Behavior-Driven Development: An Expert Panel to evaluate benefits and challenges

Nicolas Nascimento
PUCRS, School of Technology
Porto Alegre, RS, Brazil
nicolas.nascimento@pucrs.br

Alan R. Santos
PUCRS, School of Technology
Porto Alegre, RS, Brazil
alan.ricardo@acad.pucrs.br

Afonso Sales
PUCRS, School of Technology
Porto Alegre, RS, Brazil
afonso.sales@pucrs.br

Rafael Chanin
PUCRS, School of Technology
Porto Alegre, RS, Brazil
rafael.chanin@pucrs.br

ABSTRACT

Teaching modern software techniques is a challenging task as these practices tend to be collaborative and require a lot of preparation and environment setup. Among these techniques is Behavior-Driven Development (BDD), a development method which proposes software to be develop focusing primarily on its expected behavior. In this context, this paper investigates the perception of active-learning experts regarding the potential benefits and challenges of teaching BDD in active learning environments to software engineering students. To achieve this goal, we have conducted an expert panel with 28 active-learning experts from four countries. Our preliminary results indicate that experts perceive both benefits, like improvement in specification, and challenges to embrace the BDD "culture" when teaching BDD to software engineering students. Based in these findings, we found indicatives that BDD should benefit software engineering students in active learning environments, however it may require more setup, teacher preparation and engagement during the learning process.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Applied computing** → **Interactive learning environments**; Collaborative learning.

KEYWORDS

Software Engineering, Behavior-Driven Development, Agile Development, Challenge Based Learning

ACM Reference Format:
Nicolas Nascimento, Alan R. Santos, Afonso Sales, and Rafael Chanin. 2020. Behavior-Driven Development: An Expert Panel to evaluate benefits and challenges. In *34th Brazilian Symposium on Software Engineering (SBES '20), October 21–23, 2020, Natal, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3422392.3422460

SBES '20, October 21–23, 2020, Natal, Brazil

ACM ISBN 978-1-4503-8753-8/20/09.
https://doi.org/10.1145/3422392.3422460

1 INTRODUCTION

Software is ubiquitous today [9] and with the increase usage of electronic devices, such as smartphones [25], this presence does not show sign of slowing down. Specifically regarding mobile software, the trend is very pronounced. Alone, the App Store[1], a popular marketplace for mobile applications, has experienced an enormous growth, with a cumulative number of app downloads being around 180 billions from 2008 to 2017 [24].

As developing software continues to become more relevant, the education scenario is presented with the challenge to adapt quickly and to better prepare students. In this context, active learning methodologies, such as Challenge-Based Learning (CBL), are drawing attention due to reports of improved learning [4] and student engagement [1, 11, 19, 20]. The academia continues to produce relevant research related to teaching modern software development, including Mobile Application Development [10, 19].

Among modern software development techniques, is Behavior-Driven Development (BDD), a development method which proposes software to be develop focusing primarily on its expected behavior [14]. BDD improves development by both ensuring software developed is reliable and is aligned with the needs of the customer.

This work aims at assessing the perception of potential 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. Our results indicate that BDD can bring both positive impacts, specially in terms of requirements and collaboration, and negative impacts, specially regarding BDD practices and testing-culture.

The remainder of this paper is organized as follows. Section 2 briefly contextualizes important concepts. In Section 3, we present the scientific approach used for evaluation and analysis of the collected data. Following this, Section 4 depicts the results and Section 5 presents a brief discussion and highlights some important findings. Section 6 describes some threats to the study. Finally, Section 7 concludes the paper with some final thoughts and future work.

[1] https://www.apple.com/ios/app-store/

2 BACKGROUND

From its definition [14, 22], BDD is a set of practices that aim at developing software which is correct, *i.e.*, follows software engineering best practices, and right, *i.e.*, follows the client's expectations. Some practices of BDD include the augmentation of user stories, through BDD scenarios, and development by using acceptance criteria [17] and test-driven development (TDD)[2].

As a development methodology, BDD emphasizes test cases which are written in a common language, a concept derived from Domain Driven Design [6]. The specification of these test cases is done using scenarios (also known as BDD Scenarios), which should describe features of a system through examples.

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. Formatting these scenarios is usually performed using Gherkin, a structured language[3]. When specifying BDD scenarios, there are 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 additional context. This is expressed in the template by the word *"AND"*.

In terms of key characteristics associated with BDD, Solis *et al.* [23] 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.
(5) **Readable Behavior Oriented Specific Code** - As the specification of behavior is following a template, which itself is plain text, it can also be used as the system documentation and thus aid the development team to keep it updated as requirements changed.

In this context, our goal is assessing the perception of potential impacts in active learning environments. The next section details the protocol followed to achieve this goal.

3 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 [18] and aid in the improvement of hypothesis generated. In addition, experts are usually better at predicting and even preventing [8] errors when performing certain tasks. Furthermore, the opinion of experts is recognized as a valuable research artifact in Software Engineering (SE) research community [5].

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 [12]. 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:

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

3.1 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 in-person levelling activity with all participants in the study prior to asking any questions.

For this levelling activity, we have randomly split participants in two groups, due to the workshop's room capacity, and performed a one-and-half hour long 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 was structured as follows. A researcher introduced 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 (no specific type) and write lightweight specifications (*i.e.* user stories for it. These *user stories*, then, were augmented by BDD scenarios. Participants were told to write these specifications in an easy-to-exchange format so they could exchange them with other groups.

Following this structure, groups had to switch specifications and create a low-fidelity prototype using paper, pen and any prototyping app they were comfortable with. We have suggested using Proot[2]. 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 the research questionnaire and

[2]http://prottapp.com/

were told to always consider their personal opinions. Excluding demographic questions, the questions used are presented in Table 1.

Table 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 1.

Figure 1: The overall research process

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

3.2 Data collection & Analysis

The questionnaire provided to participants was available online and participants could answer either 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 [12]. 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 RESULTS

4.1 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, 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 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.

4.3 BDD - Perception of benefits

Our second result is related to positive perceptions. Results from this sub-section are divided in 6 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 below.

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

Elicitation. These results focused on 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 appear to have a potential to be impacted, 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. Participant #19 states that BDD brings up *"ideas we didn't think about"* and participant #6 states that *"[...]it makes the process more ludic and interactive (proximity with the client/user)"*.

Specification. In terms of specification, many reports from participants present positive aspects that BDD can bring to the specification of requirements. These reports range from general to specific,

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 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 reinforced by participant #13, who says *"it (BDD) helps to understand and describe in a better way the features of our apps"*.

Furthermore, 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 *"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 users.

4.3.2 User Comprehension. Five responses from experts present associated BDD with *User Comprehension*. It appears that BDD could potentially increase the team's capacity to understand the need of the user and thus properly 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 details 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 [...]"*.

4.3.3 Project. Three responses are 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 having a shared "vision" 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 could indicate that BDD could be a helpful tool for student project management.

4.3.4 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 original vision that BDD can help the entire development lifecycle [14, 22].

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.

4.3.5 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.

These 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 middleware 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 is not related to in-person communication, it is probably linked to the artefacts of BDD helping to better communicate the requirements.

4.3.6 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 indicates that BDD can potentially help not only to better communicate, but also to better collaborate in projects.

4.4 BDD - Perception of challenges

Our third result is related to negative perceptions. Results from this sub-section are divided in 6 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 below.

4.4.1 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"*.

4.4.2 Culture. Four responses were centred around the *Culture* of BDD, which itself is directly related to testing. Participant #2 says that *"beating the barrier of creation the culture, it becomes easy"*. This indicates BDD can be challenging if the culture of the educational context (or organisation) does not embrace testing. 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"*.

4.4.3 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 if students in a learning environment have difficulties or are not used to agile practices, BDD can be challenging to be implement.

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 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"*.

4.4.4 Team Engagement. Responses from three participants are centred 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)."*

4.4.5 Time. Also with three reports, time is a key factor for some participants. Participant #12 seems worried, saying that a challenge is *"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."*

4.4.6 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"*.

5 DISCUSSION

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

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 [14] and are documented by Smart [22]. They propose that BDD can have an impact in many portions of the software development lifecycle, among of which are 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.* [21], 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 requires engagement from the client with the product under development [22], this should naturally provide more space for discussion and thus increase the overall comprehension of the needs 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 organised. Implementation, however, seems to suffer a greater positive impact. Shortening the development process and helping to make abstract concepts more concrete are among the reported benefits.

This shortening of the development process, specifically, is an unexpected report. As BDD requires more collaboration and engagement from those involved in the software development process [22]. The second result, regarding an easiness when translating abstract into concrete, is most likely linked to the use of BDD scenarios, as Smart [22] reports that scenarios are useful for communication (for example, reducing the gap between the business analyst to the developer terms). 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.

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

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 the need for a greater deal of commitment from business and the development team required by BDD [22].

Another interesting challenge proposed are BDD scenarios. Previous researches have proposed a set of guidelines to evaluate scenarios [15, 16] 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 processes which use tests as driver for implementation present numerous challenges to learners [13]. 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 [7], 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 assignments or activities with tight schedules.

6 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 the participants viewpoint, thus being a subjective opinion.

Another key factor 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.

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

7 CONCLUSION

In this study, our goals were to assess the perception of impact 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.

Our indicatives provide paths to future research. As future work we will compare these expert panel results with real results obtained from a study in an environment that teaches BDD.

REFERENCES

[1] K. Ahmad and P. Gestwicki. 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* (Denver, Colorado, USA) *(SIGCSE '13)*. ACM, New York, NY, USA, 287–292.

[2] Kent Beck. 2003. *Test-driven development: by example*. Addison-Wesley Professional.

[3] David Chelimsky, Dave Astels, Bryan Helmkamp, Dan North, Zach Dennis, and Aslak Hellesoy. 2010. The RSpec Book: Behaviour Driven Development with Rspec. *Cucumber, and Friends, Pragmatic Bookshelf* (2010).

[4] Louis Deslauriers, Logan S McCarty, Kelly Miller, Kristina Callaghan, and Greg Kestin. 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 (2019), 19251–19257.

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

[6] Eric Evans. 2004. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

[7] Luis Fernandez-Sanz and Sanjay Misra. 2011. Influence of Human Factors in Software Quality and Productivity. 257–269. https://doi.org/10.1007/978-3-642-21934-4_22

[8] Soren Lauesen and Otto Vinter. 2001. Preventing requirement defects: An experiment in process improvement. *Requirements Engineering* 6, 1 (2001), 37–50.

[9] Paul Luo Li, Andrew J Ko, and Jiamin Zhu. 2015. What makes a great software engineer?. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 700–710.

[10] Qusay H Mahmoud and Pawel Popowicz. 2010. A mobile application development approach to teaching introductory programming. In *2010 IEEE Frontiers in Education Conference (FIE)*. IEEE, T4F–1.

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

[12] Joseph A Maxwell. 2008. Designing a qualitative study. *The SAGE handbook of applied social research methods* 2 (2008), 214–253.

[13] Rick Mugridge. 2003. Challenges in teaching test driven development. In *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer, 410–413.

[14] Dan North et al. 2006. Introducing bdd. *Better Software* 12 (2006).

[15] Gabriel Oliveira and Sabrina Marczak. 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)*. IEEE, 299–302.

[16] Gabriel Oliveira, Sabrina Marczak, and Cassiano Moralles. 2019. How to Evaluate BDD Scenarios' Quality?. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. 481–490.

[17] Ken Pugh. 2010. *Lean-agile acceptance test-driven development: better software through collaboration*. Pearson Education.

[18] Tony Rosqvist, Mika Koskela, and Hannu Harju. 2003. Software quality evaluation based on expert judgement. *Software Quality Journal* 11, 1 (2003), 39–55.

[19] A.R. Santos, A. Sales, P. Fernandes, and M. Nichols. 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)*. Vilnius, Lithuania, 189–194.

[20] C. Scharff, A. Wasilewska, J. Wong, M. Bousso, I. Ndiaye, and C. Sarr. 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.* Wisla, Poland, 383–389.

[21] Robbie Simpson and Tim Storer. 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)*. IEEE, 87–96.

[22] John Ferguson Smart. 2014. *BDD in Action*. Manning Publications.

[23] Carlos Solis and Xiaofeng Wang. 2011. A study of the characteristics of behaviour driven development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 383–387.

[24] Statista. 2017. Cumulative number of apps downloaded from the Apple App Store from July 2008 to June 2017 (in billions). https://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store/. [Online; accessed 29-June-2019].

[25] Statista. 2019. Smartphone Are the Bulk of Our Digital Media Diet. https://www.statista.com/chart/18347/hours-spent-on-digital-media/. [Online; accessed 29-June-2019].