaluations, of integrating more active learning techniques during the same semester in two different courses.

This paper is organized as follows: Section 2 presents works that are related to the research; Section 3 discusses how we should develop the disciplinary planning in order to successfully achieve integration between academia and industry in the role of student training; Section 4 presents the study objects that we used during the semester; Section 5 describes the context of the courses, the syllabus, the educational objectives, as well as the profile of the students and professors involved. We also present the activities developed by the students, and discuss how we performed the experience presented in this work; Section 6 presents an assessment of the outcomes of the courses by analyzing the students and professors views; and, finally, the conclusions are drawn and future work is discussed in 7.

2 RELATED WORKS

There are several papers in the literature that discuss approaches and tools for teaching courses related to software engineering [3, 7, 14, 15, 17, 20–23], but few papers [7, 14] focus on approaches to teaching software testing specifically.

Brodie et al. [6], Richardson and Delaney [20], Richardson et al. [21] and Cheiran et al. [7] report their experience applying the PBL technique to Software Engineering, Business Information Flow, Software Quality, and Software Testing classes, respectively. All three papers unanimously point out that the experience was positive for both professors and students. For students, using the PBL approach helped to develop essential skills for software engineers such as teamwork, complex problem solving, oral and written communication, critical thinking, taking initiatives, and searching for information. For the professor, PBL required new skills and the ability to adapt to change, which is not always easy.

Other works discuss serious games for teaching Software Testing [3, 22] and Software Engineering [15]. Valle et al. [22] evaluates the *Testing Game* using the *AIMED* method. The 2D game consists of levels that correspond to functional, structural, and fault-based testing techniques. Each level comprises stages corresponding to the criteria of each related technique. Beppe et al. [3] introduce *GreaTest*, a non-digital card game, in which players play the role of test analysts, and they determine the best software testing technique to run in a usage scenario. Another non-digital game, but focused on teaching the software process, is the *ProcSoft* [15]. In this board game, the goal is to teach concepts related to the definition, structure, and content of a software process that follow good software engineering practices. According to the authors, the game enables the students to understand what activities are common to the software development process and what are the relationships between the existing roles and supporting tools they can use.

Other works discuss different teaching methods [14, 17, 23]. Martinez [14] uses the *Just In Time Teaching* method to teach software testing. In this approach, the professor provides the material and a questionnaire on the topics of the lesson. Before the class, the professor analyzes the answers, and he/she works on the difficulties found based on the students' responses. He/She uses this strategy for Masters students who already have a higher degree of maturity and more motivation compared to undergraduate students. Paiva and Carvalho [17] present an approach to teaching software development following *Scrum* practices. Van Deursen et al. [23] use a collaborative approach to teaching Software Architecture in which students "adopt" an open-source system. Each week, they apply and evaluate architectural theories and present the concepts used in their systems.

This paper aims to address teaching software testing from a different perspective. The focus of this article is not to propose a new teaching methodology, but to try to develop an environment that can bring the content addressed in academia closer to the needs shown in the industry. Thus, after finishing their course, we encourage the students to have training sessions which encompass both theoretical/critical views, expected of higher education courses, and the technical/practical alignment, expected by the industry when seeking qualified professionals to deal with new challenges in the market. Moreover, we discuss the adaptations made, the main benefits and challenges that exist in the process of adopting a PBL active learning method, specific to the context of teaching software testing. There are inherent challenges in teaching software testing that we need to investigate/solve to reach the maturity of an appropriate educational pedagogical model. We believe that these challenges can only be pointed out and discussed through case studies that present experience reports showing the adaptations made, the positive points, and the issues that still need to be faced.

3 DREAMS AND CHALLENGES WHEN USING AN INDUSTRY CASE STUDY IN PBL

As previously described in Section 1, one of the major foundations for the correct application of PBL is using a real project that will be the studied object during the course. We sought partners in the industry to address this topic and try to approximate the concepts taught in class with the market demands. However, we experienced obstacles that made this approach unfeasible. The following subsections describe the initial planning accomplished - the dreams - and the challenges we faced.

3.1 The dreams

To achieve the proposed educational objectives for the courses, we established that all students would deal with the same project, a

real problem, experienced by a company, that would be the case study used throughout the semester. The students would use the project throughout the whole course to cover the syllabus aiming to understand how the contents addressed within the classroom are complementary to each other, not preclusive, as some people tend to evaluate in principle.

According to Richardson et al. [21], the success of using a project-based approach is directly related to the problem presented. Considering this, and in order to further motivate students, the initial planning included using centralized rail network and delivery control software built with service-oriented architecture and using the *C#* and *Visual Basic* programming languages. Since it is a critical system that demands high quality, this was an ideal problem and would show the importance of applying software testing techniques, as well as carrying out the test in all its phases: unit, integration and system [19].

The company was very receptive and interested in this collaboration, mainly because it was developing a critical software and, despite having a testing team, only unit and system testing were carried out in an unsystematic manner. As a consequence, the company was not able to measure how satisfactory the tests performed by them were. The scenario for the partnership seemed to be a match: academia saw a fertile ground to apply its concepts in the industry, motivate students to offer solutions that would actually be used and, thus, be able to train professionals to ensure the software quality. The industry, on the other hand, saw an opportunity to obtain the state-of-the-art in software testing methods and techniques from academia, aiming to enhance the quality of their product.

3.2 The challenges

Unfortunately, we faced some limitations that prevented us from carrying out this case study. Among them, the following can be mentioned:

- **Programming language:** The rail network management system is developed using *C#* and *Visual Basic* technologies, which makes it difficult to automatically apply structural and defect-based testing criteria and techniques. We did not find any open-source tools that would provide similar features as the standard tools that we currently use in academia and industry such as *Eclemma*, *JaBUTi*, *Major*, *PIT*, among others.
- **Confidentiality:** The company required a confidentiality agreement concerning the information provided. Some services had confidential information from partners. Although this is common practice, it would be impracticable to make a contract for each of the students. An option would be not to have access to the source code, but this would prevent us from using structural and fault-based testing.
- **System complexity:** Although the system is critical and rather interesting, it would take too long for students to understand the system as a whole.
- **Agenda unavailability:** the company has its own deadlines that make it impracticable to devote attention to meeting the deadlines of the course.
- **Absence of the person in charge:** The leader responsible for providing us with support in the courses had to be absent for a long period of time.

Due to these limitations, we decided to move forward and use open-source projects instead of the partnership initially discussed with the company. Moreover, the challenges listed above would take too long to solve, and we could not resolve them in one semester. However, to not de-characterize the initial proposal, we decided to choose projects that would develop the skills needed for the students, and thus provide them with the necessary knowledge to perform the test analyst activities. We discuss these projects in more detail in the next section.

4 THE PROJECTS

We chose the projects to simulate aspects and characteristics that students tend to deal with in real situations. Thus, we were aiming to adapt the learning process focusing on the essential knowledge and skills needed to tackle authentic and complex issues derived from a real problem. Additionally, each of the courses reported in this paper used a different open-source project as an object of study. The Computer Science course, at the Universidade de São Paulo (USP), adopted the *JabRef* project and the Information Systems course, at the Universidade Federal de Juiz de Fora (UFJF), used the *FullTeaching* project.

The project selection considered the following criteria: (i) active repository, (ii) presence of documentation, (iii) possibility of using consolidated technologies in the software testing domain, and (iv) having minimum complexity requirements. Besides, the professors' familiarity with previous projects and experiences influenced the individual decision of each professor, resulting in the choice of a different project in each course.

4.1 JabRef

JabRef is an open-source *BibTeX* reference manager written in *Java*. It can be used to structure and manage a large BibTeX/BibLaTeX database and is commonly used help users to write scientific papers. The system has been developed since 2003 and is a project that remains active up to date. We can verify the fact that the project is currently under development by the frequency of new pull requests to be added to the project daily. By 01/01/2019 the project had over 135,000 lines of code, 95 contributors and 31 releases.

The main features of *JabRef* are to enable the user to create, maintain and export bibliographic references. In summary, the principal responsibilities and capabilities of the system are:

- to enable the user to build bibliographies so that they can retrieve information from scientific studies from specialized databases;
- to facilitate the user to add, edit and delete entries in the bibliography database;
- to allow the user to export the bibliography in *BibTex/BibLaTeX* formats;
- to provide multiple language support; and
- to provide support to multiple platforms (*Linux*, *Windows* and *Mac*).

The tool's source code repository can be accessed at https://github.com/JabRef/jabref.

4.2 FullTeaching

FullTeaching is an educational web tool. One of the main reasons for choosing it was because it contains standard features found in students daily routines, such as professors, students, and course management. For each course, we can register a class schedule, the corresponding topics of each class, and the material to be made available. The tool also provides discussion forums and online classes through video conferences.

The system was developed using the *Java* and *JavaScript* languages and the *Angular* framework. The repository source code is available at https://github.com/elastest/full-teaching.

5 BACKGROUND OF COURSES AND EDUCATIONAL OBJECTIVES

Both courses have a workload of 4 hours per week, totaling 60 hours in the semester, and they are offered regularly at the undergraduate level. The third author has several years of experience as a professor of the SSC0721 – Software Testing and Inspection course, however in previous years, he followed the traditional model in which he taught theoretical content using slides. He assigned practical activities, such as exercises and projects, to the students as extracurricular activities. The second author, in turn, had been a student of the third author in this course in the past. In previous opportunities, when she offered the DCC0168 - Software Testing course, she based her teaching on the same style and material as the course she had attended. In this context, the first author, who worked as a teaching assistant in SSC0721 – Software Testing and Inspection course, helped both professors to plan the content and use the proposed approach. The syllabi in both courses were practically identical, differing only in one content: while the course at USP offered the topic Software Inspection, the course at UFJF focused on Software Testing Processes. We offered the DCC0168 - Software Testing course during the evenings. It should be mentioned that the students usually work during the day, and therefore had less time to study the previously available material. Because of this difference, we made adaptations to apply it in the DCC0168 - Software Testing course, making the approaches adopted in both courses slightly different. Every week, we introduced new content and recommended complementary material. We then followed that up by presenting the activity to be developed based on the material learnt. Then, the groups applied the concepts in the open-source projects.

In the SSC0721 – Software Testing and Inspection course, the class started with the groups giving an oral presentation about the results obtained in the previous week, besides discussing and sharing their experiences and difficulties faced while doing the activities to solve the proposed problem.

In the DCC0168 - Software Testing course, we divided the weekly classes into one hour of theory and three hours of practical lessons. During the lecture, the second author explained the content in a general way, without going into detail. She had already identified the tools that students could use. Groups could interact with each other to exchange experiences and information, but instead of weekly presentations, the students gave only one presentation comprising all the work done at the end of the course.

The following subsections point out how the courses were conducted according to the six fundamental characteristics listed by Barrows [1] and provide further details on how the approach was adopted and the main differences between both courses.

5.1 Classes

The SSC0721 – Software Testing and Inspection course consisted of 25 students (3 postgraduate and 22 undergraduate students), and one professor and one teaching assistant gave the course. The DCC0168 - Software Testing course consisted of 30 undergraduate students, most of them from the Information Systems course, and one professor led it.

5.2 Group formation

In both classes, the groups consisted of five members, but the composition was performed differently. At USP, we chose the students through a selection process to form groups with similar characteristics and homogeneous knowledge. During the selection process, we took into consideration their main technical aspects. At UFJF, we left the students to group themselves due to the difficulties to perform the extra-class assignments. Thus, these groups were formed considering affinity among the members.

5.3 Educational objectives

Both courses aim to provide an opportunity for the students to learn and experience the concepts of software validation, verification, and testing. By using Bloom [5] taxonomy, the educational objectives of the courses can be identified at three levels:

(1) At the **knowledge** level, the course enables students to gain access to crucial knowledge in the V&V domain.
(2) At the **application** level, the course enables students to apply their knowledge gained from the previous level in real projects. Thus, they can experience in practice the main difficulties that we can encounter when using theoretical knowledge.
(3) At the **assessment** level, the course enables students to evaluate and discuss solutions adopted by other groups. Thus, they can understand the different perspectives that can be used. The experience reported by each group can also contribute to homogenizing knowledge.

5.4 Syllabus

Both courses follow the "Software Testing Techniques [2]", "The Art of Software Testing [16]", and "Introdução ao Teste de Software [8]" as basic textbooks. The DCC0168 - Software Testing course also used the book "Software Testing and Analysis-Process, Principles, and Techniques [18]". The syllabi of the two courses aimed to provide an overview of software Verification, Validation, and Testing - VV&T concentrating on software testing strategies, techniques, criteria, and associated tools that we use in building software.

Among the topics covered during the semester that are common to both courses, we highlight:

- Errors in the software development life cycle;
- Terminology and basic testing concepts;
- Test phases;

- Software testing techniques;
  - Functional testing;
  - Structural testing; and
  - Defect-based testing.
- Automatic test data generation; and
- Regression testing.

The SSC0721 – Software Testing and Inspection course also addressed the topic "Verification and validation strategies: static analysis, inspections and tools for static analysis". DCC0168 - Software Testing course covered the topic "Software Testing Processes".

5.5 Course planning

In both courses, the main activities for the students were (1) to understand the complementary material related to the topic covered during the week, and (2) to apply the theoretical content proposed during the class to solve the problem. At USP, students also had to report their activities, highlighting results and difficulties; and present the results through a *pitch* at the beginning of of the following class. In the UFJF course, the students prepared only one report, in a paper format, delivered at the end of the course. This report aimed to explain the entire test process adopted and the tools used in each test phase by the students. They also gave only one presentation at the end of the course so that they could make better use of class time to develop the work.

5.5.1 Complementary material. In addition to the material given in the classroom, for each topic covered, the professors and the teaching assistant provided students with a set of supplementary material, which consisted of complementary book chapters, scientific papers, and video lessons presenting practical applications of the content covered. Moreover, at USP the students had 3 hours per week with the teaching assistant to ask questions and resolve doubts concerning the groups' practical activities. At UFJF, the students had 2 more hours per week, which were made available by the professor.

5.5.2 Application of theoretical content. At USP, students applied the concepts learned to solve the project after studying the complementary material. At UFJF, the professor used the first hour of the four hours per week to explain the concepts to the students without going into detail. As the class was offered on two days per week, the professor encouraged the students to study the material before the second class. Using more class time to solve the project helped students experience the difficulties that existed when dealing with real challenges, which is different from when dealing with an ideal controlled scenario. These strategies encouraged students to think of creative solutions to the problems encountered when applying the theoretical content.

5.5.3 Activity Report. At USP, each group had to present a report per week including the views and perspectives of the group members regarding the theme addressed, the results obtained in applying the content in the proposed project, and highlighting the main difficulties, solutions, and learning achieved to develop the given activity. An important point to mention is that for each report, the groups should specify the percentage of participation of members during the activities of that week. We used this information to give each group member a grade based on the overall grade obtained.

At UFJF, each group wrote a report, using a paper format, and delivered it at the end of the course. The report described the testing process adopted, the techniques, criteria, and tools used to test the application considering each of the testing phases. To individualize the participants' grades, each student peer reviewed another groups paper. In the review, students were encouraged to discuss the concepts presented, and so that they would not feel they were hindering another group, the group grade was taken before.

5.5.4 Presentation of results. During the oral presentation, we extended the discussion so that the groups could exchange experiences about the challenges they had faced. We also used this presentation to correlate the difficulties with the theme that they addressed.

5.6 Evaluation

We evaluated the group at USP according to the following criteria:

(1) quality of the results presented in the activity report;
(2) ability to argue the results presented during the oral presentation; and
(3) members' participation percentage while doing the activities.

After we set the group's overall grade, each group member received their individual grade based on their percentage of participation while doing the activities every week. The students' final grade corresponds to the average of the grade obtained during each content iteration.

UFJF used the following evaluation criteria:

(1) quality of the results presented during the course;
(2) ability to argue the results presented during the oral presentation;
(3) paper written by the group;
(4) individual review of a paper from another group;

For the individual grade referring to the paper review, the following scores were considered: (i) up to 20 points for correctly identifying test-related problems in the paper; (ii) up to 30 points for knowledge of testing techniques/processes when evaluating the paper; (iii) up to 30 points regarding suggestions for improvements related to the tests; and up to 20 points for the review as a whole. Each student's final grade corresponded to the simple arithmetic average of each of the items listed.

5.7 Teaching the courses

According to the activities described in Subsection 5.5, the groups went through activities that included studying the available material and using the content taught in the project they were working on.

Table 1 describes a summary containing part of the activities performed during the semester, as well as the tools used to assist in the practical application.

Regarding the heterogeneity of students, it is noteworthy that the software testing course is usually offered in the last periods of the undergraduate course and; therefore, it is expected that students have much of the knowledge needed to study the course successfully. However, courses that have students with mixed ability may follow the process adopted at USP, where some students from the

course (leaders) formed groups by analyzing resumes previously reported by their peers. The presence of the teacher assistant also contributed to mitigating the problem of heterogeneity.

6 ASSESSMENT AND DISCUSSIONS

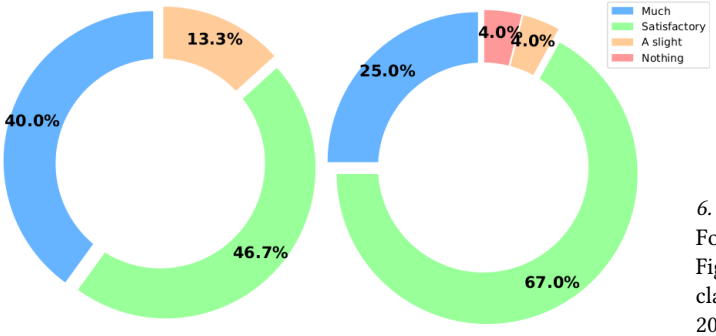
We evaluated the approach from both students' and professors' points of view. For the students' evaluation, we sent them questionnaires, and they answer them voluntarily and anonymously. A total of 39 students answered the questionnaires, 15 students from the DCC0168 - Software Testing course, taught at UFJF, and 24 students from the SSC0721 – Software Testing and Inspection course, taught at USP. The evaluation from the professors' point of view sought to show the initial and final impressions regarding the performance of the classes at the end of the courses.

6.1 Students' perspectives

An important point in assessing success in a teaching approach is to look at students' perceptions of the results obtained during the semester. Thus, this section aims to show the students understanding of both courses regarding their satisfaction with the approach adopted during the semester.

6.1.1 Regarding the degree of acquired content. For the DCC0168 - Software Testing course, according to the pie chart in Figure 1 below, 40% of students showed a substantial (much) increase in the topics of the course, while 46.7% improved partially (satisfactory) improvement, and 13.3% of the students showed a low degree of the content acquired in the course.

For the SSC0721 – Software Testing and Inspection course, 25% of students showed substantial improvement in their knowledge, 67% partial (satisfactory) growth, and 4% a slight and no increase regarding the degree of content acquired.



(a) UFJF (b) USP

Figure 1: The degree of acquired content proposed in the syllabus

Analyzing the information from both classes overall, it can be observed that most students show a partial improvement in the degree of content acquired, followed by a substantial increase. It is important to note that the percentage of students who considered having had a slight improvement plus the percentage of students who said they had no improvement is much lower compared to the positive proportion of learning. Some factors may have contributed to cases of a small degree of learning improvement, as pointed out by the students themselves (in a free translation):

> "I liked having to use tools, but I think we should have simple exercises without using them because some tools encapsulate the theoretical content and make it impossible or difficult to learn it.
> *Anonymous*"

> "To cover the concepts in a broader way and give (simple) examples during the class for a better understanding of the subject so that it can then be applied in practice throughout.
> *Anonymous*"

The students' comments show a need between balancing how to define a project that is capable of both posing challenges for students, in order to prepare them for activities that they will face in the industry, as well as the need to not exceed the level of difficulty so that it does not hinder their learning process.

This reflection meets the desire and opinion of other students about the project's degree of difficulty to be adopted in courses that use the PBL practice (in a free translation):

> "I believe that using a more basic repository, with a small, simple-to-understand program, with some primary test cases, would be better for student learning.
> *Anonymous*"

6.1.2 How do you rate the methodology used during the classes? For the course taught at UFJF, according to the graph shown in Figure 2, 46.7% of students stated that the methodology used in the classroom was excellent, 33.3% said it was a good methodology, and 20% of the students rated it as a poor methodology to be used in the classroom.

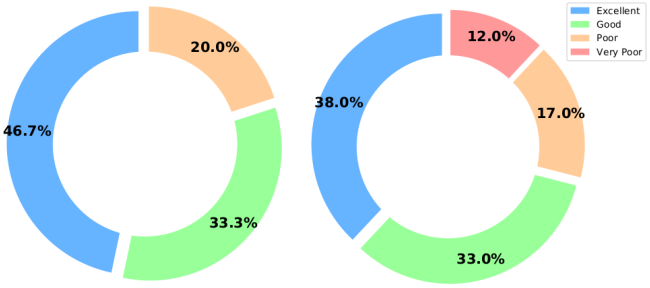
For the course taught at USP, 38% considered it to be an excellent approach, 33% thought it was a good approach, 17% viewed it as a poor approach, and 12% considered as a very poor approach to use in the classroom.

Analyzing the results as a whole, it can be concluded that most students see the use of new teaching methodologies as positive. However, it can be observed that this is a process that still needs to be further investigated to find a more suitable model.

Among students' opinions regarding the use of active methodologies, the following can be highlighted, for instance (in a free translation):

Table 1: Sample content and technologies covered during the semester.

Course units	Summary	Tools used
1. Code metrics & Static analysis	This step aims to enable the students to analyze the state of the project to define guidelines that can orient the software testing techniques activity.	*PMD & SpotBugs (FindBugs)*
2. Software inspection	By obtaining information regarding code metrics, inspection activities can be done, for example, to detect defects by reading the artifact being reviewed.	Activity performed manually
3. Structural Testing	The idea of structural testing is to evaluate test suits according to their ability to execute specific structures such as commands, deviations, or paths within the program.	*Eclemma, Code Coverage (IntelliJ), SonarQube, JaBUTi.*
4. Mutation Testing	The idea of mutation testing is to evaluate test suits according to their ability to reveal certain types of defects. The idea is that test cases that can reveal these defects are also useful for revealing other types of defects.	*PIT, Major, Proteum*
5. Functional Testing	Produce test cases based on a specification to ensure that the product behaves according to the specification for which it was designed.	*JUnit* (unit testing), *SikuliX* (system testing) *Seleniun WebDriver* (system testing)
6. Automatic Test Case Generation	An automatic test case generation approach makes it possible to reduce the effort of creating test cases significantly. The idea of this activity is to complement the test case sets produced with the other previous approaches.	*EvoSuite, Randoop*



(a) UFJF (b) USP

Figure 2: Opinions about the adequacy of the methodology used in the classroom.

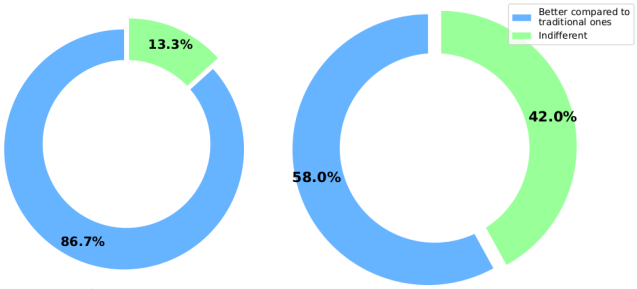
> "I like the methodology to be focused on the practical part, but it lacked theoretical content during the classes. I think we should have simpler examples and exercises that complement and consolidate the theory.
> *Anonymous*"

> "I believe the PBL-based methodology is the right way for courses like this; however, in my opinion, I also believe that it could have been better applied.
> *Anonymous*"

These comments show a balance between the use of a project-focused approach and a traditional teaching approach, as well as balancing the content difficulty level to be proposed to the students. Perhaps merging the application of the theoretical concepts into the classroom together with more straightforward problems and advancing the difficulty level of the problems as students advance in content, is the way forward.

6.1.3 How do you assess the valuation model used? Figure 3 presents the data regarding the assessment model used in the courses. For the course taught at UFJF, 86.7% of the students consider the assessment method adequate, and they consider it better than the traditional assessment model (exams), while 13.3% find the method indifferent, having no opinion between the conventional model and the model reported in this study.

For the course taught at USP, 58% of the students consider the assessment method adopted better than the traditional model, while 42% found themselves indifferent to this question.



(a) UFJF (b) USP

Figure 3: Opinions regarding the assessment model.

> "The weekly assessment without tests is good, but the projects are sometimes too long for a short time. I like the method based on problems/examples, but in the case of the course, with clearer guidelines for each project, so that we take less time than what was expected.
> *Anonymous*"

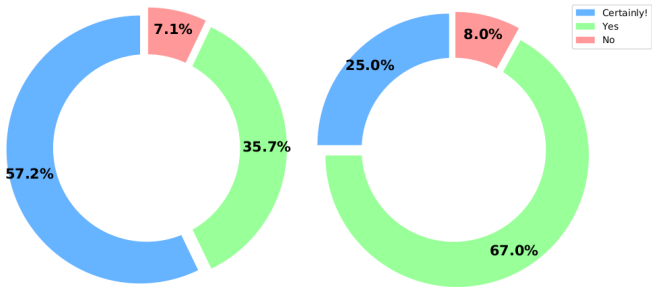
Analyzing the overall results, we can observe that most students were very open to new assessment mechanisms since the adopted model allows a more transparent reflection on the knowledge produced and steers away from the traditional model. The

last, in turn, due to assessment charges, can sometimes intimidate students, which may lead them to performing poorly [9].

6.1.4 Would you enroll in other courses with the same professor? Another interesting factor to look at is how students perceive the professor within the classroom. To investigate this perception, we asked the students if they would enroll in other courses with the same professor.

For the course taught at UFJF, as shown in Figure 4, 57.1% of students reported that they would certainly take another class with the same professor, while 35.7% said yes and only 7.1% informed that they would not enroll in another course.

For the course given at USP, 25% of the students reported that they would certainly take another course with the professor, 67% said "yes" they would take another class with the professor, and 8% said they would not enroll in another course.



(a) UFJF (b) USP

Figure 4: Would you enroll in other courses with the same professor?

6.1.5 Does having a teacher assistant help with learning? The last point analyzed is related to the presence of a Ph.D. student playing the role of teaching assistant in the classroom, helping to resolve students' doubts and providing weekly times for answering various questions. Regarding this topic, the only course evaluated was the one offered at USP, as there is no similar initiative at UFJF.

According to Figure 5, for 75% of the students interviewed, the presence of a teaching assistant helped in the learning process, 21% consider that the presence of the teaching assistant helped slightly in the learning process, and 4% believe that the presence of the teacher assistant did not support the learning process.

It is essential to highlight that the teaching assistant's activities are, among others:

- collaborate in the review of the general course planning;
- to help prepare a schedule of activities to support the classes;
- to help plan, develop and reformulate (if required) the assessment mechanisms of the taught contents, including projects and exams;
- to participate in the classes to help with discussions between the professor and students;
- to grade exercises and projects developed by the students (we discussed the criteria to grade previously, and the professor reviewed them once that had finished); and
- to offer the students help after class. This was done in places, on days and at times scheduled beforehand.

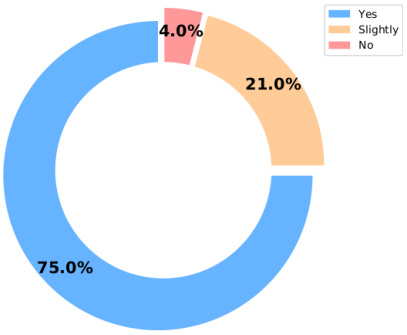


Figure 5: Does having an assistant teacher help with learning?

Among the students' opinions regarding the participation of the teaching assistant, we highlight the following:

> "The teaching assistant was extremely fundamental, and often helped more than the professor. I have no complaints.
> *Anonymous*"

> "The teaching assistant helped to bridge the gap between student and professor that exists traditionally. The advantage of having an intermediary helps when we ask doubts, which in many cases, seem to be quite "stupid" to be asked in class.
> *Anonymous*"

Based on the results of this topic, it can be observed that the presence of a teaching assistant in the classroom is a determining factor for the proper conduct of a course, serving primarily as a bridge to facilitate students' understanding of how a technical issue is resolved, which tends to occur while developing practical activities.

6.2 Professor's perspective

Based on our previous experiences adopting a teacher-centered approach, we detected the strengths and weaknesses, the problems and the limitations of applying the PBL approach.

The main advantage of this approach is that it actually engaged students more. We noticed that they were much more participative as they were asking questions and they were motivated during the class, compared to previous classes. We believe that their curiosity to find the solution to a problem and even to realize that they could discover defects not yet identified by the open-source community encouraged them to increasingly learn the other testing techniques to "break the code". Several times, they wondered how they could solve a particular problem that we would deal with in later classes.

Although the participation of working students influenced the adaptation of the approach to suit their realities, these same students contributed to making the classroom discussions more productive. They often reported how the testing process at their company was and were excited when they knew about a particular tool they could apply to their work. Furthermore, some students brought the tools they used in their jobs to class.

The results obtained from adopting the approach corroborate with the results obtained by Richardson et al. [21][21], who states that the success of PBL depends on the choice of a good problem. However, the experience presented proved to be a more significant challenge than imagined, since initially, we expected to conduct the course in partnership with a company, which became infeasible at first due to the problems listed in Subsection 3.2. In any case, shifting the perspective from being a company problem-driven approach to an open-source problem-driven approach proved to be a wise decision. In addition, a more complex project could have brought problems while applying the method.

Choosing an appropriate project to work on during the period in a course is still a challenge, and there are no guidelines in the literature to support the application of this task. In this report of the experience, each professor was responsible for selecting and defining the project to be applied. Because the two groups did not have contact with both projects, it is difficult to assess which was the best. In general, *FullTeaching* was very positive because it is a web tool whose domain academic environment is well known by the students. Thus, we noticed that because the students identified with the project, they improved their expected performance. However, giving the responsibility to students to define their projects is a difficult task. The variety of choices they could make would affect the professors' ability to assess the material produced.

Another difficulty encountered when we apply PBL is to know how to dose activities to suit different classes that have diverse student profiles. The ideal scenario would be a model in which activities could be more flexible since it is not always possible to know the students in advance nor to measure in advance how they will react to the problems investigated during the activities. The correct dose is a fundamental factor since the professor still has to cover a wide syllabus by the end of the course.

Regarding student performance, their average grades were better than those of previously offered courses. However, since the assessment model was different, no conclusions can be drawn about this. From the assessment point of view, a more flexible model allows a greater reflection on the content. This scenario is not always possible in a traditional assessment model. Besides that, measuring the assessment process is not an accurate task. One approach we applied was to measure the overall performance of the group and ask the groups of students to grade the individual participation for each activity performed. This aspect makes group members more proactive so that everyone was engaged in developing the proposed activities.

Finally, we consider that the experience has achieved a certain degree of success as it has brought to us a knowledge of successful activities, as well as a better understanding of the unsuccessful activities. Thus, in a future course, we can further refine the positive aspects and improve those activities that did not achieve the best possible outcome when adopting the approach.

7 CONCLUSION AND FUTURE WORKS

This paper has discussed the results and primary challenges faced while using an active learning methodology in the context of software testing teaching in two Brazilian universities. We presented the methodological plan used, the mechanisms adopted while giving the courses, and the main advantages and challenges found while using this teaching methodology.

We addressed issues related to the effectiveness of content adoption using active methodologies instead of traditional teaching methodologies from both students and professors points of view. We obtained the students' points of view by asking them to fill in anonymous and voluntary questionnaires. Among the issues are the content methodological adequacy and the assessment model. Finally, we also investigated the effectiveness of the role of an assistant teacher to provide support on the course, as well as resolving problems after class and in the extra hours offered to students.

Based on the students' answers, we observed that in both courses evaluated, the students positively perceived including an active methodology in the learning process. We believe that it is because this methodology approximates the students to more technical knowledge, which is, in turn, a requirement asked for by the industry. Besides, we also observed that the use of active methodologies tends to demand more from the student during the extra-class period. Since this can lead to different impacts depending on the profile of the students, we should take this into account during the course design.

Our initial proposal was to set up a partnership with the industry, but due to the limitations described during the paper, we decided to leave this for future research. To face this problem, we decided to adopt open-source projects. We consider it was the right decision as it allowed us to solve the issues and not give too many complex projects to the students.

The results obtained from the two different contexts enrich the discussions and conclusions obtained. We can highlight that scalability is impaired when weekly meetings are held to discuss results, as we found at USP due to limited class time. On the other hand, the adaptation made at UFJF, which only had one presentation at the end of the course, is easily scalable for classes with more significant numbers of students. However, the students were not aware of the development carried out by their peers in other groups, which was detrimental to their exchange of ideas. We believe that these findings indicate the need for further studies to strike a balance between student numbers, frequency, as well as the presentation format. The insights we gained are the following (i) there should be less presentations for classes with many students; (ii) courses with an average number of students may choose to keep weekly presentations more dynamic or have less presentations.

Despite different students, universities, and laboratory structures, the main challenges observed at both universities and the two courses faced the same problems in terms of using PBL adapted for Software Testing. Moreover, the results presented do not exactly corroborate with those described in the literature: the approach is not so effective for students doing an evening course, who have less time to do extra classes. Therefore, the results of the paper indicate that the success of adopting an active approach is not directly linked to infrastructure aspects.

Concerning the project selection, despite the fact that each course used a different project, the students had similar difficulties. We observed that these difficulties are not directly related to the size/complexity of the project nor the application domain. It highlights that we need to investigate an approach that can balance the use of real problems with the demand required so as not to stifle students with problems outside the scope, such as more reports and presentations.

As future work, we plan to create mechanisms to balance the choice of open source projects better, to equalize students' workload, since, although it is an excellent approach to develop students' technical skills, this type of approach requires extra-classes. In the case of evening courses, whereby most students work full time, this approach is inappropriate because the lack of extra-class time is detrimental not only in terms of doing the assignments, but also hinders communication between the members of the groups.

We also plan to work on the challenges encountered in the course in partnership with a private company. We hope that in a future course, we will be able to develop such a partnership so as to increasingly approach the needs found in the job market with the pedagogical project proposed by the universities to the students' wishes.

Finally, we understand that the experiences gained by traditional teaching techniques, the experiences reported in this work, as well as other skills developed at university, can help find a teaching model that meets the needs of all the actors involved in the process. Thus, we will be able to qualify skilled labor in the field of software testing.

ACKNOWLEDGEMENTS

This study was financed by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Stevão Andrade's research was funded by FAPESP (São Paulo Research Foundation), process number 2017/19492-1.

REFERENCES

[1] Howard S. Barrows. 1996. Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning* 1996, 68 (1996), 3–12.
[2] B. Beizer. 2003. *Software Testing Techniques.* Dreamtech.
[3] Thiago A. Beppe, Ítalo Linhares de Araújo, Bruno Sabóia Aragão, Ismayle de Sousa Santos, Davi Ximenes, and Rossana M. Castro Andrade. 2018. GreaTest: A Card Game to Motivate the Software Testing Learning. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering (SBES '18).* Sao Carlos, Brazil, 298–307.
[4] Jacob Lowell Bishop and Matthew A Verleger. 2013. The flipped classroom: A survey of the research. In *120th American Society for Engineering Education*, Vol. 30. 1–18.
[5] Benjamin S Bloom et al. 1956. Taxonomy of educational objectives. Vol. 1: Cognitive domain. *New York: McKay* 1, 1 (1956), 20–24.
[6] Lyn Brodie, Hong Zhou, and Anthony Gibbons. 2008. Steps in developing an advanced software engineering course using problem based learning. *Engineering Education* 3, 1 (2008), 2–12.
[7] Jean Felipe P. Cheiran, Elder de M. Rodrigues, Ewerson Luiz de S. Carvalho, and João Pablo S. da Silva. 2017. Problem-Based Learning to Align Theory and Practice in Software Testing Teaching. In *Proceedings of the 31st Brazilian Symposium on Software Engineering (SBES'17).* Fortaleza, CE, Brazil, 328–337.
[8] M. Delamaro, M. Jino, and J. Maldonado. 2017. *Introdução ao Teste de Software.* Elsevier Editora Ltda.
[9] Stephen H Edwards. 2003. Rethinking computer science education from a test-first perspective. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.* ACM, Anaheim, California, USA, 148–155.
[10] Cindy E. Hmelo-Silver. 2004. Problem-Based Learning: What and How Do Students Learn? *Educational Psychology Review* 16, 3 (01 Sep 2004), 235–266.
[11] Daniel E. Krutz, Samuel A. Malachowsky, and Thomas Reichlmayr. 2014. Using a Real World Project in a Software Testing Course. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14).* Atlanta, Georgia, USA, 49–54.
[12] Ling Liang. 2018. *Investigating the skills and capabilities that software testers need: A New Zealand study.* Master's thesis. School of Engineering, Computer and Mathematical Sciences, New Zealand.
[13] Yang Lu. 2017. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration* 6 (2017), 1–10.
[14] Alexandra Martinez. 2018. Use of JiTT in a Graduate Software Testing Course: An Experience Report. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '18).* Gothenburg, Sweden, 108–115.
[15] Victor Moura and Gleison Santos. 2018. ProcSoft: A Board Game to Teach Software Processes Based on ISO/IEC 29110 Standard. In *Proceedings of the 17th Brazilian Symposium on Software Quality (SBQS).* Curitiba, Brazil, 363–372.
[16] G.J. Myers, C. Sandler, and T. Badgett. 2011. *The Art of Software Testing.* Wiley.
[17] Sofia Costa Paiva and Dárlinton Barbosa Feres Carvalho. 2018. Software Creation Workshop: A Capstone Course for Business-oriented Software Engineering Teaching. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering (SBES '18).* Sao Carlos, Brazil, 280–288.
[18] M. Pezzè and M. Young. 2008. *Software Testing and Analysis: Process, Principles, and Techniques.* Wiley India Pvt. Limited.
[19] R.S. Pressman and B.R. Maxim. 2016. *Software Engineering: A Practitioner's Approach (Eighth Edition, English Abridgement).* Ji xie gong ye chu ban she.
[20] I. Richardson and Y. Delaney. 2009. Problem Based Learning in the Software Engineering Classroom. In *2009 22nd Conference on Software Engineering Education and Training.* Munich - Germany,, 174–181.
[21] I. Richardson, L. Reid, S. B. Seidman, B. Pattinson, and Y. Delaney. 2011. Educating software engineers of the future: Software quality research through problem-based learning. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE T).* Klagenfurt - Austria, 91–100.
[22] Pedro Henrique Dias Valle, Rafaela Vilela Rocha, and José Carlos Maldonado. 2017. Testing Game: An Educational Game to Support Software Testing Education. In *Proceedings of the 31st Brazilian Symposium on Software Engineering (SBES'17).* Fortaleza, CE, Brazil, 289–298.
[23] Arie Van Deursen, Maurício Aniche, Joop Aué, Rogier Slag, Michael De Jong, Alex Nederlof, and Eric Bouwers. 2017. A Collaborative Approach to Teaching Software Architecture. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17).* Seattle, Washington, USA, 591–596.