

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

NICOLAS PEREIRA DO NASCIMENTO

A study of teaching BDD in active learning environments
Um estudo sobre o ensino de BDD em ambientes de aprendizagem ativa

Porto Alegre
2021

PÓS-GRADUAÇÃO - *STRICTO SENSU*



Pontifícia Universidade Católica
do Rio Grande do Sul

**PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM**

**A STUDY OF TEACHING BDD IN
ACTIVE LEARNING
ENVIRONMENTS**

NICOLAS PEREIRA DO NASCIMENTO

Master Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Afonso Sales, Ph.D.

**Porto Alegre
2021**

Ficha Catalográfica

N244s Nascimento, Nicolas Pereira do

A Study of Teaching BDD in Active Learning Environments / Nicolas Pereira do Nascimento . – 2020.

123.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Afonso Henrique Correa de Sales.

1. Behavior-Driven Development. 2. Active Learning. 3. Software Development. I. Sales, Afonso Henrique Correa de. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051



Nicolas Pereira do Nascimento

A STUDY OF TEACHING BDD IN ACTIVE LEARNING ENVIRONMENTS

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 21 de Agosto de 2020.

BANCA EXAMINADORA:

DR. CLEIDSON RONALD BOTELHO DE SOUZA - UFPA

DR. FÁBIO GOMES ROCHA - UNIT

DRA. SABRINA DOS SANTOS MARCZAK - PPGCC - PUCRS

DR. AFONSO HENRIQUE CORRÊA DE SALES (PPGCC/PUCRS - ORIENTADOR)

*“If I have seen further it is by standing on the
shoulders of giants”*
(Isaac Newton)

UM ESTUDO SOBRE O ENSINO DE BDD EM AMBIENTES DE APRENDIZAGEM ATIVA

RESUMO

Práticas de desenvolvimento de software que melhoraram a qualidade de software e ajudam times a desenvolver melhor de forma colaborativa tem recebido atenção da comunidade acadêmica. Entre estas técnicas, está o *Behavior-Driven Development* (BDD), uma metodologia de desenvolvimento que propõe que software seja desenvolvido focando principalmente em seu comportamento esperado. Sob o ponto de vista de ensino, introduzir BDD durante cursos de engenharia de software e/ou cursos de capacitação de desenvolvedores de software se tornou importante. Neste contexto, este estudo espera criar um corpo de conhecimento sobre os impactos de ensino de BDD em ambientes de aprendizagem ativa. Para realizar isso, fez-se: (i) uma Revisão Sistemática da Literatura (RSL), objetivando entender o *estado-da-arte* da literatura sobre este tópico; (ii) um painel de especialistas, para obter a opinião de especialistas em aprendizagem ativa sobre os possíveis efeitos de BDD nestes ambientes, (iii) uma *survey* com participantes de um curso de desenvolvimento de software que ensina através de aprendizagem ativa, para entender como fatores destes ambientes impactam a satisfação dos participantes, e (iv) um estudo de caso dos efeitos de ensinar e usar BDD em ambientes de aprendizagem ativa, para avaliar os efeitos de BDD nestes ambientes. Resultados obtidos indicam que (i) há uma lacuna de estudos sobre o assunto de ensino de BDD em ambientes de aprendizagem ativa, (ii) que especialistas em aprendizagem ativa possuem mais opiniões positivas acerca do ensino de BDD nestes ambientes, (iii) que a duração de uma atividade, composição e tamanho de times podem impactar a satisfação de alunos em ambientes de aprendizagem ativa e (iv) que BDD pode ter resultados positivos, como um aumento na colaboração entre times, e resultados negativos, como dificuldades na escrita de testes unitários. Conclui-se que BDD possui mais resultados positivos do que negativos e apresenta-se um corpo de co-

Conhecimento sobre BDD em ambientes de aprendizagem ativa. Este corpo de conhecimento oferece *insights* sobre BDD nestes ambientes. Entretanto, generalização destes resultados requer mais pesquisas.

Palavras-Chave: Behavior-Driven Development, Aprendizagem Ativa, Desenvolvimento de Software.

A STUDY OF TEACHING BDD IN ACTIVE LEARNING ENVIRONMENTS

ABSTRACT

Software development practices which enhance software quality and help teams better develop collaboratively have received attention by the academic community. Among these techniques is Behavior-Driven Development (BDD), a development method which proposes software to be developed focusing primarily on its expected behavior. From a teaching standpoint, introducing BDD during software engineering classes and/or training courses for software developers has become important. In this context, this study aims at creating a body of knowledge regarding the impacts of teaching BDD in active learning environments. In order to achieve this, we have performed: (i) a Systematic Literature Review (SLR), aiming at understanding *state-of-the-art* literature regarding this topic; (ii) an expert panel, to obtain active-learning expert's opinion about the possible effects of BDD in these environments, (iii) a survey with participants in a software development course which teaches through active learning, to understand how factors from these environments impact participant satisfaction, and (iv) a case study of the effects of teaching and using BDD in an active learning environment, to assess the effects of BDD in this environment. Our results indicate that (i) there is a gap of studies regarding the matter of teaching BDD in active learning environment, (ii) that active-learning experts have more positive feelings towards teaching BDD in active learning environments, (iii) that an activity duration, team size and composition can have an impact on students perception in active learning environments, and (iv) that BDD can have positive impacts, such as an increase in collaboration among teams, and negative impacts, like difficulties in writing unit tests. We concluded that BDD has more positive than negative outcomes and we present a body of knowledge regarding BDD in active learning environments. This body of knowledge offers valuable *insights* regarding BDD in such environments. However, generalization of these results requires further research.

Keywords: Behavior-Driven Development, Active Learning, Software Development.

LIST OF FIGURES

Figure 1.1 – Research Design	23
Figure 2.1 – The XP development cycle	26
Figure 2.2 – The Scrum development cycle	31
Figure 2.3 – An example template for specifying scenarios	32
Figure 2.4 – Challenge Based Learning Framework	35
Figure 4.1 – The overall research process	48
Figure 5.1 – Demography of students	63
Figure 5.2 – Projects Score	63
Figure 5.3 – Project Score vs. Team Size	66
Figure 5.4 – Projects Score vs. Projects Duration	67
Figure 6.1 – Process used to obtain results from interviews	73
Figure 6.2 – Distribution of participants in the two interviews phases	74

LIST OF TABLES

Table 3.1 – Search string	38
Table 3.2 – Search strategy	39
Table 3.3 – Metadata information of each paper	39
Table 3.4 – Search results	40
Table 3.5 – Classification Scheme	41
Table 3.6 – Overview of results	42
Table 3.7 – Summary of results from the studies	43
Table 4.1 – Questionnaire	48
Table 5.1 – Survey results grouped by category	65
Table 6.1 – Interview questions	73
Table 6.2 – Positive aspects reported by participants	78
Table 6.3 – Negative aspects reported by participants	79
Table 6.4 – Code quality reported by participants	79
Table 6.5 – Bug category for projects	80
Table 6.6 – Documentation types reported (Pre-BDD)	80
Table 6.7 – Documentation types reported (Post-BDD)	81
Table 6.8 – Positive impacts of BDD in implementation	81
Table 6.9 – Negative impacts of BDD in implementation	82
Table 6.10 – Positive aspects of BDD	83
Table 6.11 – Negative aspects of BDD	83
Table 7.1 – Summary of results from the studies (including case study)	89
Table 7.2 – Positive aspects of BDD (including categories)	92
Table 7.3 – Positive aspects of BDD grouped by categories from experts	93
Table 7.4 – Negative aspects aspects of BDD (including categories)	95
Table 7.5 – Negative aspects of BDD grouped by categories from experts	96

CONTENTS

1	INTRODUCTION	21
1.1	OBJECTIVE	22
1.2	RESEARCH QUESTION	22
1.3	METHODOLOGY	23
2	BACKGROUND	25
2.1	AGILE METHODOLOGIES	25
2.1.1	EXTREME PROGRAMMING (XP)	26
2.1.2	FEATURE DRIVEN DEVELOPMENT (FDD)	28
2.1.3	SCRUM	29
2.2	BEHAVIOR-DRIVEN DEVELOPMENT (BDD)	31
2.3	ACTIVE LEARNING	33
2.3.1	CHALLENGE BASED LEARNING (CBL)	34
3	STATE-OF-THE-ART	37
3.1	SYSTEMATIC LITERATURE REVIEW	37
3.2	METHODOLOGY	37
3.2.1	RESEARCH QUESTION	37
3.2.2	DATA SOURCE AND SEARCH STRATEGY	38
3.2.3	DATA EXTRACTION AND CLASSIFICATION	40
3.2.4	CLASSIFICATION SCHEME	40
3.3	RESULTS	42
3.3.1	RESEARCH QUESTION ANALYSIS	42
3.4	THREATS TO VALIDITY	44
3.5	CONCLUSION	45
4	EXPERT PANEL	47
4.1	METHODOLOGY	47
4.1.1	PROTOCOL	47
4.1.2	DATA COLLECTION & ANALYSIS	49
4.2	RESULTS	49
4.2.1	DEMOGRAPHY	49
4.2.2	BDD - EXPERIENCE	50

4.2.3	BDD - PERCEPTION OF BENEFITS	50
4.2.4	BDD - PERCEPTION OF CHALLENGES	53
4.3	DISCUSSION	56
4.3.1	(RQ1) WHAT ARE THE POSITIVE ASPECTS REPORTED BY EXPERTS REGARDING TEACHING BDD?	56
4.3.2	(RQ2) WHAT ARE THE NEGATIVE ASPECTS REPORTED BY EXPERTS REGARDING TEACHING BDD?	57
4.4	THREATS TO VALIDITY	57
4.5	CONCLUSION	58
5	SURVEY	59
5.1	RELATED WORK	59
5.2	THE COURSE	59
5.3	METHODOLOGY	60
5.3.1	DATA COLLECTION & ANALYSIS	61
5.4	RESULTS	62
5.4.1	PROJECTS SCORE	62
5.4.2	STUDENT EXPERIENCE	64
5.4.3	TEAM SIZE	65
5.4.4	PROJECT DURATION	65
5.5	DISCUSSION	67
5.6	THREATS TO VALIDITY	68
5.7	CONCLUSION	68
6	CASE STUDY	71
6.1	METHODOLOGY	71
6.1.1	CASE STUDY PROTOCOL	71
6.1.2	DATA COLLECTION	72
6.2	RESULTS	75
6.2.1	PROJECT REQUIREMENTS	76
6.2.2	FEATURE DEVELOPMENT	77
6.2.3	IMPLEMENTATION QUALITY	79
6.2.4	BDD IMPACT	82
6.3	DISCUSSION	82

6.3.1	(RQ1) WHAT ARE THE POSITIVE IMPACTS OF DEVELOPING SOFTWARE USING BDD?	84
6.3.2	(RQ2) WHAT ARE THE NEGATIVE IMPACTS OF DEVELOPING SOFTWARE USING BDD?	85
6.4	LIMITATIONS	86
6.5	CONCLUSION	86
7	DISCUSSION	89
7.1	(RQ1) WHAT TOOLS, MODELS, METHODOLOGIES, FRAMEWORKS AND SOFTWARE ARE USED WHEN TEACHING BDD?	89
7.2	(RQ2) WHAT ARE THE BENEFITS AND CHALLENGES OF USING BDD IN ACTIVE LEARNING ENVIRONMENTS?	91
7.2.1	BENEFITS	91
7.2.2	CHALLENGES	94
7.3	(RQ3) ARE THERE ANY BEST PRACTICES WHEN TEACHING BDD?	97
8	CONCLUSION	99
8.1	SUMMARY	99
8.2	LIMITATIONS	100
8.3	FUTURE WORK	101
	REFERENCES	103
	APPENDIX A – Interview agreement term	111
	APPENDIX B – Complete questionnaire - Expert Panel	115
	APPENDIX C – Complete questionnaire - Survey	117
	APPENDIX D – Complete questionnaire - Case Study	119
D.1	PRE-BDD	119
D.2	POST-BDD	121

1. INTRODUCTION

Software is ubiquitous today [37] and with the increase usage of electronic devices, such as smartphones [78], this presence does not show sign of slowing down. Specifically regarding mobile software, this trend is very pronounced. Alone, the App Store¹, a popular marketplace for mobile applications, has experienced an enormous growth, with a cumulative number of app downloads being around 180 billion from 2008 to 2017 [77].

Related to this is the increase in relevance of mobile application development (MAD). Many recent studies [83, 28, 24, 1] are currently targeting specific aspects of MAD, such as tooling and development methodology. These studies report multiple indicatives that MAD presents challenges and constraints which are unique and different from traditional desktop-based software development.

In terms of software development methodology, MAD is more suited to be performed using modern software development techniques, such as Agile software development [83]. Adaptations of agile [63, 69] that focus specifically on MAD present indicatives this approach can be successfully performed and present satisfactory results.

As MAD becomes more relevant, the education scenario has to adapt to properly prepare students. Initiatives from the industry, such as *Code.org* [29], further enhance the importance of teaching such skills. In addition, many adaptations have been performed and are reported in the academia. These adaptations are diverse and range from using MAD in the teaching of introductory programming [40] in traditional classes to the teaching of MAD in combination with modern software development practices [69].

Furthermore, as educational initiatives, all the reported adaptations aim at maximizing the learning of students regarding their specific topic (such as MAD). In this sense, Active Learning [27], an engaging, collaborative and cooperative approach for learning, has been shown to output far better results in terms of student engagement and even proper understanding of concepts [61]. In addition, teaching MAD using active learning frameworks has been demonstrated to produce positive outcomes [68] and even to be suited to be combined with agile frameworks, such as Scrum [69].

Given this context, more recent approaches to develop software have augmented agile frameworks with other practices, among of which is Behavior-Driven Development (BDD) [75]. BDD is a software development methodology created by Dan North [50] which combines software engineering practices from multiple domains. As some of its principles, BDD emphasizes close collaboration with customers, which is mostly aided by adoption of an ubiquitous language [19], and the usage of scenarios for specifying system requirements, following the concept of specification by example [3].

¹<https://www.apple.com/ios/app-store/>

Among the main benefits of BDD are its ubiquitous language, which allows all project stakeholders to better communicate system requirements, and scenarios, which guide the development team in the creation of application tests by augmenting user stories.

1.1 Objective

In this context, this study has performed an investigation regarding the application of BDD when teaching in active learning environments. To achieve this main objective, a series of steps were undertaken.

The following subsections further explain the methodology applied when performing the study.

1.2 Research Question

The main research question addressed by this study is:

“What are the impacts of using Behavior-Driven Development when teaching in active learning environments?”

As a way to facilitate answering this broad research question, we have split it in subquestions. These questions are:

- (RQ1) *“What tools, models, methodologies, frameworks and software are used when teaching BDD?”*
- (RQ2) *“What are the benefits and challenges of using BDD in active learning environments?”*
- (RQ3) *“Are there any best practices when teaching BDD?”*

Each research question focuses on a specific aspect of the topic. *RQ1* aims at understanding the tools, models, methodologies and frameworks which are the *state-of-the-art* when teaching BDD. After this, *RQ2* focuses on active learning environments and the effects of using BDD in these environments. Finally, *RQ3* aims at listing practices which are shown to produce good results when teaching BDD.

1.3 Methodology

In this section, we present the proposed research methodology to be applied in this study. As such, Figure 1.1 outlines a high-level diagram of the steps which were undertaken.

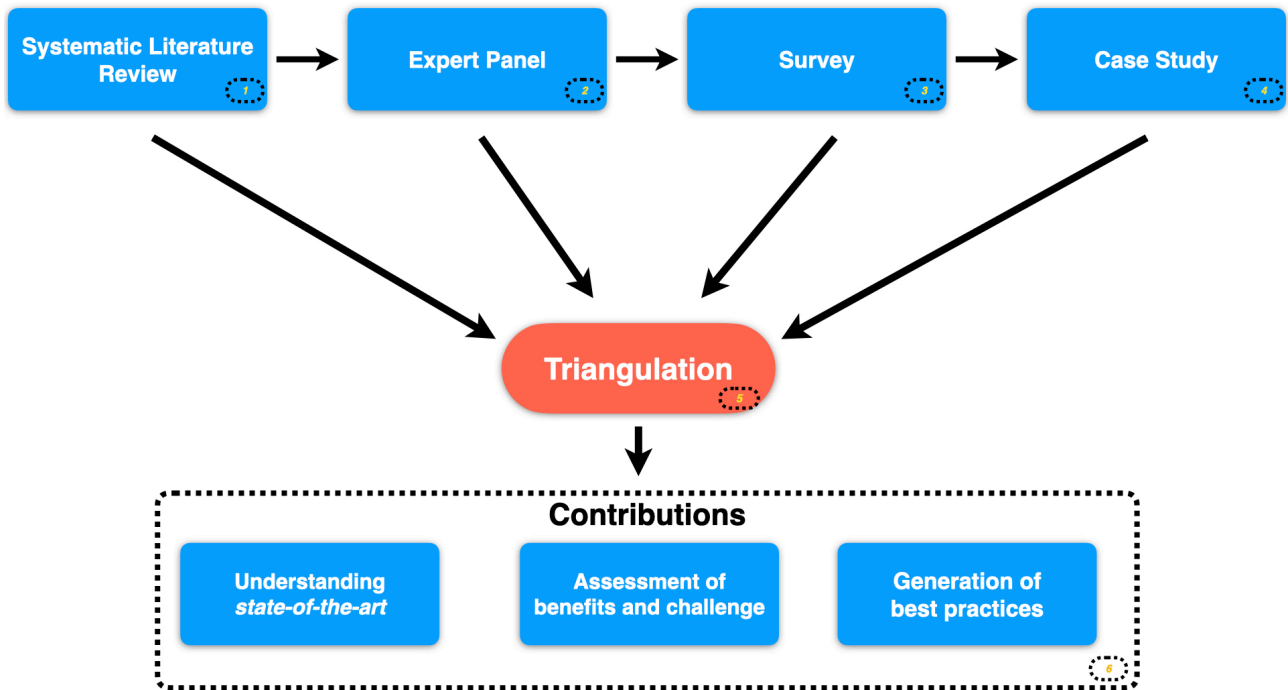


Figure 1.1 – Research Design
[Source: Author (2020)]

As an initial activity, we have sought to understand the state-of-the-art research regarding practices, tools, methods and frameworks which are current being used both in the industry and in the academia when teaching BDD. In order to achieve this, we have performed a systematic literature review (SRL). Following the standard guidelines for performing this research method [30], a SLR is suited for obtaining foundational background towards a topic, enabling other related research activities to be conducted.

Following this, we have conducted an expert panel research with active learning experts to assess potential benefits and challenges of teaching BDD in these environments. Expert panel research enables to researcher to capture expert judgment [65] and help improving hypothesis generated. Moreover, the opinion of experts is a valuable research artefact in the Software Engineering (SE) research community [18].

Complementarily, we have performed an investigation of influencing factors when teaching in active learning environments to expand our knowledge base related to these factors. This investigation was performed in a mobile application development course which uses active learning. Furthermore, it was performed through survey research, as it is appropriate when the focus of interest is on what is happening or how and why something is

happening and also applies when it is not possible to control dependent and independent variables [6].

Finally, to actively extract benefits and challenges of BDD in active learning environments, we have performed a case study research. A case study, as proposed by Kitchenham *et al.* [32], is suited for studying the effects of applying a particular method or tool, especially when piloting or introducing it. As such, ours was performed in an active learning environment which teaches mobile software development. Students were introduced to the concept of BDD, following the case study protocol, and the effects were measured.

After having collected data and findings from all these studies, we triangulated the results. According to Creswell [17], triangulation can be used as method of crossing results from different data sources, such as qualitative and quantitative. Furthermore, data from one source can be used to further inform results from another source, meaning that results can be analyzed seeking convergence and divergences.

This triangulation process allowed us to reason about our research questions. This reasoning revolved around understanding established tooling, assessing impacts (benefits and challenges) and generating a set of best practices to teach BDD in active learning environments.

The rest of this document is structured as follows. Chapter 2 presents relevant background required for a proper understanding of key concepts related to this study. Chapter 3 presents the results from the systematic literature review which focused on understanding the *state-of-the-art* literature for the topic of this study. Chapter 4 exposes the expert panel research findings and reasoning. Chapter 5 presents the result of the survey conducted which investigate influencing factors. In Chapter 6, we discuss the results from the case study which sought to understand the benefits and challenges of BDD in active learning environments. Following, we reason about the findings from each study in Chapter 7. Finally, Chapter 8 summarizes the research conducted, highlights the main contributions, discusses threats of this study and pin points future research opportunities.

2. BACKGROUND

2.1 Agile Methodologies

Among its principles, traditional software development methodologies require a good amount of effort to be put in the requirements engineering phase [55]. This approach can be difficult to maintain in scenarios where the requirements of software development projects are constantly changing. For instance, using traditional software development methodologies in an environment of constant change can lead to the development of software requirements which are misaligned with the expectations of the customer.

In modern software development, change is a constant and it is often caused by external and uncontrollable factors [8]. As markets and economies quickly and unpredictably change, traditional software development practices, tool and techniques become difficult to apply and to follow appropriately. In addition, these changes impact on the software development project, which commonly grows both in scope and cost, a phenomenon known as *Scope Creep* [8]. This results in high costs for the development, maintenance and update of software products, thus reducing the competitiveness of software development companies.

In this context, as faster and more flexible software development techniques became more necessary, in 2001, the “*Manifesto for Agile Software Development*” [23] was created. As core principles for software development, the *Manifesto* proposes valuing:

- *Individuals and interactions* over processes and tools;
- *Working software* over comprehensive documentation;
- *Customer collaboration* over contract negotiation and;
- *Responding to change* over following a plan.

Thus, Agile Methodologies, as defined by Pressman [60], are a modern approach to develop software. By embracing continuous change, emphasizing quick deployment of functional software and relying heavily on a close collaboration with the customer during the development lifecycle, agile is suitable to be applied in a variety of projects, even outside of the software development realm [71].

Although the *Manifesto* states that less value is put in the processes used for developing software, Agile has some commonly applied development approaches. Some of the most commonly applied are *Extreme Programming (XP)*, *Feature-Driven Development* and *Scrum* [5].

2.1.1 Extreme Programming (XP)

Beck [9] proposes some fundamental values to be adopted when using XP as the development approach. These include *communication*, *simplicity*, *feedback*, *courage*, and *respect*. Each one of these relies on some practices.

Communication should happen informally and closely with all stakeholders in the project. *Simplicity* is achieved by focusing on the current requirements and needs for the software, avoiding over-engineering portions of the software to handle aspects such as scalability. *Feedback* should come from the software development team, unit tests developed by software development team and from the customer. *Courage* should be a trait of the software development team and used to properly avoiding development of “*future-proof*” yet cost-full components Finally, *respect* should be a principle for all participants in the development process.

XP development is composed of a set of activities, which are iteratively performed. These activities are *Planning*, *Design*, *Coding* and *Test*. Figure 2.1 illustrates the XP development cycle, phases and activities (or artifacts) of each phase.

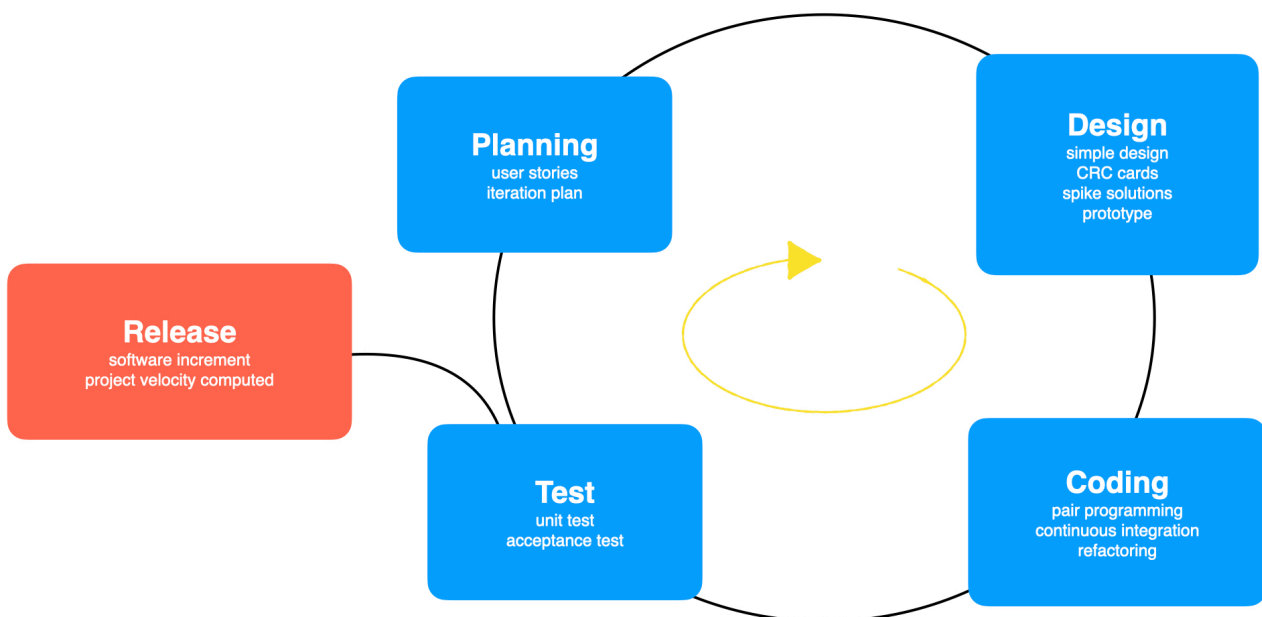


Figure 2.1 – The XP development cycle
[Source: Author (2020)]

Planning creates the set of *user stories* (descriptions of functionality, expected outputs and features of the software). These *user stories* are generated from another activity, known as *listening*, where the XP team meets and understands the business needs. Each *user story* is defined by the customer and then has its cost estimated by the XP Team. After

this, the XP team and the customer choose a subset of *user stories* to be delivered in the next release. The XP teams then organizes and prioritizes the selects stories.

Design, in XP, encourages a series of practices to be followed when developing the software. The KIS (Keep It Simple) principle, as the name implies, guides development to follow the idea of simplicity over complexity as much as possible. Engineering extra functionality considering a future scenario is discouraged.

To help with the design of functionality, XP often uses CRC (class-responsibility-collaborator) cards. These cards help structure the system and allow the team to keep the object-oriented design philosophy, a key component of XP development. The cards are also the unique design work product generated by the XP team.

As for design optimization, XP makes usage of a technique called *refactoring*, which should improve the internal quality of software while maintaining software functionality intact. Finally, it is important to note that the design process, in XP, should happen both after and before the *Coding* phase.

Coding is where the XP Team develops the functionality defined by the *user stories*. One important point aspect of this phase is that, before the *Coding* phase begins, a set of small failing unit tests is created. The testing set should reflect the expected functionality of an *user story* and developers should produce code that passes the written tests. This approach aims at ensuring that development only covers the chosen *user stories*, thus reducing the need for engineering for the future.

A practice that used in XP for the *Coding* phase is *pair-programming*. In it, two developers work closely to develop a portion of code. This technique ensures that code being developed is constantly reviewed and checked, which reduces the occurrence of bugs. Another benefit of *pair-programming* is the decentralization of knowledge in the code base, as at least two developers participate in the construction of every software portion.

Test is a constant in XP, as even before start development of code, unit tests are created to ensure correct functionality is implemented. Thus, in XP it is a practice to have an automatic and updated set of tests, which should be easily configurable and executed.

In XP, the set of unit tests play a relevant role in many phases, such as when refactoring happens. This set should be composed of unit, integration and validation tests, giving the XP team the ability to quickly diagnose (and fix) problems. Moreover, *Acceptance Tests*, which are tests specified by the customer, aim at ensuring general correctness of *user stories* which are visible and reviewable by the customer.

2.1.2 Feature Driven Development (FDD)

Feature Driven Development (FDD) is an agile development process initially proposed by Peter Coad and Jeff de Luca, who briefly outlined it in the book “*Java Modeling in Color with UML*” [16] in 1999. A more definitive manual for FDD, “*A Practical Guide to Feature Driven Development*” [56], was then created by Stephen Palmer and Mac Felsing in 2001.

FDD is centred in the idea of features. In this context, as defined by Coad [16], a *feature* is as function which is valuable for the client and which can be developed in up to two weeks.

FDD primarily focuses on two phases. The creation of a list of features to be developed, which is constantly updated, and the iterative development of individual features. As guiding principles, FDD follows the agile approach by promoting collaboration among the FDD team, relying on the decomposition of complex functionality in small features which are integrated as development proceeds and encouraging high levels of communication.

Pressman [60] states that some of benefits of FDD are the facility for the users to describe the features they want, together with any potential relations and problems that can arise from it. Besides that, features can be easily clustered by their business relation, can be easily inspected due to their small size and can drive planning, scheduling and tracking. Finally, working software is delivered every two-weeks.

As a guide for defining a feature, Coad [16] proposes the following template:

<action> the **<result>** **<by | for | of | to>** a(n) **<object>**

Where **action** is the abstraction the action that is being implemented (such as *Add, Save, Ensure, etc*), **result** is the item to be added/modified by the action (such as *Profile Information, Shopping Cart, New Item, etc*) and **object** is the affected element of the system. Considering a mobile banking application, features could be defined as:

- *Display the balance of an account.*
- *Perform the payment of taxes from an account.*
- *Store the credentials of an account.*

From this set of features, FDD creates categories aligned with the business expectations. The template for creating these categories are defined by Coad as:

<action> **<ing>** a(n) **<object>**

Following the mobile banking application example, an example category could be defined as *Managing an account*.

FDD proposes 5 framework activities (called *processes*), these being, as proposed by Coad [16]: *Develop an overall Model, Build a Features List, Plan by Feature, Design by Feature* and *Build by Feature*.

FDD, apart from other agile practices, emphasises project management during development as it is crucial for stakeholders, managers and the team to understand the project status. This allows for the identification of both achievements and problems, which in turns should guide planning for what features to be developed.

2.1.3 Scrum

Scrum is an agile software development methodology defined by Ken Schwaber and Jeff Sutherland [70], who developed it based on previous work from Takeuchi and Nonaka *et al.* [80]. In 2011, Scrum was used in over 75% of Agile implementations worldwide [79].

As guiding principle, Scrum relies heavily on the complexity theory. As such, it treats the process of developing software as a complex problem that can only be loosely controlled.

The definition of Scrum is, as stated in the Scrum Guide [70], “*A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value*”. Scrum also relies on three fundamental pillars: *Transparency, Inspection* and *Adaptation*.

The team which uses Scrum for development is also known as Scrum Team. The Scrum Team participants are divided in three roles, the *Product Owner*, the *Development Team* and the *Scrum Master*. Each one of these performs specific tasks.

The *Product Owner* maximizes the value delivered by the product. Some of the responsibilities of the *Product Owner* include ensuring *Product Backlog* items are clearly expressed, defined and understood by all team members, prioritizing items in the *Product Backlog* to maximize value delivered and ensuring the *Product Backlog* is both visible and transparent to all team members.

The *Scrum Master* is responsible for ensuring the Scrum processes and events are properly set and transparent for the development team. The person with this role should help team members to follow the Scrum Guide principles, practices and values.

The *Development Team* is composed of all professionals who work to deliver potentially shippable products at the end of the Sprint. These professionals should be self-organized and have autonomy to make decisions of how to implement certain features. The Scrum Guide suggests the size of the *Development Team* to range between 3 and 9 mem-

bers, where having less can increase the risk of skill constraints and having more leads to coordination issues.

There are 4 events defined by Scrum, these are *Sprint Planning*, *Daily Scrum*, *Sprint Review* and *Sprint Retrospective*. All these events revolve around the *Sprint*.

The *Sprint* is the core part of the Scrum framework. It is a time-boxed period of time, no longer than a month, where the Scrum team works towards developing an increment of product, which can potentially be released. During the period of the *Sprint*, Scope can be further discussed between the Product Owner and the Development team, however it is important that no changes are made that can make it difficult to achieve the Sprint Goal.

The *Sprint Planning* happens at the beginning of a Sprint. The *Product Owner* selects a set of stories from the *Product Backlog*. These stories are then added to the *Sprint Backlog*, which the Scrum team commits to work on during the Sprint. Each story from this subset of stories is, then, split into small and focused tasks. The planning is also where the team estimates the complexity of the tasks using hours or points. The estimation process helps the Scrum team to make better decisions regarding the amount of work that can be done during a Sprint, allowing the team to avoid overestimating or underestimating its working capacity.

The *Daily Scrum* is a short meeting performed by the Scrum team. In the *Daily Scrum* each team member answers three questions. As defined in [70], Those are:

1. *What did I do yesterday that helped the Development Team meet the Sprint Goal?*
2. *What will I do today to help the Development Team meet the Sprint Goal?*
3. *Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?.*

This meeting helps the Scrum team to constantly inspect the development process and allows to maintain transparency regarding both progress and impediments.

The *Sprint Review* is performed at the end of the Sprint. It is an informal meeting, usually no longer than four hours for a four-week Sprint, held by Scrum Team, *the Product Owner* and key stakeholders of the product. The main goal of this meeting is for the *Scrum Team* to demonstrate what items of the Product Backlog have been developed and what items ended up not being implemented despite being chosen in the *Sprint Planning*. The *Sprint Review* is also where the *Scrum Team*, the *Product Owner* and the stakeholders collaborate to adapt, remove or add items to the *Product Backlog*.

The *Sprint Retrospective* is an event that the Scrum Team uses to inspect and adapt itself, creating a plan of improvement. Any tools, processes and relationships are reviewed based on the events of the Sprint. It is important that Scrum Team identifies what were the strong points, which should be maintained, and weak points, which should be

corrected or eliminated. The meeting outputs a plan for improvement and the Scrum Team commits to executing the plan during the next development iteration.

The complete cycle of Scrum is illustrated in Figure 2.2.

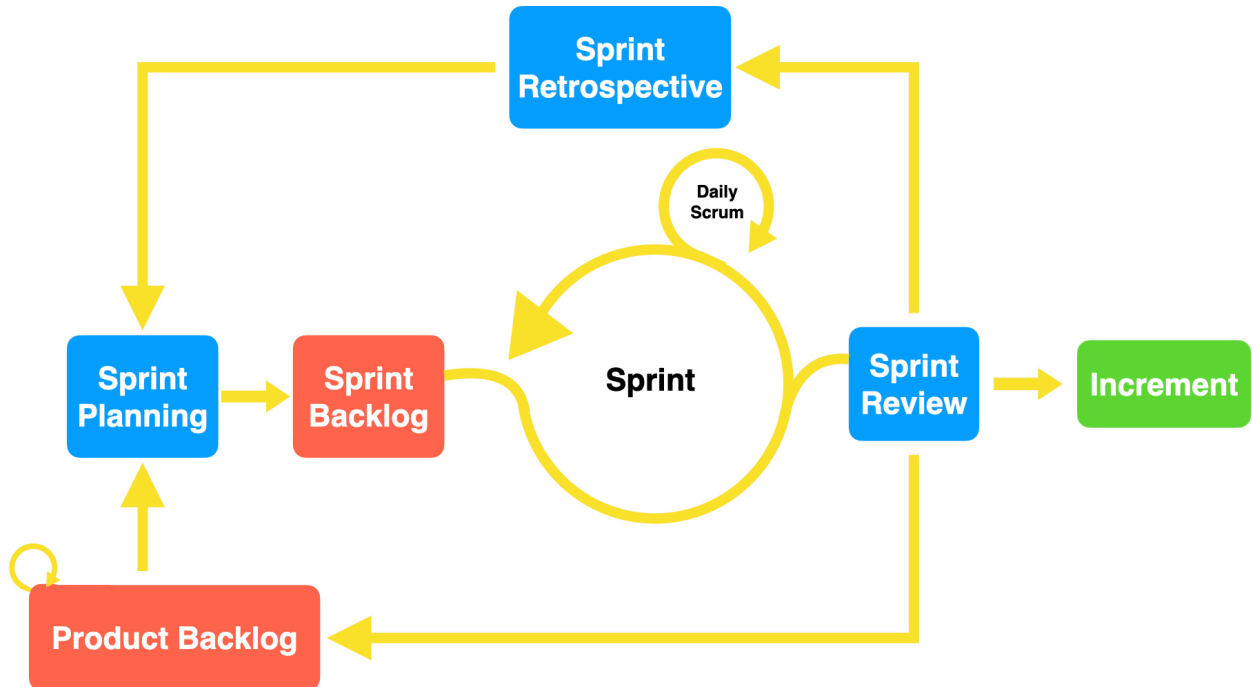


Figure 2.2 – The Scrum development cycle
[Source: Adapted from [70]]

All these agile software development methodologies help to solve some of the problems faced by the software development industry. As such, there are many research work which attempts to better frame aspects such as the effectiveness, practices and adoption of these methodologies.

However, simple adopting agile software development does not ensure that software being developed will be aligned with the needs of the customer, although it has been demonstrated to improve many aspects of the software development process.

Attempts to reduce some of these issues have been proposed in the past. One of these is what is known as Behavior-Driven Development.

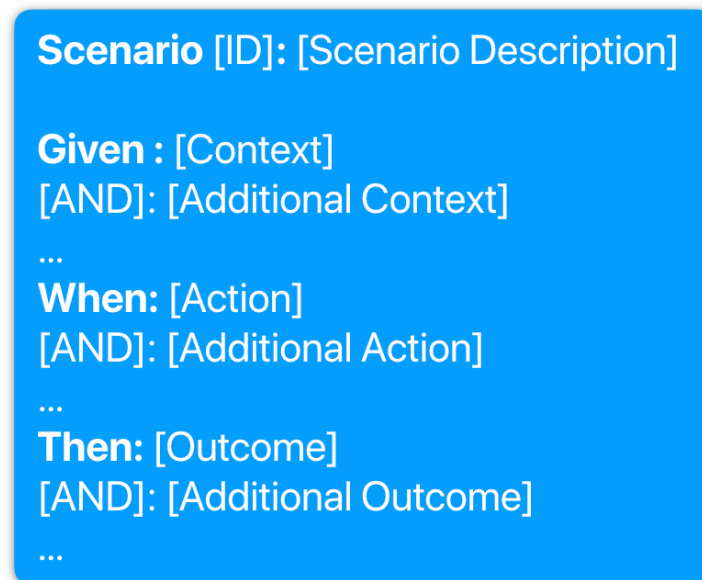
2.2 Behavior-Driven Development (BDD)

Behavior-Driven Development is a software development practice, proposed by Dan North [50], which aims at helping software development teams to build software which follows the needs of the customer.

In his initial proposal, Dan North [50] created BDD as an adaptation of Test-Driven Development (TDD) [10]. Later, however, BDD incorporated other software engineering practices, enabling it to be used across the entire software development lifecycle [75].

As a development methodology, BDD emphasizes test cases which are written in a common language, derived from Domain Driven Design [19]. The specification of these test cases is done using scenarios (also known as BDD Scenarios), which should describe a feature of a system.

BDD Scenarios are used to further enhance the descriptive capabilities of user stories, which are commonly used as lightweight requirements specification in agile software development. Regarding formatting for specifying these scenarios, a structured language is usually applied, known as Gherkin [14]. Figure 2.3 shows a possible template, based on Gherkin, which can be used to specify scenarios.



```
Scenario [ID]: [Scenario Description]

Given : [Context]
[AND]: [Additional Context]
...
When: [Action]
[AND]: [Additional Action]
...
Then: [Outcome]
[AND]: [Additional Outcome]
...
```

Figure 2.3 – An example template for specifying scenarios
[Source: Adapted from [75]]

Besides on its description, and based on this template, BDD scenarios can be split into three core elements: *Given*, *When* and *Then*.

1. *Given*: The context assumed for this scenario execution, *e.g.*: “Being logged in” or “Being on the home screen”.
2. *When*: An action or event which happens given the provided context, *e.g.*: “Press the login button” or “Type a character”.
3. *Then*: The expected outcome of the system for the provided action and context, *e.g.*: “Present a success alert” or “Redirect to Home Screen”.

In addition to this, each element can have multiple additional context. This is expressed in the template by the word “AND”.

In terms of key characteristics associated with BDD, Solis *et al.* [76] defined a few key characteristics which are inherent to BDD. Highlighting a few of those, BDD is composed of:

1. **Ubiquitous Language** - During development, stakeholders and the development team should be able to cooperate and communicate using a common language. This language should contain enough terms so that any idea regarding the software product under development could be discussed.
2. **Iterative Decomposition Process** - Development should happen iteratively with provided time slot as in the beginning of the development process both the customer and the development team are not certain about the requirements of the software being developed. This process is facilitated by using the Ubiquitous Language, also proposed by BDD.
3. **Plain Text Description with User Story and Scenario Templates** - The requirements specified should follow a pre-determined template. This template usually revolves around using plain text user stories which are augmented by BDD scenarios.
4. **Automated Acceptance Testing with Mapping Rules** - After extracting behavior using stories augmented with scenarios, it becomes a task for the development team to properly translate this behavior into actionable tests. Ideally, these tests should be easily created from the plain text specification. This is usually achieved by using a BDD development toolkit.
5. **Readable Behavior Oriented Specific Code** - When implementing the requirements specified previously, the generated code (methods, classes, etc) should clearly indicate what its purpose is. This implementation code should describe the system behavior and follow the ubiquitous language defined for the project.

BDD is also a very flexible and powerful framework, which can be used across the entire software development lifecycle [75] and help companies' software that is better aligned with the expectations of stakeholders.

2.3 Active Learning

Teaching students is an activity which has recently seen some changes. Traditionally, learning is mostly based on lectures, a teacher-centered approach which usually

provides no opportunity for interaction. As such, most activities performed in traditional lecture-based classes offer low levels of autonomy, which is highly important for student engagement [33]. In addition, many educators have recognized that although learning can happen passively, through reading and listening, it produces far better outcomes when students are learning actively [15].

In this context, active learning proposes high levels of interaction and stimulates students to perform not only low-order cognitive tasks, such as reading and writing, but also high-order ones, such as debating, analysing and decision making [21, 4, 25]. Even further, active learning has been shown to improve student motivation and engagement [15].

2.3.1 Challenge Based Learning (CBL)

Several active learning frameworks have been proposed and are used in the educational context. Some of these include Problem Based Learning, Project Based Learning, Task Based Learning and Challenge Based Learning (CBL). Each one of these frameworks present active-pedagogy characteristics, such as being student-centered and relying heavily on the usage of high-order cognitive skills, such as reasoning of data and research synthesis. It is relevant to note that, from this set of frameworks, CBL is the only one that focuses on solving real world problems [48].

As such, Challenge Based Learning (CBL) is a learning framework created by educators jointly with *Apple Inc.*. In CBL, according to Nichols *et al.* [48], the learning process happens with active engagement of participants (students, teachers, families, communities) and while solving real-world challenges. Complementarily, the focus of CBL is the entire learning process rather than the final deliverable. In order to achieve this, CBL encourages students to constantly reflect during the learning process.

As a flexible framework, CBL is divided in a set of stages. These stages are *Engage*, *Investigate* and *Act*. Figure 2.4 presents a graphical scheme of the complete CBL framework and highlights some of the sub-activities for each stage.

As such, each of the high-level stages of CBL can be further divided into a subset of activities. For each one of these stages:

- **Engage**

Big Idea: a large theme or concept which can be explored, such as *Health* or *Education*. It has to be a topic that is engaging for students;

Essential Question: a question related to the *big idea* which students eager to explore;



Figure 2.4 – Challenge Based Learning Framework
[Source: Nichols *et al.* [48]]

Challenge: a call to action derived from the essential question. It should be actionable and exciting.

- **Investigate**

Guiding Questions: all questions related to the challenge. Includes everything that needs to be learned;

Guiding Activities and Resources: the list of activities and resources that support students to pursue the challenge;

Analysis: sets the foundation to develop a solution to the challenge.

- **Act**

Solution Development: based on findings from the previous steps, a solution is implemented;

Evaluation: verifies if the solution has addressed the challenge or if it needs refinement.

Reflection: engages students into thinking about the entire learning process.

As for the execution, CBL begins with the definition of the *big idea*, which is a broad concept that can be explored in several ways. It is important for the big idea to be engaging and relevant to students. After choosing the big idea, students proceed to define the *essential question* and *challenge*. From this point, students must come up with the *guiding questions* and *guiding activities and resources*, which will guide them to develop a successful solution. The next step is *analysis*, which will set the foundation for the definition of the *solution*. Once the solution is agreed upon, the *implementation* begins. Finally, *evaluation* is undertaken in order to check out the whole process and verify if the solution can be refined.

3. STATE-OF-THE-ART

In order to have a proper understanding of the *state-of-the-art* literature regarding the topic of teaching BDD, we have chosen to perform a systematic literature review.

3.1 Systematic Literature Review

According to standard guidelines for performing this research method [30], a systematic literature review is suited for acquiring theoretical background about a certain topic. This, then, enables other related research activities to be conducted. In addition, this type of review allows for the identification of benefits and challenges being reported in other research, which can also further inform decisions regarding research activities.

In this sense, we have followed these standard guidelines and performed our search. At this point, it is important to mention that the research question used for the systematic literature review focused on the concept of teaching BDD more generally, not constrained to only mobile software development or active learning environments. This was due to a low count of studies found when more constraints were applied.

3.2 Methodology

As stated previously, the research aimed at characterizing the *state-of-the-art* literature on teaching of BDD. As such, the research question to be addressed in this study is: “*What is the state-of-the-art in the literature in regards to the teaching of BDD?*”. In order to answer this question, a systematic literature review was conducted. The review design was based on some of the most relevant studies in the area [31, 32].

3.2.1 Research Question

In order to answer the broader research question of the study, “*What is the state-of-the-art in the literature in regards to the teaching of BDD?*”, we have split it into three sub-questions, those being:

- (RQ1) *What tools, models, methodologies, frameworks and software are used when teaching BDD?*
- (RQ2) *Are there any best practices when teaching BDD?*
- (RQ3) *What are the impacts of teaching BDD?*

RQ1 aimed at discovering tooling that is applied when teaching BDD to software engineering students. Findings from this question may be either new tooling (models, tools, frameworks, *etc*) or even tooling which is not new but which sees use when teaching BDD. **RQ2** had the purpose of finding what teaching practices became specially relevant and work successfully when teaching BDD. Finally, **RQ3** intended to understand the outcomes that teaching BDD produced.

3.2.2 Data Source and Search Strategy

As a way of finding studies with high relevance to the research question proposed, we defined our search string following the guidelines proposed by Kitchenham *et al.* [31]. Consequently, the search string addresses the *Population*, *Intervention* and *Outcome* expected. We have chosen exclude *Comparision* and *Context* as the research conducted followed the principle of exploratory research. Table 3.1 summarizes the search string used.

Table 3.1 – Search string

Population	(Software Engineering OR Software Development) AND
Intervention	(Behavior Driven Development OR Behaviour Driven Development OR BDD) AND
Outcome	(Education OR Undergraduate OR Graduate OR Teaching OR Training)

In terms of search strategy, we have also followed the guidelines proposed by Kitchenham *et al.* [31]. Table 3.2 presents the summary of the applied search strategy (*i.e.*, Databases and inclusion criteria). All available databases were selected with the exception of Citeseer library, Inspec, El Compendex and Science@Direct, due to difficulties using these platforms. The minimum publication year was chosen based on the first proposal of BDD by Dan North [50] in 2006. The selection criteria for the search was defined based on the goal of the study. Work neither written in English nor published in any journal, conference, workshop or symposiums was not considered. Regarding the number of pages, which usually reflects the depth of the analysis conducted by the authors, we have chosen to accept a minimum of 4 pages, as our intention was to capture even results from preliminary studies.

From this point, we have applied our search string and used the search criteria specified above to all the specified databases. Regarding organization of the results, we have created a spreadsheet which contained the necessary information, meaning that we could apply the established inclusion criteria.

Each element in the spreadsheet contained metadata about the retrieved papers. Table 3.3 presents the attributes assigned to each paper.

Table 3.2 – Search strategy

Databases searched	ACM Digital Library IEEEExplore Scopus
Selection Criteria	available online written in English from 2006 to May 2019 in: Journals/Conferences/Workshops/Symposiums 4 pages minimum
Search applied to	Title Abstract Keywords

Table 3.3 – Metadata information of each paper

Information retrieved	Explanation
Database	ACM, Scopus, IEEEExplore
Title	Paper title
Authors	List of all authors
Type of forum	Journal, conference, workshop, symposium
Abstract	Paper abstract
Keywords	Paper keywords
Control 1	Duplicate
Control 2	Do not fit into criteria
Control 3	Is relevant

In addition to the standard information of the papers, we have added three additional attributes, which were related to the possibility of exclusion of a given paper. These attributes were assigned based on other attributes from the paper, such as abstract and keywords. The objective of these attributes was to better categorize papers, thus answering three additional questions:

- *“Is this paper duplicated?”*
- *“Is this paper relevant?”*
- *“Does this paper fit in the criteria?”*

In this sense, Table 3.4 presents the papers retrieved from each base. This generated a total of 24 papers which would be fully analyzed. It is important to note that a complete analysis of papers could give a better understanding of their research type, thus excluding or maintaining it this systematic review.

Table 3.4 – Search results

Base	Papers Found	Selected Papers
IEEEExplore	216	17
Scopus	67	3
ACM	7	4
Total	290	24

3.2.3 Data Extraction and Classification

From the initial metadata assigned to the 24 selected papers in the spreadsheet, we have also added some additional attributes. These were:

- **Contribution Facet:** Type of contribution (based on the work from Shaw [72]);
- **Research Method:** The research method applied (case study, survey *etc*);
- **Research Type:** Type of research (based on the work from Wieringa *et al.* [84]);
- **Paper Relevance:** A grade from 0 to 10 (based on the work from Salleh *et al.* [67]). For further details, see Table 3.5;
- **Contribution:** The research contribution from the study.

Further information regarding these attributes is presented in subsection 3.2.4.

3.2.4 Classification Scheme

For each proposed additional attribute assigned to the selected papers, which covered paper quality, method, research type and contribution, we have created a classification criterion.

Regarding paper relevance, we have assigned a score for each paper, based on the work from Salleh *et al.* [67]. In total, nine (9) questions were used to provide a grade to the work under analysis. Each question could be either completely answered, meaning the the work would be assigned the complete score for the question, partially answered, meaning that the work will be assigned half the points for the question, and not answered, meaning that the work will be assigned zero points for the question.

Additionally, we have chosen to better classify papers which were closely related to our research questions, thus these questions are worth 4 points in total. Once a study is graded, it is assigned a quality category, meaning that the study possesses:

- **High quality:** 8 to 10 points;
- **Medium quality:** Between 5 and 8 points (excluded);
- **Low quality:** 0 to 5 points (excluded).

In summary, we have created a completed classification scheme and present it in Table 3.5.

Table 3.5 – Classification Scheme

Category	Description
<i>Paper Relevance Facet</i> References Goal Sample Observation Research Method Clear Description Findings RQ1 RQ2 RQ3	Is the study well referred? (1 point) Is the goal clearly stated? (1 point) Data collection and sample strategy was carried out correctly? (1 point) The analysis methodology was well applied? (1 point) Is the context of the study clearly described? (1 point) Are findings credible? (1 point) Does que paper answer RQ1? (1 point) Does que paper answer RQ2? (1 point) Does que paper answer RQ3? (2 points)
<i>Research Method Facet</i> Case Study Empirical Study Experimental Study Survey	A report from a specific situation being studied [32]. Study based on empirical evidence [57]. A study in which an intervention is introduced to observe its effects [74]. A process to collect data, analyze the information and report the results [58].
<i>Research Type Facet</i> Evaluation Research Experience Paper Opinion Paper Philosophical Paper Solution Proposal Validation Research	Evaluation of a method or technique in practice. Personal experience of the author depicting how something has been done in practice. Personal opinion on whether a certain technique is good or bad. New way of looking at an existing context. The proposition of a solution to a problem. New techniques being implemented in experiments, simulations or in practice.
<i>Contribution Facet</i> Advice/Implication Framework/Method Guidelines Lessons Learned Model Tool	Recommendations based on personal opinions. Framework/Method used to teach (or to learn) behavior-driven development. Advices based on the research results. Actionable advices derived from the obtained research results. Representation of a given context based on a conceptualized process. Tools used to teach (or to learn) behavior-driven development.

Following this, Section 6.2 presents the results from the systematic literature review.

3.3 Results

A complete analysis of papers, from the 24 selected studies, has enabled us to perform a more fine-grained exclusion of studies. In this sense, after reading selected studies, we have excluded 21 non-related papers, thus reducing the selected papers to three.

All these three studies were then classified based on the classification scheme presented in Table 3.5. Results are presented in Table 3.6

Table 3.6 – Overview of results

1st Author (year)	Research method	Research type	Contribution	Paper Quality
Matthies (2017) [41]	Empirical Study	Evaluation Research	Tool	6
Hoffman (2014) [26]	Case Study	Evaluation Research	Lessons Learned	5
Simpson (2017) [73]	Empirical Study	Experience Report	Lessons Learned	3

It is important to note that even this final list of studies does not fit directly with the proposed research questions. This can be inferred by the paper quality of the studies, which is six at the most.

3.3.1 Research question analysis

After analysing and classifying studies, we have proceeded to answer the research questions proposed in this study.

(RQ1) What tools, models, methodologies, frameworks and software are used when teaching BDD?

Regarding our first research question, some of the selected studies presented relevant insights.

Mathies *et al.* [41] have demonstrated Prof. Cl., a tool used to create and reinforce testing habits when teaching Test-Driven Development (TDD) in Massive Open Online Courses (MOOCs). The tool is somewhat relevant to our research question as BDD was originally conceived from TDD and thus shares few tools.

The tool presented is open source and it is used to teach TDD concepts directly on GitHub. Students are given *issues* in a software project that they have to solve. It is relevant to mention that all issues are written as user stories and augmented using a Gherkin language (Given-When-Then).

Hoffman *et al.* [26] have conducted a case study that reported their experience when using Acceptance-Test Driven Development (ATDD), a test-driven development method-

ology which shares concepts with BDD, to develop a real-time system using Problem Based Learning (PBL) [7]. In the study, the most relevant tools reported to be used when developing the system were Robot Framework [35], Selenium, and Coverage [82].

In the context of the research question, the Robot Framework is especially relevant, as it is, according to Hoffman *et al.* [26]: “[...] a generic test automation framework for acceptance testing and ATDD. It utilizes keyword driven testing approach, as well as data driven and Behavior Driven Development [...]”.

In terms of methodologies, the study reports using agile software development and PBL. The authors justify this choice due to the interdisciplinary background of students who developed the system.

Simpson *et al.* [73] presented an experience report of changes performed to their software engineering course. After running the course multiple times, the authors noticed that the engagement of students was decreasing due to the use of unrealistic projects (*i.e.*, toy problems), outdated methods, among others.

The study mentions the changes which were applied, such as using agile software development in its practical project and changes to the material, which now includes the introduction of behavior-driven development. Besides that, students now work with real-world customers and classes are taught using a Flipped Classroom methodology.

In conclusion, although the number of studies found about the researched topic is low (*i.e.*, only three), some methodologies and tools have appeared and can be used as indicatives to what may work. A summary table of these results is presented in Table 3.7.

Table 3.7 – Summary of results from the studies

Study	Tools(s)	Learning framework	Agile framework	Dev. practice
Matthies (2017) [41]	Prof. CI.	-	Custom	TDD
Hoffman (2014) [26]	Robot, Selenium and Coverage	Problem-based learning	Scrum	ATDD
Simpson (2017) [73]	-	Flipped-classroom	Scrum	TDD

(RQ2) Are there any best practices when teaching BDD?

Regarding our second research question, as none of the studies is highly linked to this goal of this study, it is difficult to argue about it. In this sense, extracting data from the selected papers, we have indicatives that:

1. As for software development methodology, either Scrum [73, 26] or a custom agile framework [41] was applied by the selected studies. This provides indicatives that BDD was suited to be taught in an agile software development context. This is aligned with Smart *et al.* [75] which states that BDD works best in an iterative context.

2. As for tooling that can be used, studies have reported using different tools [41, 26] or have not reported the applied tools [73]. In this sense, it becomes hard to conclude if a tool can be said to generate better results.

In conclusion, although we have not found sufficient evidence that supports any best practice, some tools and software methodologies have appeared and should be considered when teaching software practices that have a strong focus on testing, such as Test Driven Development, (TDD) Acceptance-Test Driven Development (ATDD) and Behavior-Driven Development (BDD).

(RQ3) What are the impacts of teaching BDD?

As none of the studies selected applied BDD in their teaching environments, our reasoning focused on the impacts of applying a test-based development methodology, which englobes TDD, ATDD and BDD.

In a general sense, Simpson *et al.* [73] reported that the changes applied in their teaching method seem to have had a positive impact. Every participant in the study reported an improvement when executing their main task, such as students being more involved in the development and industry mentors having the chance to pass on their experience. This result is difficult to be linked to a single change, such as using TDD, due to the study reporting many modifications to the course under study.

Specifically regarding implementation, according to Hoffman *et al.* [26], it appears that developing using a combination of an active learning framework (*i.e.*, Problem-Based Learning) and ATDD requires dedication and commitment from those involved in the process. The author reinforces that ATDD was essential in the project, having an impact from the requirements specification to the software quality and developer testing importance awareness. Moreover, even though not directly linked to ATDD, the interdisciplinary approach also seems to have had a positive impact in the study.

3.4 Threats to Validity

Although results obtained by this systematic literature review provide a summary of other research being conducted, some threats to validity are present and may diminish the relevance of findings reported.

Firstly, even ensuring steps of the review were properly followed, there are many ways in which errors could have been introduced, such as when reading abstract of papers, performing the automatic search in the bases and applying the selection criteria.

In addition, even if the bases searched are standards for the type of research conducted in this study, the excluded bases may contain other papers with relevant data. As such, it is difficult to establish the results presented here as final.

3.5 Conclusion

This chapter presented the results from a systematic literature review about the teaching of BDD. The results were analysed from the perspective of three researched questions.

Our preliminary results indicate that there is a gap of research in this topic, as very few academia studies specifically address the teaching of BDD.

In this sense, we have found indicatives of tooling used when teaching topics which are closely related to BDD, such as Test Driven Development (TDD) and Acceptance-Test Driven Development (ATDD). In addition, we have not found enough evidence about best practices when teaching BDD, which is probably due to the lack of studies addressing the topic.

4. EXPERT PANEL

This chapter presents the results of an expert panel study performed with active learning environments experts. These experts were from four different countries and had proven experience teaching in active learning environments. The study aimed at obtaining insights regarding potential benefits and challenges of using BDD in active learning environments.

4.1 Methodology

In order to achieve the goal of this study, we have chosen to perform an expert panel. Expert panel research can be used to capture expert judgment [65] and aid in the improvement of hypothesis generated. In addition, experts are usually better at predicting and even preventing [36] errors when performing certain tasks. Furthermore, the opinion of experts is recognized as a valuable research artifact in the Software Engineering (SE) research community [18].

In this scenario, our study has an exploratory approach and is qualitative. To ensure reliability of results, we have followed the guidelines proposed by Maxwell [42]. As such, the goal of this research is *to understand the opinion of experts regarding the benefits and challenges of teaching BDD in active learning environments*. Thus, we have established two research questions:

- (RQ1) *What are the positive aspects reported by experts regarding teaching BDD?*
- (RQ2) *What are the negative aspects reported by experts regarding teaching BDD?*

4.1.1 Protocol

To answer these questions, we have sought the opinion of active learning teaching experts. These experts were selected due to their expertise teaching in active learning environments and due to convenience. Experts were not required to have any prior experience with BDD to participate in the study. As a result, researchers performed a levelling activity with all participants in the study prior to asking any questions.

For this levelling activity, we have split participants in two groups and performed a workshop in each group to introduce the core concepts provided by BDD. This was performed to ensure that all experts had a minimum understanding of BDD. We have decided

to conduct the levelling activity prior to presenting any research objectives primarily to avoid any biases in the answers.

The workshop conducted were structured as follows. A researcher would introduce the concepts, naming and main activities of BDD to participants through lecturing. After that, a practicing activity was performed. This activity was split in two parts. The first part had participants working together in groups to idealize a mobile application and write lightweight specifications (*i.e.*, user stories) for it. These *user stories* were augmented by BDD scenarios. Participants were told to write these specifications in an easy-to-exchange format.

Following this structure, groups had to switch specifications with other groups and create a low-fidelity prototype using paper, pen and any prototyping app with which they were comfortable. We have suggested using Proot¹. After that, participants had the chance to show their prototypes to the group which created the specifications originally.

Finally, following this activity, experts were presented to the research objective and terms. Participation in the study was clearly stated to be voluntary. Experts who were interested in participating in the study were presented to this questionnaire used in the research. Excluding demographic questions, the questions used are presented in Table 6.1.

Table 4.1 – Questionnaire

#	Question
1	Have you ever taught BDD?
2	Are you currently teaching or using BDD in your learning environment ?
3	In your opinion, what are the main benefits of BDD in your learning environment?
4	In your opinion, what are the main challenges of BDD in your learning environment?
5	Any final comments or thoughts?

A general view of the entire process is presented in Figure 4.1.

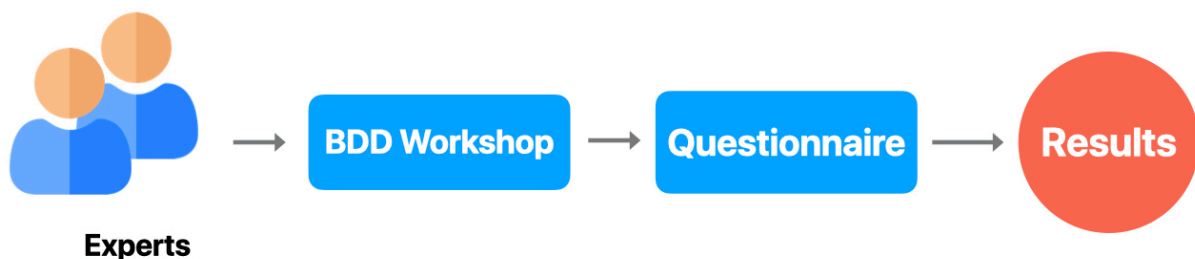


Figure 4.1 – The overall research process
[Source: Author (2020)]

The first two questions were used to assess the familiarity of participants with BDD. Our goal with these questions was to attempt extracting different perspectives from partic-

¹<https://prottapp.com/>

ipants. Thus, this would help us to assess how experience influences the perception of experts.

Questions three and four would directly aid us to answer our research questions. These questions were personal and open-ended.

Finally, the final question provided a space for additional comments to be written.

4.1.2 Data collection & Analysis

The questionnaire provided to participants was available online and participants could either answer using the mobile devices or laptops.

After collecting answers, we have organized them using a spreadsheet. These spreadsheets contained all answers from participants in a textual manner.

From this point, we began a qualitative analysis of data, following the principles of Maxwell [42]. The analysis was done using two strategies: clustering and categorization. We first clustered answers which presented similar results, counting the number of occurrences of a particular topic. In addition to this, as a qualitative study, we have collected interesting insights from participants. Afterwards, the generated clusters were categorized (which could be one or more) regarding the general topic they addressed. As a final step, we have grouped similar categories.

Finally, using the available data and generated analysis, we were able to extract insights from the data. The results of our analysis are presented in the following section.

4.2 Results

4.2.1 Demography

In total, our questionnaire had 28 valid responses. From a demography standpoint, participants in the study were 34.11 years old on average, and all participants had more than one year of experience in teaching, with the average number of years teaching being 7.43 years. In terms of active learning teaching, participants had at least one year of experience (6 participants) and at most six years (4 participants), and an average of 3 years of experience in such environments.

In terms of industry experience, participants had an average of 7.43 years of experience in the software development industry.

Finally, in terms of nationality, 18 participants were from Brazil, 6 were from Indonesia, 3 were from Italy and 1 was from France.

4.2.2 BDD - Experience

The first result of our research is related to BDD experience. In terms of experience with BDD, 9 out of 28 participants (~32%) had prior experience either teaching or using BDD and 4 (~14%) were currently teaching or using BDD.

These results indicate that most of our participants (~68%) were completely new to the methodology.

4.2.3 BDD - Perception of benefits

Our second result is related to positive perceptions. Results from this subsection are divided in six categories. These are:

1. *Requirements* - appearing in 17 responses;
2. *User Comprehension* - appearing in 5 responses;
3. *Project* - appearing in 4 responses;
4. *Implementation* - appearing twice;
5. *Communication* - appearing twice;
6. *Others* - appearing once each;

Each one of these will be further discussed as follows.

1. Requirements

As the most numerous response category, we have more finely specified it. As a result, requirements responses are divided in: *Elicitation*, *Specification* and *Validation*. Each of these is detailed below.

Elicitation

These results focused in the process of creating requirements for projects. As such, participants have mentioned many ways in which BDD could help to improve this process.

Participant #3 stated that *“scenarios help conceiving the App”*. This could imply that the process of creating scenarios could help during the initial stages of application development. This is reinforced by the report from participant #13, which states that *“it helps to understand and describe in a better way the features of our apps”*. In addition, it appears that BDD could reduce the gap between stakeholders in a project, as participant #4 states that *“the distance from Business Logic to implementation is reduced [...]”*.

Planning also appears to have an impact potential, as participant #12 states that BDD can help teams *“better understand the gaps in the planning of their solution, so they don’t find missing points later, and don’t miss test cases”*.

Even creativity could be impacted, as according to participant #19 BDD brings up *“ideas we didn’t think about”* and according to participant #6, *“[...] it makes the process more ludic and interactive (proximity with the client/user)”*.

Specification

In terms of specification, many reports from participants presented positive aspects that BDD can bring to the specification of requirements. These reports range from general improvement to requirements specifically, such as report from participant #3, *“better specification of user stories”* and #6 *“[...] help clarify requirements”*

In addition, BDD appears to have the potential of allowing teams to better understand features, with report from participant #10 stating that BDD *“helps debating features”* and from participant #20, stating that *“they (scenarios) make it easier to understand all the features of your app, and what is needed to implement them”*. This is further reinforced by participant #13, who says *“it (BDD) helps to understand and describe in a better way the features of our apps”*.

Moreover, the creation of BDD scenarios appears to have the potential to enhance the identification of gaps during planning. This is stated by participant #14, who says that a potential benefit is *“to create the scenarios and, by discussing them with the team, identify gaps that need to be addressed by the team”*, and by participant #17, who says that a benefit is *“to know what platform will serve the user scenario’s more adequately to match the story purpose”*.

Validation

These results focused in the process of validating requirements for projects. Regarding this matter, we had only one report from participant #25, saying that *“more organization and validation with the user”*. This could indicate that BDD could enhance the validation of requirements by incentivizing close work with the user.

2. User Comprehension

Five responses from experts associated BDD with *User Comprehension*. It appears that BDD could potentially increase the team's capacity to understand the needs of the user and thus enhance focus during development. This idea is provided by reports from participant #22 who says *"We can identify user needs and their expectation"*, participant #27, who says that BDD helps teams *"getting closer to user desires"* and participant #24, who says BDD could be beneficial to *"helping developers to identify user lives and needs"*.

In addition, BDD could help team more deeply focus on the detail of the context as stated by participant #18, who says a potential benefit is *"to give more detail on the context, so the students can have clearer idea on how the user behavior is [. . .]"*.

3. Project

Three responses are related project-wise topics and put BDD as beneficial. Participant #3 reports that BDD can be positive because it *"helps a more focused thinking"*. Participant #10 reports a similar perspective, saying that BDD *"helps making all students in a group to have the same vision of the App they are building"*. These reports may indicate that BDD could help manage their projects during development, by helping the team to have a common understanding of the application.

Report from participant #1 indicates that BDD can bring considerations regarding the project environment, because BDD can help *"to focus on the environment in which you stay and to benefit also of its 'nudges' [. . .]"*.

Finally, participant #25 reports that BDD can help by bringing *"more organization"*. This might indicate that BDD could be a helpful tool for student project management.

4. Implementation

In terms of implementation, two reports indicate a potential benefit of BDD. Participant #8 says that BDD *"appears to transform abstract concepts in more concrete ones and ease development"*. This is aligned with North's prediction that BDD can help the entire development lifecycle [50, 75].

In addition to this, participant #11 states that BDD can *"shorten the development process"*. This is an indicative that developing following the principles of BDD could have a positive impact in the duration of projects.

5. Communication

Communication was also a theme of two reports from participants. Participant #5 reports that BDD can help by “[...] *making it easier for development teams and POs to understand each other*”. This is probably linked to the initial phases of BDD, where *user stories* are augmented by scenarios.

This potential benefits of BDD are further cited by participant #15, who says BDD can “*mitigate the misunderstanding that can happen between designers and developers during the solution phase*”. This provides indication that BDD can be used as communication facilitator for team members with different expertises (e.g. developer and designer).

Furthermore, participant #9 reported that BDD “*simplifies a complex and ambiguous context subject into something easier to understand.*” Although this might be related to in-person communication, it might also be linked to the artefacts of BDD helping to better communicate the requirements.

6. Others

Apart from other categories, two participants reported topics once each. Participant #8 states that BDD “*is a great way to prototype value and teach students what matters in an app [...]*”. This could indicate that BDD can be used as a tool for things such as reduction of scope in a project.

Finally, participant #6 says “*the tool favours collaboration [...]*”. This indicate that BDD can potentially help not only to better communicate, but also to better collaborate in projects.

4.2.4 BDD - Perception of challenges

Our third result is related to negative perceptions. Results from this subsection are divided in six categories. These are:

1. *BDD* - appearing in 6 responses;
2. *Culture* - appearing in four responses;
3. *Requirements* - appearing in three responses;
4. *Team Engagement* - appearing in three responses;

5. *Time* - appearing in three responses;
6. *Others* - appearing once each;

These results are presented as follows.

1. BDD

These responses focus in the BDD framework and its proposed tools. Participant #4 states that the volume of scenarios can be a challenge, *i.e.*, *“maybe the volume of different scenarios to map can be a challenge to maintain a high engagement of the students”*. This can indicate that BDD potentially brings some change management concerns. The process of *“creating scenarios”*, reported by participant #28, is also a potential problem.

Participant #11 worries a challenge maybe *“to make the process more attractive, less boring”*, indicating that teams may not see value in using the methodology at all. This is related to report from participant #24, who says students should *“face it in a more visual way”*, which seems to indicate BDD lacks a visual appealing model.

Report from participant #14 focuses on how to let planning of scenarios more clear, *“Letting it clear for students how to plan scenarios (for those who are not familiar, or have been through few challenges so far, it may look kind of ‘dislocated’ to think in an App which does not exist yet”*. The participant brings the challenge of further specifying requirements which may not exist yet.

Obtaining familiarity with BDD is a concern from participant #22. The report states that *“students are not familiar with this and need to get familiar with it”*.

2. Culture

Four responses were centered around the *Culture* of BDD, which itself is directly related to testing. Participant #2 says that *“overcoming the barrier of creating the culture, it becomes easy”*. This could indicate that a challenge is creating the testing mindset in the students.

Adding to this, participant #11 points out that a challenge is *“developing the culture of using BDD”* and participant #25 complements this by saying that BDD is a *“different point to think”*.

3. Requirements

Three responses bring requirements as their main concern. Participant #3 states that BDD is difficult because *“students don’t write user stories. They just code. Including that practice may be difficult”*. This could indicate that in a learning environment where students have difficulties or are not used to agile practices, BDD can be challenging to be implemented.

Participant #8 points out that junior level developers, or students, may not be ready to extract scenarios from *user stories*, saying that *“(a challenge is) developing an abstract concept interpretation ability (which is) needed so that students are able to extract scenarios from user stories”*. This is corroborated by participant #18, who says that BDD can bring the challenge of *“thinking about actions before the app itself is idealized”*.

4. Team Engagement

Responses from three participants are centered around team engagement. Participant #9 states that *“it (BDD) is an extra step in the development process, so some students may feel it is unnecessary”*. This indicates that the steps associated with BDD may not bring more value than the effort required to execute it.

Report from participant #18 states that *“the students just want to code right away and feel lazy to take time and plan first. Because as we know, this is necessary and more important than the piece of code with no context and no purposes”*. This could indicate that BDD should be easier for student who see value in agile planning. Adding to this, participant #27 says that a challenge is *“making students understand value in it (BDD)”*.

5. Time

Also with three reports, time is a key factor for some participants. Participant #12 seems worried, saying *“Time for planning and using it, since, in the rush of challenges, they want to jump straight into coding, often forgetting best software engineering practices”*. This presents a potential challenge for BDD implementation, the reduced timeframe of educational projects. This is supported by participant #15, who says that a challenge is *“time to work with the students during the class plan”*.

Participant #13 reports something similar, saying that *“maybe the amount of time spent on it. With BDD we tend to document a lot of the features and possible scenarios, and this could also take a lot of time to do it. And considering that students have deadlines to fulfil, this could be a challenge.”*

6. Others

Some participants also reported implementation, project type and overall learning context as challenges. Participant #1 says that the background required for BDD is a challenge, *i.e.*, “*so many challenges as it depends also from different environments/ cultures / people / knowledge / skills etc*”.

Participant #5 worries that BDD can have a reversed effect in less experienced developers, the report states that “*since I have never used it I cannot say for sure, but I have the impression that it will be easier to make mistakes and write bad code this way, for a more experience developer it would be nice, and for the students this can be a problem*”.

Finally, participant #6 reports BDD can be difficult to implement in certain applications, such as games. The reports states that “*it can be used for service-oriented and functional apps. Maybe challenging for game apps, for example*”.

4.3 Discussion

4.3.1 (RQ1) What are the positive aspects reported by experts regarding teaching BDD?

According to experts, it appears that BDD can have a greater positive impact in the requirements phase of software development. These results are aligned with the proposal from North [50] and are documented by Smart [75]. They propose that BDD can have an impact in many portions of the software development lifecycle, *e.g.* requirements. This result is interesting as it may open the possibility to reduce the distance from academia to the industry, a key concern of software engineering education. Moreover, previous research results, such as Simpson *et al.* [73], provide some evidence that these indicatives are correct.

In addition, better user comprehension is also reported by experts. This result is somewhat related to previously reported improvement to the requirements phase of software development, specifically requirements elicitation. BDD brings requires engagement from the client with the product under development [75], this should naturally provide more space for discussion and thus increase the overall comprehension of the client by the software development team, even if the team is composed of students.

Regarding project, the positive results obtained do not seem to indicate anything apart from students generally being better organized. Implementation, however, seem to suffer a greater positive impact. Shortening the development process and helping to make abstract concepts more concrete are among the reported benefits.

Shortening the development process is an unexpected report. As BDD requires more collaboration and engagement from those involved in the software development process [75]. The second result, regarding and easiness to translate abstract into concrete, is most likely linked to the use of BDD scenarios, as Smart [75] reports that scenarios are useful in communication ranging from the business analyst to the developer. Supporting this, the reported benefits about communication also seem to indicate that misinterpretations can be mitigated and the distance from business and implementation can be reduced.

4.3.2 (RQ2) What are the negative aspects reported by experts regarding teaching BDD?

According to participants, the main negative aspect should be the processes of BDD. Reports vary in terms of the specific challenge of BDD, pointing aspects such as lack of experience from developers and level of formality imposed by BDD. This is aligned with the need for a greater deal of commitment from business and the development team required by BDD [75].

Another interesting challenge proposed are BDD scenarios. Previous researches have proposed a set of guidelines to evaluate scenarios [54, 52] but there is still a lack of evidence about how seniority impacts the quality of BDD scenarios. This concern is further affirmed by some challenges reported regarding requirements. Some reports appear to indicate a poor process of generating requirements from some students, in which case BDD may seem nothing more than an additional process with little to offer.

In addition, the testing-culture (*i.e.*, adding tests to the software development lifecycle) seems to be a barrier as well. This is not surprising given that development which uses tests as driver for implementation present numerous challenges to learners [45]. Furthermore, another key factor is team-engagement. As software is usually developed in teams and many human factors, such as collaboration and communication, seem to have impact in the overall software development process [20], it becomes an important factor.

In addition to all that, time is also referred to as another challenge for implementing BDD. As BDD can be applied in the entire software development lifecycle, it can increase the time required to perform certain tasks. This may not be possible in problems with tight schedules.

4.4 Threats to Validity

The study was conducted with a limited number of active-learning experts and some of those respondents had little to no previous experience with BDD. In addition, our

results are based upon participants viewpoint - experts (development teams), thus being a subjective opinion.

A relevant threat to this study is the fact that some experts had no previous experience with BDD. Moreover, there is the limitation in knowledge transfer inherent to the lecture provided to experts. This means that there is no guarantee that experts were able to properly understand BDD.

Further, another threat might be related to the experience our experts had. We decided to consider experts those who had been teaching using active learning for at least one year.

Finally, the interpretation of responses from participants may have the biases from the researches who performed the analysis.

4.5 Conclusion

In this chapter, our goals were to assess the perception of impacts of teaching BDD in active learning environments. To achieve this, we have used an expert panel with a total of 28 active learning experts to extract insights of potential benefits and challenges of teaching BDD.

Results indicate that BDD is perceived as positive in terms of requirements and collaboration. However, we have found challenges related to BDD practices and testing-culture.

5. SURVEY

This chapter presents an investigation of influencing factors when teaching MAD in an active learning environment (ALE) using CBL. We conducted this study to better understand influencing factors in ALE. The investigation was performed through survey on a two-year course that teaches MAD to undergraduate students. The factors investigated were *previous working experience*, *team size* and *project time duration* and their influence on student's perception about their projects. After discussing and exploring the results, we were able to draw relevant insights.

5.1 Related Work

Santos *et al.* [69] conducted an empirical study about the combination of CBL and Scrum. The main focus of the study is the students' perception of CBL and Scrum and its effectiveness when used for learning of MAD. Although the results indicate that combining CBL and Scrum can be highly effective in the learning process, the study does not explore other important factors for the success of the implementation, such as the background of students.

Chanin *et al.* [13] conducted a case study on the perception of students regarding whether CBL could help to improve their software engineering skills when working in software startups. Even though the findings were positive, the study lacks a further investigation of correlating factors for the success of CBL in the learning environment.

In this context, given the gaps these studies presented, there is an opportunity to understand what other factors can influence students' perception when learning on active learning environments.

5.2 The Course

This study focuses on a two-year mobile development course that teaches iOS, tvOS and watchOS development to undergraduate students. All learning in the course follows the CBL principles. In the course, students are introduced to the development ecosystem of Apple platforms and learn by working on real world problems. Besides that, students can choose to focus on development or design aspects of mobile application development. Students in the course are expected to dedicate 20 hours per week at course activities.

During the course, as CBL proposes, students learn through challenges. As described by Nichols *et al.* [48], the challenges can be classified as:

- **Nano-Challenge:** These challenges are shorter in length (1 week), focused on a particular content area or skill, have tight boundaries and are guided by the instructor. Both *Big Idea* and *Essential Question* are provided to the students. The process includes some level of investigation, but at a lower level of intensity and often stop short of implementation with an external audience;
- **Mini-Challenge:** These challenges have a longer duration (2-4 weeks) and allows learners to start with a *Big Idea* and work using the entire framework. The research depth and the reach of their solutions increase and the focus can be content specific or multidisciplinary. Mini-Challenges are good for intense learning experiences that stretch learners and prepare them for longer challenges;
- **Standard-Challenge:** These challenges are the longest (one month and longer) specified in framework and allow considerable latitude for the learners. Working together, learners identify and investigate *Big Ideas*, develop *Challenges*, do extensive investigation across multiple disciplines and take full ownership of the process. The framework is used from beginning to end, including implementation and evaluation of the solution in an authentic setting.

Due to the variety in length of the challenges, students work on a larger number of short challenges (**Nano-Challenge**) than in longer challenges (**Standard-Challenge**).

5.3 Methodology

This study was conducted through survey for data collection and classification [6]. Survey is appropriate when the focus of interest is on what is happening or how and why something is happening and also applies when it is not possible to control dependent and independent variables [6]. This study aims to answer the following research questions:

- (RQ1) *“How can previous working experience change perceptions of students on active learning environment?”*
- (RQ2) *“What is the influence of team size in student perceptions on active learning environment?”*
- (RQ3) *“What is the influence of project duration in student perception on active learning environment?”*

In order to answer these questions, the following steps were undertaken.

5.3.1 Data Collection & Analysis

A survey was conducted after the end of the two-year program in order to measure variables related to the individual perception that each student had regarding all developed projects.

Survey Protocol

The goal of the survey conducted in this study was to identify relevant aspects that influence the project development. The sample population (47 students) was composed of undergraduate students in a mobile application development program, who were chosen using the convenience criteria due to the fact that participants were selected for their availability. The sample population size was defined using the higher number of available people to participate in this study.

In order to avoid students from associating the questions presented in the survey with their individual evaluation, each student was told about the purpose of the research and about the zero-impact their answers had on their evaluation. In the survey, students were asked to individually evaluate the projects they had worked on while attending the course. While answering the survey, a researcher was available to remove any doubts regarding the survey. In addition, projects which were shorter than one week were not considered in the study.

Although the sample population is 47 students, the number of teams of students working on different projects is 90. This number ended up being larger than the number of students in the study (47) due to the multiple team configurations each student had participated.

During evaluation of projects, students could take into consideration any aspect regarding the project they found relevant. Students had to provide a grade from 1 to 5 (where 1 is the lowest and 5 the highest), using the same criteria internally applied by the instructors.

Throughout the course, instructors collected data regarding each project students have worked on. This data was obtained by averaging the instructors evaluation of a project performed by the students. This evaluation ranged from 1 to 5, following a Likert scale. The criteria used for this evaluation was:

1. Terrible performance;
2. Poor performance;
3. Average performance;

4. Good performance;
5. Amazing performance.

We have also collected students demographic data including age, gender, role (developer or designer) and previous work experience. Regarding previous working experience (prior to the course), students were classified in three categories:

1. *Full-time employment* - A student who has experience of working professionally in a software development company as a full-time software developer.
2. *Internship* - A student that has either worked professionally in a software development company as an intern software developer or worked with software development in extra-curricular projects with focus on software development.
3. *None* - A student who has no experience developing software or has only had experiences which did not fit the criteria for above categories.

Once data was collected, grouping and categorization of the information were performed. All data was stored on an Airtable database¹.

In order to perform the analysis, data from the survey, demography and project evaluation (both from the students and the instructors) were analyzed in a quantitative manner.

5.4 Results

After data collection, data analysis was performed using triangulation. From a demography standpoint, the average age was 21.5 years old, with a standard deviation of 3.18 years. Figure 5.1 presents the demography of students regarding gender, role and previous students experience.

5.4.1 Projects Score

As an initial analysis, Figure 5.2 presents the score of 9 projects developed by the students. In order to allow comparison to be performed, for each project, the left (solid) bar represents the average rating provided by the students in the survey and the right (patterned) bar represents the average rating provided by the instructors (for the same project).

¹<https://airtable.com/>

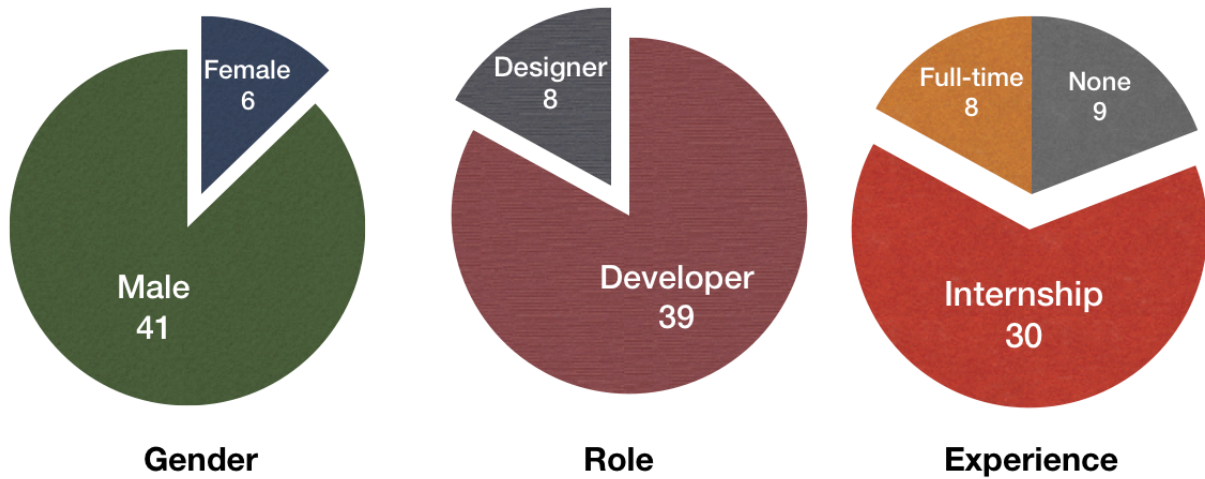


Figure 5.1 – Demography of students
[Source: Author (2020)]

Figure 5.2 demonstrates that the average instructors score regarding projects was at least 1 point higher than the students individual perception, almost reaching 2 points in some cases. Reasons behind this can be related to the initial expectation of students for a project outcome to be much higher than what is actually achieved after finishing it. Another possible reason is that students focus exclusively on the final results and end up not considering the learning experience of the project.

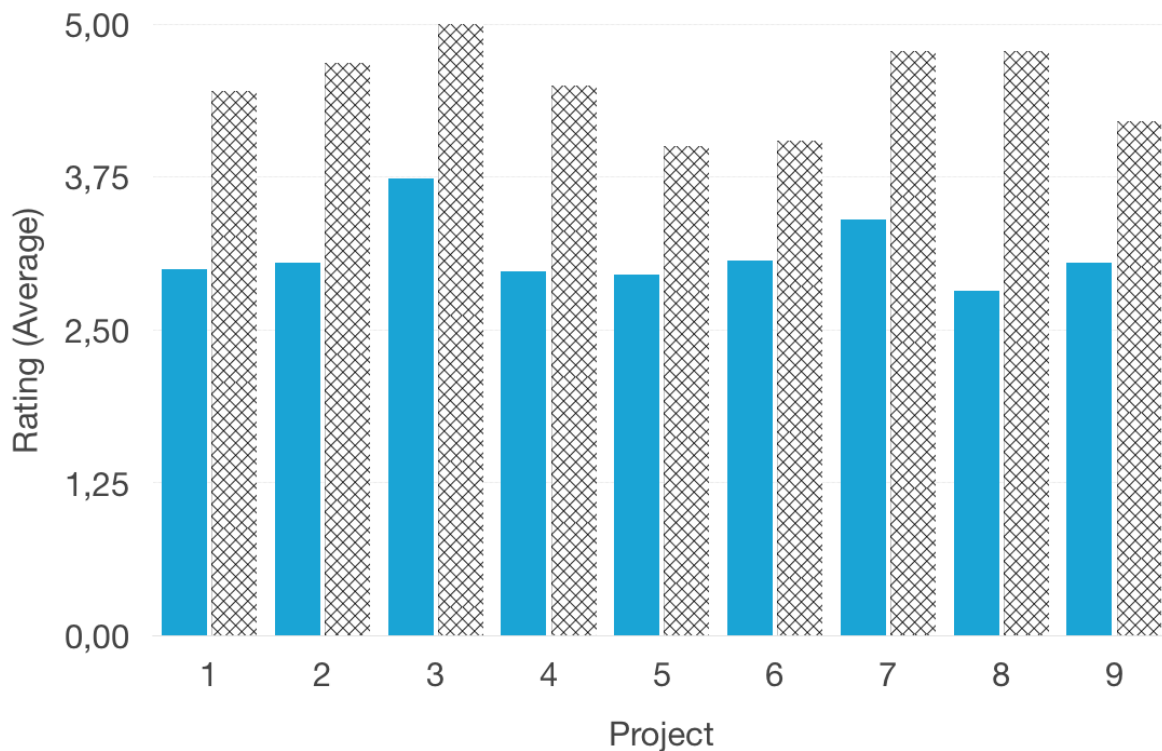


Figure 5.2 – Projects Score
[Source: Author (2020)]

In addition, reasons for these differences could also be contextual to the period of time in which activities were carried out. As the course in which students were enrolled was two years long, a project may have been impacted by external factors, such as schedule updates. Due lack of information regarding that matter for the collected data, we were unable to analyse this relation.

From this initial set of data, we continued our analysis by performing multiple grouping strategies to the collected data. Each grouping strategy provided insights regarding the research questions of this study.

5.4.2 Student Experience

The first grouping was related to **RQ1** and it was based on the previous working experience of students. In order to perform this, we have assigned each student a score associated with his/her previous working experience category (*Full-time*, *Internship* or *None*). This score was based on a system of points where a student could be assigned:

- **10 points** - If the previous working experience category of the student was *Full-time*.
- **5 points** - If the previous working experience category of the student was *Internship*.
- **0 points** - If the previous working experience category of the student was *None*.

After this step, each of the 90 teams was assigned a team score. Teams score was obtained by averaging individual scores of students participating in the team. Finally, we categorized teams in five categories as follows:

- **Novice**: if the average score of the team was under 2 (excluding 2);
- **Beginner**: if the average score of the team was between 2 and 4 points (including 2, but excluding 4);
- **Intermediate**: if the average score of the team was between 4 and 6 points (including 4, but excluding 6);
- **Advanced**: if the average score of the team was between 6 and 8 points (including 6, but excluding 8);
- **Expert**: if the average score of the team was above 8 points.

Table 5.1 presents the results from the survey after grouping teams based on this criteria.

Table 5.1 – Survey results grouped by category

Category	# Teams	Average	Std. Deviation
Novice	2	2.38	0.72
Beginner	19	3.11	0.74
Intermediate	40	2.88	0.71
Advanced	28	3.47	0.66
Expert	1	3.67	0.47

The obtained results present indicatives that the perception of students could be associated with the previous working experience from the team members. By not considering the *Novice* and *Expert* categories, which accounted for less than 4% of teams, the highest average rating was given to teams that were categorized as *Advanced*, followed by *Beginner* and *Intermediate* teams. Although these results are only indicatives, reasons can be related to a higher level of difficult of student to cooperate when the knowledge gap is large or when students face team activities in a competitive manner [38].

5.4.3 Team Size

In relation to **RQ2**, we also performed data analysis grouping all teams based on their size (*e.g.*, number of students). In this context, it is worthy to mention that students worked in teams for all projects. In those teams, they had to work at least in pairs. After this grouping strategy was performed, the results from the survey are presented in Figure 5.3.

Regarding the number of teams for each category, the grouping resulted in seven (7) teams with 3 or less students, eighteen (18) teams with 4 students, fifty (50) teams with 5 students and fifteen (15) teams with 6 or more students.

Through this perspective, it is possible to visualize that the perception of students is somehow influenced by the number of components in their group. By considering the average rating of projects, teams with 5 components had the highest average rating among students, followed by teams with “6 or more”, “4”, and “3 or less”.

5.4.4 Project Duration

Finally, regarding **RQ3** and as a final analysis, we have grouped the results from the survey considering the different duration of projects. In this sense, following the types of

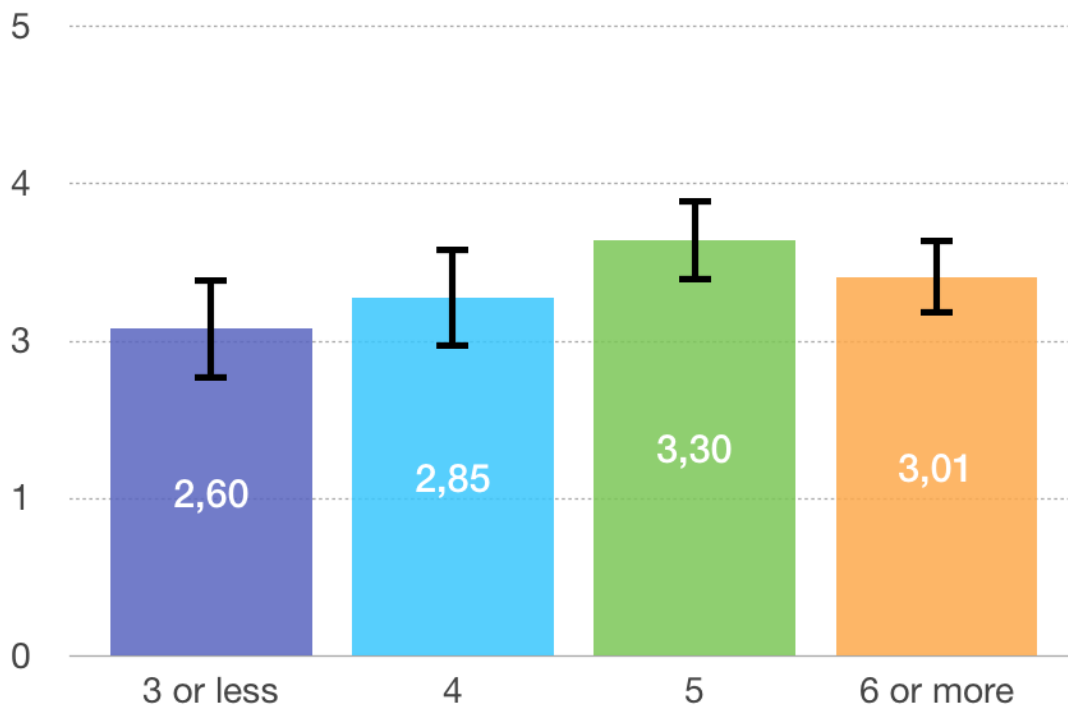


Figure 5.3 – Project Score vs. Team Size
[Source: Author (2020)]

challenges proposed by the CBL framework, teams were grouped based on the duration of the challenge to which they were associated.

This grouping have resulted in three categories: 1 week (*e.g.*, *Nano-challenges*) teams, 4-6 weeks (*e.g.*, *Mini-challenges*) teams and 26 weeks (*e.g.*, *Standard-challenges*) teams.

After applying this grouping strategy, we obtained forty-one (41) teams which were from *Nano-challenges*, forty (40) teams which were from *Mini-challenges* and nine (9) teams which were from *Standard-challenges*. In this context, the average score of teams is presented in Figure 5.4.

From this perspective, results seem to indicate that project duration may somehow influence the perspective of students. *Standard-challenges* had a better average score when compared to both *Mini-Challenges* and *Nano-Challenges*. As students are always aware of the duration of projects, some reasons behind this can be related to expectation for a project.

On short-term projects, students may have lower expectations and thus tend to be more optimistic about the final result. For intermediate projects, (*e.g.*, *Mini-Challenges*), students may have higher expectations and thus get frustrated with the final result. For longer projects, the same expectation process can happen to students, with the difference that the longer period of time allows corrections to be made.

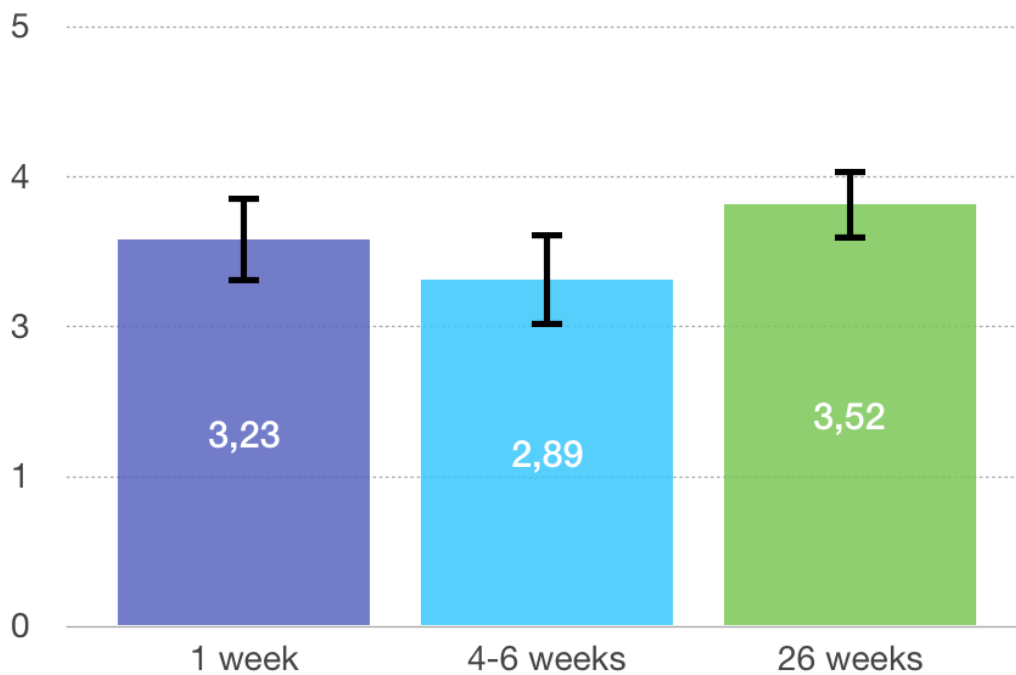


Figure 5.4 – Projects Score vs. Projects Duration
[Source: Author (2020)]

5.5 Discussion

During the course, students worked together in teams and used CBL as the learning framework and Scrum as the development framework, since the combination of these frameworks has been successfully combined and tested in a previous work [69]. In this sense, as most activities in the course required the development of small releases of software, it is fair to associate teams of students to a software development team that uses Scrum. Following this idea, the Scrum Guide [70] defines that the development team should be composed of not less than three (3) and not more than nine (9) participants. This is aligned with the results from the survey conducted with the students, where the optimal team size was five (5) people.

When the results were grouped by students' expertise, the "*intermediate*" level had the lowest rating on average. These results are in agreement with existing literature [2]. Acuña *et al.* [2] demonstrated that multiple personality traits can also have significant impact on team performance and satisfaction, such as team agreeableness impacting team cohesion. As this personality trait was not extracted from the survey, it might have had impact on students' perception.

In terms of projects' duration performed by the students, the highest average rating was given to projects with the longest duration. Previous research [85] pointed out that recognition, which is more feasible to happen on long-term projects, may play a relevant role in software development projects.

5.6 Threats to Validity

Although results obtained by this study were mostly supported by previous work presented by the literature, some threats to validity are present and may diminish the generalization of the findings reported. Firstly, assessment of projects developed by students was performed at multiple times during the course. In this sense, as students were cumulatively introduced to new concepts and had the opportunity to work in more projects, their familiarity with development practices improved. The assessment conducted by the instructors, which was subjective, may have suffered influence where projects developed later in time are more likely to present better results.

The expertise was evaluated considering the working experience students had before undertaking the course. In this sense, it does not consider the learning that occurs on during projects development and activities in the course. It is plausible, for example, to assume that a student who had no previous experience might outperform a student with a full-time job experience during development of projects. Furthermore, a student with no previous experience may put more effort in learning on projects development and surpass students with more experience.

Another important threat to the findings reported by this study is that no statistical test was performed to ensure data differences were significant. Thus, results reported serve as indicatives.

Moreover, the survey was conducted at the end of the second year of training. This may present a threat as students had to remember about projects they worked on the previous year. Even with the development artefacts available, students might not have been able to remember key occurrences of the project.

5.7 Conclusion

This chapter presented results from a survey conducted with students on a mobile application development course environment regarding their individual projects perceptions. These results were analysed through data analysis and triangulation, and they were also compared to existing literature about similar topics.

Our preliminary results demonstrated indicatives that there is an assessment difference among students and instructors' perceptions. We have found at least 1 point evaluation difference (in a scale of 5 points). Students self-assessment tends to result in lower rating. Also, it was shown that project's duration, teams composition (regarding previous work experience) and teams size play important roles in the students' individual perceptions.

It is worthy to mention that the survey was conducted at the end of the two-year program and some of the questions required students to remember projects they had developed more than one year ago. This indicative can be considered a study limitation and will require future work to explore it.

6. CASE STUDY

This chapter presents the results of a case study we have conducted in a mobile application development course. The course teaches students using an active learning framework, Challenge Based Learning (CBL) [48]. Our goal was to investigate how BDD impacts agile software development teams in active learning environments.

6.1 Methodology

In order to understand the impact of BDD in agile software development teams, we have performed a case study in the two-year mobile development course that teaches iOS, tvOS and watchOS development to undergraduate students. During the course students learn through CBL challenges as described by Nichols *et al.* [48].

Students are introduced to the development ecosystem of Apple platforms and learn by working on real world problems. Besides that, students can choose to focus on development or design aspects of mobile application development. They are expected to dedicate 20 hours per week at course activities.

As a research method, case studies can be used for software engineering research, as they allow the understanding of a certain phenomenon in its natural occurring context [66] and are suited to evaluate a method and tool [32].

As our goal was to understand the impact of BDD, with few hypotheses being previously established, we have chosen to conduct an exploratory study. As such, case study methodology is suited to be used as it enables the research to extract new insights and ideas, to understand what is happening and to generate new hypothesis for other researches [66].

6.1.1 Case Study Protocol

Our protocol follows the guidelines proposed by Kitchenham *et al.* [32]. Thus, the objective of this case study is **to identify some of the benefits and challenges of using BDD in agile software development teams**. In this sense, this paper aims at answering two research questions:

- (RQ1) *“What are the positive impacts of developing software using BDD?”*
- (RQ2) *“What are the negative impacts of developing software using BDD?”*

6.1.2 Data Collection

Regarding data collection, we have chosen to collect our data through semi-structured interviews [64]. The total number of interviews was 42. These 42 interviews were performed in two different stages in the case study: before using BDD (*e.g.*, pre-BDD) and after using BDD (*e.g.*, post-BDD). The interviews conducted in both stages were aided by interview questions. The pre-BDD interviews helped us to establish a baseline against which to compare the results of using BDD. These interviews were conducted right after teams had finished developing software in the context of CBL Nano-Challenge (*e.g.*, 1 week long). Thus, the post-BDD interviews helped us assess the benefits and challenges of developing software using BDD.

Participants in the study had been attending the course for six (6) months and thus had had the chance to work in software development projects using agile. In addition, some students had previous professional experience developing software.

The average team size in which the participants had worked on was 3.33. Furthermore, most teams were working together for the first time.

It is also relevant mentioning that the CBL Nano-Challenges performed by participants followed the conceptual model proposed by Santos *et al.* [69]. In this model, students first perform the initial phases of CBL and then use the Scrum framework for developing a software solution.

Aside from the demographic questions, the interview questions used in this case study are presented in Table 6.1. It is important to note that questions which required the interviewee to know BDD were only asked in the post-BDD stage.

In an attempt to understand the impacts of BDD in each part of the software development lifecycle, we have split the interview questions into four groups, as follows:

1. Questions 1, 2 and 3 - These questions focus on the project requirements phase, addressing elicitation, specification and the impact of BDD in this phase.
2. Questions 4, 5 and 6 - These questions focus on *feature development*. A *feature*, as defined by Coad [16], is as function which is valuable for the client and which can be developed in up to two weeks. In our context, *feature development* indicates the process of proving an implementation for a *feature*. As such, they address the presence of ambiguities during the understanding of features which suffered changes and the impact of BDD in this phase.
3. Questions 7, 8, 9 and 10: These questions focus on implementation quality. More specifically, they address overall quality, number of bugs, documentation and the impact of BDD in this phase.

4. Questions 11 and 12: These questions address the overall impact of BDD, for the entire development lifecycle.

Table 6.1 – Interview questions

#	Question	Phases
1	How did you elicit the project requirements?	Pre + Post
2	How did you specify the project requirements?	Pre + Post
3	What was the impact of BDD in the requirements elicitation/specification?	Post
4	Was there any ambiguity in the requirements?	Pre + Post
5	Was there any requirement developed which was very different or completely modified?	Pre + Post
6	What was the impact of BDD in the process of translating requirements to actual functionalities?	Post
7	How was the project in terms of code quality?	Pre + Post
8	How was the project was in terms of bugs?	Pre + Post
9	How did you document functionalities of the project?	Pre + Post
10	What was the impact of BDD in the implementation?	Post
11	What are the main benefits of developing using BDD?	Post
12	What are the main challenges of developing using BDD?	Post

In order to avoid biases, we have begun our data collection with this pre-BDD stage. This stage had a total of 27 interviews. The first 6 interviews were *meta* interviews, serving exclusively as a pilot study to improve the quality of the questions being used for actual interviews. These six interviews have enabled us to perform small improvements to the interview questions. After this, we have conducted 21 actual interviews. During this process, each participant was given the full context of the research and was told that participation was voluntary, and it had no impact on their internal assessment process. Besides that, all interviews had their audio content recorded. This process, performed in both stages, is illustrated in Figure 6.1.

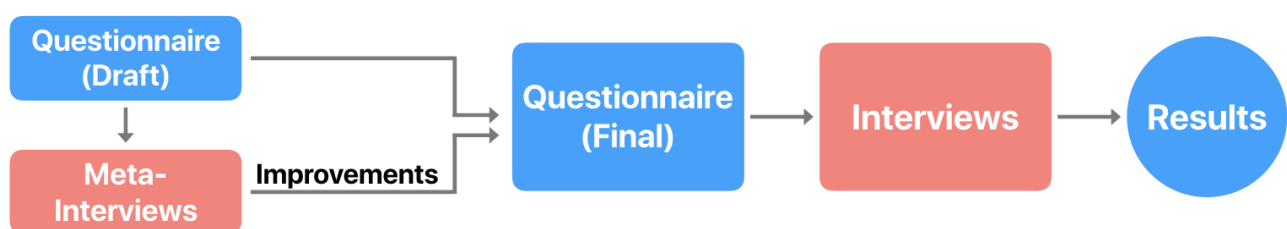


Figure 6.1 – Process used to obtain results from interviews
[Source: Author (2020)]

The pre-BDD stage interviews were performed right after teams had developed software in the context of a CBL Nano-Challenge, which lasted one week. Teams developed software using agile and no interference was performed.

In the post-BDD stage, we have also conducted 27 interviews. Maintaining the improvement procedure used in the first stage, we have conducted 6 *meta* interviews in a pilot study. These *meta* interviews, besides helping to improve the overall quality of questions in our interview questions, helped us to add two additional questions at the end of the interview. These additional questions directly address the benefits and challenges of using BDD. After adding these questions, we proceed to conduct the next 21 interviews. We have performed the same procedures as the pre-BDD stage, and these interviews were recorded as well.

The post-BDD stage interviews were performed right after teams had developed software in the context of a CBL Nano-Challenge (different from the pre-BDD stage). Teams were introduced to BDD practices through lecturing and practical activities before starting their Nano-Challenge. At any time during the development phase of the projects, participants could reach to a senior instructor in case any doubts or questions aroused. In this context, the BDD-scenarios were created in any tools that participants were comfortable working on. The only restriction was related to the executable specification aspect of BDD, in which case Quick¹ & Nimble² were used, due to their iOS-specific nature.

Another important point is the distribution of participants in the interviews. We have created three groups in which participants were distributed. The first group had 10 people whose participation was restricted to the pre-BDD interviews. The second group had 11 people and its participants were interviewed in both interview phases. The third group had 10 people who were only interviewed in the post-BDD phase. Figure 6.2 illustrates the distribution of participants in the interviews.

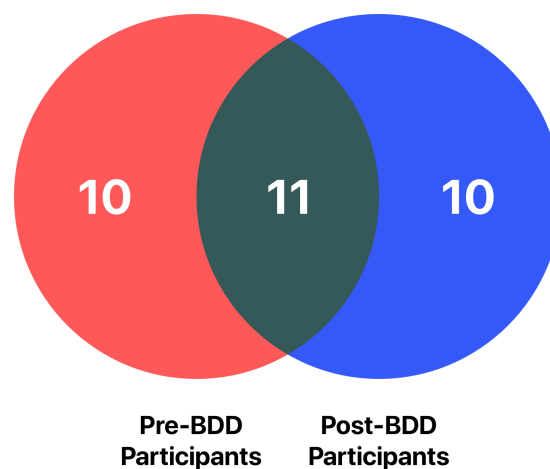


Figure 6.2 – Distribution of participants in the two interviews phases
[Source: Author (2020)]

¹github.com/Quick/Quick

²github.com/Quick/Nimble

These groups aimed at increasing the insights generated and allowed us to reduce any bias that the participants might have towards the interviews questions.

After obtaining the results from both pre-BDD and post-BDD phases, we have proceeded to the analysis phase. The analysis performed was qualitative and followed the guidelines proposed by Runeson and Höst [66]. Our analysis aimed at generating hypothesis from the data, thus, we had little to none prior hypothesis.

In this phase, we first performed transcription of the audio recorded in the interviews. In total, we have generated 8 hours, 17 minutes of recorded material. The transcripts from these recording generated 20.207 words. We have chosen to perform this process manually. This was performed as it gives the researcher the opportunity to further extract insights from the interview data during the transcription process. These transcriptions were arranged in a spreadsheet to ease the analysis process.

Using the spreadsheet, we began further analysis of the data. The analysis was done using two strategies: clustering and categorization. We first clustered answers which presented similar results, counting the number of occurrences. Afterwards, the generated clusters were categorized (which could be one or more). As a final step, we have grouped similar categories. These strategies were used for both pre-BDD and post-BDD results.

Finally, using the available data and generated analysis, we were able to extract insights from the data. The results of our analysis are presented in the following section.

6.2 Results

From a demography standpoint, all participants were actively attending undergraduate courses (they were around the 5th and 6th semester). Their average age is 23.9 years old, and all participants have less than 2 years experience in software development (having at least six months of experience on software development).

Following demographic results, we proceed to gather results from the interviews. In order to ease analysis, we have split our results into four groups:

1. **Project requirements** - The process of elicitation and specification of project requirements;
2. **Feature development** - The process of translating project requirements in actual implementations;
3. **Implementation quality** - The overall quality of code being developed;
4. **BDD impact** - The reported impact of BDD in the development process, both positive and negative.

6.2.1 Project Requirements

In terms of project requirements, we have first assessed the regular processes used by development teams and then performed a comparison of these processes after the introduction of BDD.

These was done by analysing the first three (3) questions of the interviews as they directly address aspects such as elicitation and specification of requirements. It is important to note that the last question, from this group of three, was only answered by participants who were interviewed in the post-BDD phase.

Elicitation

Prior to the introduction of BDD, the most reported (10 out of 21) way in which requirements were elicited by the teams was by building up on an individual idea of a team member who had an app idea prior to the team formation. The second most used technique was by properly understanding scope limitations, such as the timeframe.

After the introduction of BDD, the most reported (13 out of 21) way for requirements elicitation did not change and was still by following an individual idea from a team member. However, the second most reported way was through the interaction of team members (12 out of 21). This may indicate that BDD promoted more interactions among team members, thus helping in the elicitation process. However, this is just an indicative, as some reports are complementary to others and could be used in conjunction.

Specification

Prior to the introduction of BDD, participants reported using many different techniques in order to specify software requirements. The three most reported ones were *Verbalization* (9 out of 21), where requirements were not specified in any written format and were only defined through conversations, *Unstructured annotations* (7 out of 21), where requirements were specified using an informal written format, and *To-do lists* (5 out 21), where requirements were specified by defining the tasks which would be performed directly. One interesting results from this phase was that only two (2) participants reported using lightweight specification, such as User Stories.

After the introduction of BDD, every participant performed their specification using User Stories and Scenarios. One interesting result from this interview phase was that participants found it difficult to understand that scenarios were augmentation of user stories and some reported thinking scenarios encompassed user stories.

Requirements - BDD Impact

Regarding the specific impact of BDD in the project requirements phase, participants have reported many positive aspects and some negative aspects.

Starting by the positive aspects, the three most reported positive aspects were *feature clarification*, where both team and individual found it easier to correctly understand what was supposed to be developed, *ease of development*, where the overall software development process was facilitated, and *project alignment*, where the team managed to develop a common idea of the project.

In terms of negative aspects, participants reported four (4) aspects, which were *lack of impact*, *poor execution*, *initial confusion* and *difficulty to change development mindset*. The number of negative reports, in general, was low (5 reports, against 31 positive).

6.2.2 Feature Development

Following a similar procedure to the previous section, we have first assessed how participants performed the translation of requirements when implementing features in their regular process. After that, we compared the results from this stage with the ones from post-BDD.

This step was performed by analysing the answers of three questions from the interview questions (questions number 4, 5 and 6). It is important to mention that the last question analysed in this subsection was only answered by participants of the post-BDD phase.

Ambiguity

Prior to the introduction of BDD, the majority of the participants (16 out of 21) reported that their projects had some type of ambiguity when translating the requirements into concrete features. The most frequent types of comments on the ambiguities were that they were related to the application mechanics (5 out of 16), had low impact on the development process (5 out of 16) or were related to the application interaction.

After the introduction of BDD, there were less reported ambiguities, with the most frequent report (13 out of 21) being that there were not ambiguities in the translation process. In addition, the ambiguities reported by some participants was related to the creation of BDD scenarios, the creation of tests and implementation.

Feature Changes

In terms of features which were specified and changed during the development process, while in the pre-BDD phase, the majority of the participants (11 out of 21) reported not having changed any of the specified features. The remaining participants reported that their project changed during the development phase. Some of these changes included application mechanics, technological limitations and design updates.

In the post-BDD phase, results were similar. Not changing any aspect of the features remained the most reported item by the participants (13 out of 21), followed by changes in the BDD scenarios and the creation of new scenarios.

Feature Development - BDD Impact

Regarding the specific impact of BDD in this phase of development, results are very diverse. Participants described a total of 13 different positive aspects and 9 negative aspects. These aspects are presented in Table 6.2 and Table 6.3 respectively. It is important to note that these results were obtained only from the post-BDD interviews.

Table 6.2 – Positive aspects reported by participants

Aspect	Occurrences
Better understanding of features	8
Ensure correct execution	4
Team alignment	2
Reduction of unexpected changes	2
Reduction in implementation problems	1
Broader vision	1
Improvement in the development process	1
Facilitation in the requirements communication process	1
Project guide	1
Facilitation in the development process	1
Reduction in re-work	1
Reduction in user story breakdown difficulty	1
Facilitation of communication among team members with different expertise levels	1

Table 6.3 – Negative aspects reported by participants

Aspect	Occurrences
Difficult to execute	3
Difficult to adopt initially	1
More suited for large projects	1
Meaningless	1
Equivalent to a User Story	1
Value only visible at the end of the process	1
Need for previous planning	1
Easy for some scenarios, difficult for others	1
Low gain (considering learning curve)	1

6.2.3 Implementation Quality

To address implementation quality, we have followed a similar procedure to the two previous steps, comparing the pre-BDD and post-BDD reports regarding this specific aspect.

This step of the analysis considered results from four questions of the interview questions (numbers 7, 8, 9 and 10). As with the other phases, the last question was only answered by participants in the post-BDD phase.

Overall code quality

In order to assess the impact of BDD in the implementation quality, we have gathered the overall reported code quality of participants prior and after the introduction of BDD. This generated a three categories (*good*, *average* and *bad*), which summarize reports of the participants. As some of participants did not directly state their implementation quality, we have used the overall report as a matching mechanism to associate the report with one of the generated categories. Table 6.4 presents the distribution of overall reports from participants.

Table 6.4 – Code quality reported by participants

Overall Quality	Pre-BDD	Post-BDD	Difference
Good	6	11	+5
Average	8	6	-2
Bad	7	4	-3

Bugs

In terms of number of reported bugs, prior to the introduction of BDD, the majority of the participants reported code had either many bugs or some bugs (14 out of 21). In the post-BDD phase, results were somewhat similar, with a slight diminish in the number of reports of many bugs and a slight increase in the reports of no bugs. Table 6.5 presents a comparison of the results from these phases.

Table 6.5 – Bug category for projects

Overall Quality	Pre-BDD	Post-BDD	Difference
Many bugs	9	5	-4
Some Bugs	8	9	+1
No Bugs	4	7	+3

Documentation

In terms of documentation used during the development of projects, in the pre-BDD interviews, the vast majority (19 out of 21) of participants used some type of documentation. However, the most reported type of documentation was code comments. The results for this phase are presented in Table 6.6.

Table 6.6 – Documentation types reported (Pre-BDD)

Type	Occurrences
Code comments	15
Code	8
List of tasks	3
No documentation	2
Product Backlog	1
Diagrams	1

In the post-BDD phase, results have changed slightly. Among of these changes is the usage of BDD Scenarios as documentation (6 reports out of 21). The results from these phase are present in Table 6.7.

Implementation Quality - BDD Impact

Finally, in order to generate a broad vision of the impact of BDD in the implementation quality, we have used question number 11 of the interview questions. Participants re-

Table 6.7 – Documentation types reported (Post-BDD)

Type	Occurrences
Code comments	9
BDD Scenarios	6
Executable Specifications (BDD Tests)	5
User stories	5
No Documentation	4
List of tasks	1

ported many positive (16) and negative (14) influences of BDD in the implementation. Tables 6.8 and 6.9 respectively present the positive and negative aspects reported by participants.

One interesting result from this question was that the majority of those who reported negative aspects of BDD did not think it is bad. Rather, they report not being able to properly use it.

Table 6.8 – Positive impacts of BDD in implementation

Item	Occurrences
Improvement in implementation quality	5
Organization & Planning	5
Clear implementation	3
Facilitated implementation	2
Right (client expectations) implementation	2
Reduction in re-work	2
Reduction in ambiguities	1
Modularization	1
Custom execution	1
Simultaneous Development and Testing	1
Correct (functional) implementation	1
Product-oriented vision	1
Improvement in task division	1
High-level of impact in implementation	1
Perception of the need for good scenarios	1
Suited for documentation	1

Table 6.9 – Negative impacts of BDD in implementation

Item	Occurrences
No changes	3
Meaningless	2
Poor tests	2
Difficult in terms of UI	1
Difficult for unexperienced developers	1
Project too simple for BDD	1
Bigger projects can benefit more	1
Other paradigms provide better guidance	1
Low impact	1
Slower implementation	1
Testing becomes difficult	1
High learning curve	1
Lack of team commitment (to BDD)	1
Lack of time	1

6.2.4 BDD Impact

Finally, regarding the perceptions of BDD, we have gathered results from the last two (2) questions of the interview questions performed with participants. These questions were only performed in the post-BDD phase, as participants had been asked specific questions about their perception of BDD. In this sense, Table 6.10 presents the positive aspects of BDD reported by participants. In contrast, Table 6.11 presents the reported negative aspects of BDD.

6.3 Discussion

The results from our case study present a view of the impact of introducing BDD to the development lifecycle of agile software development teams in the context of a software development course. Aiming at answering the proposed research questions of this work, we further discuss these results in the following subsections.

Table 6.10 – Positive aspects of BDD

Aspect	Occurrences
Better comprehension of feature under development	4
Team alignment	3
Eased task division	3
Correct (functional) development	3
Right (client expectations) development	3
Reduction in ambiguities	3
Eased project comprehension	3
Clearer project	2
Eased project contextualization	2
Eased project organization	2
Faster development	2
Awareness of final product requirements	1
Conjunct development and testing	1
Eased project execution	1
Eased project testing	1
High market usage	1
Improved requirements	1
Improved documentation	1
Improved prioritization	1
Improved project planning	1
Value perception in larger projects	1
Automatic testing	1
Reduction in 'course' changes	1

Table 6.11 – Negative aspects of BDD

Aspect	Occurrences
Tests writing	8
Test-driven development	8
Scenario creation	4
Initial process	3
Reduction in development time	3
Lack of experience with BDD	2
Small scope	2
Team engagement with BDD	2
Scenario comprehension	1
Up-front need of complete project comprehension	1
Team size	1
Task distribution	1
Scenario precision	1
Limitation in test amount	1

6.3.1 (RQ1) What are the positive impacts of developing software using BDD?

The initial phases of development, where project requirements are generated and specified, seems to have suffered positive influence. An example is the report from participants about the elicitation phase, where the second most used technique was the interaction of team members. This result is aligned with proposed benefits of BDD [75]. Moraes [44], in her master thesis, has found that using BDD encourages team members to collaborate. This could indicate that the elicitation process has changed due to the increase in team member collaborations. In addition to this, participants have reported multiple positive aspects which may be due to the use of BDD, such as an increase in feature understanding.

Project requirement specification has also drastically changed as, prior to the introduction of BDD, participants would most-likely specify project requirements in informal ways, such as verbalization and unstructured annotations. The combination of user-stories and scenarios was the specification method used by all participants in the second phase of interviews. This result is not particularly surprising as participants were using BDD in their development lifecycle and this specification method is proposed in the BDD framework [75].

Reports from participants also indicate that the process of feature development was improved. The majority of the post-BDD participants have reported a clearer understanding of the project after adopting BDD. This could be connected with the reduction in the number of ambiguities, which were also reported by participants in the post-BDD phase.

In terms of implementation quality, BDD seems to have had a positive impact on the development of projects. The most reported positive aspect of BDD were its clarification of features which need to be developed. This means that the development team would spend less time debating a feature and more time actually implementing it as the creation of scenarios, where everyone is allowed to participate and contribute, helps to ensure a collective understanding of the user stories.

Moreover, regarding the implementation process, results from the documentation types reported show that both BDD scenarios and BDD executable specifications were considered as documentations by participants, even though code comments remained as the main documentation type. One interesting result from this was that no participant reported code as being the documentation of the project, whereas 8 participants reported this in the pre-BDD phase (Table 6.6).

Finally, analysing the results from the second to last question, BDD seems to have helped teams develop software in multiple ways. By allowing team members to better comprehend features under development and by improving team alignment, BDD helped teams have a clear vision of the project. Besides that, participants reported feeling their implementation was both correct, meaning that it had less bugs and had a better overall organization. Finally, another interesting result is an eased process of task division, a common activity in agile teams.

6.3.2 (RQ2) What are the negative impacts of developing software using BDD?

Regarding negative impacts of BDD, in the beginning of the development lifecycle, there were negative aspects reported, with problems being due to poor execution of BDD, initial confusion and a difficult to change (we are excluding the inability to see value in the methodology, which is probably related to lack of commitment of the development team). A possible cause for these problems is the fact the agile teams were using BDD for the first time and naturally went through the learning curve of the framework. This should also be considered for other negative impacts, as some might have happened because some of the participants were using BDD for the first time.

Results from the question related to the feature development were similar in terms of quantity. The number of reported features which did not change during development was very similar both in the pre and post-BDD phases (an increase of 13 reports against 11). In addition, participants reported having trouble to both create and change BDD scenarios. This may indicate that the increased levels of interaction among team members was not enough to decrease the number of features which were not completely clear during development. Furthermore, many of the negative aspects reported by participants were related to lack of experience using BDD, which could indicate more experience would remove this issue.

In terms of implementation, the BDD process seems to have been difficult to be executed, as reported by some participants. In addition to this, both the overall code quality and the number of bugs seem to have improved from the process of BDD. However, this is difficult to assume as there were no pre-established criteria for those metrics and their reports could be subjective to what a participant takes into account when classifying their code as *“good”* or *“bad”*.

Finally, results from the last question, which directly addresses the negative aspects of BDD, have shown that writing tests and developing based on tests (a shared practice between BDD and TDD) were the main challenges. In addition, the first experience with BDD can be challenging, as the learning curve is somewhat large. Furthermore, results indicate that developers have a feeling where performing the processes of BDD, such as writing

executable specification and creating scenarios, can decrease the time of development. Even with this, results indicate that participants were able to perceive the value of these processes and are not completely unhappy with them.

6.4 Limitations

Our study was conducted with a limited number of respondents and from the same iOS development course. In addition, our results are drawn based on participants viewpoint - students (development teams), thus being a subjective opinion. Therefore, the results reported in this study are dependent on the participants' honesty, perceptiveness and judgment.

It is also important to notice that part of project participants were attending a training course without previous experience with other software engineering practices and approaches. Also, as the translation and coding processes were manually performed and even ensuring they were correctly executed, errors could have been made and consequently have influenced our results.

Finally, as both projects were performed sequentially and participants were developing software in an education context, it is possible that the experience from the first project somehow improved their development skills, changing their development practices.

6.5 Conclusion

This study has performed an assessment of the impacts of BDD in agile software development teams. The case study explored the positive and negative impacts of BDD in a two-year mobile development course where participants were working in an agile software development environment.

We have found indicatives in our study that BDD can help development teams in most of the phases of the software development lifecycle. Project requirements creation may benefit as teams showed more levels of interaction in contrast to when BDD was not applied. During feature development, developers can benefit mainly by better understanding of feature encompasses. Quality implementation can also benefit as participants reported BDD improves quality implementation and better documenting their projects.

However, inherent aspects of BDD, including writing tests and creating scenarios has been reported as a challenge by participants. In addition, lack of experience and commitment with BDD is a possible influence for this negative result.

Overall, our results indicate that BDD can provide more benefits than harms to the development lifecycle. However, further studies need to be performed to address whether more experienced developers can further improve the software development lifecycle by using BDD.

7. DISCUSSION

In this chapter, we will further reason about the research questions of this study and consolidate the contributions of the work. Our reasoning will be performed based on each one of our research questions, the results obtained in our studies and some additional literature. To address these research questions, we applied triangulation, as proposed by Creswell [17].

In this sense, we have collected data from four sources types: our SLR (Section 3), Expert Panel (Section 4), Survey (Section 5), and Case Study (Section 6). It is important to notice that the participants in each study were different in order to mitigate the research bias.

7.1 (RQ1) What tools, models, methodologies, frameworks and software are used when teaching BDD?

In terms of tooling used to teach BDD, we are able to extract knowledge based on our findings from our SLR and later added our own set of tools, which were applied in the case study.

In order to enhance our analysis we have classified our case study following the same summary table presented in our SLR (Table 3.7, page 43). The updated table is presented in Table 7.1.

Table 7.1 – Summary of results from the studies (including case study)

Study	Tools(s)	Learning framework	Agile framework	Dev. practice
Matthies (2017) [41]	Prof. CI.	-	Custom	TDD
Hoffman (2014) [26]	Robot, Selenium and Coverage	Problem-based learning	Scrum	ATDD
Simpson (2017) [73]	-	Flipped-classroom	Scrum	TDD
Case study (Section 6)	Quick & Nimble and Xcode	Challenge-based learning	Scrum	BDD

Initially, we have found that there is a lack of studies which address the specific topic of *teaching BDD* in active learning environments. Using the quality guidelines, we specified during the SLR, we have only found 3 (three) studies which intersected with proposed research questions. Even though none of the studies directly address the usage of BDD in active learning environments, they provide some good indicatives.

In terms of tools, Prof CI., reported by Mathies *et al.* [41], and the Robot framework (which was used alongside other tools, reported by Hoffman *et al.* [26]), were the only tools found in our SLR.

Prof CI. is a continuous integration tool that aids the teaching and development process. It is reported as a successful tool to teach TDD to undergraduate students. As TDD practices can be incorporated in the BDD development lifecycle, during the implementation of *low-level specifications* [75], the tool is something which can be adapted and used for teaching some aspects of BDD, specially during the implementation phase.

The Robot framework is another tool which is reported by the literature. It helped in the development of a real-time system and it was used alongside Problem Based Learning (PBL), an active learning framework, and Acceptance-Test Driven Development (ATDD), a development methodology which shares similarities with BDD. This study is particularly interesting as it provides a framework to be used when teaching a methodology very similar to BDD.

As the only tools which had reports of being used in active learning environments, both Robot and Prof. CI focus primarily in the implementation phase. This result is aligned findings from Solis [76] *et al.*, who found that most BDD tools had the implementation phase as their main objective.

Specifically during our case study planning, we used a set of tools (*i.e.*, Quick & Nimble and Xcode) which also were meant to be used during the implementation phase. Our reasoning revolved around adapting a set of tools which would be easy to master and that had sufficient material to be studied by our study participants. In addition, these tools were also chosen due to natural fit in the learning environment of the case study, *i.e.*, development for Apple platforms. The tools were successfully applied in our case study and thus expand the scientific reports about what can be used to teach BDD.

As a summary for tools, although there are others that can be used to develop software using BDD, such as Jbehave¹, it seems that there is a lack in reports for these tools in the education scenario.

Regarding methodologies, two of the studies we analysed, *i.e.*, [73, 26], applied BDD in an agile development context. Furthermore, the case study we performed had agile development during execution. This indicates that teaching of BDD is suited to be performed in an agile context. This can be linked to both the increase in agile software development relevance, as it improves the success chance of a project [46, 71], and BDD's natural fit in agile development contexts [50, 75].

¹<https://jbehave.org/>

7.2 (RQ2) What are the benefits and challenges of using BDD in active learning environments?

To address our second research question, we decided to use the results from our case study primarily. This is due to the low number (3) of papers found in our SLR, which indicated a gap in the literature regarding the matter of teaching BDD.

To enhance our analysis, we have crossed results from the case study with our expert panel results regarding expectations and some insightful data from the papers we found in our SLR. As the proposed research question has a natural duality associated to it, we decided to reason about each one its foci separately.

7.2.1 Benefits

Benefits found by the SLR

Simpson *et al.* [73] report using BDD to increase the realism in software engineering projects in a software engineering program. As a result, engagement of students was improved. Furthermore, engagement of mentors (professors, senior students and external developers) was also enhanced. This is reported to be an effect of the improved teaching experience promoted by the changes in the course.

Although this result presents benefits, it is not clear whether these benefits were caused by BDD or any of the other employed techniques, such as agile methods, which are known to promote more positive outcomes [71]. However, these results can be taken as indicatives to what would be found later in our case study.

Adding to this, results from the expert panel also provided relevant insights. According to experts, requirements would be the most positively-impacted aspect of the development cycle for students. This improvement expectation is reasonable is probably linked to the usage of a ubiquitous language [76] as it enhances communication and improvements the requirements engineering process [44, 75].

Overall, results from the expert panel indicate that experts were optimistic about the positive impact that BDD can have in active learning environments, especially in the initial phases of development. As such, we proceeded to perform our case study and were able to draw conclusions from it.

Benefits found in case study and expert panel

As an initial step, we tagged the benefits reported by the participants in our case study using the categories reported by the experts during the expert panel study. As a reminder, these categories were generated using *Card Sorting*, a method explained in Chapter 4 (page 50), and were *requirements*, *user comprehension*, *project*, *implementation*, *communication* and *others*. The result is presented in Table 7.2.

Table 7.2 – Positive aspects of BDD (including categories)

Aspect	Occurrences	Category
Better comprehension of feature under development	4	Requirements
Team alignment	3	Requirements; Communication
Eased task division	3	Project
Correct (functional) development	3	Implementation
Right (client expectations) development	3	User comprehension
Reduction in ambiguities	3	Requirements; Communication
Eased project comprehension	3	Project; Communication
Clearer project	2	Project; Communication
Eased project contextualization	2	Project; Communication
Eased project organization	2	Project
Faster development	2	Implementation
Awareness of final product requirements	1	Requirements
Conjunct development and testing	1	Implementation
Eased project execution	1	Project; Implementation
Eased project testing	1	Implementation
High market usage	1	Others
Improved requirements	1	Requirements
Improved documentation	1	Project; Implementation
Improved prioritization	1	Requirements; Project; User comprehension
Improved project planning	1	Project
Value perception in larger projects	1	Others
Automatic testing	1	Implementation
Reduction in 'course' changes	1	Project; Communication

After adding the categories, we notice that most reports from participants are aligned with the expectations from the experts, which indicates that these expectations were reasonable. Following this, we decided group the aspects in the categories and sum the number of occurrences of each category. Data from this table should allow us to better infer the benefits of BDD. The result is presented in Table 7.3.

As presented in Table 7.3, BDD can be beneficial when learning to develop software in active learning environments in many aspects.

Table 7.3 – Positive aspects of BDD grouped by categories from experts

Category	Occurrences (Sum)	% of total
Project	17	28.33
Communication	14	23.33
Requirements	13	21.67
Implementation	10	16.67
User comprehension	4	6.67
Others	2	3.33
Total	60	100.00

To enhance clarity and reason about each category of benefits, we present as follows a short statement of a BDD benefit followed by a short explained about why this appears to be the case.

1. **BDD benefits project management** - There are many risks involved in software development. Proccacino *et al.* [62] state that poor requirements, lack of management support and customers who are unavailable can have negative impacts in a project. Moreover, when training software developers, this can get even worse due to the lack of experience from students.

In this context, the processes promoted by BDD, such as the creation of scenarios and the definition of a ubiquitous language, are probably linked to the results we found in our case study of improved project management practices, specially regarding requirements, which play an important role in a software project [62].

2. **BDD benefits communication** - Communication is a success factor in software projects and the impacts of modern development practices, such as Scrum and XP, in it have been to positive outcomes [59].

BDD enhances communication by more than one mean. The ubiquitous language, which lowers the barrier among project stakeholders, an active participation of the client, which improves communication between client and the development team, and the creation of scenarios, which can drive the implementation are all reported as beneficial by Smart [75] *et al.* and may explain why communication has been a key benefit in our case study results.

3. **BDD benefits requirements** - A key indicative of a software project's success is the level to which it satisfies the stakeholders' expectations [51]. This indicative is tightly associated with an effective requirements engineering phase.

Through our studies, we were able to see that BDD promoted better management of requirements both as an expectation from the experts, which was reported in our

expert panel, and actual results, which were reported by the participants in our case study.

This may be a result from the relatively easiness in the usage of scenarios, which are plain-text real-world examples of a user story. This may be a reason as one the main problems regarding requirements engineering is the difficulty in using the available tools [43].

4. **BDD benefits implementation** - A poorly implemented software is a problem where the developed code is faulty, unreliable, difficult to maintain, difficult to modify, among other flaws. Software with this characteristics has been reported to cause numerous problems, ranging from financial fees to human deaths [81, 34]. In the education scenario, these problems also happen but tend to have mild consequences and learning is a process where mistakes are expected and even encouraged [39, 22].

In the context of our case study, it appears that BDD provides enough tooling for students so that they are able to implement code properly (*i.e.*, functional), faster and even maintaining a test policy. This indicates that a positive outcome in implementation is a possibility when using BDD in the education scenario.

5. **BDD benefits user comprehension** - Among its principles, BDD promotes high levels of interaction between stakeholders in a software project [75], which includes the end-user. Moreover, interactions with the end-user are not exclusive to BDD and can even be used to develop real-world applications [12, 49].

Results from our case study suggest that there is an increase in user-comprehension, where the expectations of the end-users were met by the final software product. This result is an indicative as many of the products developed by participants in our case study had no external end-users.

7.2.2 Challenges

Challenges found by the SLR

Mathies *et al.* [41] reported teaching TDD using Prof. CI. required extra-care to ensure tasks were clear and understandable. This challenged relates to the usage of Prof. CI. and the work-model it imposes. Thus, it does not appear to be a challenge which relates to our reasoning matter.

Simpson and Storer [73] stated that scalability, *i.e.*, performing the proposed changes in a larger cohort, can be a challenge. This is explained as related to studio-based learning [11], an approach where software engineering is based on collaboration and thus requires inter-personal relationships to take place.

Hoffman *et al.* [26] reinforced that teaching using ATDD requires dedication and commitment and thus it should be a concern of those who want to use it in combination with active learning methods.

Overall, the challenges discussed previously appear to be reasonable and help clarify which concerns should be discussed by those who are considering the implementation of test-driven methodologies in learning environments.

Challenges found in case study and expert panel

Applying the same principle as the benefits, we tagged the challenges reported by the participants in our case study using the categories reported by the experts during the expert panel study. As a reminder, these categories were generated using *Card Sorting*, a method explained in Chapter 4 (page 53), and were *BDD*, *Culture*, *Requirements*, *Team Engagement*, *Time* and *Others*. The result is presented in Table 7.4.

Table 7.4 – Negative aspects aspects of BDD (including categories)

Aspect	Occurrences	Category
Tests writing	8	BDD; Culture
Test-driven development	8	BDD; Culture
Scenario creation	4	BDD
Initial process	3	Culture; Requirements
Reduction in development time	3	Time
Lack of experience with BDD	2	Others
Small scope	2	BDD; Time
Team engagement with BDD	2	Team engagement
Scenario comprehension	1	BDD
Up-front need of complete project comprehension	1	Requirements
Team size	1	Others
Task distribution	1	Others
Scenario precision	1	BDD
Limitation in test amount	1	BDD; Culture

Following this, we proceeded to group the aspects in the categories reported by the experts and sum the number of occurrences of each category. Data from this table should allow us to better infer the challenges of BDD. The result is presented in Table 7.5.

Table 7.5 displays that BDD can be challenging when learning to develop software in active learning environments in numerous aspects.

Continuing to follow the methodology applied for the benefits, to enhance clarity and reason about each category of benefits, we present as follows a short statement of a BDD challenge followed by a short explanation about why this appears to be the case.

Table 7.5 – Negative aspects of BDD grouped by categories from experts

Category	Occurrences (Sum)	% of total
BDD	27	45.00
Culture	20	33.33
Time	5	8.33
Requirements	4	6.67
Others	2	3.33
Team Engagement	2	3.33
Total	60	100.00

1. **BDD processes may be challenging** - As BDD promotes the usage of processes throughout the software development lifecycle [75], it can be challenging for the development team.

In the education scenario, this challenge is probably higher given that most students do not possess industry experience. In addition to this, even though it is expected that student fail during the learning process [39, 22], students tend to report being frustrated when learning something new through active learning

2. **BDD culture may be difficult to implement** - When Dan North introduced BDD [50], it was a based in TDD [10]. As a consequence, BDD inherits the concept of using tests to drive development and its associated challenges, such as an unwillingness of the students to apply the development paradigm of TDD and the need of secondary skills, such as testing, designing and refactoring [45].

3. **BDD may not work in time-restricted environments** - In education, most activities have a timeframe, either a day, a week or an entire course. This timeframe ensures the objectives of a class are achieved in a reasonable time and allow the teacher/mentor to manage the schedule of activities.

In this scenario, BDD is challenging to be taught and/or used due to its many time-demanding activities. Customer collaboration, defining the ubiquitous language and automating acceptance tests are activities which have to be performed when using BDD [75] and increase the amount of time a development activity requires.

4. **BDD may hinder requirements** - According to Smart [75], BDD revolves around not only building a functional software, but one that addresses the needs of the customer. Among the main mechanisms provided by BDD, the augmentation of user stories using scenarios, a process which appears simple, but requires deep knowledge about business goals or customer expectations.

Creating good scenarios is a topic yet to be more deeply explored in the academia. Oliveira *et al.* [54, 53, 52] has performed a few studies on this topic with relevant con-

tribution, such as a question-based checklist to evaluate the quality of the generated scenario. However, further work is required to establish characteristics of a “good” scenario.

7.3 (RQ3) Are there any best practices when teaching BDD?

To address our third research question, we have crossed findings from our SLR and all studies performed in this work.

Best practices found by the SLR

Our SLR found three studies which had some intersection with our topic under study. By using these studies, we were able to extract some indicatives regarding setting and tooling. It is important to note that as none of the studies selected applied BDD in their teaching environments, reasoning from these studies can only provide indicatives and not final statements. In this sense, we found:

1. As for software development methodology, agile was applied by two of the selected studies [73, 26], which indicates that BDD is suited to be taught in an agile software development context. In addition, Smart [75] states that BDD works best in an iterative context, further promoting this indicative.
2. As for tooling that can be used, studies have reported using different tools [41, 26] or have not reported the applied tools [73]. In this sense, it becomes hard to conclude if a specific tool can be said to generate better results. As the tooling used was different, we can assume there is no standard tool.

As a final thought, we have not found sufficient evidence that supports any best practice, some tools and software methodologies have appeared and should be considered when teaching software practices that have a strong focus on testing, such as Test Driven Development (TDD), Acceptance-Test Driven Development (ATDD), and Behavior-Driven Development (BDD).

Best practices derived from our the case study

Our case study was conducted in a mobile development course which combined active learning and agile software development. Participants had the chance to work with and without BDD. Detailed results were reported in Chapter 6 (page 71) and seem to indicate that BDD can have positive and negative impacts. After crossing our study setting and

outcomes with some of the reported influencing factors presented in Chapter 5 (page 59), we have created the following set of best practices.

1. **Deliver BDD processes through short activities** - During our study on influencing factors, we were able to find indicatives that students had a better engagement with short activities (*e.g.*, CBL Nano-Challenge). Uniting this with some of the reported issues from our case study, such as a difficulty to write tests, a possible mitigation strategy would be introduced BDD processes and/or requirements individually prior to using the entire set of practices.
2. **Introduce BDD after agile has been introduced** - BDD foster collaboration and co-operation. As a consequence, according to Smart [75], it thrives in agile contexts. This indicates that BDD should fit more naturally if students have already experience agile software development.
3. **Introduce a test-driven development mindset prior to introducing BDD** - The two main problems reported by students during our case study were related to test-driven development mindset, where one starts development by creating a test case.

Due to the high dependence of BDD on this concept, it is recommendable that students are given a chance to experience the mindset of test-driven development prior to using BDD.

8. CONCLUSION

This chapter summarizes the work conducted by this study, highlight its relevant contributions and present opportunities for future work.

8.1 Summary

This study has performed a study of teaching BDD in active learning environments. Through four complementary studies, we have been able to better understand the implications of teaching BDD in these environments.

Initially, we have conducted a SLR regarding teaching BDD more broadly. We have found only three studies which were somewhat related to the proposed research question. Consequently, we have seen that the literature is scarce regarding this topic and there is a contribution opportunity available.

Next, we have conducted an expert panel with active learning experts aiming at eliciting expectations regarding benefits and challenges of BDD in active learning environments. Through this study, we have been able to explore the perspective of teachers regarding BDD and to draw conclusions towards it. Experts have reported that BDD should have benefits impacts, such as improving requirements engineering, but negative ones were expected too, such as the practices proposed by BDD.

Following this, we have sought to comprehend what influencing factors are important in active learning environments. To achieve this, we have conducted a survey in an active learning environment that teaches MAD. Through it, we have been able to increase comprehension regarding other important factors that could have impact when teaching in such environments. We have seen that project duration, group composition and the team size can have impacts in the individual perception of students.

As a final study, we have performed a case study in an active learning environment and were able to extract benefits and challenges from using BDD in this specific setting. Since BDD touches almost every aspect of the software development lifecycle, we have analysed its impact regarding: *requirements*, *feature development* and *implementation quality*. Consequently, we have been able to extract facets of BDD that are tangent to each topic. In a more general sense, we have seen that BDD can be beneficial, *e.g.*, by allowing team members to better comprehend the features under development, and challenging, *e.g.*, by requiring students to embrace test-driven development. Overall, however, we conclude that the benefits outweigh the challenges.

Finally, gathering knowledge obtained from these studies and following the triangulation method proposed by Creswell [17], we have been able to cross data from these

studies, solidify benefits and challenges BDD should have in active learning environments. Furthermore, this same principle allowed us to establish a set of best practices to be used when teaching BDD in active learning environments. These findings, provided the gap of studies addressing BDD, seem novel and should help teachers, schools, universities and any environment using active learning to teach software development to have a set of guidelines to add BDD to its curriculum.

8.2 Limitations

The reader must bear in mind that any findings and insights from this study are limited by the nature of the sub-studies of which it is composed. Each study has associated limitation, but some of these are more generally applied.

First, this research could have been submitted to the research bias, where the researcher expects a result and thus interprets findings from this perspective. This study has put in practice techniques which reduce this bias, such as multi-author analysis and discussion of results.

In addition, given that this study has made use of interviews, recordings and transcriptions, there is the limitation towards transcription of interviews and interpretation of responses by the researcher. This is mitigated by the scientific procedures adopted, but cannot be fully eliminated.

Moreover, personal opinions of participants may have suffered impact once the research purpose is made explicit. Attempts to avoid this effect revolved around providing the research objective neutrally, ensuring participants understand the lack of research impact towards their personal evaluations and generating non-biased interview questions.

Finally, aiming at reducing these biases, we have submitted all our individual studies to peer-reviewed conferences. These include:

1. **Behavior-Driven Development Teaching: A Systematic Literature Review** - Submitted for SBES 2020.
2. **Behavior-Driven Development - An Expert Panel to evaluate benefits and challenges** - Submitted for SBES 2020;
3. **An Investigation of Influencing Factors when Teaching on Active Learning Environments** [47];
4. **Behavior-Driven Development - A case study on its impacts on agile development teams** - Presented and published at CHASE 2020;

8.3 Future Work

This study was able to address relevant information regarding BDD in active learning environments. Naturally, however, it is limited by uncontrollable factors, such as time-constraints. Furthermore, as scientific study, it not only serves to answer the proposed research questions, but also to open space for future study opportunities.

Consequently, we have outlined still-open topics to be discussed and that escape the scope of this study. Some of these studies include, but are not limited to:

1. **Assessing the impacts and effectiveness of individual practices of BDD** - BDD is a hub of practices which impact the entire development lifecycle, from requirement to documentation. As a consequence, it promotes a series of practices, such the augmentation of user stories using scenarios [75].

Each individual practice has specific impacts and understanding these individual impacts could lead to better usage of these practices and maximize the outcomes promoted by them.

2. **Adapting BDD practices for software engineering education** - The educational context has its peculiarities and one of the roles of a teacher is to adapt industry practices to work in such context.

BDD is not different. It proposes the usage of industry specific tooling, such as JBehave¹ and Gherkin [14]. Adapting these tools to maximize efficiency but remain loyal to the original concept of BDD is still a study to be conducted.

Further, to address the issue of BDD being too cumbersome for students, many techniques could be applied, such as Gamification. Studies which

3. **Comparing BDD and other test-driven development methodologies** - Test-driven development (TDD), proposed by Beck [10], originated many development methodologies, such as ATDD and BDD. Each one promoting slightly different practices but maintaining the overarching concept of developing based on test cases.

These methodologies have an overarching similarity and their specific impacts and differences are topics which the scientific community is yet to address. Consequently, studies which compare these methodologies are not numerous and an in-depth study is a work still to be performed.

¹<https://jbehave.org/>

REFERENCES

- [1] Abrahamsson, P.; Hanhineva, A.; Hulkko, H.; Ihme, T.; Jääliñoja, J.; Korkala, M.; Koskela, J.; Kyllönen, P.; Salo, O. “Mobile-d: an agile approach for mobile application development”. In: Proceedings of the 19th annual Companion to the Conference on Object-Oriented Programming Systems, Languages, and Applications, 2004, pp. 174–175.
- [2] Acuña, S. T.; Gómez, M.; Juristo, N. “How do personality, team processes and task characteristics relate to job satisfaction and software quality?”, *Journal of Information and Software Technology*, vol. 51–3, March 2009, pp. 627–639.
- [3] Adzic, G. “Specification by example”. Citeseer, 2011, 20p.
- [4] Ahmad, K.; Gestwicki, P. “Studio-based learning and app inventor for android in an introductory cs course for non-majors”. In: Proceeding of the 44th Technical Symposium on Computer Science Education, 2013, pp. 287–292.
- [5] Anand, V.; Dinakaran, M. “Popular agile methods in software development: Review and analysis”, *International Journal of Applied Engineering Research*, vol. 11–5, November 2016, pp. 3433–3437.
- [6] Babbie, E. “Survey research methods”. Universidade Federal de Minas Gerais, 2005, 383p.
- [7] Barrows, H.; Tamblyn, R.; et al.. “Problem-based learning: an approach to medical education”. Springer Publishing Company, 1980, 228p.
- [8] Barry, E. J.; Mukhopadhyay, T.; Slaughter, S. A. “Software project duration and effort: an empirical study”, *Journal of Information Technology and Management*, vol. 3–1-2, January 2002, pp. 113–136.
- [9] Beck, K. “Extreme programming explained: embrace change”. Addison-Wesley Longman, 2000, 224p.
- [10] Beck, K. “Test-driven development: by example”. Addison-Wesley Professional, 2003, 240p.
- [11] Bull, C. N.; Whittle, J.; Cruickshank, L. “Studios in software engineering education: towards an evaluable model”. In: Proceedings of the 35th International Conference on Software Engineering, 2013, pp. 1063–1072.
- [12] Buller, D. B.; Berwick, M.; Shane, J.; Kane, I.; Lantz, K.; Buller, M. K. “User-centered development of a smart phone mobile application delivering personalized real-time

advice on sun protection”, *Journal of Translational Behavioral Medicine*, vol. 3–3, September 2013, pp. 326–334.

- [13] Chanin, R.; Sales, A.; Santos, A. R.; Pompermaier, L.; Prikladnicki, R. “A collaborative approach to teaching software startups: findings from a study using challenge based learning”. In: *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, 2018, pp. 9–12.
- [14] Chelimsky, D.; Astels, D.; Helmkamp, B.; North, D.; Dennis, Z.; Hellesoy, A. “The RSpec book: behaviour driven development with rspec”. Pragmatic Bookshelf, 2010, 25p.
- [15] Chi, M. T.; Wylie, R. “The icap framework: linking cognitive engagement to active learning outcomes”, *Journal of Educational Psychologist*, vol. 49–4, October 2014, pp. 219–243.
- [16] Coad, P.; Luca, J. d.; Lefebvre, E. “Java modeling color with UML: enterprise components and process with cdrom”. Prentice Hall, 1999, 221p.
- [17] Creswell, J.; Creswell, J. “Research design: qualitative, quantitative, and mixed methods approaches”. Sage Publications, 2017, 304p.
- [18] Dyba, T. “An instrument for measuring the key factors of success in software process improvement”, *Journal of Empirical Software Engineering*, vol. 5–4, December 2000, pp. 357–390.
- [19] Evans, E. “Domain-driven design: tackling complexity in the heart of software”. Addison-Wesley Professional, 2004, 560p.
- [20] Fernandez-Sanz, L.; Misra, S. “Influence of human factors in software quality and productivity”. In: *Proceedings of the 11th International Conference on Computational Science and Its Applications*, 2011, pp. 257–269.
- [21] Fetaji, M.; Fetaji, B. “Analyses of mobile learning software solution in education using the task based learning approach”. In: *Proceedings of the 31st International Conference on Information Technology Interfaces*, 2009, pp. 373–378.
- [22] Fischer, M. A.; Mazor, K. M.; Baril, J.; Alper, E.; DeMarco, D.; Pugnaire, M. “Learning from mistakes”, *Journal of General Internal Medicine*, vol. 21–5, May 2006, pp. 419–423.
- [23] Fowler, M.; Highsmith, J.; et al.. “The agile manifesto”, *Journal of Software Development*, vol. 9–8, August 2001, pp. 28–35.
- [24] Gavalas, D.; Economou, D. “Development platforms for mobile applications: status and trends”, *Journal of Software*, vol. 28–1, December 2010, pp. 77–86.

- [25] Gestwicki, P.; Ahmad, K. "App inventor for android with studio-based learning", *Journal of Computing Sciences in Colleges*, vol. 27–1, October 2011, pp. 55–63.
- [26] Hoffmann, L. F.; de Vasconcelos, L. E.; Lamas, E.; Cunha, A. M.; Dias, L. A. "Applying acceptance test driven development to a problem based learning academic real-time system". In: Proceedings of the 11th International Conference on Information Technology, 2014, pp. 3–8.
- [27] Johnson, R. T.; Johnson, D. W. "Active learning: cooperation in the classroom", *Journal of the Annual Report of Educational Psychology in Japan*, vol. 47, March 2008, pp. 29–30.
- [28] Joorabchi, M. E.; Mesbah, A.; Kruchten, P. "Real challenges in mobile app development". In: Proceedings of the 7th International Symposium on Empirical Software Engineering and Measurement, 2013, pp. 15–24.
- [29] Kalelioğlu, F. "A new way of teaching programming skills to k-12 students: code. org", *Journal of Computers in Human Behavior*, vol. 52, November 2015, pp. 200–210.
- [30] Keele, S.; et al.. "Guidelines for performing systematic literature reviews in software engineering", Tech report, Keele University, 2007, 57p.
- [31] Kitchenham, B.; Charters, S. "Guidelines for performing systematic literature reviews in software engineering", Tech report, Keele University and Durham University, 2007, 57p.
- [32] Kitchenham, B.; Pickard, L.; Pfleeger, S. L. "Case studies for method and tool evaluation", *Journal of Software*, vol. 12–4, July 1995, pp. 52–62.
- [33] Klem, A. M.; Connell, J. P. "Relationships matter: Linking teacher support to student engagement and achievement", *Journal of School Health*, vol. 74–7, September 2004, pp. 262–273.
- [34] Krasner, H. "The cost of poor quality software in the us: A 2018 report", Tech report, Consortium for IT Software Quality, 2018, 44p.
- [35] Larman, C.; Vodde, B. "Acceptance test-driven development with robot framework", *Journal of Eurostar*, vol. 1, May 2010, pp. 10.
- [36] Lauesen, S.; Vinter, O. "Preventing requirement defects: An experiment in process improvement", *Journal of Requirements Engineering*, vol. 6–1, February 2001, pp. 37–50.
- [37] Li, P. L.; Ko, A. J.; Zhu, J. "What makes a great software engineer?" In: Proceedings of the 37th International Conference on Software Engineering, 2015, pp. 700–710.

- [38] Lie, A. "Cooperative learning (cover baru)". Grasindo, 2002, 86p.
- [39] Lundquist, R. "Critical thinking and the art of making good mistakes", *Journal of Teaching in Higher Education*, vol. 4–4, October 1999, pp. 523–530.
- [40] Mahmoud, Q. H.; Popowicz, P. "A mobile application development approach to teaching introductory programming". In: Proceedings of the 40th Frontiers in Education Conference, 2010, pp. T4F–1.
- [41] Matthies, C.; Treffer, A.; Uflacker, M. "Prof. ci: employing continuous integration services and github workflows to teach test-driven development". In: Proceedings of the 47th Frontiers in Education Conference, 2017, pp. 1–8.
- [42] Maxwell, J. "Designing a qualitative study". Sage Thousand Oaks, 2008, chap. 7, pp. 214–253.
- [43] Memon, R. N.; Ahmad, R.; Salim, S. S. "Problems in requirements engineering education: a survey". In: Proceedings of the 8th International Conference on Frontiers of Information Technology, 2010, pp. 1–6.
- [44] Moraes, L. "An empirical study on the use of bdd and its support to requirements engineering", Master's dissertation, Pontifical Catholic University of Rio Grande do Sul, 2016, 139p.
- [45] Mugridge, R. "Challenges in teaching test driven development". In: Proceedings of the 2nd International Conference on Extreme Programming and Agile Processes in Software Engineering, 2003, pp. 410–413.
- [46] Murphy, B.; Bird, C.; Zimmermann, T.; Williams, L.; Nagappan, N.; Begel, A. "Have agile techniques been the silver bullet for software development at microsoft?" In: Proceedings of the 7th International Symposium on Empirical Software Engineering and Measurement, 2013, pp. 75–84.
- [47] Nascimento, N.; R. Santos, A.; Sales, A.; Chanin, R. "An investigation of influencing factors when teaching on active learning environments". In: Proceedings of the 33rd Brazilian Symposium on Software Engineering, 2019, pp. 517–522.
- [48] Nichols, M.; Cator, K.; Torres, M. "Challenge based learning guide". Digital Promise, 2016, 59p.
- [49] Nilsson, S. "Augmentation in the wild: user centered development and evaluation of augmented reality applications", Phd thesis, Linköping University Electronic Press, 2010, 173p.
- [50] North, D. "Introducing bdd", *Journal of Better Software*, vol. 12, March 2006, pp. 5.

- [51] Nuseibeh, B.; Easterbrook, S. "Requirements engineering: a roadmap". In: Proceedings of the 1st Conference on the Future of Software Engineering, 2000, pp. 35–46.
- [52] Oliveira, G.; Marczak, S. "On the empirical evaluation of bdd scenarios quality: preliminary findings of an empirical study". In: Proceedings of the 25th International Conference Workshops on Requirements Engineering, 2017, pp. 299–302.
- [53] Oliveira, G.; Marczak, S. "On the understanding of bdd scenarios' quality: preliminary practitioners' opinions". In: Proceedings of the 12th International Working Conference on Requirements Engineering: Foundation for Software Quality, 2018, pp. 290–296.
- [54] Oliveira, G.; Marczak, S.; Moralles, C. "How to evaluate bdd scenarios' quality?" In: Proceedings of the 33th Brazilian Symposium on Software Engineering, 2019, pp. 481–490.
- [55] Paetsch, F.; Eberlein, A.; Maurer, F. "Requirements engineering and agile software development". In: Proceedings of the 12th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003, pp. 308–313.
- [56] Palmer, S.; Felsing, M. "A practical guide to feature-driven development". Pearson Education, 2001, 304p.
- [57] Perry, D.; Porter, A.; Votta, L. "Empirical studies of software engineering: a roadmap". In: Proceedings of the 1st Conference on the Future of Software Engineering, 2000, pp. 345–355.
- [58] Pfleeger, S. L.; Kitchenham, B. "Principles of survey research: part 1: turning lemons into lemonade", *Journal of Software Engineering Notes*, vol. 26–6, November 2001, pp. 16–18.
- [59] Pikkarainen, M.; Haikara, J.; Salo, O.; Abrahamsson, P.; Still, J. "The impact of agile practices on communication in software development", *Journal of Empirical Software Engineering*, vol. 13–3, June 2008, pp. 303–337.
- [60] Pressman, R. "Software engineering: a practitioner's approach". Palgrave Macmillan, 2005, chap. 3, pp. 65–93.
- [61] Prince, M. "Does active learning work? a review of the research", *Journal of Engineering Education*, vol. 93–3, July 2004, pp. 223–231.
- [62] Procaccino, D.; Verner, J. M.; Overmyer, S. P.; Darter, M. E. "Case study: factors for early prediction of software development success", *Journal of Information and Software Technology*, vol. 44–1, January 2002, pp. 53–62.

- [63] Rahimian, V.; Ramsin, R. "Designing an agile methodology for mobile software development: A hybrid method engineering approach". In: Proceedings of the 2nd International Conference on Research Challenges in Information Science, 2008, pp. 337–342.
- [64] Robson, C. "Real world research". Wiley Chichester, 2011, 608p.
- [65] Rosqvist, T.; Koskela, M.; Harju, H. "Software quality evaluation based on expert judgement", *Journal of Software Quality*, vol. 11–1, May 2003, pp. 39–55.
- [66] Runeson, P.; Höst, M. "Guidelines for conducting and reporting case study research in software engineering", *Journal of Empirical Software Engineering*, vol. 14–2, April 2009, pp. 131.
- [67] Salleh, N.; Mendes, E.; Grundy, J. "Empirical studies of pair programming for cs/se teaching in higher education: a systematic literature review", *Journal of Transactions on Software Engineering*, vol. 37–4, June 2011, pp. 509–525.
- [68] Santos, A. R.; Kroll, J.; Sales, A.; Fernandes, P.; Wildt, D. "Investigating the adoption of agile practices in mobile application development". In: Proceedings of the 18th International Conference on Enterprise Information Systems, 2016, pp. 490–497.
- [69] Santos, A. R.; Sales, A.; Fernandes, P.; Nichols, M. "Combining challenge-based learning and scrum framework for mobile application development". In: Proceedings of the 20th Conference on Innovation and Technology in Computer Science Education, 2015, pp. 189–194.
- [70] Schwaber, K.; Sutherland, J. "The scrum guide". Scrum Alliance, 2011, 19p.
- [71] Serrador, P.; Pinto, J. K. "Does agile work?—a quantitative analysis of agile project success", *International Journal of Project Management*, vol. 33–5, July 2015, pp. 1040–1051.
- [72] Shaw, M. "Writing good software engineering research papers: minitutorial". In: Proceedings of the 25th International Conference on Software Engineering, 2003, pp. 726–736.
- [73] Simpson, R.; Storer, T. "Experimenting with realism in software engineering team projects: an experience report". In: Proceedings of the 30th Conference on Software Engineering Education and Training, 2017, pp. 87–96.
- [74] Sjøberg, D. I.; Hannay, J. E.; Hansen, O.; Kampenes, V. B.; Karahasanovic, A.; Liborg, N.-K.; Rekdal, A. C. "A survey of controlled experiments in software engineering", *Journal of Transactions on Software Engineering*, vol. 31–9, October 2005, pp. 733–753.

- [75] Smart, J. "BDD in action". Manning Publications, 2014, 353p.
- [76] Solis, C.; Wang, X. "A study of the characteristics of behaviour driven development". In: Proceedings of the 37th Euromicro Conference on Software Engineering and Advanced Applications, 2011, pp. 383–387.
- [77] Statista. "Cumulative number of apps downloaded from the apple app store from july 2008 to june 2017 (in billions)". Source: <https://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store/>, 29-june-2019.
- [78] Statista. "Smartphone are the bulk of our digital media diet". Source: <https://www.statista.com/chart/18347/hours-spent-on-digital-media/>, 29-june-2019.
- [79] Sutherland, J.; Schwaber, K. "The scrum papers: nut, bolts, and origins of an agile framework". Scrum Inc, 2011, 224p.
- [80] Takeuchi, H.; Nonaka, I. "The new new product development game", *Journal of Business Review*, vol. 64–1, January 1986, pp. 137–146.
- [81] Tassej, G. "The economic impacts of inadequate infrastructure for software testing", Tech report, National Institute of Standards and Technology, 2002, 309p.
- [82] Turnquist, G. "Python testing cookbook". Packt Publishing, 2011, 364p.
- [83] Wasserman, A. I. "Software engineering issues for mobile application development". In: Proceedings of the 1st Workshop on Future of Software Engineering Research, 2010, pp. 397–400.
- [84] Wieringa, R.; Maiden, N.; Mead, N.; Rolland, C. "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion", *Journal of Requirements Engineering*, vol. 11–1, March 2005, pp. 102–107.
- [85] Zwikael, O.; Unger-Aviram, E. "Hrm in project groups: The effect of project duration on team development effectiveness", *International Journal of Project Management*, vol. 28–5, July 2010, pp. 413–421.

APPENDIX A – INTERVIEW AGREEMENT TERM

TERMO DE CONSENTIMENTO PARA AS ENTREVISTAS TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

Nós, Nicolas Pereira do Nascimento (aluno de Mestrado) e Afonso Henrique Sales (professor orientador), responsáveis pela pesquisa ***A study of teaching BDD in active learning environments***, estamos fazendo um convite para você participar como voluntário nesse estudo.

Este estudo espera criar um corpo de conhecimento sobre os impactos de ensino de *behavior-driven development* (BDD) em ambientes de aprendizagem ativa. Para tanto, há a necessidade do entendimento dos benefícios e desafios do uso desta metodologia sob a ótica daqueles que estão inseridos no contexto de aprendizagem ativa. Não há benefícios a curto prazo para os participantes dessa pesquisa, contudo, ao término desse estudo a seguinte contribuição é esperada: A criação de um corpo de conhecimento a cerca dos efeitos do uso de BDD em ambientes de aprendizagem ativa que possa servir de guia para professores, instrutores e/ou times de desenvolvimento de software que desejam introduzir BDD no seu contexto.

Para a coleta dos dados serão utilizadas entrevistas semi-estruturadas. Entendemos que há riscos mínimos durante essas atividades como: divulgação de dados confidenciais (quebra de sigilo), desconforto e/ou constrangimento durante gravações de áudio e/ou vídeo. Lembrando que o objetivo deste estudo não é avaliar o participante, mas, sim, analisar os processos de trabalho relacionados ao tópico da pesquisa. O uso que se faz dos registros efetuados durante o teste é estritamente limitado a atividades acadêmicas e buscaremos garantir seu anonimato e confidencialidade

Outras informações importantes:

As informações desta pesquisa serão confidenciais, e serão divulgadas apenas em eventos e/ou publicações científicas, não havendo identificação dos participantes, a não ser entre os responsáveis pelo estudo, sendo assegurado o sigilo sobre sua participação.

Você tem garantido o seu direito de não aceitar participar e/ou de retirar sua

permissão, a qualquer momento, sem nenhum tipo de prejuízo ou retaliação, pela sua decisão.

Se por algum motivo você tiver despesas decorrentes da sua participação neste estudo com transporte e/ou alimentação, você será reembolsado adequadamente pelos pesquisadores. Você tem o direito de pedir uma indenização por qualquer dano que resulte da sua participação no estudo.

Ao assinar este termo de consentimento, você não abre mão de nenhum direito legal que teria de outra forma.

Somente assine este termo de consentimento caso tenha tido a oportunidade de fazer perguntas e tenha recebido respostas satisfatórias para suas dúvidas.

Se você concordar em participar deste estudo, você rubricará todas as páginas e assinará e datará duas vias originais deste termo de consentimento. Você receberá uma das vias para seus registros e a outra será arquivada pelo responsável pelo estudo.

Durante todo o período da pesquisa você tem o direito de esclarecer qualquer dúvida ou pedir qualquer outro esclarecimento, bastando para isso entrar em contato com:

Nicolas – (51) 993641782; E-mail: nicolas.nascimento@pucrs.br
Afonso – (51) 981438256; E-mail: afonso.sales@pucrs.br

Caso você tenha qualquer dúvida quanto aos seus direitos como participante de pesquisa, entre em contato com Comitê de Ética em Pesquisa da Pontifícia Universidade Católica do Rio Grande do Sul (CEP-PUCRS) em (51) 33203345, Av. Ipiranga, 6681/prédio 50, sala 703, Porto Alegre – RS, e-mail: cep@pucrs.br, de segunda a sexta-feira das 8h às 12h e das 13h30 às 17h. O CEP é um órgão independente constituído de profissionais das diferentes áreas do conhecimento e membros da comunidade. Sua responsabilidade é garantir a proteção dos direitos, a segurança e o bem-estar dos participantes por meio da revisão e da aprovação do estudo, entre outras ações.

Eu, _____, após a leitura deste documento e de ter tido a oportunidade de conversar para esclarecer todas as minhas dúvidas, acredito estar suficientemente informado, ficando claro para mim que minha participação é voluntária e que posso retirar este consentimento a qualquer momento sem penalidades e/ou perda de qualquer benefício. Estou ciente também dos objetivos da pesquisa, dos procedimentos aos quais serei submetido, dos possíveis danos ou riscos deles provenientes e da garantia de confidencialidade e esclarecimentos sempre que desejar.

Diante do exposto expresso minha concordância de espontânea vontade em participar deste estudo.

Assinatura do participante da pesquisa

Assinatura de uma testemunha

DECLARAÇÃO DO PROFISSIONAL QUE OBTEVE O CONSENTIMENTO

Expliquei integralmente este estudo ao participante. Na minha opinião e na opinião do participante, houve acesso suficiente às informações, incluindo riscos e benefícios, para que uma decisão consciente seja tomada.

Data: _____


Nicolas Pereira do Nascimento

Aluno de Mestrado em Ciência da
Computação Politécnica
PPGCC – Escola Politécnica


Afonso Henrique Sales


Professor do PPGCC – Escola
Politécnica


APPENDIX B – COMPLETE QUESTIONNAIRE - EXPERT PANEL


*  1 How old are you ?

*  2 Where are you from (Country and state/province) ?

*  3 How many years of academic experience do you have ?


*  4 How many years of industry experience do you have ?


*  5 For how long have you been a part of the Apple Developer Academy (years) ?

*  6 Have you ever taught or used Behavior-Driven Development (BDD) ?

*  7 Are you currently teaching or using BDD in your Academy ?

*  8 In your opinion, What are the main benefits of BDD in the academy ?

*  9 In your opinion, What are the main challenges of BDD in the academy?

 10 Any final comments or thoughts ?

APPENDIX C – COMPLETE QUESTIONNAIRE - SURVEY

Mini Challenge 1

+ Add

Mini Challenge 1 (Rating)

★★★★★

Mini Challenge 2

+ Add

Mini Challenge 2 (Rating)

★★★★★

Nano Challenge - Games

+ Add

Nano Challenge - Games (Rating)

★★★★★

Mini Challenge 3

+ Add

Mini Challenge 3 (Rating)

★★★★★

Nano Challenge - Apple TV

+ Add

Nano Challenge - Apple TV (Rating)

★★★★★

Mini Challenge 4

+ Add

Mini Challenge 4 (Rating)

★★★★★

Nano Challenge - Arduino

+ Add

Nano Challenge - Arduino (Rating)

★★★★★

Nano Challenge - Watch

+ Add

Nano Challenge - Watch (Rating)

★★★★★

Final Challenge

+ Add

Final Challenge (Rating)

★★★★★

Submit

APPENDIX D – COMPLETE QUESTIONNAIRE - CASE STUDY

D.1 Pre-BDD

Idade

Curso

Semestre

Anos de experiência em desenvolvimento de software

Em relação aos requisitos do projeto, como vocês elicitaram estes?

Em relação aos requisitos do projeto, como vocês especificavam estes?

Em relação aos requisitos originalmente planejados, houve algum requisito desenvolvido que acabou ficando bastante diferente ou foi alterado completamente?

Em relação aos requisitos originalmente planejados, houve algum aspecto ambíguo? Se sim, como vocês resolviam as ambiguidades destes requisitos?

Em termos de qualidade de código (separação de responsabilidades, modularização, tratamento de erros, etc), como você diria que o seu projeto foi?

Em relação a bugs em seu aplicativo (comportamento inesperado), como você diria que seu projeto foi? Quão críticos são estes bugs?

Como vocês realizavam a documentação das funcionalidades do projeto?

Submit

D.2 Post-BDD

Idade

Curso

Semestre

Anos de experiência em desenvolvimento de software

Em relação aos requisitos do projeto, como vocês elicitaram estes?

Em relação aos requisitos do projeto, como vocês especificavam estes?

Qual foi o impacto do uso de BDD em relação ao processo de elicitação/especificação dos requisitos ?

Em relação aos requisitos originalmente planejados, houve algum aspecto ambíguo? Se sim, como vocês resolviam as ambiguidades destes requisitos?

Em relação aos requisitos originalmente planejados, houve algum requisito desenvolvido que acabou ficando bastante diferente ou foi alterado completamente?

Qual fo o impacto do uso de BDD neste processo de tradução de requisitos para funcionalidades ?

Em termos de qualidade de código (separação de responsabilidades, modularização, tratamento de erros, etc), como você diria que o seu projeto foi?

Em relação a bugs em seu aplicativo (comportamento inesperado), como você diria que seu projeto foi? Quão críticos são estes bugs?

Como vocês realizavam a documentação das funcionalidades do projeto? Você consideraria o Scenario como uma documentação?

Qual foi o impacto de BDD referente a implementação ?

Em relação ao desenvolvimento utilizando BDD, quais foram os principais benefícios?

Em relação ao desenvolvimento utilizando BDD, quais foram os principais desafios?

Submit