Enhancing the Student Learning Experience by Adopting TDD and BDD in Course Projects

Fabio Gomes Rocha, Layse Santos Souza, Thiciane Suely Silva and Guillermo Rodríguez

2021 IEEE Global Engineering Education Conference (EDUCON) | 978-1-7281-8478-4/20/$31.00 ©2021 IEEE | DOI: 10.1109/EDUCON46332.2021.9453916

***Abstract*—*Contribution:* A demonstration of the application and contribution of Test-Driven Development (TDD) and Behavior-Driven Development (BDD) in the student learning experience in the context of a Software Engineering course. *Background:* Software testing is an activity for ensuring software quality. Although teaching testing rigorously to students is a priority in academia, undergraduate students often encounter difficulties performing testing tasks effectively. *Intended Outcomes:* To increase satisfaction rate and course grades, and reduce delivery time. *Application Design:* We experimented with the Software Engineering Laboratory (LES) course of a Private University in the Bachelor of Computer Science and Information Systems courses. This experiment corroborated the learning difficulties of students. Collected data were assessed both quantitatively and qualitatively. *Findings:* Backed up with statistical tests, the results showed a reduction in student absences, higher student satisfaction rate, and higher grades in the courses. Furthermore, our approach allowed students to deliver a product in a short period, representing a possibility of adoption of BDD due to their successful learning experience. Finally, we aim to foster a discussion of appropriate teaching methods of software testing.**

***Index Terms*—Software Testing, Student Learning Experience, Software Engineering Education, course Project, Agile Software Development.**

I. Introduction

THE teaching of Software Engineering has been a widely discussed issue; Dawson [1] and Regev et al. [2] have stated that projects used in teaching move away from reality, thus forming students with limited preparation for their professional future.

According to Lethbridge et al. [3], companies sometimes have to supplement students' knowledge with training to provide necessary skills for software development demanded by the industry. According to Yamaguti et al. [4], nowadays, there is a significant challenge in the teaching of Software Engineering in Brazil, due to a demand for well-qualified professionals to develop software efficiently.

According to Yamaguti et al. [4] innovative approaches should be adopted to enable the integration of content through interdisciplinarity. Some research has evidence the need to teach software development, by simulating a professional context [5], [6].

Furthermore, universities have begun to incorporate the teaching of agile methods into the academic training of software engineering (SE) students [7]. There is a growing need to introduce students to agile methods. Devedzic and Milenkovic described their eight years of experiences of teaching agile software methodologies to various groups of students at different universities by combining Scrum and extreme programming (XP) [8]. Rico reported a course course based on agile methods, in which students were able to choose an agile method out of Scrum, XP, feature-driven development (FDD), the dynamic systems development method (DSDM) and Crystal Clear, and more [9].

In the pedagogical approach reported in [10], the authors proposed a hybrid Information Systems (IS course) based on XP, Scrum, and FFD. Mahnic introduced Scrum in a software engineering course as a framework for planning and managing student projects, focusing on the students' learning of Scrum [11]. A previous study revealed that students enjoyed working with Scrum and had a successful hands-on experience [12].

By simulating a software engineering environment as close to the real-world as possible, Scharf et al. presented a Scrum undergraduate course complemented with useful tools to implement this agile method [13]. Similarly, Paasivaara et al. introduced a simulation game to teach Scrum roles, events, and related concepts and achieving promising results according to student feedback [14]. A more detailed overview of the use of Scrum in software engineering education can be found in [15].

A typical problem of the aforementioned approaches can be the lack of assistance to students as they perform agile practices. Students might not have adequate skills, and even when working in an agile context, they may tend to write lengthy requirements documents, engage in waterfall-like issues, follow a plan instead of responding to change, and focus on delivery dates instead of product quality. Students must be helped to understand Scrum rules, to overcome programming obstacles, and to tackle non-technical issues such as management and teamwork. Moreover, software testing is an essential skill for software developers; however, this component has been subtly undermined in the computer science curriculum [16].

In this work, we aim to analyze the application and the contribution of the use of Test Oriented Development (TDD) and Behavior Oriented Development (BDD) in the process of teaching Software Engineering in the Bachelor of Information Systems and Computer Science courses in the Software Engineering Laboratory (SEL) discipline in the fourth period (second year) for a software development task, to meet the changing requirements during a course project development.

To experiment, 35 students from the Bachelor of Information Systems and Computer Science courses were divided into 5 groups detailed better in section V. Each group had as a task

F. Rocha, L. Souza and T. Silva are with GPITIC, Tiradentes University, Aracaju, Brasil, e-mail:gomesrocha@gmail.com, laysesantossouza@gmail.com, thicianecouto@gmail.com

G. Rodriguez is with ISISTAN (UNICEN-CONICET) Research Institute. e-mail: gmrodrigue2@uade.edu.ar

978-1-7281-8478-4/21/$31.00 ©2021 IEEE

the development of a product, which should be developed in eight weeks. The groups used BDD or BDD Partial (for some functionalities was used the BDD), or TDD, or did not use any of the agile practices. The results have evidenced that the use of BDD improves the student's software development capacity, as well as their learning experience. After several iterations, we observed a lower number of failures in the deliveries. Furthermore, there was a reduction in the number of absences of students, higher student satisfaction rate, and higher grades in the course.

The rest of the article is organized as follows: Section II presents the theoretical foundation of the agile practices adopted in the experiment. Section III details the related work. Section IV described the methodology applied to conduct the study, the planning, and the results obtained. Section V describes the experience report. Section VI presents the discussion of the results. Our lessons learned are drawn in Section VII. Section VIII concludes our work with final considerations.

II. Agile practices

According to Cohn [17], agile practices are essential for the delivery of products systematically and incrementally. Thus, agile teams have been successfully using consolidated practices, such as Test-Driven Development (TDD), refactoring, collective ownership, continuous integration, continuous delivery, pair programming, and also Behavior-Driven Development (BDD) [18].

A. Test-Driven Development (TDD)

Test-Driven Development (TDD) is a programming technique that aims to deliver a quality product in a short period. Usually adopted in agile contexts, TDD aims to prevent possible errors and to facilitate the understanding of the system requirements [19]. Idealized by Kent Beck as part of the XP Methodology, TDD is an XP practice to analyze and test that allows for refactoring and continuous integration.

According to Bissi et al. [20], TDD has been gaining popularity beyond XP; besides, the authors point to a significant increase in quality when using this practice in software development. TDD is the most used agile practice in the software development sector due to the simplicity of its life cycle [20]. TDD starts with the creation of a failed unit test, aiming to ensure that the test works and captures errors. In this way, the focus is on the developer knowing the problem so that he can implement a functional solution that corrects. Later, this solution evolves through refactoring.

Besides, according to Jazen and Saiedian [21], TDD is a strategy that requires the writing of automated tests before the development of the functional code and its iterations to verify if they are correct, complete and what their level of quality is. Despite appearing as part of the XP method, TDD has received attention as an independent practice, being adopted by other methods and having several tools for TDD support in various programming languages. Numerous books have been written defining its concept. Moreover, researchers and educators began to analyze the effects of TDD in the reduction of defects and improvements in the quality of academic and professional environments, and in the integration of this strategy in the courses of Bachelor of Information Systems, Computer Science and Software Engineering.

According to D. North [22], the use of TDD generated some questions, such as: What would be the initial tests? What should be tested? How to understand the failure of a test? Such questions ended up leading to the appearance of doubts and, consequently, impacting software development. To solve this problem, North presents Behavior-Driven Development (BDD), which is based on TDD and Acceptance Test-Driven Development (ATDD).

B. Behavior-Driven Development (BDD)

The BDD allows extracting specifications provided by the client during the research of the needs. Thus, it is possible to say that the TDD has evolved into the BDD with the following questions: Where to start? What to test? What not to test? When to test? How to name the tests? Why does a test fail? In his study, North highlights that the BDD differs from other approaches in describing a system's behavior from the perspective of its stakeholders at all levels of granularity [22], thus ensuring a focus on describing the behavior of the system and provide better communication. Besides, Smart [23] states that the BDD has a broader view of agile analysis, automated testing, and consequently assisting teams in building, delivering value, and software quality. In the study of Solis and Wang [24], the BDD is described as a combination of the following features and agile practices:

- Ubiquitous language: It should be similar to common language, with terms that are easy to understand, so any member of the team can understand and discuss the product;
- Automated acceptance tests: After understanding the acceptance criteria, it is the team's job to transcribe software tests from these criteria;
- Readable code writing: The implementation must express the objective of the BDD scenarios clearly. At the time of implementation, the code (methods, classes, etc.) must indicate its objective, according to the specifications made available in the user stories.

Still, according to Solis and Wang [24], the BDD is a flexible framework that can be used throughout the software development life cycle helping the development team better understand the client's interests. Therefore, it can be highlighted that the great contribution of the BDD is the integration, which allows everyone involved in the project to have access to any aspect of the system (requirement, code, testing) is highlighted that the BDD has benefits such as reducing waste, communication, cost reduction, change more comfortable, safe, and **releases** faster. Moraes reports more advantages of the comic book, for example, communication between teams, overview, knowledge sharing, and dynamic documentation [25].

Smart [23] also explains the BDD cycle, as illustrated in Fig. 1: the cycle is initiated with the meeting between stakeholders and customers for the presentation of those involved and

978-1-7281-8478-4/21/$31.00 ©2021 IEEE

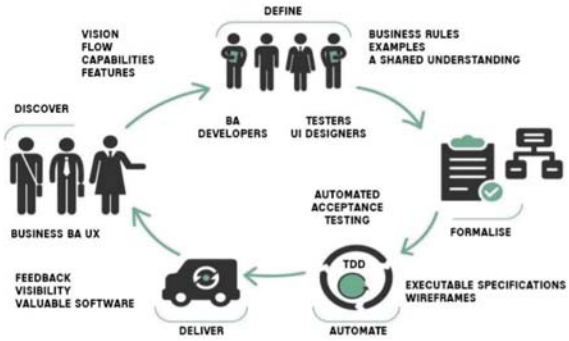

Fig. 1. BDD process according to [23]

discussion of the business overview. After defining the usage scenarios that express the *features* and are defined in the format of *user stories*, with this, the user requirements are represented. In the next step, the scenarios are structured according to the ubiquitous language Gherkin (Given that "context", When "event" and Then "result"). Because the scenarios are jointly defined between stakeholders and customers, the software's function.

After the BDD process is completed, the scenarios must be of acceptable quality to avoid problems with requirements such as documentation failures, inconsistency, and incomplete scenarios. To evaluate the user stories documents' quality, Oliveira and Marczak [26] propose the following *checklist*: concise, estimable, feasible, negotiable, prioritized, small, testable, understandable, unequivocal, and valuable.

Therefore, it can be said that the BDD is a specification technique that links the functional requirements to the source code, connecting the textual representation of requirements to automated tests. Thus, the BDD extends the TDD, since the TDD aims to obtain the correct solution that precisely corresponds to the business problem at the unit and integration levels. Simultaneously, the BDD takes care of the acceptance level, where the requirements reside (Fig. 2).

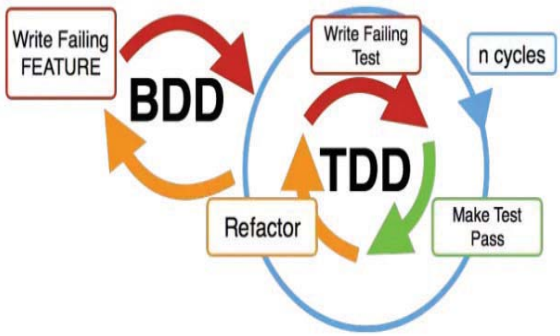

Fig. 2. BDD and TDD integration

III. Related Work

The literature reports several strategies for addressing project-based learning experiences in software engineering courses. Gestwicki and Mcnely defined a Scrum project-oriented model that provided a self-organized and multifunctional structure for teams [27]. The authors pointed out the model as beneficial for the student learning experience.

George and Williams conducted a case study with professional programmers. The authors state that the use of TDD increased programming time by 16% [28]. Along this line, the case studies presented in the works of Nagappan et al. [29] and Canfora et al. [30] presented similar conclusions with the developers of companies such as Soluziona Software Factory, IBM, and Microsoft.

Edwards' work has stated the need to teach software testing skills to computer students, who generally, according to the authors, are poorly prepared for this task [31]. Thus, the authors proposed the use of TDD as a way to require students to test their codes and point out that students produced codes with 45% fewer defects per thousand lines of code when using TDD. Likewise, Eierman and Iversen [32] compared TDD with pair programming, analyzing, by the students' perception, which of these more contributed to the learning of programming. Results showed that both techniques are considered useful; thus, TDD is an essential and healthy practice.

Santos et al. [33] presented TesterDS, a game focused on the teaching of Data Structures that uses techniques of TDD and Structural Software Testing. The game focuses on students from more advanced programming disciplines and aims to facilitate and stimulate the learning of Data Structures.

Seroa et al. [34] proposed a software development process to implement the demands of a software factory in an academic environment. The proposed methodology is based on an iterative-incremental model and includes methods such as Scrum, XP, RUP, and BDD, among others. The authors applied the process as a practical teaching methodology in a Software Engineering course. Along this line, the authors aimed at assessing the impact of TDD and BDD techniques on software development by analyzing students' commitment regarding quality and productivity [35].

Unlike the aforementioned research works, our work presents an experience report in a discipline of the Bachelor's degree courses in Information Systems and Computer Science, in the development of a product, comparing groups of students who employed TDD and BDD with a group that did not use agile practices.

IV. Design of the Course

In this research work, we conducted an experience report on the verification of students' learning difficulties. Also, the report shows the exploration of TDD and BDD as a teaching methodology, as an alternative to the teaching of Software Engineering practices. The analysis of the results was conducted under a qualitative and quantitative approach, associating the data obtained in the pedagogical activity and the references adopted in the study.

The report presented covers editions of the curriculum component Software Engineering Laboratory (LES), which has 80 hours of class in the Bachelor's degree courses in

978-1-7281-8478-4/21/$31.00 ©2021 IEEE 21-23 April 2021, Vienna, Austria

Information Systems and Computer Science, from January to April 2019. The LES is part of the Computer Science and Information Systems courses of a Brazilian Northeast University, focusing on the integration of content from the modeling, software engineering, and programming courses in a practical and applied way. This course is located in the fourth semester of the courses, and its objective meets the operational stages defined by Piaget [36].

The course, composed of 35 students, was divided into 5 groups being group 1 with 7 students, group 2 with 6 students, group 3 with 7 students, group 4 with 8 students, and group 5 with 7 students. The first and second group used the BDD, the third group used the Partial BDD and TDD, the fourth group used TDD and the fifth group did not use any of the agile practices. Each group had the task of developing a software product in 8 weeks. Thus, the students presented the progress of the project every week.

Students are active participants in the problem-solving process. They face concrete situations by interacting with professors and colleagues, who assist the students in a controlled academic environment [37]. This activity usually occurs in the classroom, where the interaction with knowledge and the actors of the educational activities take place, leading to the development of the collective intelligence [38].

The participation of the professor is especially relevant at times of passage from the theoretical approach to the detailed explanation on how to find a solution to the proposed problem. Besides, the professor must clarify the possibility of the proposed solution being reviewed by students, constituting a continuous cycle of improvement of the initial solution. The process of teaching and learning thus becomes continuous in the reconstruction of the experience itself [39].

After the introduction of the work environment, the course worked as follows. First, students receive three classes, totaling 12 hours of fundamentals of the TDD and BDD methodologies. Second, the students, divided into groups, are stimulated to develop a project proposed by the professors. Based on this project, each group must make methodological decisions; that is, they choose which method they will adopt. Third, during the development of the software solution, the professor plays two roles. The first role is of the Coach, in which s/he must perform quick hands-on training on environment tools and recommended practices. The second role is of the client, in which s/he communicates the needs related to the project to each team. After 40h of course development, each team presents the project's progress to the professors. Finally, when the course development reaches 78h of course, a new presentation is carried out to all the teams and professors. The goal behind this strategy is that not only the professors examine the results, but all students understand the positive and negative points of the choices. It is worth noting that the professors accompany the students weekly through Github and Trello to evaluate each team and its members.

Hence, we defined the following hypothesis for the evaluation and validation of the students' results by group:

H0: There is no difference between the grades of students of each group.

H1: There is a difference in the distribution of grades of students of each group.

The data collection was carried out through GitHub[1]. This tool was used in the course project by the students for versioning the source code. Additionally, we used Trello[2] to create, administrate, and monitor the tasks developed by the students. To apply TDD, we used JUnit[3], whereas we used JBehave[4] for BDD. Furthermore, we used Slack[5] for communication between members of the groups and holding daily meetings.

V. Experience report

The effectiveness of our teaching approach on undergraduate software engineering students was studied and reported. Five teams (35 students) participated in the study over the second semester of 2019. Ninety-seven percent of the students were male students. All participants were of similar age, ranging between 18 and 24 years old.

The students had basic concepts previously acquired in subjects such as Software Engineering, Introduction to Programming, and Object-Oriented Programming. It is worth mentioning that these students had no practical or advanced experience in agile software development. Besides, the student groups were monitored in the form of coaching by the teacher during the requirement changes after sprints. We observed that the understanding of the problem and the ability to separate stable and unstable requirements was essential for BDD, in which user stories and scenarios are part of the development, allowing students to get a better understanding of the system components.

We experimented as depicted by Fig. 3. In the first week, students were given a lecture on Agile Methods and how these impact software development, discussing tools, such as Trello, Github, Slack, JUnit, and JBehave. In the second week, we developed practices using TDD and BDD, along with the tools discussed in the lecture. In the third week, the students started with the development of the project. After an introductory presentation of the course project, the duration of the project was stipulated as 10 weeks. Finally, at the end of the project, students conducted a demo to present the final solution to the project.

The groups created the project in Trello as shown in Fig. 4 and added teachers as collaborators. The groups were self-organized with the Scrum roles, namely Scrum Master, Product Owner, and Team. Besides, one teacher was a customer responsible for information about the product and features. Another teacher also helped as a coach during iterations [40], allowing students to make decisions and assisting them during the process.

Concerning the course project, the domain of the chosen problem was **Software for medical clinic management**. We

[1]https://github.com/
[2]https://trello.com/
[3]https://junit.org/junit5/
[4]https://jbehave.org/
[5]https://slack.com/

978-1-7281-8478-4/21/$31.00 ©2021 IEEE

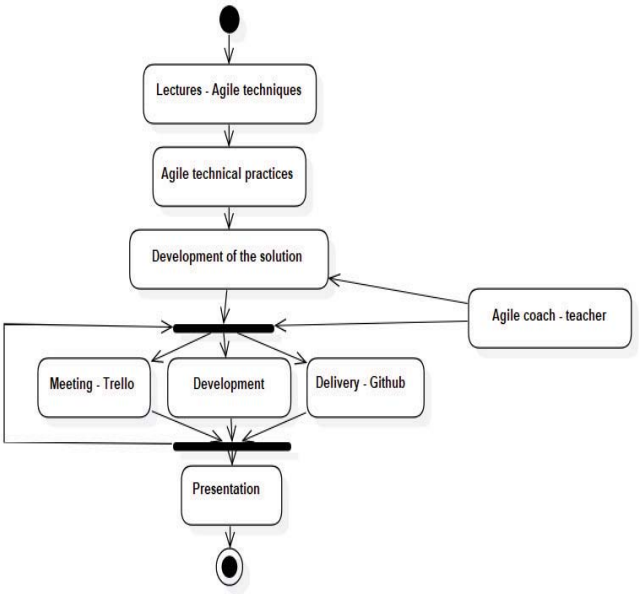

Fig. 3. Course methodology.

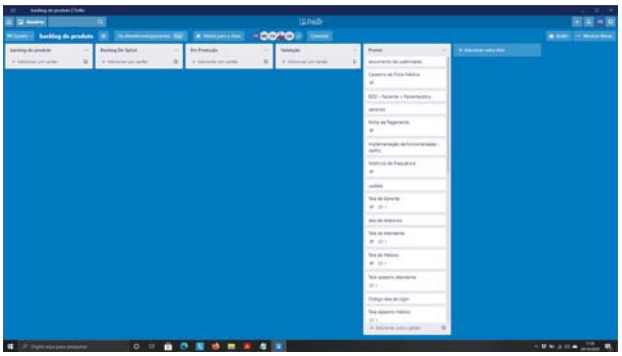

Fig. 4. Trello board used by the groups.

delivered the overview of the project, i.e., an epic, and the students should create user stories to build the initial Product Backlog. The students were responsible for interviewing the customers to obtain information about functional and non-functional requirements.

The customer was MedClin Medical Clinic[6], who has requested a management system able to track the enrollment of patients, physicians, staff, controlling the physician's schedule, consultation pay, and staff and physician payroll. The system must be in the Cloud, allowing remote management access and monitoring of the schedule by the doctor and patient. It should generate monthly consultation reports for the financial monitoring of the clinic.

During the course, the students had to develop a course project, organize and prioritize the Product Backlog. The students were expected to work for about two hours a day). For student assessment, individual and group analyses were performed, using the following data:

- *Commits* carried out individually;
- Individually developed tasks;

TABLE I
DISTRIBUTION OF GROUPS

	Group 1	**Group 2**	**Group 3**	**Group 4**	**Group 5**
Agile technique	BDD	BDD	BDD Partial and TDD	TDD	Not used
Number of students	7	6	7	8	7
Male	7	5	7	8	7
Female	0	1	0	0	0
Language	Java	Java	Java	Java	Java
IDE	Free selection	Free selection	Free selection	Free selection	Free selection
Framework	JBehave	JBehave	JBehave and JUnit	JUnit	Not used

- Individual rework quantity;
- Delivery produced by the group;
- Product Presentation;
- Problem solving by the group;
- Theoretical evaluation of the developed practice.

As for setup, each group chose the programming language, Database Management System (open-source), and the division and assignment of tasks among team members. Once a week, teams were required to perform functional deliveries and define tasks for the next iteration. In this context, the professor should evaluate if the deliveries met the customer's needs; otherwise, it would return as rework (user stories were pushed back). Consequently, the teams were organized by choosing the agile techniques to be adopted, the programming language, the Integrated Development Environment (IDE), and the frameworks as shown in Table I.

Fig. 5 describes the iteration cycle followed in our proposed teaching model. In the first iteration, the first stage consisted of a Planning Meeting, in which the students understand the problem, discuss, and define the tasks. This meeting takes 1 hour and 30 minutes, followed by the development stage. The subsequent iterations begin with a meeting to discuss the Delivery and present it to the customer. The review of the iteration is carried out to see what they have learned and how they have addressed issues. Afterward, the teams plan the next iteration and started the Development. This series of actions are conducted in all other iterations. Finally, in the final iteration, the students review what they have produced, meet to check the final delivery, and present the product to the customer and product owner. After that, as a retrospective, they create a report with the lessons learned. During the entire cycle, the coach (another teacher) help and assist the teams.

The teams, in the first iteration (working week), failed to obtain significant results due to lack of practical experience. They delivered few tasks and with several errors. The teams that decided to use agile techniques during the first cycle delivered less tasks than the team that did not use agile techniques. According to the students, the reason was the lack of practice with the chosen tool. Thus, only the team "Group 5" obtained considerable initial results, and the Fig. **??** shows these results.

During the second iteration, the fourth team (Group 4) made

978-1-7281-8478-4/21/$31.00 ©2021 IEEE
http://www.medclinmed.com.br

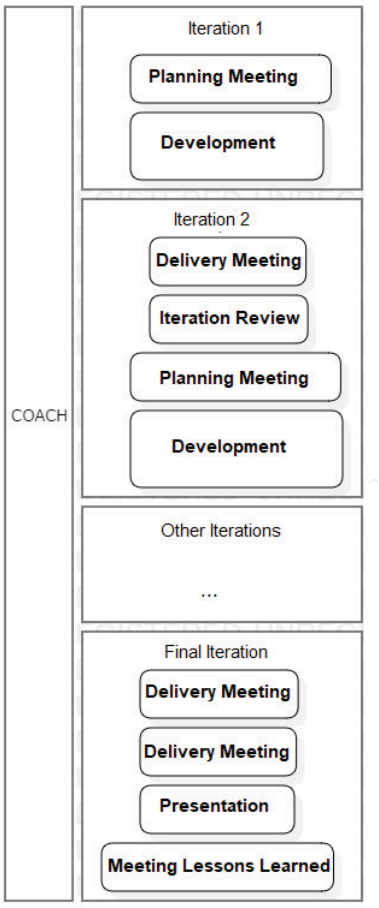

Fig. 5. Iteration cycle

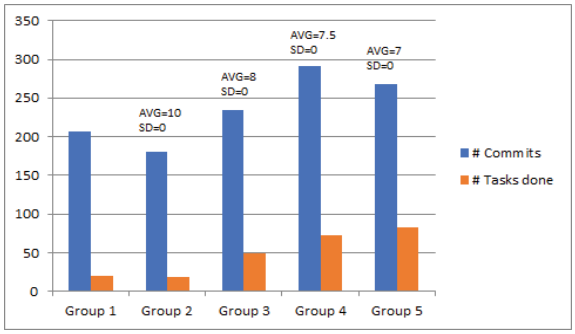

Fig. 6. Performance of groups in terms of commits, task done and professors' qualifications.

a higher number of deliveries, followed by Group 1 and finally, Groups 1 to 3. At this point, Group 4 reported that they had understood the development process. Group 3 stated that the task deliveries were smaller, nonetheless, they delivered high-quality documentation, such as the user stories and the scenarios. In the following cycle, Groups 1 to 3 performed a higher number of deliveries, followed by Group 4. Group 5 had a higher number of rework related to the tasks delivered due to a lack of integration with the new features.

Changes in requirements emerged in the third iteration. As the project was unique and started from scratch, all teams had the same starting point and the same changes requested by the customer. The use of agile practices and the use of JUnit by the group that adopted the TDD, and the use of JBehave and JUnit by the group that adopted the BDD, considerably allowed teams to reduce the rework.

As a result, Groups 1 to 3 began to make a higher number of deliveries, with less rework, followed by Group 4. According to the students of Groups 1 to 3, the initial work related to writing the user stories and scenarios allowed them to obtain a prior understanding of the features. This issue took considerable time to be understood at the beginning of the iteration, but as advantage, this activity allowed for reducing the work in the following stages. Group 4 reported that the task of creating the tests contributed to reduce the rework because they had to think about the inputs and outputs of each functionality, before its construction.

Groups 1 to 3 reported that they initially thought that the use of BDD was a burdensome and time-consuming task. However, they noticed that during the iterations, BDD helped teams reduce rework. Besides, the scenarios helped in the validation of the features, even before running the test, which facilitated the development work.

In the penultimate iteration, Groups 1 to 3 had completed the tasks and was only working on improvements, and Group 4 made a group of deliveries. In the last iteration, only Group 5 still had deliveries to be made. At the end of the project, the results were analyzed through Trello and Github.

VI. Results and Discussion

The practices adopted in the course project contributed to students to bridge the gap between academic and professional contexts. Such action is essential for training, as pointed out in the Delors Report [41]. In our study, the adopted model brought the principle of autonomy, where the groups had to decide which practices they would adopt, and how they would use.

We observed at the end of the execution of this work that during the theoretical presentation of the content, the students showed little interest [42]. This caused Group 5 to decide not to use agile practices during the development, and the other four groups defined the practices only during the practical class. The students were responsible for finding the best solution to the problem presented.

At the end of the tasks, Groups 1 to 3, which adopted BDD, achieved superior performance, followed by Group 4, which adopted TDD. Finally, Group 5 was in the third place and did not adopt any agile practice.

We noticed that deliveries of functionalities were different by the group. Groups 1 to 3 started more slowly in deliveries, but from the third iteration, the students maintained a high number of deliveries, improving their performance. Group 4 started with better delivery time, but during iterations, there was no significant evolution. Group 5 obtained the best result in the time of the initial deliveries. However, in the subsequent iterations, the team failed to maintain stability due to numerous errors, thus reducing the overall performance of the group, as depicted in Fig. 6.

978-1-7281-8478-4/21/$31.00 ©2021 IEEE

TABLE II
DISTRIBUTION OF PRODUCT SCORES PER STUDENT.

	Group 1	Group 2	Group 3	Group 4	Group 5
Student 1	10	10	8	7,5	7
Student 2	10	10	8	7,5	7
Student 3	10	10	8	7,5	7
Student 4	10	10	8	7,5	7
Student 5	10	10	8	7,5	7
Student 6	10	10	8	7,5	7
Student 7	10	-	8	7,5	7
Student 8	-	-	-	7,5	-

TABLE III
KRUSKAL-WALLIS TEST RESULTS

x	df	p-value
32.062	7	3.955e-05

When comparing students' results by the group in the range of 0-10, it is possible to note that the use of BDD allowed Groups 1 and 2 to maintain a higher score, followed by Groups 3, 4, and 5, as shown in Table II.

Table II shows the distribution of product scores per student. Given the size of our sample, we used the analysis of variance with the Kruskal-Wallis test [43], also known as Test H. This test compares three or more groups with independent samples and analyzes the existence of significant differences.

The table III shows that the result of the H test (p.value = 3.955e-05, chi-square = 32.062). It is worth mentioning that for this test, the significance level of 0.05 and the confidence interval of 95% were adopted. The chi-squared is a measure of divergence between the distribution of data and an expected distribution. In this experience report, it was chosen to determine if the statistical model adjusts the data adequately. The degrees of freedom (df) is equal to the sample number minus 1, that is, eight students, on average, per evaluated group evaluated per grade minus one equal to five.

The p-value is the probability of obtaining a statistical test equal to or more extreme than that observed in a sample, under the null hypothesis (H0). A confidence interval is a range of values derived from sample statistics that are likely to contain the value of an unknown population parameter. The significance level is the probability of incorrectly rejecting the null hypothesis (H0) when it is true.

Thus, in this case, it is possible to conclude that if the p-value associated with its chi-square statistic is lower than its selected significance level, the test rejects the null hypothesis that the model adjusts the data. Therefore, the **H0 - There is no difference between the grades of students of each group** is rejected because the p-value < 0.05, that is, there is not a probability of obtaining this result by chance when the treatment has no real effect.

Thus, when analyzing the graph in Fig. 7, it is possible to visualize the differences in grades of the groups of students; it was noted that Groups 1 and 2 have a starting point; that is, all the students had a more significant prominence in comparison with the students of the other groups.

Regarding groups commits in Github, Table IV shows that all the groups presented considerable work, despite having

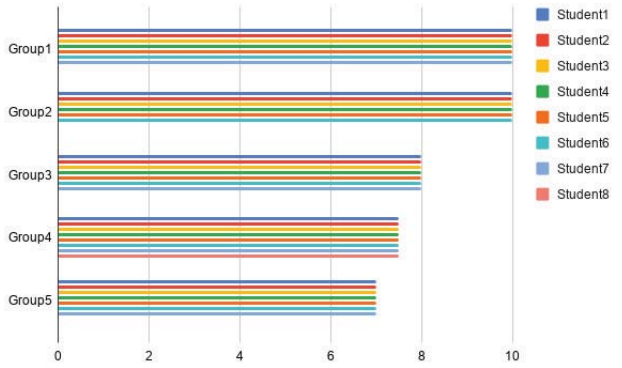

Fig. 7. Comparison of notes by group.

TABLE IV
Commits PER GROUP OF STUDENTS

	Group 1	Group 2	Group 3	Group 4	Group 5
Student 1	30	32	33	40	39
Student 2	33	30	37	39	33
Student 3	19	31	38	28	38
Student 4	30	29	29	38	35
Student 5	27	31	33	37	38
Student 6	28	28	30	38	40
Student 7	29	-	35	39	45
Student 8	-	-	-	33	-

more commits. In group 1, students 1 and 4 performed the same amount of commits (30). In group 2, students 3 and 5 performed the same amount of commits (31). In group 3, students 1 and 5 performed the same amount of commits (33). In group 4, students 2 and 7 carried out the same amount of commits (39), as well as students 4 and 6 carried out the same amount of commits (38). In group 5, students 3 and 5 performed the same amount of commits (38).

All the groups had a collaborative behavior among the members. However, Groups 1 to 4 obtained better feedback during the development, as a result of the adoption of agile practices. The conflicts generated during iterations were solved with the support of the Coach, without hindering the overall progress of the groups. Groups 1 to 3 stated that the scenarios enabled a fast and efficient communication with the team members and with the customer, also facilitating the final presentation. Besides, at the end of the course there is a professor qualification, with a score from 0 to 10. Groups 1 and 2 obtained outstanding results, and Group 3 obtained a sufficient qualification (Fig. 6). We can state that BDD allowed students to have a superior performance in comparison with other groups. Group 5 that did not use any agile technique obtained the lowest qualification. Thus, it was possible to answer the two research questions:

Q1: What do professors need to teach students to deal quickly and effectively with changing requirements during project development?

By using our proposed model, we obtained outstanding results. Although the students were taught the main concepts

978-1-7281-8478-4/21/$31.00 ©2021 IEEE

useful for the course project in other disciplines, they stated the lectures provided at the beginning of the course were crucial. Furthermore, the students proposed new lectures complemented with professionals of the software industry to have the real experience of software developers.

We note that groups that adopted agile practices had fewer issues with changing requirements and were able to implement adaptations more quickly. The students pointed out that the professors act as customers and Coach during the iterations, assisting in agile practices. Students also reported difficulties in coping with changing requirements due to team members' lack of communication, which was improved through Slack, which was significantly used to hold daily meetings.

Thus, in response to the question, students need to understand the requirements and how they impact the product. Such an understanding was acquired through agile practices. Students also need to learn that communication among team members is essential. Thus, a professor, acting as a coach, was essential to explain to the teams that, although they meet in the classroom only once a week, the communication should be continuous.

Q2: How to evaluate individual performance when teaching a project based on TDD and BDD?

The evaluation model used in the group's deliveries, based on Trello and Github, facilitated both the professor's work and the monitoring of students. Moreover, Coach could easily help students with low productivity. Github commits, when integrated with Trello, facilitated the work of individual and group productivity analysis. Thus, the integration of these tools can serve as a method for assessing individual student performance.

VII. Lessons Learned

This paper presented the experience of applying TDD and BDD in a Software Engineering course from a theoretical and practical standpoint. The students had the possibility of applying in a course project skills and knowledge previously acquired. However, as a lesson learned, it can be stated that to leave under-experienced students to decide to select Agile techniques failed to be the best option. We should have determined the use of BDD by students during the execution of the course project. Another point to be observed is the first iteration; usually, the low performance was obtained, which is normal due to the lack of experience of students. Hence, we should take this issue into account for the design of the lectures given to the students at the beginning of the course.

Additionally, permitting the groups to select tools resulted in differences at the beginning of the tasks, because it was necessary to understand the tools before using them. Fig. 6 allowed to evidence that initially Group 5 started with the highest amount of practical tasks and Group 2 with the lowest number of tasks (students had a focus on planning). However, at the end of the estimated time, Groups 1 to 4 completed the project even before the last iteration, while the same did not occur with Group 5. Thus, we emphasize that the planning stage is critical to avoid rework, deliver products on time, and especially, maintain the quality of the product.

There are some threats to validity that need to be addressed. Firstly, the fact that students were randomly grouped might have impacted student performance in comparison with other strategies for group formation. Secondly, the Agile Coach role strongly depended on the ability of the teacher to address group management, resource allocation, and leadership. Probably, other teachers with different background and expertise could impact the results obtained. The Agile Coach in this study had work experience in coaching agile teams, and this skill might have decreased the impact on the threats to validity. Teachers with limited coaching experience might find it difficult to act without affecting the self-organization of the team and might find themselves telling students the right course of action, rather than guiding them to learn from their own mistakes.

It is worth noting that students tackled some challenges, such as an inability to make accurate estimates of workload, a limited experience in software testing, and an inability to discard code inappropriate situations. These issues enriched students' learning experience; the Agile Coach could have reduced the impact of these challenges on student performance. Moreover, students having limited software development background knowledge might find it difficult to appreciate the proposed training approach, so this course is offered in the last year of the program.

To generalize the findings, other issues may have biased the results of the experiments. First, an industrial environment was simulated, in which the teaching professionals did their best to replicate typical software engineering problems. Second, our approach seems to apply to other case studies, if students receive training in software testing, Scrum, and agile coaching. Finally, the performance of students is related to the understanding of the problems; BDD helped in this process, however, due to the lack of experience of students, some scenarios that were created might not have real utilities for the project.

VIII. Conclusions

In this work, we introduced an experience report of adopting TDD and BDD through a course project in the context of a Software Engineering course. The report covers the 2019 edition of the curricular component of the Software Engineering Laboratory of the Bachelor's degree courses in Computer Science and Information Systems of a private University in the Northeast of Brazil. In this course project, students were able to integrate several contents, such as software design, development, testing, and validation, by applying agile methodologies.

Baked up with the results, we could state that BDD has better adherence to the integration of the different contents. Furthermore, BDD allowed students to better understand the context of the course project, thus providing benefits to student learning. The results also showed that adopting agile practices helped the students in the development process, reducing the noise related to the understanding of the system, and the rework of software components. Thus, our proposal enhanced

978-1-7281-8478-4/21/$31.00 ©2021 IEEE 21-23 April 2021, Vienna, Austria

the student learning experience along with the development of the course project.

It is worth noting that agile practices were complemented with a continuous delivery model and weekly review meetings. On the one hand, Trello and Github allowed students to manage tasks; on the other hand, these tools allowed professors to assess and audit team performance and delivery of artifacts. Also, the Agile Coach facilitated the student progress and the following of our proposed methodology.

In future work, we aim to apply techniques of scenario analysis to improve the quality of work. Along this line, we propose to use a tool for managing defects (i.e., bugs), such as Mantis. Furthermore, we are planning to design guidelines to replicate our study in other academic contexts. Finally, we aim to foster a discussion of appropriate teaching methods of software testing.

REFERENCES

[1] R. Dawson, "Twenty dirty tricks to train software engineers," in *Proceedings of the 22nd international conference on Software engineering*. ACM, 2000, pp. 209–218.

[2] G. Regev, D. C. Gause, and A. Wegmann, "Experiential learning approach for requirements engineering education," *Requirements engineering*, vol. 14, no. 4, p. 269, 2009.

[3] T. C. Lethbridge, J. Diaz-Herrera, J. Richard Jr, J. B. Thompson *et al.*, "Improving software practice through education: Challenges and future trends," in *Future of Software Engineering (FOSE'07)*. IEEE, 2007, pp. 12–28.

[4] M. H. Yamaguti, F. M. de Oliveira, C. A. Trindade, and A. Dutra, "Ages: An interdisciplinary space based on projects for software engineering learning," in *Proceedings of the 31st Brazilian Symposium on Software Engineering*. ACM, 2017, pp. 368–373.

[5] B. Bruegge, S. Krusche, and L. Alperowitz, "Software engineering project courses with industrial clients," *ACM Transactions on Computing Education (TOCE)*, vol. 15, no. 4, p. 17, 2015.

[6] M. Marques, S. F. Ochoa, M. C. Bastarrica, and F. J. Gutierrez, "Enhancing the student learning experience in software engineering project courses," *IEEE Transactions on Education*, vol. 61, no. 1, pp. 63–73, 2017.

[7] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," 2012.

[8] V. Devedzic *et al.*, "Teaching agile software development: A case study," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 273–278, 2010.

[9] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," in *2009 Agile Conference*. IEEE, 2009, pp. 174–179.

[10] C.-H. Tan, W.-K. Tan, and H.-H. Teo, "Training students to be agile information systems developers: A pedagogical approach," in *Proceedings of the 2008 ACM SIGMIS CPR conference on Computer personnel doctoral consortium and research*. ACM, 2008, pp. 88–96.

[11] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 99–106, 2011.

[12] ——, "Teaching scrum through team-project work: Students' perceptions and teacher's observations," *International Journal of Engineering Education*, vol. 26, no. 1, p. 96, 2010.

[13] A. Scharf and A. Koch, "Scrum in a software engineering course: An in-depth praxis report," in *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2013, pp. 159–168.

[14] M. Paasivaara, V. Heikkilä, C. Lassenius, and T. Toivola, "Teaching students scrum using lego blocks," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 382–391.

[15] V. Mahnič, "Scrum in software engineering courses: an outline of the literature," *Global Journal of Engineering Education*, vol. 17, no. 2, pp. 77–83, 2015.

[16] G. Fraser, A. Gambi, M. Kreis, and J. M. Rojas, "Gamifying a software testing course with code defenders," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 571–577.

[17] M. Cohn, *Succeeding with agile: software development using Scrum*. Pearson Education, 2010.

[18] S. Hammond and D. Umphress, "Test driven development: the state of the practice," in *Proceedings of the 50th Annual Southeast Regional Conference*. ACM, 2012, pp. 158–163.

[19] S. Fraser, D. Astels, K. Beck, B. Boehm, J. McGregor, J. Newkirk, and C. Poole, "Discipline and practices of tdd:(test driven development)," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. ACM, 2003, pp. 268–270.

[20] W. Bissi, A. G. S. S. Neto, and M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," *Information and Software Technology*, vol. 74, pp. 45–54, 2016.

[21] D. Janzen and H. Saiedian, "Does test-driven development really improve software design quality?" *Ieee Software*, vol. 25, no. 2, pp. 77–84, 2008.

[22] D. North, "Introducing bdd (2006)," *Verfügbar unter: http://dannorth.net/introducingbdd*, 2019.

[23] J. F. Smart, *BDD in Action*. Manning Publications, 2014.

[24] C. Solis and X. Wang, "A study of the characteristics of behaviour driven development," in *Proceedings of the International Conference on Software Engineering and Advanced Applications, IEEE, Oulu, Finland, 2011, pp. 383-387*.

[25] L. C. P. Moraes *et al.*, "Um estudo empírico sobre o uso do bdd e seu apoio a engenharia de requisitos," 2016.

[26] G. Oliveira, S. Marczak, and C. Moralles, "How to evaluate bdd scenarios' quality?" in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering, ACM, Salvador, Brazil, 2019, pp. 481–490*. [Online]. Available: https://doi.org/10.1145/3350768.3351301

[27] P. Gestwicki and B. McNely, "Interdisciplinary projects in the academic studio," *ACM Transactions on Computing Education (TOCE)*, vol. 16, no. 2, pp. 1–24, 2016.

[28] B. George and L. Williams, "A structured experiment of test-driven development," *Information and Software Technology*, vol. 46, pp. 337–342, 04 2004.

[29] N. Nagappan, E. M. Maximilien, T. Bhat, and L. Williams, "Realizing quality improvement through test driven development: Results and experiences of four industrial teams," *Empir Software Eng*, pp. 289 – 302, June 2008.

[30] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio, "Evaluating advantages of test driven development: A controlled experiment with professionals," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ser. ISESE'06, ACM. New York, NY, USA, 2006, pp. 364–371.

[31] S. H. Edwards, "Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance," in *Proceedings of the international conference on education and information systems: technologies and applications EISTA*, vol. 3. Citeseer, 2003.

[32] M. A. Eierman and J. Iversen, "Comparing test-driven development and pair programming to improve the learning of programming languages," *Journal of the Midwest Association for Information Systems— Vol*, vol. 2018, no. 1, p. 23, 2018.

[33] G. L. R. dos Santos and E. C. Neto, "Um processo de desenvolvimento de software para o ensino técnico baseado em uma fábrica de software escola," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 29, no. 1, 2018, p. 1741.

[34] I. Seroa, H. Bertoldo, and V. Neves, "Testerds: uma maneira fácil e estimulante para aprender estruturas de dados," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 29, no. 1, 2018, p. 864.

[35] L. A. Cisneros, C. I. Reis, M. Maximiano, and J. A. Quiña, "An experimental evaluation of itl, tdd and bdd," in *ICSEA 2018, The Thirteenth International Conference on Software Engineering Advances*. ThinkMind, 2018, pp. 20–24.

[36] J. Piaget and D. Elkind, *Six psychological studies*. Vintage Books, 1968, vol. 462.

[37] M. L. Bigge, *Learning theories for teachers*. Harper & Row, 1982.

[38] V. M. Kenski, *Educação e tecnologias*. Papirus editora, 2007.

[39] J. Dewey, *Experience and education: The kappa delta pi lecture series*. Touchstone, 1997.

[40] G. Rodríguez, Á. Soria, and M. Campo, "Measuring the impact of agile coaching on students' performance," *IEEE Transactions on Education*, vol. 59, no. 3, pp. 202–209, 2016.

[41] J. Delors and Z. Nanzhao, "Educação um tesouro a descobrir," 1998.

978-1-7281-8478-4/21/$31.00 ©2021 IEEE

[42] B. Gadelha and A. Castro, "A reference model for teaching collaborative mobile systems," in *Proceedings of the 31st Brazilian Symposium on Software Engineering*. ACM, 2017, pp. 374–383.
[43] C. Kim and W. S. Shin, "Does information from the higher education and rd institutes improve the innovation efficiency of logistic firms?" vol. 35, no. 1, pp. 70–76, 2019.

Fabio Gomes Rocha Master in Computer Science - Federal University of Sergipe (UFS), PhD in Education - Tiradentes University (UNIT), Adjunct Professor I of computer courses at Tiradentes University (UNIT), Coordinator of the Advanced Computer Laboratory and Artificial Intelligence - ITP and member of Interdisciplinary Research Group in Information and Communication Technology (GPITIC).

Layse Santos Souza is graduated in Computer Science - Tiradentes University (UNIT), Master Student in Computer Science - Federal University of Sergipe (UFS), member of Interdisciplinary Research Group in Information and Communication Technology (GPITIC).

Thiciane Suely C. Silva is graduated in Computer Science - Tiradentes University (UNIT), Master Student in Computer Science - PUC-RS, member of Interdisciplinary Research Group in Information and Communication Technology (GPITIC).

Guillermo Rodríguez received the Ph.D. degree in computer science at the Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Tandil, Argentina, in 2014. Since 2008, he has been part of ISISTAN Research Institute (CONICET - UNICEN). Since 2018 he has been part of Universidad Argentina de la Empresa (UADE).

978-1-7281-8478-4/21/$31.00 ©2021 IEEE 21–23 April 2021, Vienna, Austria
2021 IEEE Global Engineering Education Conference (EDUCON)