Teaching BDD in Active Learning Environments: A Multi-study Analysis

Nicolas Nascimento[a], Alan Santos[b], Afonso Sales[c] and Rafael Chanin[d]
Polytechnical School, Pontifical Catholic University of Rio Grande do Sul, Avenida Ipiranga 6681, Porto Alegre, Brazil

Keywords: Software Engineering Education, Behavior-Driven Development, Agile Development, Active Learning.

Abstract: Software development practices to 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 approach which proposes software to be developed focusing primarily on its expected behavior. Teaching-wise, introducing BDD on software engineering classes and/or training courses for software developers has become important. In this context, this study presents a body of knowledge on the impacts of teaching BDD in active learning environments (ALE). To achieve this, we have triangulated data from four data sources: (i) a systematic literature review; (ii) an expert panel with active-learning experts, (iii) a survey with participants in a software development course which teaches through active learning, and (iv) a case study on the effects of teaching and using BDD in an ALE. This study results are (i) the-state-of-the-art literature on this topic, (ii) an assessment of benefits and challenges of BDD in ALEs, and (iii) a set of best practices when teaching BDD in ALEs. We concluded that BDD has more positive than negative outcomes and we present a body of knowledge regarding BDD in ALEs.

1 INTRODUCTION

The omnipresence of software is undeniable (Li et al., 2015) and many modern services and business would not be able to operate without software solutions. This trend increases the need for software developers and development practices which enhance software quality and help teams better develop collaboratively. Among modern software development techniques, is Behavior-Driven Development (BDD), a development method which proposes software to be develop based on its expected behavior (North et al., 2006). BDD improves development by both ensuring software developed is reliable and is aligned with the needs of the customer (Smart, 2014). Consequently, the software engineering education scenario is presented with the challenge to adapt quickly and to better prepare students. To address this, active learning methodologies, such as Challenge-Based Learning (CBL), are drawing attention due to reports of improved learning (Deslauriers et al., 2019) and student engagement (Santos et al., 2015; Scharff et al., 2009; Ahmad and Gestwicki, 2013; Matos and Grasser, 2010).

This paper presents a multi-study of teaching BDD in active learning environments (ALEs) aiming at understanding the impacts of teaching BDD in ALEs. To achieve this, we have used triangulation (Creswell and Creswell, 2017), crossing the results of four previous studies: (i) a Systematic Literature Review, (ii) an Expert Panel, (iii) a Survey, and (iv) a Case Study. We concluded that BDD has more positive than negative outcomes and we present a body of knowledge regarding BDD in ALEs.

2 BACKGROUND

Behavior-Driven Development is a software development practice, proposed by Dan North (North et al., 2006) and derived from Test-Driven Development (TDD) (Beck, 2003), which aims at helping software development teams to build software which follows the needs of the customer (Smart, 2014). As a development methodology, BDD emphasizes test cases which are written in a common language (Evans, 2004). The specification of these test cases revolves around *"BDD scenarios"*. These scenarios are created from concrete examples of a system's user stories. User stories are a lightweight specification

[a] https://orcid.org/0000-0002-0080-8822
[b] https://orcid.org/0000-0001-8323-3472
[c] https://orcid.org/0000-0001-6962-3706
[d] https://orcid.org/0000-0002-6293-7419

Nascimento, N., Santos, A., Sales, A. and Chanin, R.
Teaching BDD in Active Learning Environments: A Multi-study Analysis.
DOI: 10.5220/0011057300003182
In *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022) - Volume 2*, pages 268-275
ISBN: 978-989-758-562-3; ISSN: 2184-5026

method commonly used in agile software development. BDD Scenarios are used to further enhance the descriptive capabilities of these user stories.

3 METHODOLOGY

This paper presents an investigation regarding the application of BDD when teaching in active learning environments. To achieve this main objective, a series of steps were undertaken.

3.1 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.

3.2 Research Protocol

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. 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. Complementarily, we have performed a survey research to investigate influencing factors when teaching in active learning environments. Finally, to actively extract benefits and challenges of BDD in active learning environments, we have performed a case study research.

After having collected data and findings from all these studies, we triangulated the results. According to Creswell (Creswell and Creswell, 2017), 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.

4 SUB-STUDIES

4.1 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 (SLR). According to standard guidelines for performing this research method (Keele et al., 2007), a SLR 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.

Given that this work focuses on the triangulation process and due to page limitation, we have decided to omit the SLR research design and present the papers found only. The three studies which were selected after conducting the complete SLR processes were Mathies *et al.* (Matthies et al., 2017), Hoffman *et al.* (Hoffmann et al., 2014) and Simpson *et al.* (Simpson and Storer, 2017).A summary table of these studies is presented in Table 1 (It is worth mentioning that the final row in the table is our own work).

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.

4.2 Expert Panel

This subsection presents a summary of the results obtained from an expert panel study (Nascimento et al., 2020b) performed with active-learning environments experts. this summary is brief as the the focus of this paper is the triangulation of results and due to page limitations.

Expert panel research enables to researcher to capture expert judgment (Rosqvist et al., 2003) and help improving hypothesis generated. Moreover, the opinion of experts is a valuable research artefact in the Software Engineering (SE) research community (Dyba, 2000).

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. In total, 28 experts participated in the study.

Drawing on the results obtained from this study, we were able to extract the following categories of potential benefits and challenges reported the experts:

1. **Benefits**
   (a) *Requirements* - *i.e.*, elicitation, specification and validation;
   (b) *User Comprehension* - *i.e.*, the understanding of the needs of user of the application;
   (c) *Project* - *i.e.*, project management;
   (d) *Implementation* - *i.e.*, coding and documenting the application;
   (e) *Communication* - *i.e.*, communication among team members;
   (f) *Others* - *i.e.*, other subjects.
2. **Challenges**
   (a) *BDD* - *i.e.*, processes of BDD;
   (b) *Culture* - *i.e.*, test-driven development mindset;
   (c) *Requirements* - *i.e.*, elicitation, specification and validation;
   (d) *Team Engagement* - *i.e.*, the team's willingness to embrace BDD;
   (e) *Time* - *i.e.*, project duration and time constraints;
   (f) *Others* - *i.e.*, other subjects.

These results seem to indicate the main categories of benefits and challenges that teaching BDD should present. Following this study, we have conducted a survey with students in an active learning environment.

4.3 Survey

This subsection presents a summary of an investigation of influencing factors when teaching MAD in an active learning environment (ALE) using Challenge based learning (CBL) (Nascimento et al., 2019). We conducted this study to better understand influencing factors in ALEs. The investigation was performed through survey on a two-year course that teaches MAD to undergraduate students. 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 (Babbie, 2005). 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.

Focusing on the results obtained from the study due to page limitations, 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. Following this study, we have decided to conduct a case study in an ALE.

4.4 Case Study

This subsection presents a summary of the results of a case study we have conducted in a mobile application development course (Nascimento et al., 2020a). The course teaches students using an active learning framework, Challenge Based Learning (CBL) (Nichols et al., 2016). Our goal was to investigate how BDD impacts agile software development teams in active learning environments.

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 (Runeson and Höst, 2009) and are suited to evaluate a method and tool (Kitchenham et al., 1995).

Focusing on the main results obtained, we present the positive aspects reported in Table 2 and the negative aspects in Table 3.

Overall, the 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 their software development lifecycle by using BDD.

5 DISCUSSION

In this section, we will further reason about the research questions of this study and consolidate the contributions of the work. This will be performed based on each one of our research questions and using the results obtained in our studies and additional literature. To address these research questions, we applied

Table 1: Summary of results from the studies (including case study).

Study	Tools(s)	Learning framework	Agile framework	Dev. practice
Matthies (2017) (Matthies et al., 2017)	Prof. CI.	-	Custom	TDD
Hoffman (2014) (Hoffmann et al., 2014)	Robot, Selenium and Coverage	Problem-based learning	Scrum	ATDD
Simpson (2017) (Simpson and Storer, 2017)	-	Flipped-classroom	Scrum	TDD
Author (2020) (Nascimento et al., 2020a)	Quick & Nimble and Xcode	Challenge-based learning	Scrum	BDD

triangulation, as proposed by Creswell (Creswell and Creswell, 2017).

In this sense, we have collected data from four sources types: our SLR (4.1), Expert Panel (4.2), Survey (4.3), and Case Study (4.4). It is important to notice that the participants in each study were different in order to mitigate the research bias.

5.1 (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. The updated table is presented in Table 1.

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.* (Matthies et al., 2017), and the Robot framework (which was used alongside other tools, reported by Hoffman *et al.* (Hoffmann et al., 2014)), 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* (Smart, 2014), 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 similar development methodology which is similar to 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 (Solis and Wang, 2011) *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 Jbehave[1], 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.*, (Simpson and Storer, 2017; Hoffmann et al., 2014), 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 (Murphy et al., 2013; Serrador and Pinto, 2015), and BDD's natural fit in agile development contexts (North et al., 2006; Smart, 2014).

5.2 (RQ2) What Are the Benefits and Challenges of using BDD in Active Learning Environments?

To answer the second research question, we have de-

[1]https://jbehave.org/

cided to use the results from our case study primarily. This is due to the low number (3) of papers found in our SLR.

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.

5.2.1 Benefits

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 used in the expert panel 4.2, and were *requirements*, *user comprehension*, *project*, *implementation*, *communication* and *others*. The top five results are presented in Table 2.

Table 2: Pos. aspects of BDD (with categories).

Aspect	Occ.	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

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.

After doing this, we have noticed that BDD can be beneficial when learning to develop software in active learning environments in many aspects. 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.* (Procaccino et al., 2002) 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 (Procaccino et al., 2002).

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 (Pikkarainen et al., 2008).

   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 (Smart, 2014) *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 (Nuseibeh and Easterbrook, 2000). 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 (Memon et al., 2010).

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 (Planning, 2002; Krasner, 2018). 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 (Lundquist, 1999; Fischer et al., 2006).

   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 (Smart, 2014), 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 (Buller et al., 2013; Nilsson, 2010).

   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.

5.2.2 Challenges

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 the expert panel (4.2), and were *BDD*, *Culture*, *Requirements*, *Team Engagement*, *Time* and *Others*. The top five results are presented in Table 3.

Table 3: Neg. aspects aspects of BDD (with categories).

Aspect	Occ.	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

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.

This indicated 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.

1. **BDD Processes May Be Challenging** - As BDD promotes the usage of processes throughout the software development lifecycle (Smart, 2014), 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 (Lundquist, 1999; Fischer et al., 2006), 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 (North et al., 2006), it was based in TDD (Beck, 2003). 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 (Mugridge, 2003).

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 (Smart, 2014) and increase the amount of time a development activity requires.

4. **BDD May Hinder Requirements** - According to Smart (Smart, 2014), 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.* (Oliveira et al., 2019; Oliveira and Marczak, 2018; Oliveira and Marczak, 2017) has performed a few studies on this topic with relevant contribution, 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.

5.3 (RQ3) Are There Any Best Practices When Teaching BDD?

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 subsection 4.4 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 subsection 4.3, we have created the following set of best practices.

1. **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.
2. **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.
3. **Introduce BDD after Agile Has Been Introduced** - BDD foster collaboration and cooperation. As a consequence, according to Smart (Smart, 2014), it thrives in agile contexts. This indicates that BDD should fit more naturally if students have already experience agile software development.

   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.

6 CONCLUSION

This paper presented a multi-study of teaching BDD in ALEs. By combing the results of four complementary studies, we have been able to understand the implications of teaching BDD in ALEs.

Gathering knowledge obtained from these studies and following the triangulation method proposed by Creswell (Creswell and Creswell, 2017), we have been able to cross data from these studies and solidify benefits and challenges BDD should have in ALEs. Furthermore, this same principle allowed us to establish a set of best practices to be used when teaching BDD in ALEs. These findings, helped to fill a gap of studies addressing BDD. It seem novel and should help teachers, schools, universities and active learning environments, which teaches software development, to have a set of guidelines to add BDD to their curriculum.

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

REFERENCES

Ahmad, K. and Gestwicki, P. (2013). Studio-based learning and app inventor for android in an introductory cs course for non-majors. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 287–292, New York, NY, USA. ACM.

Babbie, E. (2005). *Survey research methods*. UFMG.

Beck, K. (2003). *Test-driven development: by example*. Addison-Wesley Professional.

Buller, D. B., Berwick, M., Shane, J., Kane, I., Lantz, K., and Buller, M. K. (2013). User-centered development of a smart phone mobile application delivering personalized real-time advice on sun protection. *Translational behavioral medicine*, 3(3):326–334.

Creswell, J. W. and Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Deslauriers, L., McCarty, L. S., Miller, K., Callaghan, K., and Kestin, G. (2019). Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. *Proceedings of the National Academy of Sciences*, 116(39):19251–19257.

Dyba, T. (2000). An instrument for measuring the key factors of success in software process improvement. *Empirical software engineering*, 5(4):357–390.

Evans, E. (2004). *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

Fischer, M. A., Mazor, K. M., Baril, J., Alper, E., DeMarco, D., and Pugnaire, M. (2006). Learning from mistakes. *Journal of general internal medicine*, 21(5):419–423.

Hoffmann, L. F. S., de Vasconcelos, L. E. G., Lamas, E., da Cunha, A. M., and Dias, L. A. V. (2014). Applying acceptance test driven development to a problem based learning academic real-time system. In *2014 11th International Conference on Information Technology: New Generations*, pages 3–8. IEEE.

Keele, S. et al. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, Ver. 2.3 EBSE Technical Report. EBSE.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. *IEEE software*, 12(4):52–62.

Krasner, H. (2018). The cost of poor quality software in the us: A 2018 report. *Consortium for IT Software Quality, Tech. Rep.*

Li, P. L., Ko, A. J., and Zhu, J. (2015). What makes a great software engineer? In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, pages 700–710. IEEE Press.

Lundquist, R. (1999). Critical thinking and the art of making good mistakes. *Teaching in Higher Education*, 4(4):523–530.

Matos, V. and Grasser, R. (2010). Building Applications for the Android OS Mobile Platform: A Primer and Course Materials. *Journal of Computing Sciences in Colleges*, 26(1):23–29.

Matthies, C., Treffer, A., and Uflacker, M. (2017). Prof. ci: Employing continuous integration services and github workflows to teach test-driven development. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE.

Memon, R. N., Ahmad, R., and Salim, S. S. (2010). Problems in requirements engineering education: a survey. In *Proceedings of the 8th International Conference on Frontiers of Information Technology*, pages 1–6.

Mugridge, R. (2003). Challenges in teaching test driven development. In *International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 410–413. Springer.

Murphy, B., Bird, C., Zimmermann, T., Williams, L., Nagappan, N., and Begel, A. (2013). Have agile techniques been the silver bullet for software development at microsoft? In *2013 ACM/IEEE international symposium on empirical software engineering and measurement*, pages 75–84. IEEE.

Nascimento, N., Santos, A. R., Sales, A., and Chanin, R. (2019). An investigation of influencing factors when teaching on active learning environments. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 517–522.

Nascimento, N., Santos, A. R., Sales, A., and Chanin, R. (2020a). Behavior-driven development: A case study on its impacts on agile development teams. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 109–116.

Nascimento, N., Santos, A. R., Sales, A., and Chanin, R. (2020b). Behavior-driven development: An expert panel to evaluate benefits and challenges. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. Proceedings have not been published yet.

Nichols, M., Cator, K., and Torres, M. (2016). *Challenge Based Learning Guide*. Digital Promise, Redwood City, CA, USA.

Nilsson, S. (2010). *Augmentation in the wild: user centered development and evaluation of augmented reality applications*. PhD thesis, Linköping University Electronic Press.

North, D. et al. (2006). Introducing bdd. *Better Software*, 12.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46.

Oliveira, G. and Marczak, S. (2017). On the empirical evaluation of bdd scenarios quality: preliminary findings of an empirical study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 299–302. IEEE.

Oliveira, G. and Marczak, S. (2018). On the understanding of bdd scenarios' quality: Preliminary practitioners' opinions. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 290–296. Springer.

Oliveira, G., Marczak, S., and Moralles, C. (2019). How to evaluate bdd scenarios' quality? In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 481–490.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., and Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3):303–337.

Planning, S. (2002). The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology*.

Procaccino, J. D., Verner, J. M., Overmyer, S. P., and Darter, M. E. (2002). Case study: factors for early prediction of software development success. *Information and software technology*, 44(1):53–62.

Rosqvist, T., Koskela, M., and Harju, H. (2003). Software quality evaluation based on expert judgement. *Software Quality Journal*, 11(1):39–55.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131.

Santos, A., Sales, A., Fernandes, P., and Nichols, M. (2015). Combining Challenge-Based Learning and Scrum Framework for Mobile Application Development. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'15)*, pages 189–194, Vilnius, Lithuania.

Scharff, C., Wasilewska, A., Wong, J., Bousso, M., Ndiaye, I., and Sarr, C. (2009). A model for teaching mobile application development for social changes: Implementation and lessons learned in senegal. In *Int. Multiconf. on Computer Science and Information Technology, 2009. IMCSIT '09.*, pages 383–389, Wisla, Poland.

Serrador, P. and Pinto, J. K. (2015). Does agile work?—a quantitative analysis of agile project success. *International Journal of Project Management*, 33(5):1040–1051.

Simpson, R. and Storer, T. (2017). Experimenting with realism in software engineering team projects: An experience report. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 87–96. IEEE.

Smart, J. F. (2014). *BDD in Action*. Manning Publications.

Solis, C. and Wang, X. (2011). A study of the characteristics of behaviour driven development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 383–387. IEEE.