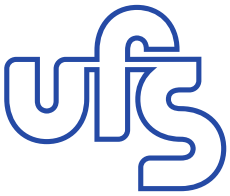

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Guidelines for the adoption of Behavior-Driven Development (BDD): An approach with Design Science Research

Dissertação

Shexmo Richarlison Ribeiro dos Santos

São Cristóvão – Sergipe

2025

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Shexmo Richarlison Ribeiro dos Santos

Guidelines for the adoption of Behavior-Driven Development (BDD): An approach with Design Science Research

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Fabio Gomes Rocha
Coorientador(a): Guillermo Rodriguez

São Cristóvão – Sergipe

2025

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Santos, Shexmo Richarlison Ribeiro dos
S237g Guidelines for the adoption of behavior-driven development (BDD): an approach with design science research / Shexmo Richarlison Ribeiro dos Santos ; orientador Fabio Gomes Rocha. - São Cristóvão, 2025.
98 f.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2025.

1. Ciência da computação. 2. Framework (Arquivo de computador). I. Rocha, Fabio Gomes orient. II. Título.

CDU 004

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: SHEXMO RICHARLISON RIBEIRO DOS SANTOS

Em 16 dias do mês de janeiro do ano de dois mil e vinte cinco, com início às 16hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Shexmo Richarlison Ribeiro dos Santos**, que desenvolveu o trabalho intitulado: **"Guidelines for the adoption of Behavior-Driven Development (BDD): An approach with Design Science Research",** sob a orientação do Prof. Dr. **Fábio Gomes Rocha.** A Sessão foi presidida pelo Prof. Dr. **Fábio Gomes Rocha** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Afonso Henrique Correa de Sales** (PUCRS) e, em seguida, o Prof. Dr. **Michel dos Santos Soares** (Procc/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) Aprovado. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 16 de janeiro de 2025.

Prof. Dr. Fábio Gomes Rocha
(PROCC/UFS)
Presidente

Prof. Dr Afonso Henrique Correa de Sales
(PUCRS)
Examinador Externo

Prof. Dr. Michel dos Santos Soares
(PROCC/UFS)
Examinador Interno

Shexmo Richarlison Ribeiro dos Santos
Candidato

Acredito que a maior merecedora desta dedicatória seja minha mãe, que conseguiu me educar e me fazer uma pessoa de respeito em meio as dificuldades da vida. Se estou aqui hoje escrevendo esta dedicatória é graças a senhora, que abriu mão de tanto para cuidar de mim.
Te amo, mãe.

Acknowledgements

Meus agradecimentos vão para todos que estiveram comigo durante a minha jornada no mestrado.

Deus, que me sustentou durante todo o mestrado e me ajudou a superar cada obstáculo.

Minha mãe Lourdes, que me educou e sempre me incentivou a voar mais alto.

Minhas irmãs Valda e Luiza e minha prima Paula, que se fizeram presente nos meus dias, ajudando a deixá-los mais leves.

Meus amigos Marcos Venícius e Marcos César, por toda ajuda e apoio durante o mestrado, todos os desabafos ouvidos e conhecimentos compartilhados.

Meu namorado Marckson, por todo o apoio e compreensão nos momentos turbulentos.

Meu orientador prof. Fabio, que me ouviu, ajudou e orientou muito mais do que deveria, mostrando o caminho que precisava seguir para ter sucesso.

Minha profa. Rosimeri, que mesmo não tendo mais ligação direta com minha formação, me ouvia e orientava sempre que precisava de ajuda.

A todas as pessoas não citadas, amigos, colegas de curso, e todas as outras que fizeram parte da minha trajetória no mestrado, direta ou indiretamente, muito obrigado.

Gratidão pela experiência vivida (mesmo com os novos fios de cabelo branco).

No, try not. Do or do not. There is no try.
(Master Yoda)

Abstract

Context: In Requirements Engineering, it is necessary to look for ways to adopt methods and tools that contribute to the quality of functionalities delivered. **Problem**: Lack of clarity regarding the aspects necessary for adopting BDD. **Solution**: Present guidelines for implementing BDD to guide developers in adopting this framework. **Method**: Identify and present via a mind map such guidelines through a Design Science Research approach. **Results**: State-of-the-art and state-of-the-practice regarding BDD were identified, a case study was carried out to focus on eliciting non-functional requirements, and an experiment using BDD with LLMs. **Conclusions**: It was possible to validate the guidelines identified through the Design Science approach through a new case study, ensuring the effectiveness of the guidelines presented. **Main contributions**: To the scientific community, this work advances the study regarding BDD, presenting guidelines for teams that do not use it in their work activities; to the industry, this work presents a roadmap for the adoption of BDD, assisting professionals who have not yet had contact with the framework.

Keywords: Behavior-Driven Development (BDD). BDD adoption. Design Science Research. Guidelines.

List of Figures

List of Tables

Contents

1 Introduction

The software architect is responsible for building the software, considering internal and external factors such as the architectural standards and clients, respectively (KRUCHTEN, 2008). For an architect to successfully develop software, one must know the aspects inherent to its creation, such as the existing frameworks, to achieve specific purposes and maintain quality software.

A framework is a structure that is a basis for systematically building applications. There is also a need for frameworks used in software architecture to meet such expectations external to agile methods, for example, framework Behavior-Driven Development (BDD), used in the life cycle of a system (BRUSCHI et al., 2019).

To ensure software development achieves quality, functional and non-functional (or quality) requirements must be agreed upon based on the software's expected behavior. BDD, created in 2003 by Dan North (NORTH, 2006), came to mitigate flaws found in Test-Driven Development (TDD), a traditional method. The way TDD is proposed causes failures when the software is delivered due to a lack of communication among those involved in the process. Based on the requirements elicited, each development team member performs their task as they understand the request. The probability of the software being delivered with misinterpreted requirements is greater.

On the other hand, BDD focuses on behavior following what the software is expected to accomplish, so there is better interaction among those involved regarding the elicitation of requirements to contribute with greater assertiveness in the delivery of releases. BDD framework defines software behavior using the **given-when-then** pattern, which can be expressed in natural language, domain-specific, and then executed through automated tests. In this way, there is better communication between those involved in the process to improve the delivery of the final product concerning the validation of the elicited requirements.

According to Brooks (BROOKS; BULLET, 1987), no single software meets all require-

ments; software is built based on customer needs, needing to maintain the quality inherent in its execution. Furthermore, according to Boehm (BOEHM, 2006), over time, software needs to meet new demands, so there is a need to try to achieve what the customer expects regarding their final product.

To seek improvements regarding the development and evaluation of software (SHAW, 2002; SHAW, 2003), the objective of this research is: "**Analyze** the BDD framework, **with the purpose of** presenting guidelines, **with respect to** BDD adoption, **from the point of view of** researchers and professionals, **in the context of** software development". To achieve the proposed objective, the following research questions are defined:

ID	Question	Justification
Q1	What is the current state of the art of BDD?	Present what has been searched regarding BDD through a multivocal literature review.
Q2	What is the current state of practice of BDD?	Present the perception of BDD through a survey.
Q3	Is BDD effective in eliciting non-functional requirements regarding ISO/IEC/IEEE 25010?	Present BDD effectiveness to elicit non-functional requirements.
Q4	Is BDD effective for creating prompts with LLMs?	Analyze the effectiveness of BDD for creating standard prompts for automation tests.
Q5	Are the discovered guidelines replicable in teams that do not use BDD?	Validate results found through a case study.

Table 1 – Research questions

Based on the results found in the four research studies among Q1 to Q4, it was possible to carry out a case study with a team that did not use BDD to validate the guidelines identified by the studies regarding adopting BDD, answering Q5.

This study has the following Chapters: **Theoretical Concepts**, where the main concepts regarding BDD are addressed; **Design Science Research**, where the parts that make up this study are described to outline how it achieved the objective of this research; **Results**, where it presents the information concluded in the studies; and, finally, **Conclusion**, where it concludes with this study's main contribution to software development.

2

Theoretical Concepts

This Chapter addresses the main concepts regarding BDD.

2.1 Behavior-Driven Development (BDD)

BDD emerged in 2003, conceived by Dan North (NORTH, 2006) as a framework of notable flexibility. North outlined the need for a framework focused on solving communication and team integration issues. Test-Driven Development (TDD) does not address these gaps while focusing on test automation, a factor that contributes significantly to product quality. Thus, while TDD is primarily dedicated to technical aspects, BDD seeks to improve communication and collaboration between the technical team and stakeholders. In this context, BDD is a framework mainly oriented towards observing system behavior.

BDD introduced gherkin to promote collaboration and communication, a remarkable way of expressing specifications culminating in user stories and acceptance scenarios in natural language. This approach enables integration with testing tools, allowing the adoption of BDD to monitor system requirements and evaluate its behavior, ensuring that it is validated as ready at the end of each iteration. BDD has demonstrated significant benefits, such as more effective communication, reduced release delivery time, and greater accuracy in requirements elicitation (PEREIRA et al., 2018).

BDD uses the “3 amigos” technique, where a meeting is held between the product owner, tester, and developer to objectively and concisely specify the expected behavior by the system when eliciting the requirements, in this case, user stories and their acceptance scenarios, to outline the behavior expected by the software. As such stories need to be written directly and transparently to achieve the proposed objective, BDD needs to make its scenarios testable (SILVA; FITZGERALD, 2021). In this way, it is clear how BDD can collaborate to elicit non-functional requirements.

From this point on, the features begin to be written, implemented, and validated, seeking to maintain the behavior expected by the requirements elicited in each release to deliver what stakeholders expect. In addition to helping with communication and collaboration between the parts involved (COUTO et al., 2022), as it is written in English, the BDD also helps with the dynamic documentation of the system (PEREIRA et al., 2018; SILVA; FITZGERALD, 2021; NASCIMENTO et al., 2020b). Dynamic documentation, or living documentation, refers to the agility required for updates to process documentation. With these aspects, it is expected that the use of BDD will save time and money, as it reduces the need for rework due to poor quality releases, has better quality validation (BINAMUNGU; EMBURY; KONSTANTINOU, 2018; MOE, 2019), in addition to a better understanding of code (GUERRA-GARCIA et al., 2023). Figure 1 shows the BDD process.

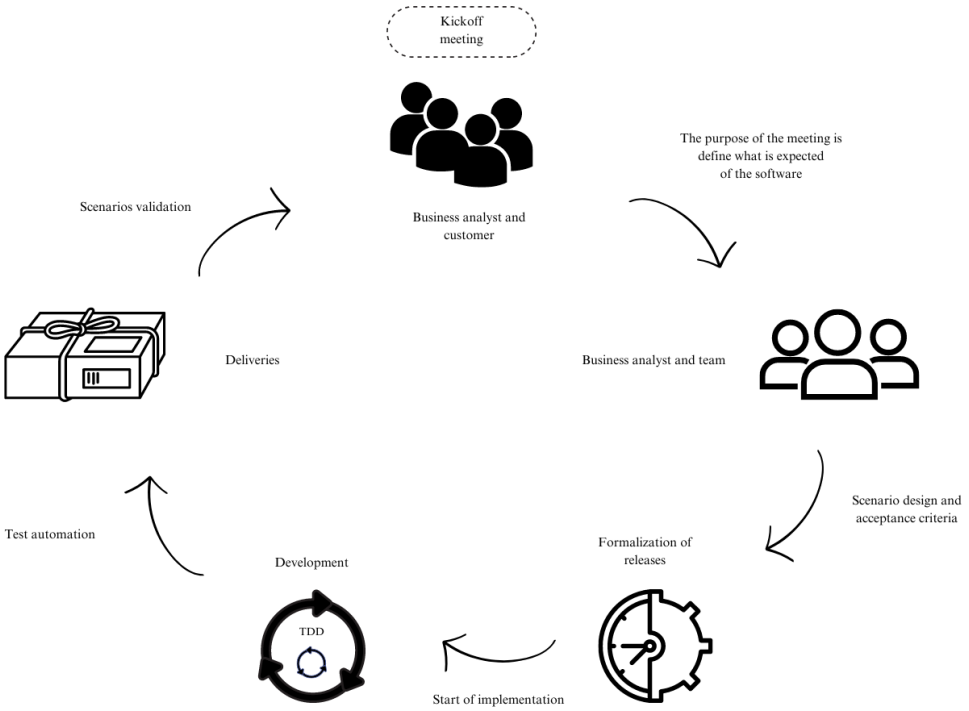

Figure 1 – BDD process [adapted (SMART; MOLAK, 2023)]

BDD is used throughout the software life cycle, so it is useful from requirements elicitation through validation and documentation to software maintenance. From requirements elicitation, the developer and tester perform the automation and validation of the functionalities requested by the product owner to deliver small parts of software throughout the process to put it into operation in the shortest possible time. Verification and validation of requirements are carried out to identify whether what was requested was carried out, as well as implementation, respectively (BRUSCHI et al., 2019).

3

Design Science Research

Design Science is a methodology that has been used over the years in the area of Information Systems (EA et al., 2020). According to Hevner et al. (HEVNER et al., 2010), research in Information Systems is composed of two strands: Design Science and Behavioral Science. Furthermore, the authors reported that both strands are complementary; thus, they correlate, as shown in Figure 2.

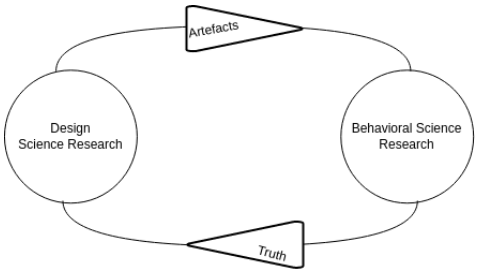

Figure 2 – Relationship between Design Science and Behavioral Science

Design Science Research seeks to create innovative artefacts through methods inherent to this type of research to improve the artefacts' effectiveness and usefulness in the real context (EA et al., 2020). While Design Science seeks to deliver artefacts, Behavioral Science, through the delivered artefacts, seeks to present the truth about adoption of artefacts. Thus, the correlation between the two approaches is perceived as being at different stages of the same cycle. Figure 3 presents the methodology approach inherent to this study.

Through the knowledge bases presented in Figure 3, it is possible to identify the execution of different studies inherent to the Design Science presented here. Each of the knowledge bases was studied to achieve this dissertation's general objective. The methodology used to execute each of the respective studies is presented in their respective Sections. Through the knowledge bases, it is possible to analyze the effectiveness of the artifacts discovered through a real study to validate the results obtained.

Figure 3 – Design Science Research

Through a multivocal literature review (SANTOS; RODRIGUEZ; ROCHA, 2024), it was identified how BDD has been researched and used to explain the perspective of researchers and professionals. With the results found, a survey was conducted that presented the point of view of professionals who use BDD in their work routines to explain their perception regarding this framework. Furthermore, a case study identified how BDD elicited non-functional (quality) requirements through the performance efficiency characteristic expressed in the ISO/IEC/IEEE 25010 Standard (SANTOS et al., 2024). Finally, by an experiment using BDD and LLMs to generate standard prompts, it was possible to verify the effectiveness of using this framework with artificial intelligence.

Through these studies, it was possible to outline guidelines for adopting BDD, which will finally be validated through a case study with a team that does not use this framework.

The following Sections present the methodologies used in this research, namely: Multivocal Literature Review 3.1, Survey 3.2, Case Study - ISO/IEC/IEEE 25010 3.3, Experiment 3.4, and Case study - Guidelines 3.5.

3.1 Multivocal Literature Review (MLR)

Multivocal Literature Review (MLR) is a type of Systematic Literature Review that seeks to cross-reference information found in white and gray literature (GAROUSI; MÄNTYLÄ, 2016). White literature comprehends articles published on traditional scientific bases, while gray literature is the open Internet, such as websites and blogs. This study used the guidelines presented by Garousi, Felderer, and Mantyla (GAROUSI; FELDERER; MÄNTYLÄ, 2019) regarding the adoptionn of 2nd tier for gray literature. According to these authors, there are three tiers of gray literature, as shown in Table 2.

According to Rodriguez (RODRÍGUEZ; GONZÁLEZ-CAINO; RESETT, 2021), one of the benefits of an MLR is improving the understanding of the topic addressed in terms of

Tiers	Examples
1st (High credibility)	Books, magazines.
2nd (Moderate credibility)	Annual reports, sites.
3rd (Low credibility)	Blogs, emails, tweets.

Table 2 – Tiers of Gray Literature

researchers and practitioners through the gain obtained by gray literature. Furthermore, it is essential to emphasize that information from gray literature is discarded when only it uses white literature (GAROUSI; FELDERER; MÄNTYLÄ, 2016). This MLR[1] was carried out aiming to present the relevance of adopting BDD for software development.

To carry out the MLR, Parsifal tool[2] was used to structure the entire research, from the definition of the title to the analysis of the results collected. Thereby, this study was carried out from the Goal-Question-Metric approach according to Basili (BASILI, 1992): "**To analyze** studies on BDD, **with the purpose to** characterize, **respect to** its adoption and application, **from the point of view of** researchers and professionals, **in the context of** theoretical and applied researches from 2013 to 2023".

This study used the time between 2013 and 2023. The book "Cucumber Recipes: Automate Anything with BDD Tools and Techniques" (DEES; WYNNE; HELLESOY, 2013) was used as a starting point since Cucumber is one of the most important frameworks for BDD. Also, it is worth mentioning Dan North's article "JBehave. A framework for Behavior-Driven Development (BDD)" (NORTH, 2012) BDD's creator. Through these frameworks, it was possible to advance with BDD. In addition, in 2014, the company Sucumbe was created. This company may also have impacted the development of BDD.

Based on such information as Pai et al. (PAI et al., 2004), PICOC criteria were used to define the terms: **Population**: BDD; **Intervention**: Requirements, Test; **Outcome**: Tools; **Context**: Software development. It discarded the **C**omparison criteria because no intent related to this term existed. According to the presented keywords and their synonyms, it was created the following string to use in the selected databases:

("**BDD**" OR "**Behavior-Driven Development**") AND ("**requirements**" OR "**documentation**" OR "**quality**" OR "**test**") AND ("**tools**" OR "**methods**" OR "**models**" OR "**techniques**" OR "**software development**")

The search string was used in 8 databases, 5 of which were white literature and 3 of gray literature, with the period between 2013 and 2023 as a cut.

[1] Link for the study: https://publicaciones.sadio.org.ar/index.php/JAIIO/article/view/998.
[2] https://parsif.al/

3.1.1 Studies identification

The chosen white literature databases were: ACM Digital Library[3], IEEEXplore Digital Library[4], Web of Science[5] and Scopus[6]. These databases were chosen because they are the most relevant bases for the computing field, it selected them as the path to follow in this investigation. In addition, the science@direct[7] base was initially also selected. However, when the string was entered, the base did not return any articles, justifying that the string was too large, so it was deleted.

As the basis for the gray literature regarding the 2nd tier as mentioned in Table 2: Google Scholar[8] (top five tabs) because it is a website with an academic purpose, InfoQ[9] (top five tabs) which is a portal with a variety of content related to SE, and Dan North[10] who is the BDD's creator. The top five tabs of Google Scholar and InfoQ were defined to investigate the main results with the developed string. Also, Dan North's website was selected because BDD is the main focus of this investigation, which is studied in the tab "**tag**" subject "**bdd**" found on the website.

As previously mentioned, the MLR crosses the data found in the white and gray literature to observe what has been researched by the academy and published by relevant authors regarding BDD on a non-scientific basis. Table 3 identifies the number of articles returned from each base when performed with the search string.

Basis	Quantity
ACM Digital Library	800
IEEEXplore Digital Library	100
Web of Science	293
Scopus	603
Google Scholar	50
InfoQ	50
Dan North	4
Total	1900

Table 3 – Quantity of articles

The following Subsection continues with the definition of the research questions.

3.1.2 Definition of research questions

As a guide for this study, the research questions were used:

[3] https://dl.acm.org/
[4] https://ieeexplore.ieee.org
[5] https://www.isiknowledge.com
[6] https://www.scopus.com
[7] https://www.sciencedirect.com/
[8] https://scholar.google.com/
[9] https://www.infoq.com/
[10] https://dannorth.net/

- RQ1 - What type of results?
- RQ2 - What type of validation?
- RQ3 - What type of methodology?
- RQ4 - In which application domains is BDD most used?
- RQ5 - In which context is BDD most used?
- RQ6 - Are benefits of adopting BDD presented? Which ones?
- RQ7 - Are harms of adopting BDD presented? Which ones?
- RQ8 - For the adoption of BDD, which are tools most used?

These research questions will help us achieve the goal proposed for this study. Table 4 presents the inclusion and exclusion criteria used to screen the articles.

Criteria	**Definition**
Inclusion	The article presents the factors related to BDD
Exclusion	Duplicate articles Articles published only as journal abstracts or forewords Articles that are not in English Articles that do not partially answer at least one of the research questions Articles that do not deal with BDD Behavior-Driven Development

Table 4 – Criteria

Using the criteria presented in Table 4, it was possible to screen the articles as per the following Subsection.

3.1.3 Selection of articles

Three researchers participated in the selection of articles. One researcher initially carried out the screening, while the other two researchers carried out the validation. During the validation step, there was no issues among researchers.

Initially, the Parsifal tool was used to search for duplicate articles automatically. Soon after, reading the titles and abstracts to verify if the paper responded, even partially, to at least one of the research questions. Finally, read the remaining articles thoroughly. Figure 4 presents the screening steps' numbers.

Figure 4 presents 157 selected articles, which 18.47% from gray literature and 81.53% from white literature. Articles were extracted and the data collected were grouped in the graphics presented in the following Subsection to answer the research questions.

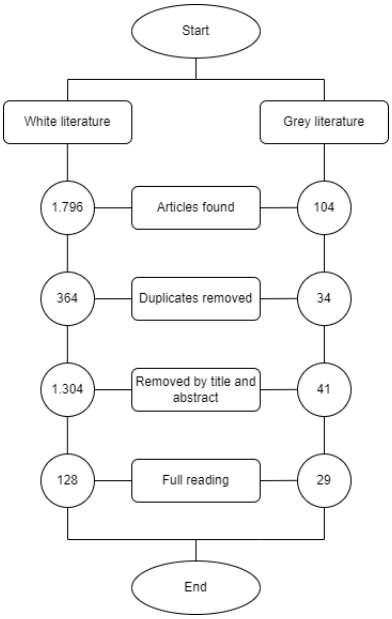

Figure 4 – Selection of articles

3.2 Survey

According to Boehm (BOEHM, 2006), software changes over time to meet new demands that arise intrinsic to emerging needs. In this research, a questionnaire was used as a form of data collection, with some of its benefits being the relatively low cost and ease of collection, as everything is done virtually (PUNTER et al., 2003).

A study focused on BDD was conducted, where a survey was applied with the aim of better understanding its practice from the point of view of professionals in the job market. According to Shaw (SHAW, 2003), procedures or techniques are new or better ways to perform tasks in Software Engineering. Therefore, identifying aspects related to BDD becomes essential to understanding the framework, which, based on such results, can help professionals and researchers find a better way of using and understanding it for new research, respectively.

3.2.1 Research goal

The guiding objective of this study was designed following part of the *Goal-Question-Metric* (MASHIKO; BASILI, 1997) model, namely: "**Analyze** aspects of BDD, **with the purpose of** characterizing, **about** the use of this framework, **from the point of view** of professionals, **in the context of the** software life cycle". A survey consisting of 12 closed and mandatory questions was created in Table 5 to achieve the proposed goal.

When creating the questions for this research, the population was defined as professionals who use BDD in their work activities (KITCHENHAM; PFLEEGER, 2002). The survey presented in this study aimed to identify aspects of BDD in practice to understand it through the collected information better (PFLEEGER; KITCHENHAM, 2001).

ID	Question	Justification
Q1	How long have you used the Behavior-Driven Development (BDD) framework?	Understand the professional's experience.
Q2	What is the main tool you use for BDD?	Identify the most used tools.
Q3	Where do you use BDD most?	Explain in which part of the process BDD has been most applied.
Q4	What are the main benefits related to adoption of BDD?	Elicit the main positive characteristics of BDD.
Q5	What are the main difficulties encountered for adoption of BDD?	Identify whether there are negative points inherent to adopting BDD.
Q6	On a scale of 0 to 10, how much is the BDD language understandable (Gherkin)?	Identify the effectiveness of the Gherkin language.
Q7	On a scale of 0 to 10, considering the purpose of the BDD regarding readability, according to its creator, Dan North, how much does BDD achieve in this regard?	Check readability related to the use of BDD.
Q8	On a scale of 0 to 10, considering the purpose of the BDD regarding communication, according to its creator, Dan North, how much does BDD achieve in this regard?	Check communicability related to the use of BDD.
Q9	On a scale of 0 to 10, how much does BDD perform faster delivery?	Realize the efficiency in using BDD.
Q10	On a scale of 0 to 10, how much does BDD perform to deliver with higher quality?	Understand the effectiveness of using BDD.
Q11	On a scale of 0 to 10, how practical is BDD to reperform updates in the code (living documentation)?	Explain the practicality of BDD in relation to living documentation.
Q12	On a scale of 0 to 10, how much does BDD contribute to the relationship to the company's economy when having living documentation?	Check whether living documentation can generate financial savings for the company.

Table 5 – Survey questions

3.2.2 Pilot and execution

A pilot was carried out with three respondents, and minor adjustments were made regarding the formatting and clarity of some questions, which were incorporated into the published version of the survey.

Initially, the research was released, in its English and Portuguese versions, only to companies through *Whatsapp* groups and the *LinkedIn* platform, and, later, the *snowball* method was used to get more respondents. The *snowball* method asks respondents to send the questionnaire to acquaintances who may be part of the research's target audience.

3.3 Case Study - ISO/IEC/IEEE 25010 Standard

Case studies have been increasingly used in Software Engineering (MELEGATI; WANG, 2020) because they use empirical data collection that seeks to understand a specific population. Thus, Software Engineering is based on observing actions aimed at software and their respective responses (WOHLIN; HÖST; HENNINGSSON, 2003), as, in this way, it is possible to use the observed aspects to validate how the system should behave. One of the research purposes in Software Engineering focuses on how software engineers and other professionals in the field carry out development, operation, and maintenance in different situations (RUNESON; HÖST, 2009). Therefore, according to Perry, Sim, and Easterbrook (PERRY; SIM; EASTERBROOK, 2006), a case study is a scientific method carried out in a planned manner, from the elaboration of questions to the presentation of results.

This case study is exploratory and descriptive (RUNESON; HÖST, 2009; YIN, 2012), given the inherent need to elicit non-functional requirements through agile frameworks to integrate quality validation. Some guidelines pointed out by Melegati and Wang (MELEGATI; WANG, 2020) and Peterson (PETERSEN, 2020) were followed, which involve a clear definition of research questions and goals, appropriate selection of relevant cases for research, collection, and analysis of data from various sources such as interviews, observations, and documentation. There is also a need to ensure the validity and reliability of the data through triangulation and member checking.

The presented study analyzed the case of a team that adopted BDD for non-functional requirements and test automation based on the performance efficiency characteristic of the ISO/IEC/IEEE 25010 Standard. Experienced teams are understood to have more than one year of experience in adopting BDD; however, they still need to apply it to non-functional requirements. The case study focused on analyzing requirements elicitation in one of the products developed by a Brazilian software manufacturer. The tested product is for customer management and product pricing. This Section describes the steps followed during the research, highlighting the team members' participation.

Below, for better understanding, a BPMN diagram of the steps of this research is presented in Figure 5.

The protocol used to direct this case study was formalized using part of the *Goal-Question-Metric* (WOHLIN et al., 2012) model defined as follows: "**To analyze** the application of BDD, **with the purpose of** automating the elicitation and testing, **in relation to** non-functional requirements based on the characteristics of the ISO/IEC/IEEE 25010 Standard, **from the point of view** of those involved in the software product, **in the context of** the performance characteristic". Table 6 presents the research questions proposed to achieve this purpose.

In order to maintain software quality, there is a need for the structures used in its development to be designed in such a way as to contribute to the delivery of the final product,

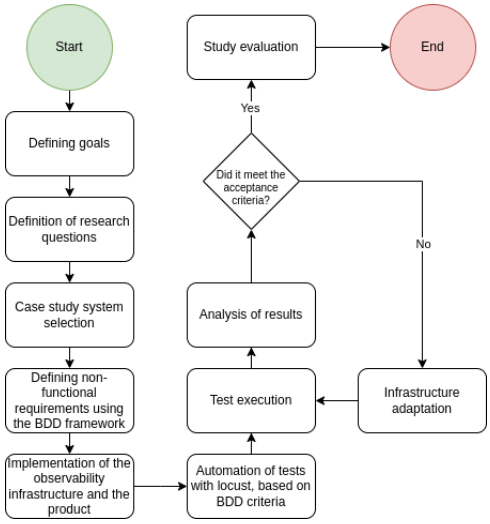

Figure 5 – Step by Step of the research

ID	Question	Justification
RQ1	How does BDD address difficulties in ensuring non-functional requirements?	This question aimed to understand how BDD can mitigate situations arising from the elicitation of non-functional requirements.
RQ2	How can BDD be used to ensure quality related to the performance efficiency characteristic of the ISO/IEC/IEEE 25010 Standard?	This question aimed to explain how this framework could contribute to eliciting non-functional requirements.
RQ3	What are the benefits of using BDD in identifying and automating non-functional tests?	This question aimed to bring positive points to adopting BDD in the testing cycle.

Table 6 – Research questions

aiming to achieve what the user expects from the product and reduce rework arising from flaws in these structures (KARAGÖZ; SÖZER, 2017). As described in the case study as follows, it validates the proposed scenarios in a company that uses the BDD framework for test cycles. Thus, it ensures security in the quality of the product delivered according to testable scenarios for non-functional requirements in a real example.

This case study was applied to a system with mixed CRM and people management characteristics, so it was chosen it due to its relevance to the company. In addition, the systems only adopt BDD for functional requirements and do not cover non-functional ones, which, according to the quality team, lacks information for tests to be carried out, generating retests in several cases. A multidisciplinary team was involved during the case study with the following team members:

- A software architect responsible for defining the architecture and non-functional requirements;

- Two developers in charge of implementing the acceptance scenarios;
- A Team Lead, responsible for leading the process and helping with communication between team members;
- A test analyst tasked with automating performance tests using Locust in Python;
- A DevOps, responsible for configuring the infrastructure and deploying the system.

The software architect created user stories and acceptance scenarios that described non-functional performance requirements based on the characteristics of the ISO/IEC/IEEE 25010 Standard, following the gherkin language, a standardized and widely adopted language in BDD frameworks. These user stories and acceptance scenarios were presented to the team and relevant stakeholders to gain their approval, ensuring a common understanding of the non-functional requirements.

The tests were automated after the user stories were written, and the acceptance and approval criteria were set by those involved. The test analyst was responsible for automating performance tests using the Locust tool in Python, ensuring that acceptance scenarios were translated into executable tests.

The tests were carried out, and the metrics available in the acceptance criteria were monitored using analysis tools. During the execution of performance tests, the multidisciplinary team, including the software architect, test analyst, and DevOps, tracked the results and metrics generated by monitoring and analyzing the results using Grafana and Jaeger tools.

3.3.1 Interview

Before executing the automation tests, one of the authors interviewed the team lead who used BDD in this case study. In a brief conversation, the team lead informed us of how the requirements were approved by interested parts to achieve the behavior expected by the software. Furthermore, he mentioned the importance of integration between the team, the process, and the quality of the software, aiming to obtain releases that meet what was requested, stating the importance of using BDD from the elicitation of the requirement to the execution of the test cases.

In order to guarantee the performance efficiency characteristic expressed in the ISO/IEC/IEEE 25010 Standard, as well as its sub-characteristics, based on the elicitation of requirements, user stories were created with their respective acceptance criteria reported in the following Section.

3.3.2 Case execution

The execution of the case study is based on the functionalities and scenarios defined following the ISO/IEC/IEEE 25010 standard using BDD. The case study intended to evaluate the

system's performance efficiency concerning three sub-characteristics: time behavior, resource utilization, and capacity. User stories and acceptance criteria following the gherkin language are described as follows. This standardization ensures the reliability and consistency of these concepts, instilling confidence in their use. Each subsection corresponds to one of the sub-characteristics addressed in this study in relation to the ISO/IEC/IEEE 25010 Standard. Figures 6, 7, and 8 as follows were extracted from Locust and are part of the BDD automation.

3.3.3 User story: System efficiency (time behavior)

Description: As a system's administrator, the team seeks to ensure the system can handle 100 concurrent users for 7 days, ensuring efficiency is consistent during periods of high demand.

Scenario: Time behavior test

Given that the system is working correctly and can accommodate 100 simultaneous users, as shown in Figure 6.

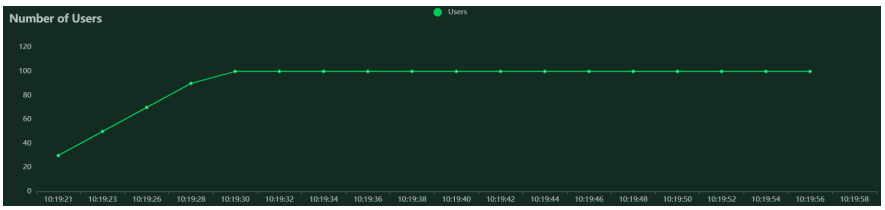

Figure 6 – Number of users

When 100 simultaneous users access the system for 7 consecutive days, each user creates an opportunity every 5 minutes, as shown in Figure 7.

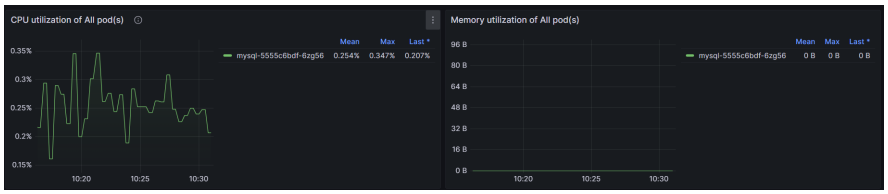

Figure 7 – Memory usage

Then the system must maintain an average response time of less than 5 seconds, not present critical errors during the test, and the use of server resources must remain below 80%, as shown in Figure 8.

3.3.4 User Story: Resource utilization

Description: As a system's administrator, the team seeks to ensure that the system can handle 1000 concurrent users for 30 minutes, ensuring that server resource utilization remains below 85% during load testing.

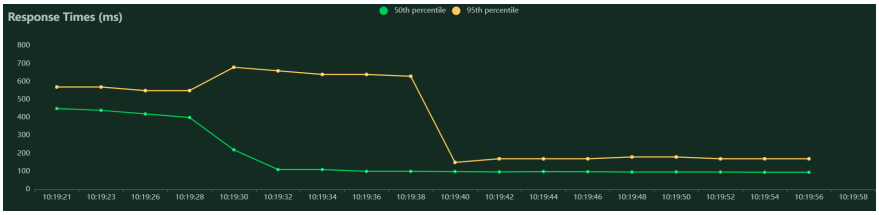

Figure 8 – Time to response

Scenario: Resource utilization test

Given that the system is working correctly and has capacity for 1000 simultaneous users.

When 1000 simultaneous users access the system for 30 minutes.

Then server resource utilization should not exceed 85%, and the system should not experience critical errors during testing.

3.3.5 User Story: Capacity

Description: As a system's administrator, the team seeks to ensure that the system can handle 2,700 concurrent users for 30 minutes, ensuring that system capacity is sufficient to meet demand during peak usage.

Scenario: System capacity test

Given that the system is working correctly and has a capacity for 2,700 simultaneous users.

When 2,700 simultaneous users access the system for 30 minutes.

Then server resource utilization should not exceed 90%, and the system should not experience critical errors during testing.

3.4 Experiment

Experiments are an empirical method that aids in the evaluation and validation of research results (WOHLIN; HÖST; HENNINGSSON, 2003). In Software Engineering, experiments aim to identify the outcomes of certain situations and seek to benefit the field with potential discoveries.

For this experiment, a library provided by a company containing 34 user stories was used. Of these, 30 stories had three acceptance criteria, and four had only one criterion, adding 94 acceptance criteria. The stories were written in Gherkin language using the BDD framework in the native language of the researchers (Brazilian Portuguese). Based on these, a prompt was created for each scenario in the selected Large Language Models. The stories and their respective scenarios and prompts can be viewed at the link[11].

[11] https://doi.org/10.5281/zenodo.13155965

The main goal of this research is to "**Analyze** the efficiency of user stories using BDD, **aiming at** automatic generation of test code, **in comparison to** Large Language Models (LLMs), **from an** academic **perspective, in the context of** software development". To achieve this goal, the posed the following research questions.

The objective at this point is to compare the effectiveness of LLMs Grok, Gemini, ChatGPT, and GitHub Copilot in generating automated tests based on user stories and acceptance criteria, following BDD in the Gherkin standard, compared to computerised tests performed by a development team. This objective can be divided into five sub-objectives: each part is related to one of the research questions and has their own metrics, and hypotheses detailed as follows:

Objective 1: Measure the similarity of responses from different LLMs

- **RQ1** - Question: What is the similarity of responses among the different AIs when generating automated tests?
- *Metric:* Similarity coefficient (*Cosine Similarity*).
- Null Hypothesis (H0.1): There is no significant difference in the similarity of responses among the different AIs.
- Alternative Hypothesis (H1.1): There is a significant difference in the similarity of responses among the different AIs.

Objective 2: Validate whether the results generated by the LLMs cover the acceptance criteria.

- **RQ2** - Question: What is the coverage of acceptance criteria by the tests generated by each AI?
- *Metric:* Acceptance criteria coverage (percentage of acceptance criteria covered by the generated tests).
- Null Hypothesis (H0.2): There is no significant difference in the coverage of acceptance criteria among the different AIs.
- Alternative Hypothesis (H1.2): There is a significant difference in the coverage of acceptance criteria among the different AIs.

Objective 3: Evaluate the accuracy of the tests generated by the different LLMs

- **RQ3** - Question: What is the accuracy of the generated tests compared to a reference test set?

- *Metric:* Test accuracy (percentage of correspondence between the generated tests and the reference test set).
- Null Hypothesis (H0.3): There is no significant difference in the accuracy of the tests generated among the different AIs.
- Alternative Hypothesis (H1.3): There is a significant difference in the accuracy of the tests generated among the different AIs.

Objective 4: Evaluate the efficiency, in terms of time, for generating the tests

- **RQ4** - Question: How much time is required to generate the tests by each LLM?
- *Metric:* Test generation time (average time required to generate tests).
- Null Hypothesis (H0.4): There is no significant difference in the test generation time among the different AIs.
- Alternative Hypothesis (H1.4): There is a significant difference in the test generation time among the different AIs.

Objective 5: Evaluate the clarity of responses among different executions of each AI

- **RQ5** - Question: What is the clarity of responses among different executions of each AI?
- *Metric:* Clarity of responses (evaluated by subjective criteria such as readability, comprehensibility, and adherence to acceptance criteria).
- Null Hypothesis (H0.5): There is no significant difference in the clarity of responses among the different AIs.
- Alternative Hypothesis (H1.5): There is a significant difference in the clarity of responses among the different AIs.

3.4.1 Experiment Execution

The methodological steps taken for executing this experiment are outlined next:

1. Submit each user story and their respective acceptance criteria to the LLMs Grok, Gemini, ChatGPT, and GitHub Copilot using a standard prompt;
2. Generate and document the test code returned by each source;
3. Execute the generated tests and record the results;

Metric	Definition	Data Collection
Accuracy	It refers to the proportion of tests that passed (correct results) among all executed tests.	After executing the generated tests, the number of tests that passed and failed was recorded.
Coverage	It refers to the proportion of requirements or acceptance criteria covered by the generated tests.	The number of acceptance criteria covered by the generated tests was checked for each user story.
Clarity	It refers to the readability and comprehension of the generated tests. It can be qualitatively evaluated by a group of developers or through automatic readability metrics.	Developers could assign a score from 1 to 5 for each generated test. Alternatively, readability metrics such as *Flesch Reading Ease* could be used.
Efficiency	It refers, in the context of this paper, to the time required to generate the tests.	The time from the test request to its generation and recording was measured.

Table 7 – Evaluated Metrics

4. Statistically evaluate the results.

Table 7 shows the metrics adopted in this study, which are partially related to the need to improve alignment between acceptance criteria and test automation. The acceptance criteria stage must have a positive result for the automation of its respective tests to be carried out. Thus, they must be in line with the behavior expected by the software. The metrics used were selected with the aim of achieving the objectives proposed in this study. In addition, the efficiency measured is associated with the time needed to create the tests, thus reducing the manual workload, as identified as a gap in the literature review.

Through these metrics, results can be more assertive, increasing the reliability of results and providing clarity and effectiveness to the conclusions of this research.

3.5 Case Study - Guidelines

Case studies are used when analyzing a specific sample is needed. With this case study, exploring the guidelines for applying BDD to understand better how and why to use it (PERRY; SIM; EASTERBROOK, 2006) will be possible. This study is characterized as exploratory and descriptive (RUNESON; HÖST, 2009), as it seeks to explore and describe the adoption of BDD by teams that do not use it.

Therefore, a case study was conducted to evaluate the possibility of applying guidelines to adopting BDD and to validate previous studies regarding this framework. Part of the Goal-Question-Metric protocol (WOHLIN et al., 2012) was used, namely: "**To analyze** the adoption of BDD, **with the purpose of** implementing, **in relation** to agile teams, **from the point of view**

of professionals , **in the context of** software development". To achieve this goal, the research questions are presented in Table 8.

ID	Definition	Justification
Q1	What are the guidelines inherent to adopting BDD?	Present the guidelines to facilitate the adoption of BDD.
Q2	Did BDD achieve the expected purpose concerning software development?	Identify whether BDD applies to teams that have never used it to develop software.
Q3	What is the respondents perception that adopted BDD for this case study?	Present positive and negative points through a survey regarding the adoption of BDD.
Q4	Are the guidelines presented effective for adopting BDD?	Validate the characteristics found by the authors in previous studies.

Table 8 – Research questions

To answer the research questions and achieve the general objective of this study, a case study and a survey were conducted with a team that did not use BDD in its work activities at a private Brazilian security software development company.

3.5.1 Survey

Surveys are used to understand how a group behaves in a given situation. The benefits of surveys are that they are easy to apply and low cost since everything is done online (PUNTER et al., 2003). The objective of this survey is to analyze whether the guidelines presented were effective in their adoption by agile teams to improve the quality of software development (WOHLIN; HÖST; HENNINGSSON, 2006). This survey is exploratory (WOHLIN; HÖST; HENNINGSSON, 2006) because it aims to collect information related to the adoption of BDD that can help with the guidelines found.

3.5.2 Execution

The Survey was open for seven days in October 2024. The target audience for this research was the team leaders of a private Brazilian company in the security software development sector. A team leader is understood as a professional responsible for supervising his team's activities and productivity. 4 responses were obtained, reaching 100% of the respondents from the population. The Survey consisted of 13 closed questions expressed in Table 9.

ID	Definition	Justification
SQ1	Did BDD manage to achieve the objective expected by the team?	Present whether the BDD was effective.
SQ2	What is your perception regarding the adoption of BDD for software development?	Use the participant's expertise to check how easy it is to understand how to use BDD.
SQ3	What is your perception regarding the adoption of BDD as technological support?	Analyze the effectiveness of adopting BDD as technological support.
SQ4	What is your perception regarding the adoption of BDD with the gherkin pattern?	Analyze the effectiveness of the gherkin pattern for using BDD.
SQ5	What is your perception regarding the adoption of BDD using Large-Language Models (LLM) to support writing?	Analyze the result of writing through the use of LLMs for scenarios using BDD.
SQ6	What is your perception regarding the adoption of BDD using LLM to support test automation?	Present aspects aimed at test automation through the use of LLMs together with BDD.
SQ7	What is your perception regarding the adoption of BDD in relation to living documentation?	Present the importance of living documentation.
SQ8	8. What is your perception regarding the adoption of BDD in human/behavioral relationships?	Present the ease of understanding and human interaction when using BDD.
SQ9	What is your perception regarding the adoption of BDD to elicit functional and non-functional requirements?	Present effectiveness in the requirements elicitation stage with BDD.
SQ10	What is your perception regarding the adoption of BDD for the "3 amigos" meeting?	Present the need for the initial meeting with a focus on aligning the software development process.
SQ11	What is your perception regarding the adoption of BDD on scenarios review?	Present the ease of using BDD to review scenarios.
SQ12	What is your perception regarding the adoption of BDD in relation to communication between stakeholders?	Present the importance of communication in the software development process.
SQ13	What is your perception regarding the adoption of BDD in relation to standardized writing?	Present the need for standardized writing in relation to the adoption of BDD.

Table 9 – Survey Questions

4

Results and Discussion

This Chapter presents the results and discussions of this research.

4.1 Multivocal Literature Review (MLR)

Definitions mentioned by Shaw (SHAW, 2003) were used in this study, aiming to evaluate each type of result according to the definition provided in her article. Shaw's article was chosen to guide some of our research questions as she addresses the necessary components for exemplary research in computing and is a reference in the field. The answers to each of the research questions of this study are below.

4.1.1 RQ1 - What type of results?

According to Figure 9, the type **Report** was the primary type obtained with 90 articles, followed by **Procedure or Technique** with 39, showing cases where it uses BDD. Moreover, the type **Qualitative or descriptive model** also presented a considerable number of 30 articles, followed by **Tool or notation** with 21. The other types not mentioned had a lower quantity than those presented, showing possible types to address in new investigations.

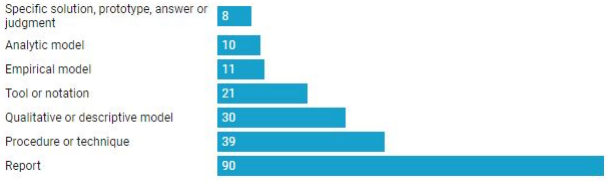

Figure 9 – RQ1 - What type of results?

4.1.2 RQ2 - What type of validation?

Figure 10 answers the second research question. The types of validation of the selected articles were obtained, with the primary type being **Analysis** mentioned in 148 articles, followed by **Evaluation** with 98. It received 80 results for **Example** to infer a type of validation widely used when dealing with BDD. In addition, 26 articles with **Experience** validation were found, so it is possible to carry out even more investigations with this type of validation in future works.

Figure 10 – RQ2 - What type of validation?

4.1.3 RQ3 - What type of methodology?

Figure 11 identifies the methodologies used in the articles, answering the third research question. It was noticed that the **Descriptive** methodology stood out, being mentioned in 80 articles, denoting the need for a good description of how to carry out an investigation, followed by the **Case Study** with 55 and **Experiment** with 28, so that it could observe that case studies, while carried out with a more abstract perception, are less solid than experiments, which aim at more assertive data pointed at their realization.

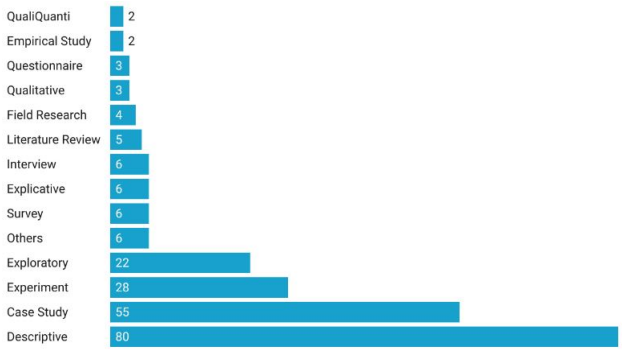

Figure 11 – RQ3 - What type of methodology?

It identified 22 articles with the **Exploratory** type, which infers the importance of exploring actions aimed at BDD. Finally, it is worth mentioning that it found six articles for the **Survey** type of methodology so that there could be more applications of this type aimed at better understanding the behavior of those who use BDD, whether in research at the academy or everyday life in the industry.

4.1.4 RQ4 - In which application domains is BDD most used?

Figure 12 identifies which domains apply BDD, the fourth research question. 72 articles **Not mentioned** which domain it can be understood as a point to improve in future investigations. In addition, it had 36 articles with the domain **Information Systems** and 29 with **Workflow**, thus denoting the main domains found. It is necessary to carry out studies in specific domains to validate the framework.

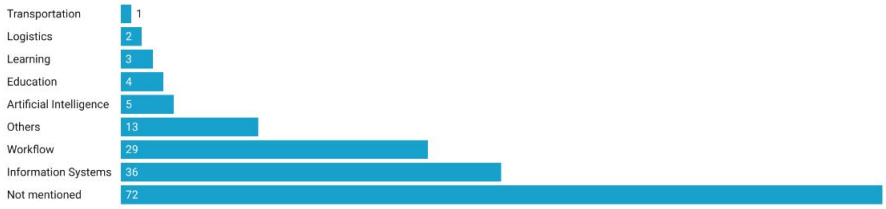

Figure 12 – RQ4 - In which application domains is BDD most used?

To better explain the use of BDD, academia, and industry would benefit from such aspects, as they would be able to identify which domains the framework would be most effective.

4.1.5 RQ5 - In which context is BDD most used?

Figure 13 identifies the context where BDD applies, answering the fifth research question. 77 articles mentioned that BDD applies as **Test**, followed by **Requirement** with 43 and **Not mentioned** with 27, detonating, in the latter, the need to mention the context in which adopts BDD, aiming to contribute to clearer methodological elaborations.

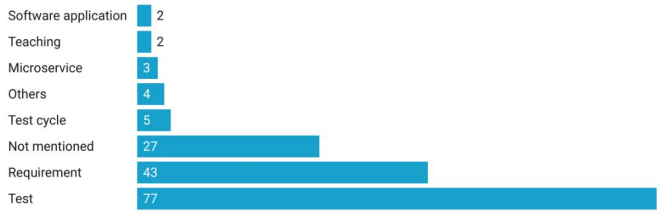

Figure 13 – RQ5 - In which context is BDD most used?

According to North (NORTH, 2006), BDD is a tool aimed at behavioral testing of the system. The BDD framework is present in requirements elicitation and test automation throughout the software lifecycle. The context that uses BDD confirms what Dan North intended when he created the framework.

4.1.6 RQ6 - Are benefits of adopting BDD presented? Which ones?

Figure 14 presents the benefits related to the adoption of BDD, obtaining 54 mentions for **Readability**, followed by 52 for **Communication**, corroborating what North (NORTH, 2006)

mentioned when characterizing BDD, in addition to what Purkayastha et al. (PURKAYASTHA et al., 2020) found. Moreover, 51 articles **Not mentioned** if there are benefits related to the adoption of BDD so that there is a need to improve the characterization of BDD in the investigations carried out.

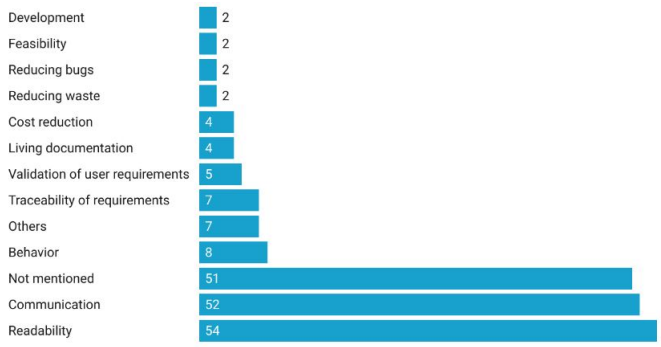

Figure 14 – RQ6 - Are benefits of adopting BDD presented? Which ones?

BDD has positive aspects in readability and communication because it uses the Gherkin language. Features written using BDD follow the Given-When-Then pattern, which improves everyone involved in the process's understanding, as well as the execution of more precise and assertive functionality.

4.1.7 RQ7 - Are harms of adopting BDD presented? Which ones?

Figure 15 presents some negative points related to the adoption of BDD. As it can be seen, the most significant number found were articles that **Not mentioned** harms in its adoption with 131 responses, followed by **Others** with 52, with this number relatively high due to the small mentions focused on details such as aspects related to large-scale projects or delayed return on investment applied to their adoption. It also got six mentions for **Cost**, initially associated with the cost of implementation concerning other tools, 3 for **Lack of experience**, and 3 for **Productivity**, so the last two are linked, where it can be inferred that if there is no experience the probability of Productivity becomes low.

Figure 15 – RQ7 - Are harms of adopting BDD presented? Which ones?

As the most significant number found in the articles analyzed were **Not mentioned** related to negative aspects inherent to the use of BDD in 15, it can be understood from this point that the framework has a good resourcefulness in what was proposed by Dan North (NORTH, 2006), as its use continues to be adopted by the industry.

4.1.8 RQ8 - For the adoption of BDD, which are tools most used?

Figure 16 identifies the main tools used for the application of BDD, resulting in the **Cucumber** tool in 71 articles, in which it is worth mentioning that this tool is the oldest, followed by 67 articles **Not mentioned**, highlighting, once again, the lack of clarity regarding the characterization of BDD. Also, it got 32 mentions for **Jbehaviour**, a tool created by Dan North.

Figure 16 – RQ8 - For the adoption of BDD, which are tools most used?

These results identified the main points related to the adoption of BDD. We, therefore, proceed with the Discussion of the results found in the next Subsection.

4.1.9 Discussion

The main aspects related to the adoption of BDD could be identified, for example, the types of validation about its adoption, as well as tools that support the execution of this framework, in addition to positive points aimed at its implementation. Related to Farooq et al. (FAROOQ et al., 2023), Solis (SOLIS; WANG, 2011) and Arnyndiasari, Ferdiana, and Santosa (ARNYNDIASARI; FERDIANA; SANTOSA, 2022), it could advance on the state-of-the-art by characterizing BDD through a Multivocal Literature Review.

It could identify Dan North's primary goal when he created BDD: the focus on communication and readability (NORTH, 2006) being achieved within the literature through the validation found in Figure 11. It also identified a need for experimental investigations to identify cases better individually to obtain more accurate results in the studies carried out. However, due to the lack of experimental studies, results can be generalized regarding the same situation.

Regarding the negative aspects related to the adoption of BDD, it was inferred from Figure 15 that most of the articles did not mention negative aspects related to its adoption. It could be inferred that BDD is a good framework used by the industry, mainly regarding requirements elicitation and tests, as shown in Figure 13. Moreover, as mentioned by Garousi and Zhi (GAROUSI; ZHI, 2013), who surveyed the industry and found the BDD used as a testing tool, besides Yang, Costa, and Zou (YANG; COSTA; ZOU, 2019) and Scandaroli (SCANDAROLI et al., 2019), in mentioning how BDD works in describing features, contributing to the clarity process.

Figure 14 concludes the importance of people adopting BDD. Having presented the improvements using this framework, Communication and Readability in delivering the final

product will have been more assertive, helping with less cost and rework, as long as the product developed will have more quality.

Through Figure 16, it could realize the Cucumber and Jbehaviour frameworks' impact on BDD. It could identify these most used frameworks related to BDD, inferring their importance to adopting BDD better.

Besides reaffirming what Nascimento et al. (NASCIMENTO et al., 2020a) reported that there are more positive than negative results related to the adoption of BDD, it has advanced in studies on the subject, so that it contributes to the scientific community with the main aspects found in the investigated databases, in addition, to contribute for the industry in terms of understanding the functionality of BDD.

4.2 Survey

The survey was open for responses for a month, and 43 respondents were reached. The answers to the research questions in Table 5 are as follows.

4.2.1 Q1 - How long have you been using the Behavior-Driven Development (BDD) framework?

According to Figure 17, a decreasing order can be observed concerning the time of use of BDD and the number of professionals who use it, so that the longer the experience, the fewer respondents one can obtain.

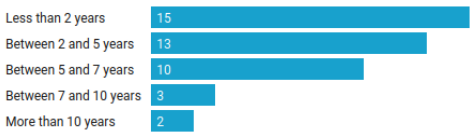

Figure 17 – Time of use of BDD

Interestingly, this gradual increase can be seen among the respondents, so it can be inferred that BDD has become more popular for software development in recent years.

4.2.2 Q2 - What is the main tool you use for BDD?

According to Figure 18, 12 tools were identified, so the four most cited ones can be highlighted, with Cucumber being the most used, where it is worth emphasizing that it works in several languages, such as Java, for example; Behavior, a framework for the Python language; Specflow, which is a framework for .Net; and JBehaviour, which is the Java framework created by Dan North.

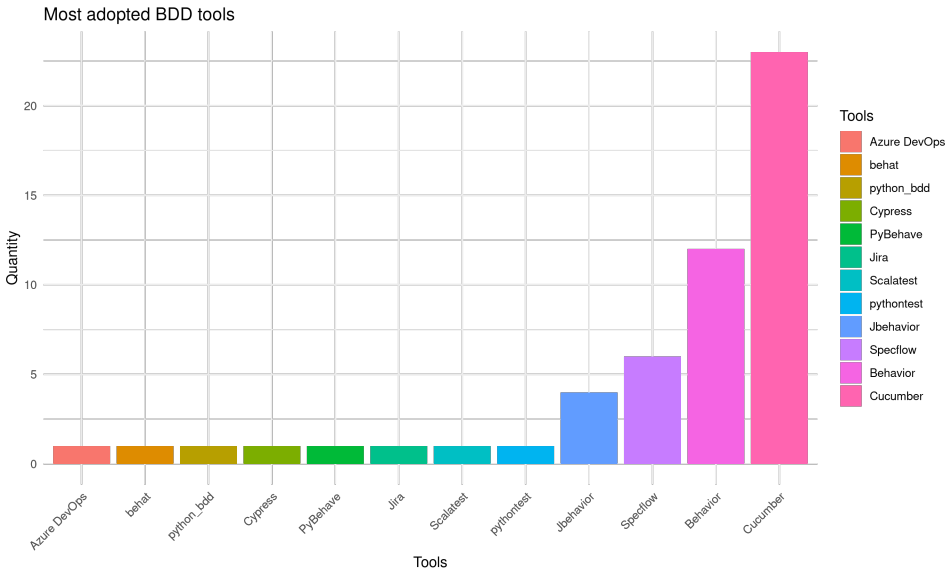

Figure 18 – Most used tools when using BDD

Through this graph, it is possible to understand which tools BDD is most used for to infer which ones can work more effectively and more straightforwardly.

4.2.3 Q3 - Where do you use BDD most?

According to Figure 19, one can observe the frequency of BDD use in the software development process stages to highlight the test stages, requirements, and test cycles with the highest number of mentions.

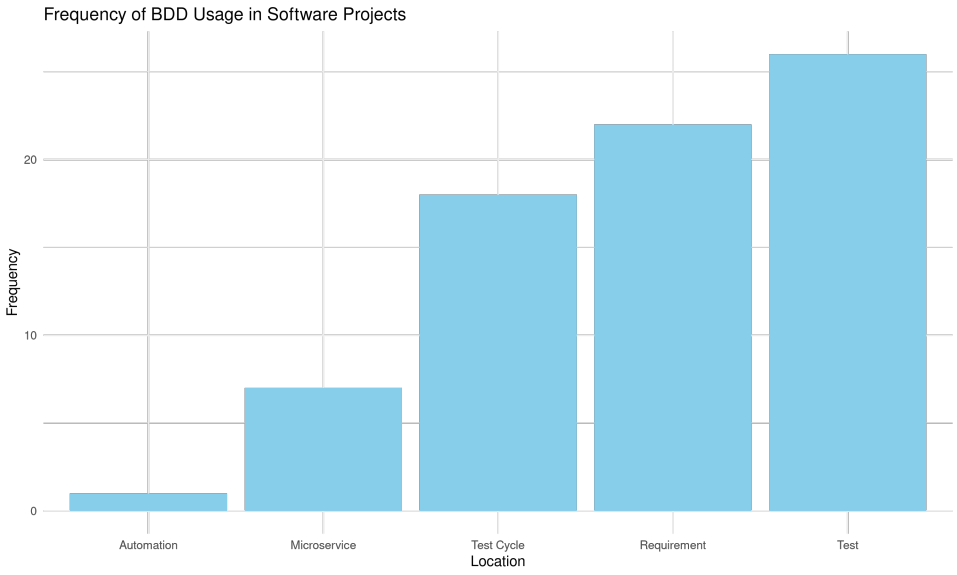

Figure 19 – Places where BDD is most used

As BDD is a framework used throughout the software life cycle, it was also mentioned in other stages but on a smaller scale. Furthermore, it is interesting to highlight the following

response from one of the participants:

> *Despite being aware of the ability to automate tests with BDD, such as Cucumber, I use BDD to specify requirements, detail the user story narrative, and identify user acceptance criteria that can be used as scenarios or test cases, patterns, and exceptions.*

With this, it can be seen the use of BDD in terms of its readability to help professionals when dealing with user story narratives and their respective acceptance criteria. Therefore, it can be inferred that the use of BDD demonstrates as a positive point the use of the *gherkin* language and the "given, when, then" pattern to be used in a practical way to achieve behaviors desired, the primary purpose of Dan North (NORTH, 2006).

4.2.4 Q4 - What are the main benefits related to adoption of BDD?

According to Figure 20, it is possible to observe the main benefits according to the respondents, highlighting communication as the main benefit, followed by traceability and requirements, and behavior and readability.

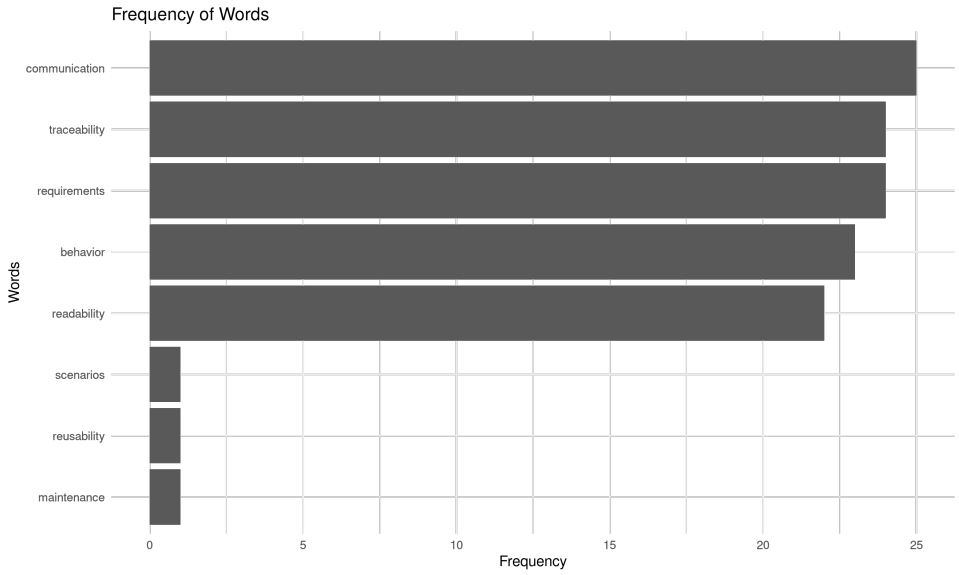

Figure 20 – Benefits to adopting BDD

In this way, it can be seen that the aspects highlighted by Dan North remain concerning the use of BDD, these being communication and readability (NORTH, 2006). It is also interesting to note that the other three aspects mentioned appeared with a high number of mentions, so it is possible to infer that BDD also has as positive characteristics the assertiveness in the behavior expected by the system to meet the elicited requirement in addition to the traceability of such requirements.

Furthermore, it is worth highlighting the response of one of the participants regarding the benefits of using BDD, namely: “Reusing the use of similar scenarios in test automation.” With this, the effectiveness of using BDD is to help professionals to identify the similarity between the automated tests, where it is possible to infer the saving of time for the professional and money for the company through this statement.

4.2.5 Q5 - What are the main difficulties encountered for adoption of BDD?

According to Figure 21, it is possible to see that the main difficulties mentioned were the lack of experience and writing. The lack of knowledge can be linked to the other factors mentioned as negative points, so if there is no user experience, it becomes challenging to maintain a positive result in different aspects.

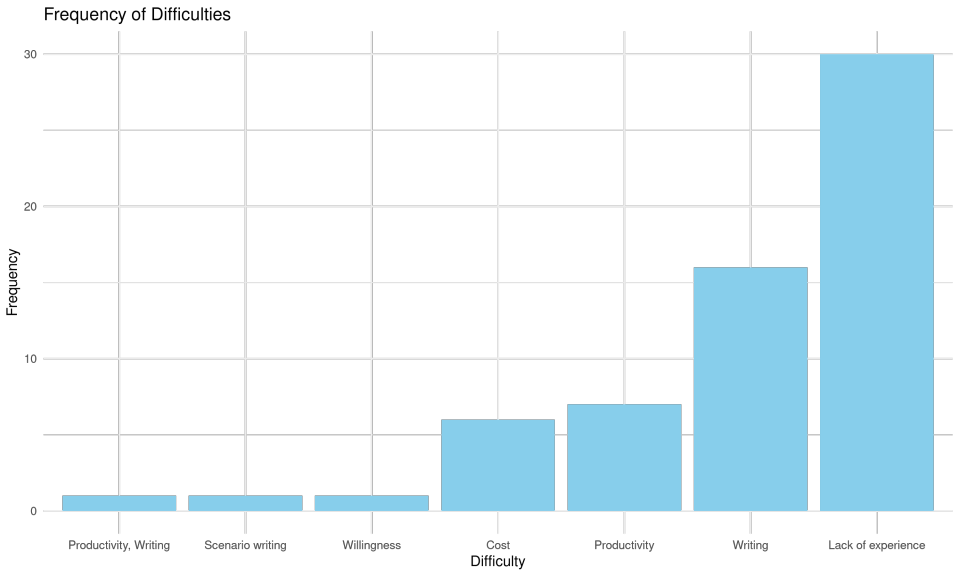

Figure 21 – Difficulties in adopting BDD

It is worth highlighting the response of one of the participants regarding the difficulties in adopting BDD, as follows:

> *Understanding how it is done, as BDD must be used as a behavioral reference for development using the 3 friends for writing to occur. Thinking about BDD for any test is not BDD, it is already a test scenario using Gherkin. Many people confuse Gherkin with BDD.*

Therefore, it is possible to infer that there is a need for an initial meeting of the “3 friends” where the requirements can be elicited appropriately, aiming for the behavior expected by the software. The meeting aims to illustrate the necessary behaviors of software so that their respective acceptance scenarios are outlined later, according to the participant’s response.

Still, another participant stated the difficulty: "*Learning curve to understand the concept and apply it efficiently and effectively.*" Therefore, it is essential to have the necessary understanding inherent to adopting BDD since, if misapplied, the benefits arising from this framework cannot be used.

Finally, the last answer to be highlighted is:

> *I believe that a "pre-concept" was generated because it was used little or in a certain way incorrectly. When we look at implementation with bdd, there is only an additional layer where more verbose words are written for more understanding, and it does not make execution take longer or cause configuration hangs. I've seen people work, for example, with Cypress and Cuca, which is highly costly because Cypress itself ends up blocking the reuse of the cucumber.*

This answer may be linked to both cost and lack of experience. It is possible to infer that when choosing other tools to complement the use of BDD, if chosen incorrectly, the budget and quality of software delivery can be compromised, causing losses instead of benefits.

4.2.6 Q6 to Q12

According to Figure 22, one can see the graphs referring to questions 6 to 12.

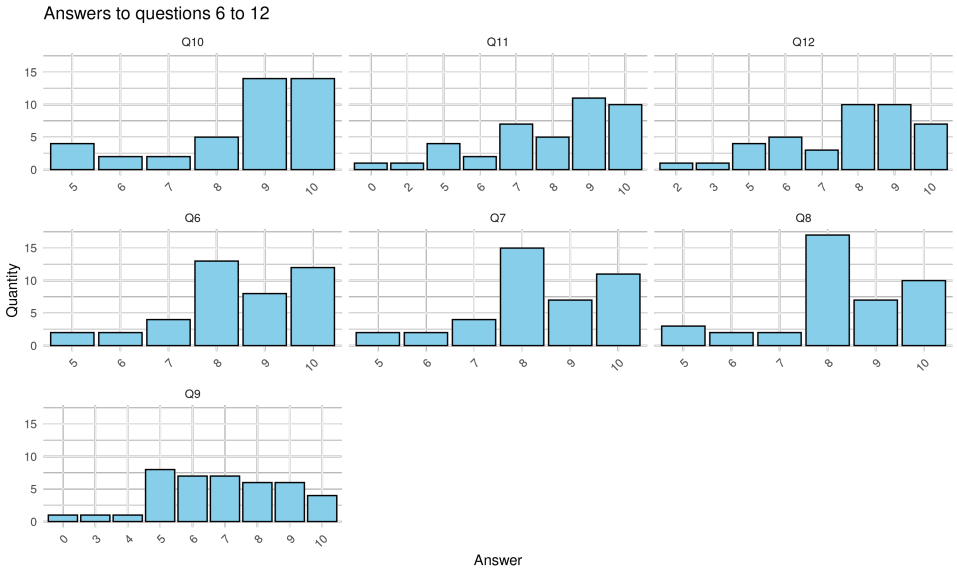

Figure 22 – Answers to questions 6 to 12

These data will be discussed in the following Subsections.

4.2.7 Q6 - On a scale of 0 to 10, how much is the BDD language understandable (Gherkin)?

According to the graph referring to Q6, most respondents chose 8, 10, and 9, respectively, and the rest chose 5 or more. In this way, it sheds light that the BDD language is, in this case, considered understandable, and it can be inferred that the use of *gherkin* remains positive in the face of recurring work activities.

4.2.8 Q7 - On a scale of 0 to 10, considering the purpose of the BDD regarding readability, according to its creator, Dan North, how much does BDD achieve in this regard?

According to the graph referring to Q7, all respondents chose the value 5 or more, with the highest number of votes being 8, 10, and 9, respectively. Thus, it is inferred that the initial objective of BDD outlined by Dan North is successfully maintained.

4.2.9 Q8 - On a scale of 0 to 10, considering the purpose of the BDD regarding communication, according to its creator, Dan North, how much does BDD achieve in this regard?

According to the graph referring to Q8, all respondents chose the value 5 or more, with the highest number of votes being 8, 10, and 9, respectively. It is noted that the communication aspect also maintains quality concerning the use of BDD. This factor may be related to the fact that there is a meeting of the “3 friends” so that the requirements and the time for delivering the releases are planned.

4.2.10 Q9 - On a scale of 0 to 10, how much does BDD perform faster delivery?

According to the graph referring to Q9, it can be seen that there was a distribution between the scores, ranging from 0 to 10 among the respondents. This factor can be linked to the professional’s experience, so the more experience they have, the faster delivery can be carried out.

4.2.11 Q10 - On a scale of 0 to 10, how much does BDD perform to deliver with higher quality?

According to the graph referring to Q10, it is clear that most respondents opted for values 9 and 10, assuming delivery effectiveness with the use of BDD. Additionally, all other participants responded with 5 or more.

4.2.12 Q11 - On a scale of 0 to 10, how practical is BDD to reperform updates in the code (living documentation)?

According to the graph referring to Q11, the highest concentration of respondents was between 9 and 10. This aspect can be linked to the fact that the professionals who carry out the code also carry out the documentation in the correct way so that when the code needs to be updated due to a problem, for example, with documentation appropriately done, it will make the work easier, saving resources such as time and money.

4.2.13 Q12 - On a scale of 0 to 10, how much does BDD contribute to the relationship to the company's economy when having living documentation?

According to the graph referring to Q12, responses ranged from values 2 to 10, obtaining their highest concentration between values 8 and 9 with an equal number of respondents and value 10 shortly after that. The relationship between the savings that the company can generate through dynamic documentation is due to the need to document the code correctly, as there is always support in case of a problem with the software.

4.2.14 Discussion

The open survey obtained 43 respondents. According to Figure 23, when analyzing questions 6 to 12, it can be identified that all of them have an average above 7.5 in the answers. Only Q9 and Q11 have discrepant values with answers below grade 5.

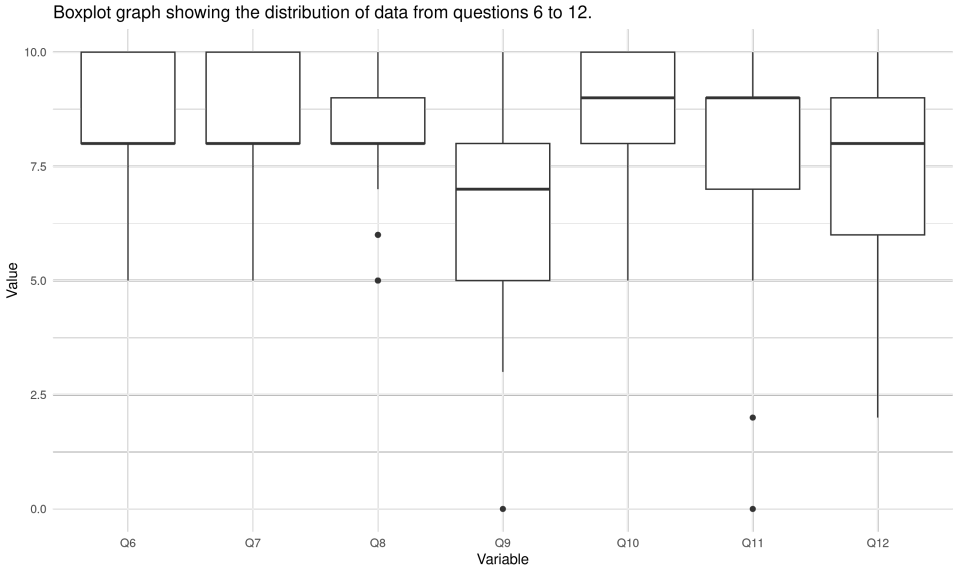

Figure 23 – Distribution of answers to questions 6 to 12

According to Figure 24, the relationship between the participants' experience and understanding of BDD can be observed, with a positive correlation of 0.238661. In the Experience line, there are five columns with values from 1 to 5, where 1 is less than 2 years; 2 is equivalent to between 2 and 5 years; 3 is between 5 and 7 years; 4 is between 7 and 10 years; and, finally, 5 is equivalent to more than 10 years.

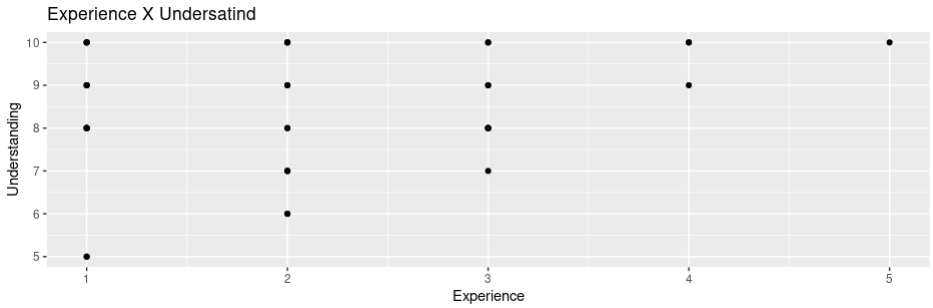

Figure 24 – Correlation between experience and understanding

Analyzing the correlation between experience time and perceived difficulty in using a framework is essential to understanding the impact of developers' experience on the effectiveness of using computational tools. This study calculated the correlation coefficient between the developers' experience time and the perceived difficulty in using the framework as 0.238661. This value suggests a weak positive correlation between these two variables, indicating that, in general, an increase in experience time is associated with a slight increase in perceived difficulty in using the framework.

Although this correlation is statistically significant, its magnitude is relatively low, suggesting that factors other than length of experience may have a more substantial influence on developers' perception of difficulty in using the framework. Therefore, additional studies are necessary to promote a better understanding of the relationship between developers' experience and the effectiveness of using the framework, considering a more comprehensive range of variables and development contexts.

Identifying this correlation between experience and understanding can contribute to the study carried out by Silva and Fitzgerald (SILVA; FITZGERALD, 2021) regarding the understanding of BDD so that in this new study, it can be observed that experience directly impacts the understanding of use of BDD, in a way that can lead to future problems such as poorly elicited requirements, for example, which could have been mitigated initially if there had been the necessary understanding inherent to the use of the framework.

Regarding experience and readability, a positive, but very weak, correlation of 0.09434189 was obtained, as shown in Figure 25. This correlation between time spent using the BDD framework and the readability objective, as outlined by its creator Dan North, suggests a positive but weak association between these two variables.

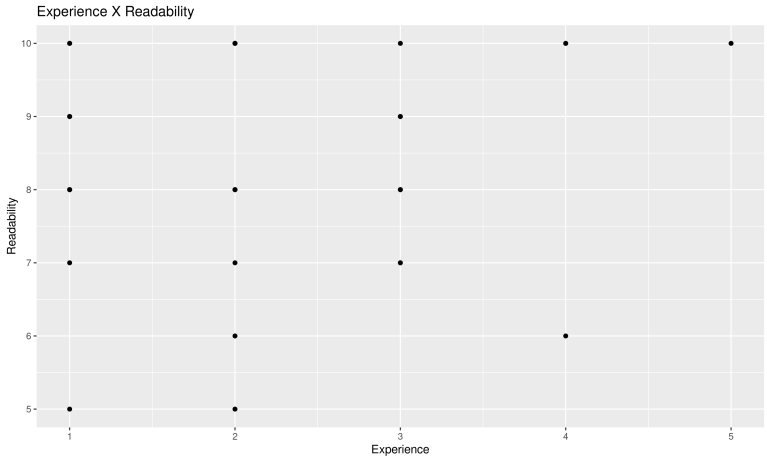

Figure 25 – Correlation between experience and readability

This implies that, in general, an increase in BDD usage time may be associated with an improvement in the readability of tests and software specifications, as recommended by BDD. However, it is essential to note that the correlation is relatively low, indicating that factors other than usage time may significantly impact on readability. Therefore, to maximize the effectiveness of BDD in promoting code and specification readability, it is critical to consider not only usage time but also other aspects of the software development process and BDD adoption.

Figure 26 shows the relationship between experience and communication, with a weak positive correlation of 0.2447518.

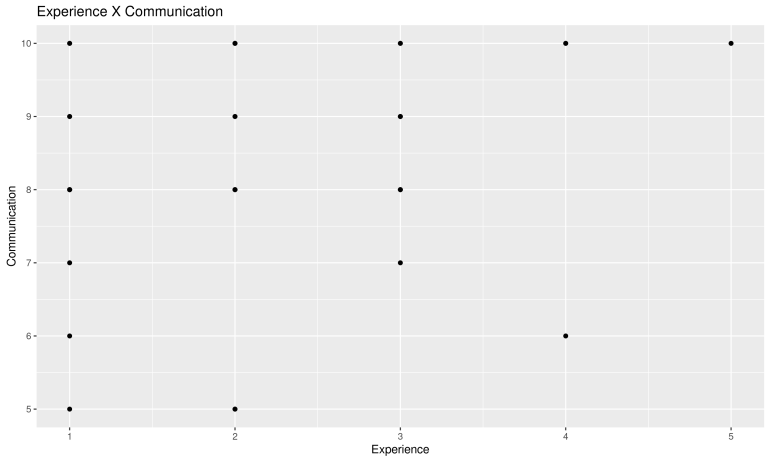

Figure 26 – Correlation between experience and communication

It can be seen that experience directly impacts the communication factor as well, so the weak positive correlation demonstrates that based on the experience obtained by professionals, communication can be improved to assist in delivering the final product, also acting as an essential factor in the process.

Making a correlation between the data collected, it is clear that the level of experience and speed of delivery, as shown in Figure 27, presents a weak positive correlation of 0.1963602.

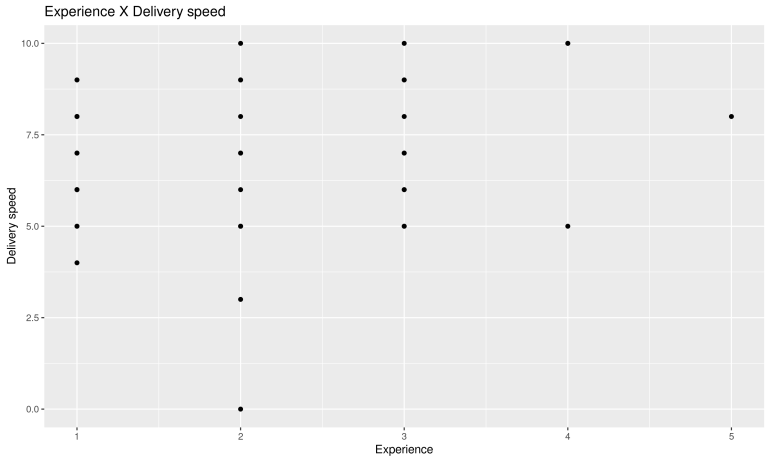

Figure 27 – Correlation between time of experience and speed of delivery

One can observe that the most significant relationship between both aspects was between columns 1 and 3, representing the most important number of professionals who, even with little experience using BDD, can identify that the framework has a fast delivery.

Regarding the relationship between experience time and delivery quality, despite the relationship being positive, it is a very weak correlation with a value of 0.06831631, as can be seen in Figure 28.

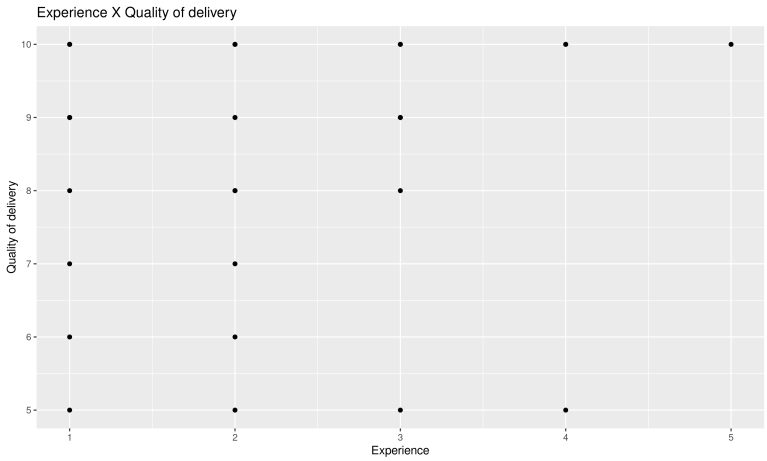

Figure 28 – Correlation between experience time and delivery quality

The correlation of 0.06831631 between time using the BDD framework and quality software delivery suggests a positive but weak association between these variables. Although the correlation indicates that an increase in BDD uptime may be loosely related to an improvement in the quality of software delivery, it is essential to note that this association is relatively low. This implies that factors other than BDD usage time may significantly influence the quality of software delivery, such as the team's expertise, the quality of requirements captured in tests, and the effectiveness of adopted agile development practices.

In the Experience line, columns 1 to 5 represent the experience time, as explained previously. There is a good distribution among respondents, ranging from the professional with the least experience and who sees a low-quality delivery to the professional with the most experience and sees a better quality of delivery. Furthermore, there is a concentration of respondents with low experience who also identify quality in delivery, as shown in the first three columns of the graph.

Therefore, to achieve superior software delivery, it is crucial to consider BDD uptime and adopt a holistic approach that encompasses multiple aspects of the software development process and BDD adoption.

Next, the length of experience was analyzed about document updating, as shown in Figure 29. There is an almost insistent correlation, despite being positive, of 0.03743474.

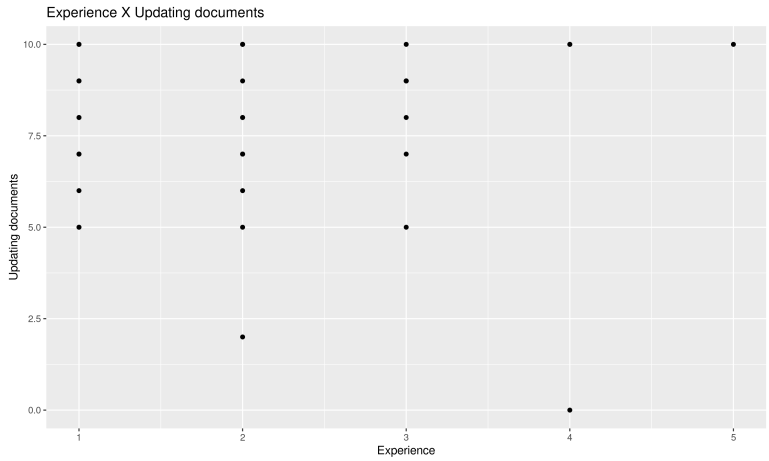

Figure 29 – Correlation between length of experience and document updating

There is a concentration of respondents in the first three columns so that even with little experience, professionals can understand the effectiveness of BDD concerning document updating.

Finally, regarding the relationship between experience and savings, there is a negative correlation, although weak, of -0.06944453, being the only negative correlation found, as shown in Figure 30.

This result may be linked to the aspect that initially, the implementation of BDD is seen as a relatively high cost, so if there is the adoption of something new in a company, there is a need for time to adapt, and, therefore, contribute effectively to the economy where the new project is being implemented.

With the information exposed, it was possible to characterize and measure the characteristics along with the adoption of BDD to contribute to the work of Rauf and AlGhafees (RAUF; ALGHAFEES, 2015) regarding the understanding of agile practices. Furthermore, it was possible to advance the knowledge regarding the state of practice in which BDD finds itself.

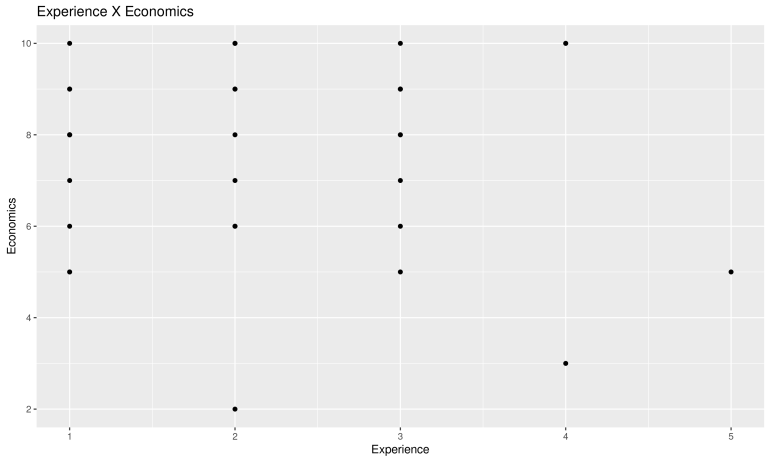

Figure 30 – Correlation between experience time and savings

Although all correlations were low, all were positive except for the correlation between experience and economy. In this way, it is clear at which points BDD can prove to be stronger and at which points there is a need for greater attention, demonstrating the need for knowledge about its scope.

4.3 Case Study - ISO/IEC/IEEE 25020

Below are the answers to the research questions presented in Table 6.

4.3.1 QP1 - How does BDD address difficulties in ensuring non-functional requirements?

BDD framework intended for test automation, such as Cucumber and JBehave, was initially designed to address the system's functional units, resulting in significant challenges in integrating user stories and their corresponding scenarios and subsequent automation. However, as evidenced in this investigation, it is possible to considerably mitigate the difficulties associated with the practical application of BDD in contexts involving non-functional requirements by adopting complementary tools.

The combination of Pytest-BDD and Locust was chosen for the present study. While the former makes it possible to apply BDD in Python environments, the latter facilitates performance testing. As a result, an integration enables the validation of the acceptance criteria established by the BDD, focusing on evaluating the system's performance. This approach demonstrated that adopting complementary tools, such as Locust, allows the framework adopted for BDD to validate the acceptance criteria efficiently, providing more excellent reliability to the results obtained.

Adopting BDD combined with complementary tools is an adequate procedure to satisfy

the system's non-functional requirements, allowing efficient compliance monitoring with the proposed scenarios.

4.3.2 QP2 - How can BDD be used to ensure quality related to the performance efficiency characteristic of the ISO/IEC/IEEE 25010 Standard?

Solis and Wang (SOLIS; WANG, 2011) identified essential characteristics of BDD, highlighting scenario models and automated acceptance tests with mapping rules as preponderant elements. Such characteristics play a fundamental role in ensuring software quality. This study adopted a set of tools based on user stories and acceptance criteria, enabling test automation and monitoring of user perspectives. To this end, Locust was used to evaluate system performance and Jaeger to analyze communication between services, as illustrated in Figure 31.

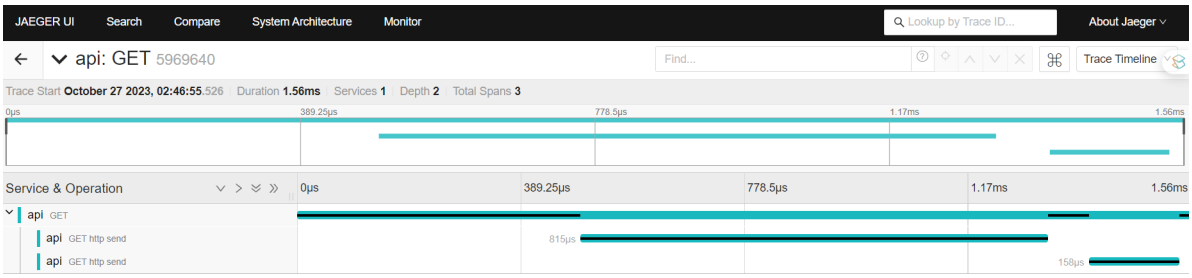

Figure 31 – Communication using Jaeger

Figure 31 presents Jaeger. This system shows communication between services, demonstrating performance and how each microservice communicates, allowing one to understand if there is an internal problem between services. This approach facilitates the work that links the need for performance that BDD pointed out and the possibility of verifying the behavior, not just of the response, but of the components linked to the response.

Figure 32 shows Grafana analyzing the containers of running services, allowing one to understand memory and processor consumption during tests, thus allowing one to cross-reference information from the point of view of the user with Locust, the service with Jaeger and the server with Grafana, having a 360 view of the system's performance.

The assessment of infrastructure components, monitored by Grafana, provided a comprehensive and end-to-end view of the system's performance.

4.3.3 QP3 - What are the benefits of using BDD in identifying and automating non-functional tests?

The application of BDD to ensure the quality of a system's characteristics offers a series of advantages, particularly highlighting the substantial improvement in communication between

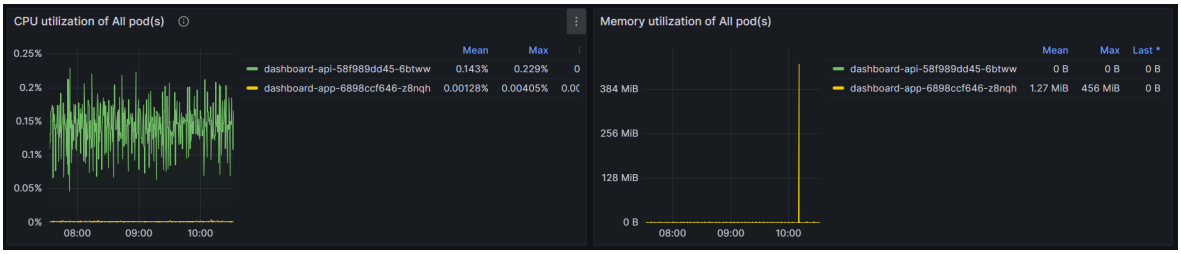

Figure 32 – Evaluation of infrastructure components

the architect and the product development team. Furthermore, its intrinsic integration with automation processes provides greater transparency and ease in validating the system, ensuring that it effectively fulfills the characteristics outlined during the design process.

As evidenced in Section 3.3, BDD becomes more transparent and accurate in requirements elicitation when using the gherkin language. In this way, the behaviors expected by the software achieved success in automating non-functional tests. Furthermore, as demonstrated in Figure 5, the importance of acceptance criteria meeting the tests to guarantee the quality initially proposed by eliciting the requirement becomes evident.

4.4 Discussion

The execution of these test scenarios made it possible to evaluate the system's performance concerning the sub-characteristics inherent to the performance efficiency characteristic of the ISO/IEC/IEEE 25010 Standard and ensure that the defined non-functional requirements were met. Thus, scripts were created for testing in Python with Locust.

As mentioned by Haoues et al. (HAOUES et al., 2017), to maintain software quality following ISO/IEC/IEEE 25010 there is a need to focus on the quality aspect throughout the entire software life cycle. In this way, BDD presented positive results from eliciting requirements, carried out objectively among team members, and automating tests to meet the proposed acceptance criteria.

According to Estdale and Georgiadou (ESTDALE; GEORGIADOU, 2018), non-functional requirements present improvements regarding the functioning of the software. As they are considered more complex, they are expected to have difficulty eliciting non-functional requirements. Through the gherkin language used by BDD, there was a better understanding of what to expect from the software's behavior so that user stories were described with a clear purpose.

According to Jarzkebowicz and Weichbroth (JARZĘBOWICZ; WEICHBROTH, 2021), it is necessary to understand how non-functional requirements are elicited. This study perceived the steps required for the elicitation of non-functional requirements, as shown in Figure 5, to realize the need to validate the requirement through user stories and their respective acceptance criteria to identify whether what was expected by the software was achieved or not.

Regarding the Systematic Literature Review with a focus on non-functional requirements carried out by Olsson, Sentilles, and Papatheocharous (OLSSON; SENTILLES; PAPATHEOCHAROUS, 2022), the authors observed the need to carry out studies that validate what has been researched in academia to deliver answers to assertions consistent with the reality of the industry. This research contributed to understanding the elicitation of non-functional requirements with BDD in a current situation.

As North (NORTH, 2006) mentioned, BDD was created to mitigate problems arising from Test-Driven Development (TDD), so BDD focuses on releases' behavior, while TDD focuses directly on testing. Thus, the author also states that BDD has the benefits of improving communication and readability, aspects perceived in this research in the elicitation of non-functional requirements so that there was the participation of the parts involved in the development of releases, such as was described in Section 3.3.

4.5 Experiment

The results related to each of the objectives outlined in Section 3.4 with the aim of answering their respective RQs outlined in the same Section are presented as follows in Figures 33 and 34.

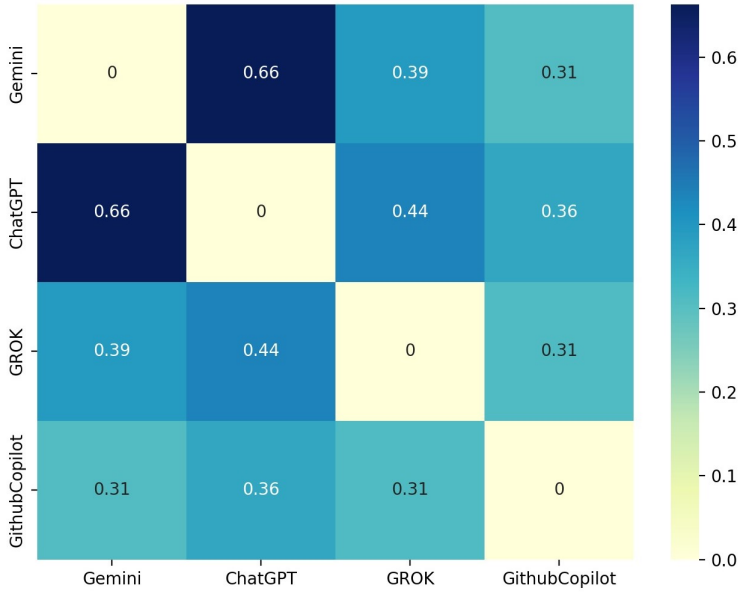

Figure 33 – Similarity Matrix

Figure 34, "A" refers to LLM Gemini, "B" refers to ChatGPT, "C" refers to GROK, and "D" refers to GitHub Copilot.

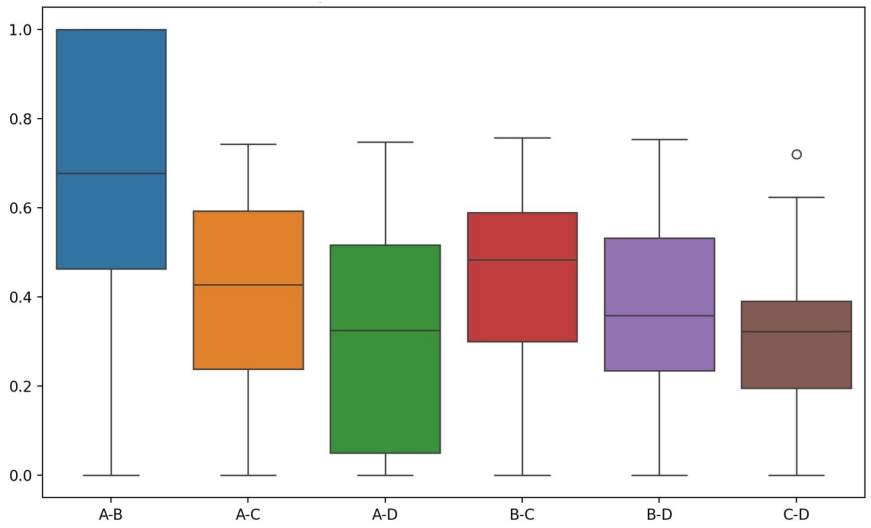

Figure 34 – Distribution of similarities

4.5.1 Objective 1

To measure the similarity of responses from different LLMs, the Kruskal-Wallis test was employed. This non-parametric test is appropriate for comparing independent distributions when the assumptions of normality are not met.

Results of the Kruskal-Wallis Test:

- Statistic: 36.2464
- p-Value: 0.0000

Results indicated a Kruskal-Wallis statistic of 36.2464 and a p-value of 0.0000. The extremely low p-value (less than 0.05) suggests that there is a statistically significant difference in the similarity of responses among the different LLMs.

Hypotheses:

- Null Hypothesis (H0.1): There is no significant difference in the similarity of responses among the different AIs.
- Alternative Hypothesis (H1.1): There is a significant difference in the similarity of responses among the different AIs.

Answering RQ1: Therefore, since the p-value is less than 0.05, it rejects the null hypothesis (H0.1). Thus, it supports the alternative hypothesis (H1.1), which asserts that there is a significant difference in the similarity of responses among the different LLMs. This implies that the LLMs Grok, Gemini, ChatGPT, and GitHub Copilot produce responses with statistically significant varying levels of similarity when generating automated tests based on user stories and acceptance criteria.

4.5.2 Objective 2

To validate if the results generated by the LLMs cover the acceptance criteria, a coverage analysis and an ANOVA test were conducted, followed by*post-hoc* and Tukey HSD.

Coverage Means:

- Grok: 0.4054
- Gemini: 0.5943
- ChatGPT: 0.7670
- GitHub Copilot: 0.7315

Coverage ANOVA:

- Sum of Squares (Model): 7.419452
- Sum of Squares (Residual): 13.830678
- F-Value: 65.089134
- p-Value: 1.025147e-33

ANOVA results indicated an F-value of 65.089134 and a p-value of 1.025147e-33, which is extremely low (less than 0.05). This suggests that there is a statistically significant difference in the coverage of acceptance criteria among the different LLMs.

Tukey HSD Test for Coverage:

- ChatGPT vs. GitHub Copilot: p = 0.6063 (not significant)
- ChatGPT vs. Gemini: p < 0.001 (significant)
- ChatGPT vs. Grok: p < 0.001 (significant)
- GitHub Copilot vs. Gemini: p < 0.001 (significant)
- GitHub Copilot vs. Grok: p < 0.001 (significant)
- Gemini vs. Grok: p < 0.001 (significant)

Results of the *post-hoc* Tukey HSD test showed that the differences in coverage of acceptance criteria are significant among most LLMs, except for ChatGPT and GitHub Copilot, whose differences were not significant (p = 0.6063).

Hypotheses:

- Null Hypothesis (H0.2): There is no significant difference in the coverage of acceptance criteria among the different AIs.
- Alternative Hypothesis (H1.2): There is a significant difference in the coverage of acceptance criteria among the different AIs.

Answering RQ2: Therefore, since the p-value of the ANOVA is less than 0.05, it rejects the null hypothesis (H0.2). Thus, it supports the alternative hypothesis (H1.2), which states that there is a significant difference in the coverage of acceptance criteria among the different AIs. The *post-hoc* tests indicate that, although ChatGPT and GitHub Copilot do not show significant differences between each other, all other comparisons between the LLMs are significantly different.

4.5.3 Objective 3

To assess the accuracy of tests generated by different LLMs, precision means were calculated and an analysis of variance (ANOVA) was conducted, followed by the Tukey HSD *post-hoc* test.

Precision Means:

- Grok: 0.3391
- Gemini: 0.5373
- ChatGPT: 0.7670
- GitHub Copilot: 0.7239

ANOVA of Precision:

- Sum of Squares (Model): 10.57519
- Sum of Squares (Residual): 12.50989
- F-Value: 102.56869
- P-Value: 3.882251e-48

Results of the ANOVA indicated an F-value of 102.56869 and a p-value of 3.882251e-48, which is extremely low (less than 0.05). This suggests that there is a statistically significant difference in the accuracy of tests generated by the different LLMs.

Tukey HSD Test of Precision:

- ChatGPT vs. GitHub Copilot: p = 0.3944 (not significant)

- ChatGPT vs. Gemini: $p < 0.001$ (significant)
- ChatGPT vs. Grok: $p < 0.001$ (significant)
- GitHub Copilot vs. Gemini: $p < 0.001$ (significant)
- GitHub Copilot vs. Grok: $p < 0.001$ (significant)
- Gemini vs. Grok: $p < 0.001$ (significant)

Results of the Tukey HSD *post-hoc* test showed that the differences in test accuracy are significant between ChatGPT, Gemini, and Grok. There were no significant differences in accuracy between GitHub Copilot and ChatGPT ($p = 0.3944$).

Hypotheses:

- Null Hypothesis (H0.3): There is no significant difference in the accuracy of tests generated among the different AIs
- Alternative Hypothesis (H1.3): There is a significant difference in the accuracy of tests generated among the different AIs.

Answering RQ3: Given that the p-value of the ANOVA is less than 0.05, it rejects the null hypothesis (H0.3). Therefore, it supports the alternative hypothesis (H1.3), which states a significant difference in the accuracy of tests generated by the different LLMs. Results of the *post-hoc* test indicate that ChatGPT has a significantly different accuracy compared to the other LLMs tested, except for GitHub Copilot. At the same time, the differences between GitHub Copilot, Gemini, and Grok are also statistically significant.

4.5.4 Objective 4

To assess the efficiency of the different LLMs in test generation, the mean generation times were calculated and an analysis of variance (ANOVA) was conducted, followed by the Tukey HSD *post-hoc* test.

ANOVA of Efficiency:

- Sum of Squares (Model): 0.712254
- Sum of Squares (Residual): 0.028181
- F-Value: 3066.558824
- P-Value: 6.633292e-258

ANOVA results indicate a significant difference between the groups, as the p-value is extremely low (6.633292e-258). This means that at least one of the models has a significantly different efficiency performance compared to the others.

Tukey HSD Test of Efficiency:

- ChatGPT vs. GitHub Copilot: p = 1.0 (not significant)
- ChatGPT vs. Gemini: p < 0.001 (significant)
- ChatGPT vs. Grok: p = 0.1858 (not significant)
- GitHub Copilot vs. Gemini: p < 0.001 (significant)
- GitHub Copilot vs. Grok: p = 0.1858 (not significant)
- Gemini vs. Grok: p < 0.001 (significant)

Significant Differences:

- ChatGPT vs. Gemini: The difference is significant, indicating that the generation time of ChatGPT is significantly shorter than that of Gemini.
- GitHub Copilot vs. Gemini: The difference is significant, indicating that the generation time of GitHub Copilot is significantly shorter than that of Gemini.
- Gemini vs. Grok: The difference is significant, indicating that the generation time of Gemini is significantly longer than that of Grok.

Non-Significant Differences:

- ChatGPT vs. GitHub Copilot: There is no significant difference, suggesting that the generation time of ChatGPT is similar to that of GitHub Copilot.
- ChatGPT vs. Grok: There is no significant difference, indicating that the generation time of ChatGPT is similar to that of Grok.
- GitHub Copilot vs. Grok: There is no significant difference, indicating that the generation time of GitHub Copilot is similar to that of Grok.

Interpretation:

- ChatGPT e GitHub Copilot: Both have comparable and efficient test generation times, with no significant differences between them.

- Gemini: The Gemini model exhibits significantly longer generation times compared to all other models, indicating inefficiency in its test generation process.
- Grok: The Grok model performs efficiently, similar to ChatGPT and GitHub Copilot, and significantly better than Gemini.

Results suggest that, in terms of test generation time efficiency, both ChatGPT and GitHub Copilot are effective and comparable. However, the Gemini model is significantly slower, indicating that it may not be the best choice when generation time is a critical factor. The Grok model also demonstrates efficiency and is comparable to ChatGPT and GitHub Copilot.

Hypotheses:

- Null Hypothesis (H0.4): There is no significant difference in the test generation time among the different AIs.
- Alternative Hypothesis (H1.4): There is a significant difference in the test generation time among the different AIs.

Answering RQ4: Given that the p-value of the ANOVA is less than 0.05, it rejects the null hypothesis (H0.4). Therefore, it supports the alternative hypothesis (H1.4), which states that there is a significant difference in the test generation time among the different AIs.

4.5.5 Objective 5

To assess the clarity of responses across different executions of each AI, clarity means were calculated and an analysis of variance (ANOVA) was conducted, followed by the Tukey HSD *post-hoc* test.

ANOVA of Clarity:

- Sum of Squares (Model): 145030.766421
- Sum of Squares (Residual): 124106.926135
- F-Value: 141.789559
- P-Value: 7.339233e-61

Results of the ANOVA indicate a significant difference in the clarity of responses among the different AIs, as the p-value is extremely low (7.339233e-61). This means that at least one of the AIs has significantly different clarity than the others.

Means of Clarity:

- Grok: 44.6511
- Gemini: 67.0604
- ChatGPT: 92.2609
- GitHub Copilot: 92.2609

Tukey HSD Test of Clarity:

- ChatGPT vs. GitHub Copilot: p = 1.0 (not significant)
- ChatGPT vs. Gemini: p < 0.001 (significant)
- ChatGPT vs. Grok: p < 0.001 (significant)
- GitHub Copilot vs. Gemini: p < 0.001 (significant)
- GitHub Copilot vs. Grok: p < 0.001 (significant)
- Gemini vs. Grok: p < 0.001 (significant)

Significant Differences:

- ChatGPT vs. Gemini: The difference is significant, indicating that the clarity of responses from ChatGPT is significantly higher than that of Gemini.
- ChatGPT vs. Grok: The difference is significant, indicating that the clarity of responses from ChatGPT is significantly higher than that of Grok.
- GitHub Copilot vs. Gemini: The difference is significant, indicating that the clarity of responses from GitHub Copilot is significantly higher than that of Gemini.
- GitHub Copilot vs. Grok: The difference is significant, indicating that the clarity of responses from GitHub Copilot is significantly higher than that of Grok.
- Gemini vs. Grok: The difference is significant, indicating that the clarity of responses from Gemini is significantly higher than that of Grok.

Non-Significant Differences:

- ChatGPT vs. GitHub Copilot: There is no significant difference, suggesting that the clarity of responses from ChatGPT is similar to that of GitHub Copilot.

Interpretation

- ChatGPT e GitHub Copilot: Both AIs have responses with comparable clarity and significantly higher than the other AIs evaluated.
- Gemini: The Gemini model exhibits intermediate response clarity, being significantly better than Grok but worse than ChatGPT and GitHub Copilot.
- Grok: The Grok model has the lowest response clarity among all evaluated AIs.

Thus, the results suggest that ChatGPT and GitHub Copilot are adequate and comparable in terms of response clarity. The Gemini model is intermediate, exhibiting significantly higher clarity than Grok but lower than ChatGPT and GitHub Copilot. The Grok model shows the lowest clarity among all AIs.

Hypotheses:

- Null Hypothesis (H0.5): There is no significant difference in the clarity of responses among the different AIs.
- Alternative Hypothesis (H1.5): There is a significant difference in the clarity of responses among the different AIs.

Answering RQ5: Since the p-value of the ANOVA is less than 0.05, it rejects the null hypothesis (H0.5). Therefore, it supports the alternative hypothesis (H1.5), which states a significant difference in the clarity of responses among the different AIs.

4.6 Discussion

Figure 35 was created using the model developed by Rajbhoj et al. (RAJBHOJ et al., 2024). The adaptation created for this study made it possible to follow a step-by-step method inherent to executing the automatic test generator created here.

There is initially a stakeholder who has a desire for a particular behaviour performed by the software. Thus, the user story initially emerges, which, in turn, leads to one or more acceptance criteria, scenarios testable that can guarantee the return desired by the software. From this point on, the programming language followed by the testing framework is selected, intending the standard prompt to be used in LLMs can be created, and finally, the automatic code can be generated.

Furthermore, this study allowed reviewing an integrated cycle for using LLMs associated with the BDD points. The LLMs are incorporated as feedback points after formalizing the user story and its acceptance criteria. During development, LLMs may be asked to review and improve the source code during the TDD refactoring stage. This cycle is illustrated in Figure 36.

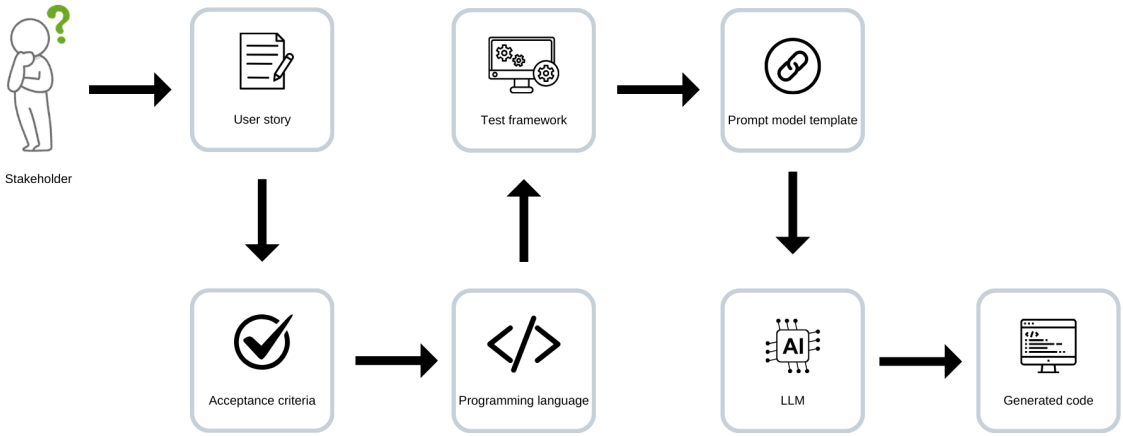

Figure 35 – Prompt model for BDD test automation

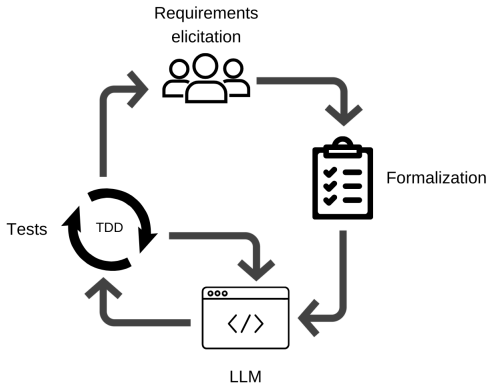

Figure 36 – Automatic code generation

In addition to the previous cycle, the work began with the formalization of user stories and their respective acceptance scenarios provided by the company. These stories and scenarios were integrated into the 4.1 prompt, making it possible to carry out the necessary tests in the LLMs Grok, Gemini, ChatGPT and GitHub Copilot, using the Python programming language to observe the similarity resulting from each one. Therefore, during the development cycle, after test generation, it was implemented a new interaction that can benefit the development process by using LLM, allowing LLM feedback during the TDD refactoring stage.

```
1 You are a test analyst, responsible for creating unit tests, based
    on the user story {us} and the acceptance criteria {ac} based on
    gherkin. Create unit tests for the programming language {pl}
    based on the {fw} framework and explain how to use the code.
```

Listing 4.1 – Prompt for the Tests Generation

The model generates the code. Otherwise, as TDD and BDD themselves highlight refactoring as a crucial factor, a cycle involving new requests to LLM may be necessary, automating the development, execution and refactoring process with the support of LLM.

Regarding accuracy, ChatGPT and GitHub Copilot performed the best, being very close to each other. This result is because GitHub Copilot uses parts of the ChatGPT model. On the other hand, Gemini and Grok had significantly lower accuracies, suggesting that the different models were more effective, in part due to the use of the free version but also because, in general, Gemini has difficulty delivering tests for three scenarios in one single command. Therefore, it is suggested submitting one scenario at a time during refactoring.

As for clarity, Grok had the worst performance, mainly because it often generates results in English, which is in line with the platform's main focus being the English language. Another point to consider is that Grok focuses more on conversation and research based on data from *Twitter*, not having code generation as its primary objective. In the free version, Gemini presents good clarity when analyzing a single acceptance criterion but has difficulty generating code for multiple scenarios. There is a need to evaluate the advanced version to see if this issue is resolved.

The results showed that ChatGPT and the development team were effective and comparable in terms of test generation time efficiency. The Gemini model, however, was significantly slower, indicating that it may not be the best choice when generation time is a critical factor. The Grok model proved efficient and comparable to ChatGPT and the development team.

It was observed that low-quality stories and scenarios negatively impact automatic code generation. This occurs due to ambiguities or unclear texts, making it difficult for LLMs to read and causing confusion in delivering the expected text. BDD was developed to be objective and concise; however, if a scenario is prepared with poor-quality writing, the return will certainly not be as expected.

The correct use of BDD and an LLM can benefit software development by helping developers automate test code. However, it is essential to emphasize that using this technology does not mean replacing the professional with the machine but instead taking advantage of existing technologies to assist in the necessary work.

4.7 Case Study - Guidelines

Through the studies carried out by the authors mentioned in Section 2.1, it was possible to outline guidelines for the adoption of Behavior-Driven Development (BDD). The guidelines were summarized in Figure 37.

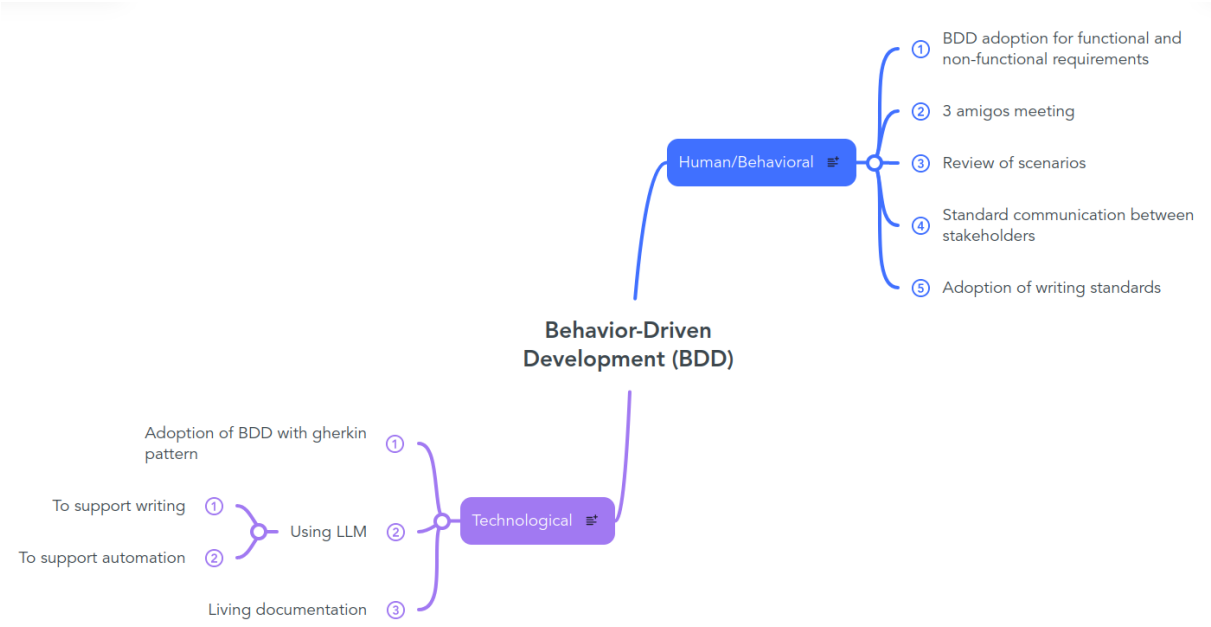

Figure 37 – Mind map for BDD adoption

The guidelines presented in Figure 37 for adopting BDD are divided into human/behavioral and technological. Regarding the human/behavioral aspect, it is possible to observe 5 points:

- Adoption of BDD for functional and non-functional requirements: since BDD is a framework used throughout the software life cycle, it is possible to adopt it both for eliciting functional and non-functional requirements;
- 3 amigos meeting: initial meeting held by the product owner, tester, and developer to idealize what is expected of the functionalities inherent to the software development;
- Scenario review: necessary review inherent to the software development process to see if what is being developed is following what is expected of the final product;
- Standardization of communication between stakeholders: use of everyday language without the need for jargon or technical terms;
- Adoption of writing standards: to maintain everyone's understanding of what is being discussed.

Regarding the technological aspects, 3 points are presented:

- Adoption of BDD with the Gherkin standard: to contribute to the standardization of the steps inherent to the use of BDD, that is, the terms "given, when, then" used in the requirements elicitation stage;

- Use of LLMs to support writing and automation: BDD can contribute to the refinement of the prompt to bring quality to the delivery of the output performed by the LLM. Part of the software development process was automated, an aspect that contributed to the adoption of BDD;
- Living documentation: documentation of the code performed concomitantly with its development so that it is not a stage performed only when the final product is delivered.

Figure 37 presents 13 points which were used for serving as the basis for the questionnaire applied to the target team of this study. Since the team had not used BDD before, creating a script to understand the topics presented in the questions was necessary. The script is expressed in Figure 38 and presents the components that it indicates as good practices for adopting BDD.

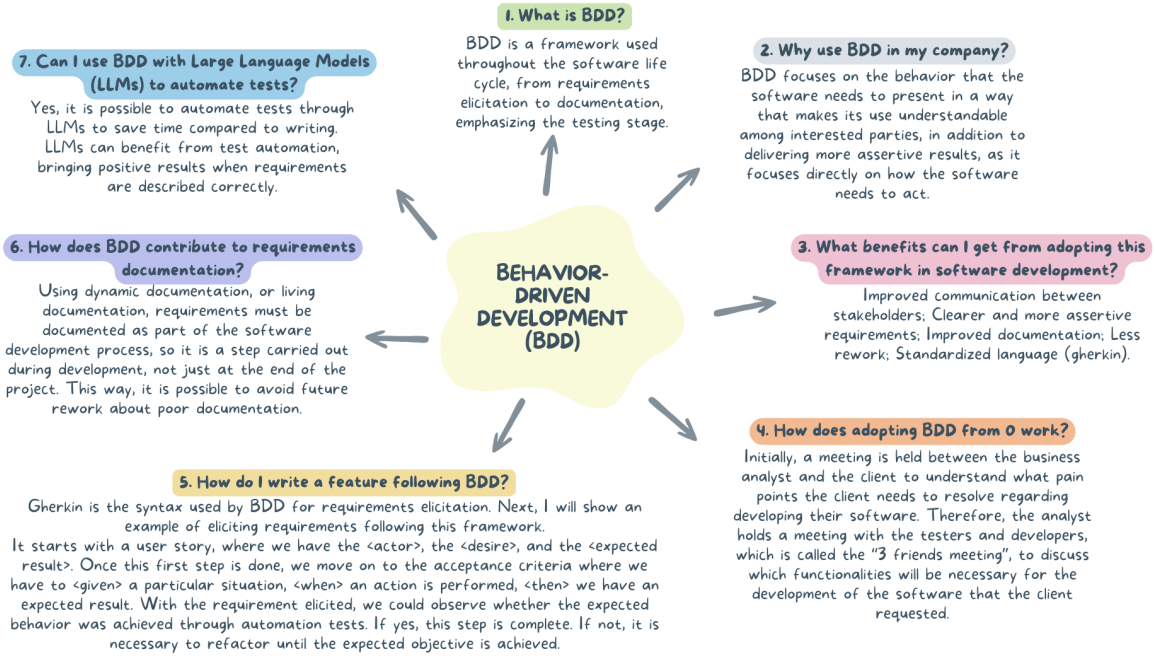

Figure 38 – Script for BDD adoption

The script guided respondents with basic terms in the context of BDD, such as gherkin language.

4.7.1 Survey Results

Responses collected in the survey are presented here[1]. Since the focus of this study was not to have a quantitative analysis, there is no need to include a graph for each research question since our target audience was only the 4 participants who answered the questionnaire. The email of one of the authors was made available in case any of the respondents had comments to add.

[1] https://abre.ai/liTx.

4.7.1.1 SQ1 - Did BDD manage to achieve the objective expected by the team?

The responses obtained were 50% for **yes**, 50% for **partially**, and no response for **no**. One of the respondents added the following comment on the use of BDD.

> *BDD was an effective tool for achieving the team's technical goals, such as increasing clarity in requirements and promoting the creation of automated tests based on use cases. However, a significant part of the goals were not achieved, and this was not due to the BDD methodology but rather to the lack of integration and collaboration between the development team and stakeholders. The lack of consistent alignment between both parties resulted in a lack of communication and compromised delivery of some expected benefits. Therefore, the problem is not with BDD but stakeholders' lack of support and active involvement throughout the process.*

The adoption of BDD itself is not a problem for teams that do not use it, but rather the collaboration between stakeholders so that it can directly affect the quality of the final product. Through the script presented in the Figure 38, it is possible, in theory, to achieve the expected objective from the software's behavior.

From questions SQ2 to SQ13, the response options, following the Linkert scale, were very bad, bad, good, very good, and excellent.

4.7.1.2 SQ2 - What is your perception regarding the adoption of BDD for software development?

Responses obtained were 50% for **excellent**, and 50% for **very good**. From the point of view of the 4 team leaders, the adoption of BDD occurs satisfactorily in the context of software development.

4.7.1.3 SQ3 - What is your perception regarding the adoption of BDD as technological support?

Responses obtained were 50% for **very good**, and 50% for **good**. It were able to ingest positive aspects in the adoption of BDD as technical support, given its adoption in a succinct and easy-to-understand manner among stakeholders.

4.7.1.4 SQ4 - What is your perception regarding the adoption of BDD with the gherkin pattern?

Responses obtained were 50% for **very good**, 25% for **excellent**, and 25% for **good**. Using the Gherkin pattern, it is possible to achieve the expected behavior of the software more directly than with traditional frameworks. Thus, using the Gherkin pattern can bring quality to

software development and save time since rework can be reduced, and the time spent redoing one step of the process can be used in other steps.

4.7.1.5 SQ5 - What is your perception regarding the adoption of BDD using Large-Language Models (LLM) to support writing?

Responses obtained were 50% for **very good**, and 50% for **good**. Using prompts written in a way that is appropriate to the desired behavior of the software, BDD can contribute to requirements elicitation and test automation. As shown in the Figure 37, BDD can enhance its capabilities using LLMs when used correctly.

4.7.1.6 SQ6 - What is your perception regarding the adoption of BDD using LLM to support test automation?

Responses obtained were 50% for **very good**, and 50% for **good**. Using BDD together with LLMs can contribute to the quality of the final product since the monotonous work of requirements elicitation, for example, can be automated through new or reused prompts for specific software behaviors.

4.7.1.7 SQ7 - What is your perception regarding the adoption of BDD in relation to living documentation?

Responses obtained were 50% for **very good**, 25% for **excellent**, and 25% **good**. Software documentation is an essential step, as any other step, in developing software, although it is often neglected. By having documentation carried out simultaneously with software development, there are benefits such as generating savings if it is necessary to return the software to a certain point where there was no bug, for example, because with documentation, it is known precisely what, why and how a particular step in the process was done.

4.7.1.8 SQ8 - What is your perception regarding the adoption of BDD in human/behavioral relationships?

Responses obtained were 50% for **good**, 25% for **very good**, and 25% **bad**. The need for stakeholder participation in software development is essential. For software to be developed correctly and to meet stakeholder requirements, stakeholders must be present at meetings required during the development process to validate whether what is being done is what stakeholders expect. When stakeholders neglect development, the possibility of delivering unsatisfactory software is greater than if they had participated during the process.

4.7.1.9 SQ9 - What is your perception regarding the adoption of BDD to elicit functional and non-functional requirements?

Responses obtained were 50% for **very good**, 25% for **excellent**, and 25% **good**. Non-functional requirements are more complex than functional requirements. The requirements elicitation stage needs to be well defined so that the software achieves the behaviors expected by stakeholders. In the elicitation of non-functional requirements, it is necessary to have clarity and conciseness in the requirements, given the need to translate them into testable scenarios. The gherkin language of BDD contributes to the required conciseness inherent in eliciting non-functional requirements.

4.7.1.10 SQ10 - What is your perception regarding the adoption of BDD for the "3 amigos" meeting?

Responses obtained were 75% for **good**, and 25% for **bad**. The "3 friends" meeting was one of the guidelines identified for adopting BDD. Although it is an essential step for its execution, it is vital that the design made by the product owner, developers, and testers is harmonious since if the meeting is not well done, this will reflect negatively on the subsequent stages of software development, and features may be built for pains that do not exist.

4.7.1.11 SQ11 - What is your perception regarding the adoption of BDD on scenarios review?

Responses obtained were 25% for **excellent**, 25% for **very good**, 25% for **good**, and 25% for **bad**. BDD uses user stories created with testable scenarios once requirements that can be implemented in the software are elicited. Reviewing these scenarios is objective to the proposed requirement since the Gherkin language is done directly. It is important to emphasize the need for adequate writing for the elicitation of requirements since the review of scenarios is an additional step to guarantee the final product's quality, and it is not necessary to carry out the requirements elicitation step in its entirety once again.

4.7.1.12 SQ12 - What is your perception regarding the adoption of BDD in relation to communication between stakeholders?

Responses obtained were 50% for **good**, 25% for **excellent**, and 25% for **bad**. For the final product to be high quality, all those involved must actively participate in developing the software. Suppose some parts do not understand their role in the process. In that case, the software will likely fail to achieve satisfactory quality since each stakeholder plays a different role. The communication proposed by BDD is that the process be carried out clearly with all stakeholders. If communication is flawed or presents ambiguities, the adoption of BDD is not being done correctly.

4.7.1.13 SQ13 - What is your perception regarding the adoption of BDD in relation to standardized writing?

Responses obtained were 75% for **excellent**, and 25% for **very good**. Standardized writing facilitates stakeholders' understanding during the software development process. The use of jargon, for example, can make it difficult for stakeholders from different fields to understand. BDD must be adopted using everyday language, without technical terms, so everyone can understand what a given part of the software needs to achieve, for example. Using the "given, when, then" pattern helps to assertively deliver the requirement so that it is outlined within a testable scenario.

4.7.2 Research questions

4.7.2.1 Q1 - What are the guidelines inherent to adopting BDD?

Guidelines for adopting BDD are presented in Figure 37. These guidelines were outlined through a Systematic Multivocal Literature Review, a Case Study, a Survey, and an Experiment. The adoption of BDD can contribute to the quality of the software as long as it is carried out correctly and follows the guidelines presented.

This work can guide companies that are not using BDD in their work activities to adopt it in the software development process. The Figure 38 presents the roadmap for adopting BDD and explains its step-by-step use.

4.7.2.2 Q2 - Did BDD achieve the expected purpose concerning software development?

As it can be seen in the survey results, BDD presents positive aspects in all steps of its implementation, seen by the team leaders as a framework that promotes quality for the final product. The point that all four respondents mentioned as unfavorable concerning the adoption of BDD was regarding the interaction between stakeholders, so the problem lies in something other than BDD itself, but rather in the communication between the partakers.

4.7.2.3 Q3 - What is the respondents perception that adopted BDD for this case study?

The survey showed positive results in all questions except for specific responses related to interpersonal relationships and scenarios review. Suppose BDD is not transparent to all stakeholders. In that case, the probability of the software being developed in a way that is not expected is greater than when the functionalities have precise purposes. An additional comment was made by one of the participants regarding the Meeting of the "3 Amigos".

> *We noticed a lack of genuine collaboration in the "3 Amigos" meeting, which was supposed to facilitate alignment between developers, testers, and business stakeholders. Although the physical presence of participants was guaranteed, active participation*

> *and engagement, especially from a business perspective, were insufficient. This created gaps in understanding between the development team and stakeholders, leaving some issues open and compromising the clarity needed for agile and well-directed development. In this format, the meeting could benefit from more detailed preparation and more proactive involvement of business representatives.*

Once again, the lack of collaboration from one of the parties was mentioned negatively concerning the adoption of BDD. Suppose the meeting of the "3 amigos" is not outlined correctly. In that case, there will likely be problems in the behaviors developed for the software since the functionalities were not elicited objectively.

Regarding scenarios review, this step can be seen as indispensable considering that the team already has experience in the requirements elicitation stage, for example. However, suppose the team trusts that the functionality is elicited correctly and skips this step. In that case, there may be a possibility of losing money due to the implementation of a poorly developed functionality, so it is better to invest a little more time in reviewing the scenarios rather than having to redo the functionality from scratch, in addition to wasting time with the elicitation of the correct functionality, as well as money and human capital. An additional comment was made by one of the participants regarding scenarios review.

> *During the scenario review, one challenge identified was the lack of involvement from external stakeholders in validating that the team was on track for all the proposed scenarios. While BDD made creating and automating tests easier based on these scenarios, the process could be improved by having more active stakeholder participation in the reviews. This collaboration would ensure that the scenarios were validated against business expectations and that feedback was provided continuously. Thus, BDD worked technically, but the whole value was not achieved due to the lack of consistent external validation.*

Therefore, the perception of team leaders regarding adopting BDD is positive. The problem mentioned only occurs concerning the communications required during the process.

4.7.2.4 Q4 - Are the guidelines presented effective for adopting BDD?

Yes, the guidelines presented are effective. Adopting BDD has proven effective for teams that are not using it in their software development activities. Because it has an everyday language, it has shown that when used correctly, as shown in the Figure, BDD contributes to the quality of the delivery of the final product.

The crucial point raised by all respondents was the lack of communication between stakeholders. One of the respondents made an additional comment.

> *BDD provides a good basis for improving communication between stakeholders, as it transforms business requirements into language accessible to both the development team and those involved. However, we realized that the process could be optimized using visual tools and a more interactive approach, simplifying communication and maintaining stakeholders' interest over time. For example, adopting diagrams or visual dashboards could facilitate the understanding of usage scenarios and promote more fluid and efficient interaction between stakeholders and the development team.*

It is interesting to highlight the recommendation of one of the respondents regarding using visual media for better collaboration among stakeholders, given the perceived failure in communication. The respondents viewed the guidelines positively, so the main obstacle was how stakeholders neglected communication during software development.

4.7.3 Discussion

Survey results showed the effectiveness of the guidelines presented for adopting BDD. When BDD is used in its entirety, without leaving out any essential step, such as the meeting of the "3 amigos", for example, there is a great chance of delivering a quality product in less time than expected.

As a framework used throughout the software life cycle, BDD can go through the steps of the software development process, from elicitation to documentation of requirements. It is important to realize how complete BDD is, with a standardized language that helps both in testing and in the validity and verification of features, assuming an essential role in software quality.

Ågren et al. (ÅGREN et al., 2019) stated the need for well-designed test cases. Through our case study, it was possible to outline a roadmap that can be used by companies seeking to adopt BDD in their work activities so that they can use this framework systematically using the guidelines related to its adoption.

Following the same idea by Bruschi et al. (BRUSCHI et al., 2019), Nascimento et al. (NASCIMENTO et al., 2020a), and Santos et al. (SANTOS et al., 2024), this study verified the effectiveness of BDD through a case study, in this case, the effectiveness of the guidelines that involve its adoption. There is a need to improve communication between stakeholders so that BDD can present the maximum positive results.

Binamungo and Maro (BINAMUNGU; MARO, 2023) sought to analyze the state of the art that BDD was in, while Kudo et al. (KUDO et al., 2023) showed the need for aligned writing standards for requirements and tests. Our study was able to advance the understanding of BDD regarding the visual need that can be implemented concerning stakeholders, as well as showing the benefits of standardized writing that BDD has.

Through the responses obtained, it was observed that the only point to be improved does

not concern BDD, and all four respondents stated this, but the need for collaboration on the part of stakeholders. There is no point in the team being competent in its activities if no stakeholders can validate what is being developed, so the chance of functionalities being developed correctly, but for problems that do not exist is high.

It is essential to highlight that the only negative point regarding adopting BDD is not directly linked to the framework but to the stakeholders who leave the development process at the mercy of the technical team. For BDD to be used with quality, all parties involved in the process must be aligned so that everyone commonly understands the objective. The suggestion of one of the respondents who mentioned using dashboards could be beneficial in this context, as it helps stakeholders understand.

5
Conclusion

Methodologies such as Systematic Literature Reviews are a good fit when there is a relatively new subject in which one seeks to systematize what exists published. A Multivocal Literature Review (MLR) was carried out to synthesize what has been researched and published in the white and gray literature regarding Behavior-Driven Development (BDD).

By characterizing BDD through the MLR presented in Section 3.1, it was possible to contribute to a better understanding of how to use this framework by industry and, on the other hand, how researchers could advance studies related to BDD. It could realize aspects of adopting BDD as mentioned in Section 4.1 through our eight research questions and answers.

Respect to the survey presented in Section 3.2, it was possible to observe the point of view of 43 professionals who use BDD in their work activities to characterize this framework, making it possible to advance the understanding regarding BDD and its strengths and weaknesses inherent to its adoption.

It was inferred that it was necessary to have experience using BDD to achieve the potential expected by the framework, so its adoption has benefits such as quality in delivery and readability. The lack of experience in using BDD directly impacts the performance of work activities, so the more experience is gained in using the framework, the fewer situations will be considered harmful regarding its adoption.

Regarding the case study presented in Section 3.3, the BDD framework was adopted for the elicitation of non-functional requirements and the ISO/IEC/IEEE 25010 Standard, precisely the performance efficiency characteristic with its three sub-characteristics, namely, time behavior, resource utilization, and capacity. Each subcharacteristics was associated with a user story, accompanied by one or more acceptance scenarios, as exemplified in Section 3.3.2. This approach made it possible to guide user testing, establishing links between efficiency characteristics and how requirements are being elicited through the BDD Gherkin language. Using BDD to align non-functional requirements is crucial for the company, as it facilitates product approval.

Furthermore, it fosters closer integration between the Quality team and the product since BDD requirements are directly linked to the product.

The use of BDD as a framework in the software life cycle leads to improved communication on the part of the members involved in the process to improve the software development stages, from requirements elicitation to documentation and maintenance. This research brought a holistic approach to the system's quality requirements by relating BDD to the ISO/IEC/IEEE 25010 Standard. Through this study, BDD showed its effectiveness following the performance efficiency characteristic expressed in the ISO/IEC/IEEE 25010 Standard as per Section 4.3.

Regarding the experiment presented in Section 3.4, the study adopted an strategy using statistical evaluation based on 34 user stories and a total of 94 acceptance scenarios to analyze the similarity between the responses, the coverage of the tests generated to the indicated scenarios, the accuracy of the tests, the efficiency to generation time and, finally, the clarity of responses.

Creating automated tests using LLMs through BDD has proven to be a relevant approach in software development. However, it was identified that faster LLMs currently do not provide satisfactory results in clarity and accuracy, which suggests that speed should not be the main criterion when choosing an LLM.

The LLMs used in the study could understand and generate natural language text with precision and quality based on well-described user stories and acceptance scenarios. This aspect allows software engineers and quality assurance teams to automate the creation of their tests based on BDD acceptance scenarios, using natural language descriptions, speeding up development and ensuring that tests more accurately reflect the precise requirements of the business.

About the last case study regarding guidelines for BDD adoption presented in Section 3.5, the main contribution was to show that even though BDD presents itself as a framework for effective use in software development if there is inadequate communication between the team and external stakeholders, the delivery of the final product will suffer negative consequences. Furthermore, the recommendation of one of the respondents to use diagrams and dashboards to help visualize external stakeholders is a crucial point to be mentioned, as it can lead to a significant improvement in the communication inherent to the process.

All parties involved in software development are essential, as each one plays a different role. If there is no client to verify the features being developed, the features will likely end up achieving purposes that do not exist.

Through the Design Science Research approach, it was possible to present guidelines for adopting BDD. To the scientific community, this work advances the study regarding BDD, because it presents guidelines for teams that do not use it in their work activities; to the industry, this work presents a roadmap for the adoption of BDD, assisting professionals who have not yet had contact with the framework.

As suggestions for future work, studies will be carried out with the BDD on the

EventStorming methodology so that it will be possible to create standard prompts for use in LLMs during the requirements elicitation stage. By automating the stage of eliciting functional and non-functional requirements, it may be possible to save time for developers, who can use their time with other steps inherent to software development, in addition to generating financial savings for the company.

5.1 Published papers

The papers published while studying for the master's degree are presented as follows. I, Shexmo, am the primary author of the papers highlighted in bold. Papers that are not highlighted are those in which I collaborated.

1. **Lei Geral de Proteção de Dados Pessoais (LGPD): Efetividade das ações desenvolvidas pela Universidade Federal de Sergipe (UFS)**[1]
2. **Identifying aspects related to Behavior-Driven Development (BDD) using a survey**[2]
3. Métricas de qualidade em sistemas críticos: um mapeamento sistemático[3]
4. **Adopting Behavior-Driven Development (BDD) in software development: a multivocal review**[4]
5. **Using Behavior-Driven Development (BDD) for non-functional requirements**[5]
6. Systematic Mapping: Microfrontend[6]
7. **Guidelines for the adoption of Behavior-Driven Development (BDD): An approach with Design Science Research**[7]
8. **Increasing Test Coverage by Automating BDD Tests in Proofs of Concepts (POCs) using LLM**[8]

5.2 Submitted papers

The articles to be published have already been finalized and await a decision by event/journal.

[1] https://e-revista.unioeste.br/index.php/expectativa/article/view/30653.
[2] Poster presentation for the 3rd Latin American School of Software Engineering (Latam) at the Brazilian Software Congress (CbSoft) 2023.
[3] https://revistacaribena.com/ojs/index.php/rccs/article/view/3587.
[4] https://ojs.sadio.org.ar/index.php/JAIIO/article/view/998.
[5] https://www.mdpi.com/2674-113X/3/3/14.
[6] https://www.insticc.org/node/TechnicalProgram/webist/2024/presentationDetails/130154.
[7] $https://sol.sbc.org.br/index.php/sbqs_estendido/article/view/31829$.
[8] DOI: 10.1145/3701625.3701637. (Awaiting publication.)

1. **Perception of professionals regarding the adoption of Behavior-Driven Development (BDD): a descriptive and statistical study through a Survey**
2. **Automated Test Generation Using LLM Based on BDD: A Comparative Study**[9]
3. **An experiment with focus on security through Large-Language Models using Behavior-Driven Development**
4. **Automated Test Generation Using LLM Based on BDD**[10]
5. **Guidelines for adopting Behavior-Driven Development (BDD): a case study**
6. Using Event Storming in Attack SUrface analysis: A report of experience for a microservices architecture system[11]
7. A Systematic Review and Survey of Metrics for Microservices
8. Software Architecture Patterns in Microservices: A Systematic Mapping of the Literature
9. Petri Nets with Time in the Past Two Decades: A Systematic Mapping Study
10. Requirements with Event Storm in Agile Teams : A case study considering remote work teams
11. Using Event Storming in Attack SUrface analysis: A report of experience for a microservices architecture system
12. Use Case - Catalog of Metrics Applied to Microservices
13. A Review on Microservices Metrics for Managing Software Development Teams
14. Using Event Storming in attack surface analysis: a report of experience for a microservices architecture system[12]

[9] Short paper.
[10] Short paper.
[11] Full paper.
[12] Full paper.

Bibliography

ÅGREN, S. M. et al. The impact of requirements on systems development speed: a multiple-case study in automotive. *Requirements Engineering*, Springer, v. 24, p. 315–340, 2019. Citado na página 71.

ARNYNDIASARI, D.; FERDIANA, R.; SANTOSA, P. I. Software practices for agile developers: A systematic literature review. In: IEEE. *2022 1st International Conference on Information System & Information Technology (ICISIT)*. [S.l.], 2022. p. 238–243. Citado na página 38.

BASILI, V. R. *Software modeling and measurement: the Goal/Question/Metric paradigm*. [S.l.], 1992. Citado na página 19.

BINAMUNGU, L. P.; EMBURY, S. M.; KONSTANTINOU, N. Maintaining Behaviour Driven-Development specifications: Challenges and opportunities. In: IEEE. *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. [S.l.], 2018. p. 175–184. Citado na página 16.

BINAMUNGU, L. P.; MARO, S. Behaviour driven development: A systematic mapping study. *Journal of Systems and Software*, v. 203, p. 111749, 2023. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121223001449>. Citado na página 71.

BOEHM, B. A view of 20th and 21st century software engineering. In: *Proceedings of the 28th international conference on Software engineering*. [S.l.: s.n.], 2006. p. 12–29. Citado 2 vezes nas páginas 14 and 22.

BROOKS, F. P.; BULLET, N. S. Essence and accidents of software engineering. *IEEE computer*, v. 20, n. 4, p. 10–19, 1987. Citado na página 13.

BRUSCHI, S. et al. Behavior Driven-Development (BDD): a case study in healthtech. In: *Pacific NW Software Quality Conference*. [S.l.: s.n.], 2019. Citado 3 vezes nas páginas 13, 16, and 71.

COUTO, T. et al. On the characterization of behavior-driven development adoption benefits: A multiple case study of novice software teams. *Anais do XXI Simpósio Brasileiro de Qualidade de Software, 2022, Brasil.*, 2022. Citado na página 16.

DEES, I.; WYNNE, M.; HELLESOY, A. *Cucumber Recipes: Automate Anything with BDD Tools and Techniques*. [S.l.]: Pragmatic Bookshelf, 2013. Citado na página 19.

EA, F. H. et al. Design science in digital innovation: A literature review. In: *XVI Brazilian Symposium on Information Systems*. [S.l.: s.n.], 2020. p. 1–7. Citado na página 17.

ESTDALE, J.; GEORGIADOU, E. Applying the ISO/IEC 25010 quality models to software product: 25th european conference, EuroSPI 2018, bilbao, spain, september 5-7, 2018, proceedings. In: ____. [S.l.: s.n.], 2018. p. 492–503. ISBN 978-3-319-97924-3. Citado na página 52.

FAROOQ, M. S. et al. Behavior driven development: A systematic literature review. *IEEE Access*, IEEE, 2023. Citado na página 38.

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In: *Proceedings of the 20th international conference on evaluation and assessment in software engineering*. [S.l.: s.n.], 2016. p. 1–6. Citado na página 19.

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology*, Elsevier, v. 106, p. 101–121, 2019. Citado na página 18.

GAROUSI, V.; MÄNTYLÄ, M. V. When and what to automate in software testing? a multi-vocal literature review. *Information and Software Technology*, Elsevier, v. 76, p. 92–117, 2016. Citado na página 18.

GAROUSI, V.; ZHI, J. A survey of software testing practices in canada. *Journal of Systems and Software*, Elsevier, v. 86, n. 5, p. 1354–1376, 2013. Citado na página 38.

GUERRA-GARCIA, C. et al. Iso/iec 25012-based methodology for managing data quality requirements in the development of information systems: Towards data quality by design. *Data and Knowledge Engineering*, v. 145, p. 102152–102152, 2023. Citado na página 16.

HAOUES, M. et al. A guideline for software architecture selection based on ISO 25010 quality related characteristics. *International Journal of System Assurance Engineering and Management*, Springer, v. 8, p. 886–909, 2017. Citado na página 52.

HEVNER, A. et al. Design science research in information systems. *Design research in information systems: theory and practice*, Springer, p. 9–22, 2010. Citado na página 17.

JARZĘBOWICZ, A.; WEICHBROTH, P. A qualitative study on non-functional requirements in agile software development. *IEEE Access*, IEEE, v. 9, p. 40458–40475, 2021. Citado na página 52.

KARAGÖZ, G.; SÖZER, H. Reproducing failures based on semiformal failure scenario descriptions. *Software Quality Journal*, Springer, v. 25, p. 111–129, 2017. Citado na página 25.

KITCHENHAM, B.; PFLEEGER, S. L. Principles of survey research: part 5: populations and samples. *ACM SIGSOFT Software Engineering Notes*, ACM New York, NY, USA, v. 27, n. 5, p. 17–20, 2002. Citado na página 22.

KRUCHTEN, P. What do software architects really do? *Journal of Systems and Software*, Elsevier, v. 81, n. 12, p. 2413–2416, 2008. Citado na página 13.

KUDO, T. N. et al. Aligning requirements and testing through metamodeling and patterns: design and evaluation. *Requirements Engineering*, Springer, v. 28, n. 1, p. 97–115, 2023. Citado na página 71.

MASHIKO, Y.; BASILI, V. R. Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement. *Journal of Systems and Software*, Elsevier, v. 36, n. 1, p. 17–32, 1997. Citado na página 22.

MELEGATI, J.; WANG, X. Case survey studies in software engineering research. In: *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. [S.l.: s.n.], 2020. p. 1–12. Citado na página 24.

MOE, M. M. Comparative study of test-driven development tdd, behavior-driven development bdd and acceptance test–driven development atdd. *International Journal of Trend in Scientific Research and Development*, v. 3, p. 231–234, 2019. Citado na página 16.

NASCIMENTO, N. et al. 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*. [S.l.: s.n.], 2020. p. 109–116. Citado 2 vezes nas páginas 39 and 71.

NASCIMENTO, N. et al. Behavior-Driven Development: an expert panel to evaluate benefits and challenges. In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. [S.l.: s.n.], 2020. p. 41–46. Citado na página 16.

NORTH, D. *Introducing BDD*. 2006. Citado 7 vezes nas páginas 13, 15, 36, 37, 38, 41, and 53.

NORTH, D. *JBehave. A framework for behaviour driven development (BDD)*. 2012. Citado na página 19.

OLSSON, T.; SENTILLES, S.; PAPATHEOCHAROUS, E. A systematic literature review of empirical research on quality requirements. *Requirements Engineering*, Springer, v. 27, n. 2, p. 249–271, 2022. Citado na página 53.

PAI, M. et al. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. *The National medical journal of India*, v. 17, n. 2, p. 86–95, 2004. Citado na página 19.

PEREIRA, L. et al. Behavior-Driven Development benefits and challenges: reports from an industrial study. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*. [S.l.: s.n.], 2018. p. 1–4. Citado 2 vezes nas páginas 15 and 16.

PERRY, D. E.; SIM, S. E.; EASTERBROOK, S. Case studies for software engineers. In: *Proceedings of the 28th international conference on software engineering*. [S.l.: s.n.], 2006. p. 1045–1046. Citado 2 vezes nas páginas 24 and 31.

PETERSEN, K. Guidelines for case survey research in software engineering. *Contemporary empirical methods in software engineering*, Springer, p. 63–92, 2020. Citado na página 24.

PFLEEGER, S. L.; KITCHENHAM, B. A. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes*, ACM New York, NY, USA, v. 26, n. 6, p. 16–18, 2001. Citado na página 22.

PUNTER, T. et al. Conducting on-line surveys in software engineering. In: IEEE. *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* [S.l.], 2003. p. 80–88. Citado 2 vezes nas páginas 22 and 32.

PURKAYASTHA, S. et al. Continuous security through integration testing in an electronic health records system. In: IEEE. *2020 International Conference on Software Security and Assurance (ICSSA)*. [S.l.], 2020. p. 26–31. Citado na página 37.

RAJBHOJ, A. et al. Accelerating software development using generative ai: Chatgpt case study. In: *Proceedings of the 17th Innovations in Software Engineering Conference*. [S.l.: s.n.], 2024. p. 1–11. Citado na página 61.

RAUF, A.; ALGHAFEES, M. Gap analysis between state of practice and state of art practices in agile software development. In: IEEE. *2015 agile conference*. [S.l.], 2015. p. 102–106. Citado na página 49.

RODRÍGUEZ, G.; GONZÁLEZ-CAINO, P. C.; RESETT, S. Serious games for teaching agile methods: A review of multivocal literature. *Computer Applications in Engineering Education*, Wiley Online Library, v. 29, n. 6, p. 1931–1949, 2021. Citado na página 18.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, Springer, v. 14, p. 131–164, 2009. Citado 2 vezes nas páginas 24 and 31.

SANTOS, S. et al. Using behavior-driven development (bdd) for non-functional requirements. *Software*, v. 3, n. 3, p. 271–283, 2024. ISSN 2674-113X. Disponível em: <https://www.mdpi.com/2674-113X/3/3/14>. Citado 2 vezes nas páginas 18 and 71.

SANTOS, S. Ribeiro dos; RODRIGUEZ, G.; ROCHA, F. G. Adopting behavior-driven development (bdd) in software development: a multivocal review. *Memorias de las JAIIO*, v. 10, n. 2, p. 14–27, sep. 2024. Disponível em: <https://publicaciones.sadio.org.ar/index.php/JAIIO/article/view/998>. Citado na página 18.

SCANDAROLI, A. et al. Behavior-driven development as an approach to improve software quality and communication across remote business stakeholders, developers and qa: two case studies. In: IEEE. *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. [S.l.], 2019. p. 105–110. Citado na página 38.

SHAW, M. What makes good research in software engineering? *International Journal on Software Tools for Technology Transfer*, Springer, v. 4, p. 1–7, 2002. Citado na página 14.

SHAW, M. Writing good software engineering research papers. In: *25th International Conference on Software Engineering, 2003. Proceedings.* [S.l.: s.n.], 2003. p. 726–736. Citado 3 vezes nas páginas 14, 22, and 34.

SILVA, T. R.; FITZGERALD, B. Empirical findings on BDD story parsing to support consistency assurance between requirements and artifacts. In: *Evaluation and Assessment in Software Engineering*. [S.l.: s.n.], 2021. p. 266–271. Citado 3 vezes nas páginas 15, 16, and 46.

SMART, J. F.; MOLAK, J. *BDD in Action: Behavior-driven development for the whole software lifecycle*. [S.l.]: Simon and Schuster, 2023. Citado 2 vezes nas páginas 6 and 16.

SOLIS, C.; WANG, X. A study of the characteristics of behaviour driven development. In: IEEE. *2011 37th EUROMICRO conference on software engineering and advanced applications*. [S.l.], 2011. p. 383–387. Citado 2 vezes nas páginas 38 and 51.

WOHLIN, C.; HÖST, M.; HENNINGSSON, K. Empirical research methods in software engineering. *Empirical methods and studies in software engineering: Experiences from ESERNET*, Springer, p. 7–23, 2003. Citado 2 vezes nas páginas 24 and 28.

WOHLIN, C.; HÖST, M.; HENNINGSSON, K. Empirical research methods in web and software engineering. *Web engineering*, Springer, p. 409–430, 2006. Citado na página 32.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012. Citado 2 vezes nas páginas 24 and 31.

YANG, A. Z.; COSTA, D. A. da; ZOU, Y. Predicting co-changes between functionality specifications and source code in behavior driven development. In: IEEE. *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. [S.l.], 2019. p. 534–544. Citado na página 38.

YIN, R. K. Case study methods. In: . [S.l.]: American Psychological Association, 2012. Citado na página 24.