IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 July 2023, accepted 25 July 2023, date of publication 7 August 2023, date of current version 23 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3302356

RESEARCH ARTICLE

Behavior Driven Development: A Systematic Literature Review

MUHAMMAD SHOAIB FAROOQ[1], UZMA OMER[2], AMNA RAMZAN[3], MANSOOR AHMAD RASHEED[3], AND ZABIHULLAH ATAL[4]

[1]Department of Artificial Intelligence, School of System and Technology, University of Management and Technology, Lahore 54000, Pakistan
[2]Department of Information Science, Division of Science and Technology, University of Education, Lahore 54000, Pakistan
[3]Department of Computer Science, School of System and Technology, University of Management and Technology, Lahore 54000, Pakistan
[4]Department of Computer Science, Kardan University, Kabul 1007, Afghanistan

Corresponding author: Zabihullah Atal (z.atal@kardan.edu.af)

ABSTRACT Behavior Driven Development (BDD) is a widely adopted agile methodology for software development that emphasizes the behavior of an application as a series of test cases, using the keywords, which include "Given," "When," and "Then." It involves writing requirements in a structured and testable format that can be evaluated to ensure compliance with the expected behavior. Although a significant amount of research has been conducted to examine the impact of using BDD on software development process yet rare work is observed to synthesize these studies and identify areas for future exploration. This study presents a review of the state-of-the-art BDD by synthesizing the recent advancements in its uses and applications. It aims to systematically investigate the impact of BDD on software development process as well as on product quality by aiding to bridge the communication gap between the stakeholders. The results reveal that BDD is an effective technique to clarify requirements during the software development process as it helps minimizing the intrinsic ambiguities. This work proposes a taxonomy based on the role and applications of BDD in various contexts. It suggests a framework for applying BDD in software development and defines a workflow for its application in software development. Finally, this work highlights some pertinent future directions for the use of BDD in software development.

INDEX TERMS Behavior-driven development, software testing, agile methodology, ubiquitous language, automation testing, test driven development.

I. INTRODUCTION

Behavior Driven Development (BDD) is a method of software development that involves collaboration among both technical and non-technical team members working on a software product or project [1]. BDD is based on the approaches used in Test-Driven Development (TDD) [2]; however, it operates at a higher level than TDD and can be considered as a refinement of TDD as it shifts the focus from testing to the identification of the expected behaviors of a system. Furthermore, it is considered a valuable approach for managing client requirements and scenarios, which are expressed in the form of test scenarios. Test scenarios that are derived through requirements play very important role for verbal interactions between the agile team and end-user. In BDD, a series of inputs is provided to interact with application or software system written in a form of plain language phrases organized in defined pattern "Given-When-Then" [3], depicting the details as:

- **Given** - shows the initial context.
- **When** - presents an occurring of an event.
- **Then** - demonstrates a promise of an outcome as expected.

BDD scripting utilizes a pattern that facilitates the automation of testing. Traditional methods of clarifying requirements or reducing ambiguities often resulted in miscommunication between project stakeholders and the technical team, making direct communication with stakeholders difficult [4]. BDD addresses this issue by incorporating a narrative collaboration between all team members and the client. The three objectives

The associate editor coordinating the review of this manuscript and approving it for publication was Taehong Kim.

of BDD are events, outcomes and context. Events refer to the actions that take place to achieve the final outcome, which is the expected result. In other words, an event is an action a user performs, and an outcome is the expected result that will be obtained from the action. The primary goal of BDD is to determine the set of behaviors that the system can expect from the user. BDD tests can be written and executed at any stage of the development process, whether it be before, during or after the development process. This results in a specification that is easily understandable by the consumers. The examples in BDD are also executable due to the ''glue code'' which connects plain human-readable language statements to a test which is a piece of code. This not only defines the program's requirements but also serves as an acceptance test for comparing the implementation to the specification.

One major challenge for software developers is determining the starting point or deciding which aspects should be tested and which should be omitted. It has been observed that the medium or language used to express tests, such as class and operation names, plays a crucial role in both creating test cases and identifying issues in the event of a failed test. BDD utilizes natural language as a universal mode of communication to describe the expected outcomes using scenarios in a test. This process of mapping sentences to code based on scenarios is usually done manually, which is time-consuming and prone to errors. The data collection is necessary to execute this process on a basic level previously given in the natural description. The purpose of BDD is to retrieve a system specification as output which is executable [5]. Six major features of BDD were identified in [6], which include iterative decay method, client story, and the templates of the scenarios. Additionally, automatic testing, evaluation acceptance with planning rules, clear behavior code, and behavior-driven at various stages are supported by various toolkits, including JBehave [7], Cucumber [8], and RSpec [9]. However, the BDD technique is still in its early stages. Similarly, as many other agile strategies, the adoption and consistent utilization of BDD is directly associated with organizations, individuals, process, and technical aspects [10], [11]. Despite notable research is produced to examine BDD impact on software development process, efforts are required to synthesize these research studies in order to examine whether and how BDD supports the software development process and how its applications in software development can be improved. To fill the gaps in existing BDD research, this work provides synthesis of the BDD literature exhibits evidence-based insights into its impact on software development process. The main objective of this study is to synthesize the state-of-the-art BDD and examine the applicability of this technique in software development process. It further aims to identify any areas that lead to improve existing approaches of BDD and determine the future prospects of BDD in software development. This study will be useful for practitioners and researchers to incorporate the effective practices during the software development process and identify areas for further exploration of BDD.

This article presents a thorough and comprehensive review of the latest advancements in BDD and evaluate its potential to improve various issues in software development process such as ambiguities in requirements resulting from misunderstandings between business stakeholders and the development team. This study presents a BDD taxonomy to assist practitioners and researchers in identifying various techniques for evaluating software behavior. Additionally, a framework is proposed to highlight the stage wise explanation of the key aspects of applying BDD during the software development process. Furthermore, a workflow for application of BDD in software industry has been proposed for the explanation of overall working of the proposed framework. Finally, the problems and research gaps have been identified to guide future research of the application of BDD during the software development process.

The structure of this article is as follows: Section II presents a review of the related work. Section III outlines the methodology for conducting this work by presenting the study objectives, research questions, the strategy for identifying relevant literature, the process for selecting studies, and the criteria used to assess the quality of the studies. Also demonstrates the findings obtained in response to investigating the research questions. Section IV shows an in-depth discussion and analysis of the research discoveries. Section V represent the limitations. Finally, Section VI presents the conclusion of this work.

II. RELATED WORK

BDD has been shown to be an effective method for addressing the disconnect between business stakeholders and developers. It has been used by the software development teams to improve the quality of software development processes and the quality of the final product. Several studies have been conducted to evaluate the effectiveness of BDD. Some of these studies exhibit systematic evaluations, which aimed to examine the techniques, methods, and tools associated with BDD, as well as any problems that may arise when using this approach. Additionally, these evaluations sought to determine the most appropriate workflow for implementing BDD in software development. Vijaysarathy et al. investigated the justifications for why individuals and associations accept are implement agile methods, as well as the advantages and issues that improvement groups experience in the beginning stages of adopting specific agile methodologies [11]. In other research, Senpathi et al. utilized the Agile Usage Model to decide the factors that influence the effective application of agile methodologies after they have been acknowledged by organizations, as well as the effect of acceptance [12]. In [13], few descriptions of the organizational, individuals, process, and technological viewpoints that impact the adoption of agile approaches can be found.

In [14], Binamungu et al. investigated the extent to which the market utilizes BDD and its associated benefits, as well as the common and specific challenges faced, particularly in the duplication of material. They proposed ten different

options for researchers to aid in re-evaluating and modifying their BDD practices using both quantitative and qualitative data collected via a web survey. The study primarily focuses on the opportunities and level of BDD usage. However, it does not discuss the limitations of the approach. Another study [6] conducted by Solis and Wang analyzed the features of BDD, the accompanying tool support, and the weaknesses and strengths of current applications as previously reported by Okolnychyi and Fogen in [7]. However, the study does not explore other important tools and their strengths and weaknesses. Rahman and Gao discussed the restructuring of issues that may arise when adapting BDD tests to multiple configurations in [15]. Rai [16] evaluated the effectiveness of the BDD technique in practice, as well as the ease with which individuals can acquire and understand Gherkin, the most popular medium of communication in BDD. However, the study does not focus on its impact on development. The improvement of time, cost, communication, and software quality through the use of BDD was discussed in [17]. However, it failed to address the weaknesses and issues where the targeted approach was not effective or where the approach did not resolve the problem.

Previous research has presented four principles for determining the quality of BDD suites, as surveyed from 75% of practitioners in [1]. However, these studies did not provide a comprehensive identification of BDD techniques and impacts through any taxonomy to assist researchers and developers in their respective domains. Additionally, it has been shown that BDD and TDD practices, when compared to TDD, BDD can be costlier in terms of time and money. However, BDD is generally more effective in terms of user acceptance, as opposed to TDD [18]. A case study presented in [19] demonstrated that, compared to the Scrum approach used in the EAMS-CBALM development, the BDD process improved communication between the product owner and developers or designers throughout the development process, while also maintaining a normal level of communication targeting the education sector, through the use of acceptance tests and BDD scenarios. Despite these improvements, there remains potential for further enhancement of such approaches.

This study contributes to the field of software development by presenting a review of the state-of-the-art BDD and assesses its potential for addressing critical issues, such as requirements ambiguities resulting from misunderstandings between business stakeholders and the development team. A taxonomy is presented to assist practitioners and researchers in identifying various aspects of BDD. Furthermore, a workflow is proposed to illustrate the application of BDD in the software industry, offering a comprehensive overview of how the proposed framework can be implemented in practice. By providing step-by-step guidance, this workflow aids in understanding the overall working and benefits of BDD in software development. The problems and gaps in current approaches have also been highlighted to guide future research directions for the application of BDD in software development.

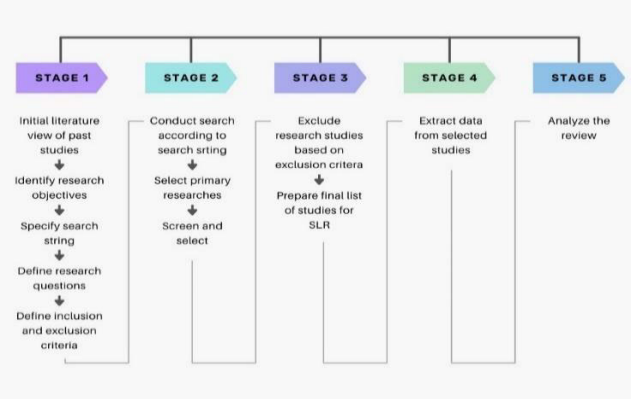

FIGURE 1. SLR process model.

III. RESEARCH METHODOLOGY

This study mainly follows the widely established Systematic Literature Review (SLR) technique outlined in [20] and the techniques observed in [21]. An SLR is an approach that includes the systematic collection and analysis of literature in a particular field. Finding significant research gaps and overlooked areas that demand more research is the primary objective. It can be used to determine future research priorities and to identify areas where more research is needed. Conducting an SLR requires a significant amount of time and effort, but a rigorous methodology can help ensure that the review is comprehensive.

This study followed a multi-stage methodology of conducting the review. In Stage 1, a comprehensive review of previous studies was conducted to gain an understanding of the existing literature in the field. This helped to identify the research objective and formulate a specific research question. A search string was developed, and inclusion and exclusion criteria were established. Stage 2 involved an extensive search using the defined search string across relevant databases followed by the selection of primary research studies that met the criteria outlined in the inclusion and exclusion criteria and screening out irrelevant ones. In Stage 3, studies were further excluded based on the predefined exclusion criteria, resulting in a final list of studies for the SLR. The Stage 4 concentrated on data extraction from the chosen research, in which important information and key findings were identified. Finally, in Stage 5, the selected data after screening were considered to determine the synthesis of findings and to address the research question, and also highlight relevant findings as shown in Figure 1.

A. RESEARCH OBJECTIVES (RO)

Following are the main objectives of this research study:

1. To identify state-of-the-art BDD techniques and approaches.
2. To determine the role of BDD in solving problems discovered in communication between business stakeholders.

TABLE 1. Research questions and major motivations.

	Research Question	Major Motivation
RQ1	What techniques and methods are used to reduce intrinsic ambiguity and communication gap between the members of agile team?	To find the solution of the communication gaps between the business people and stakeholders .
RQ2	What are the BDD impacts on software development stages?	To identify the BDD impact on software development process
RQ3	What challenges and opportunities have been reported to use BDD in software development?	To identify major opportunities and problems of using BDD approach

3. To identify BDD impact on software development process.

B. RESEARCH QUESTIONS (RQ)

The primary objective of this work is to explore the state-of-the-art BDD in order to identify the future research dimensions in the specified area. Considering the major objective of this study, the research questions to investigate the specific aspects of BDD are devised. Table 1 presents the research questions evaluated in this review as well as their major motivation.

C. SEARCH SCHEME

The development of a comprehensive search plan to accurately identify and retrieve relevant articles within the nominated subject is a crucial aspect of an SLR. This process encompasses the formulation of a search string, the utilization of bibliographic databases to execute the search, and the implementation of a screening plan to recover highly pertinent articles from the collection. A portion of the selected materials were subjectively and critically evaluated to address multiple perspectives on the subject.

1) SEARCH STRING

An in-depth study was conducted to formulate a search string that would aid in the retrieval of relevant studies from digital repositories. The research began with an initial literature review on the subject. To select all potential systematic literature review (SLR) research studies, a search string comprising of primary, secondary, and additional keywords was employed. Google Scholar has been demonstrated to be an efficient tool for conducting bibliometric research [22]. To perform the automated search of relevant literature, the following search string was utilized:

((Behavior-Driven Development'' OR ''BDD'') AND (''tool'' OR ''software'' OR ''requirement'' OR ''agile'' OR ''process'' OR ''automation'' OR ''framework'' OR ''approach'' OR ''specification'') NOT (''binary decision diagram''))

To construct an effective search string, logical operators such as ''AND'' and ''OR'' were used to combine the final keywords and alternative terms. The wildcard operator ''*'' was also employed to represent zero or more characters as necessary. However, if no additional or tertiary keywords were required in the string, the use of the wildcard operator was unnecessary. The utilization of the ''OR'' operator enabled a broader range of search options, while the ''AND'' operator linked phrases to depict search alternatives and narrowed the query to yield relevant search results.

TABLE 2. Applied search strings on different repositories.

Repository	*Search String*
Springer Link	*("BEHAVIOR-DRIVEN DEVELOPMENT") AND ("AUTOMATION" OR "SOFTWARE" OR "FRAMEWORK" OR "SPECIFICATION" OR "AGILE" OR "PROCESS" OR "REQUIREMENT" OR "APPROACH") AND NOT ("BINARY DECISION DIAGRAM")*
Science Direct	*((BEHAVIOR-DRIVEN DEVELOPMENT" OR "BDD") AND ("SOFTWARE" OR "REQUIREMENT" OR "AGILE" OR "AUTOMATION" OR "FRAMEWORK" OR "SPECIFICATION") NOT ("BINARY DECISION DIAGRAM"))*
IEEE Xplore	*("ALL METADATA":BEHAVIOR-DRIVEN DEVELOPMENT OR "ALL METADATA":BDD) AND ("ALL METADATA":TOOL OR "ALL METADATA":SOFTWARE OR "ALL METADATA":REQUIREMENT OR "ALL METADATA":PROCESS OR "ALL METADATA":SPECIFICATION OR "ALL METADATA":AUTOMATION OR "ALL METADATA":FRAMEWORK OR "ALL MATADATA":AGILE OR "ALL METADATA":APPROACH) NOT ("ALL METADATA":BINARY DECISION DIAGRAM)*
ACM Digital Library	*"BEHAVIOR-DRIVEN DEVELOPMENT" AND ("TOOL" OR "SOFTWARE" OR "REQUIREMENT" OR "PROCESS" OR "SPECIFICATION" OR "AUTOMATION" OR "FRAMEWORK" OR "AGILE" OR "APPROACH") AND NOT "BINARY DECISION DIAGRAM"*

2) LITERATURE RESOURCES

The most prominent and domain related literature resources have been considered to search the research articles for carrying out this review. Details of the selected sources and search phrases applied are listed in Table 2.

3) INCLUSION AND EXCLUSION CREITERIA

The method of collecting data for this study was designed to identify articles that were relevant to the objective of this research. When the same material appeared in multiple publications according to our search, it was only included once. The initial task was to classify the research from each source using the specified search string. Articles with matching titles or those that were not directly related to the review were excluded.

IC1: Studies that examine the role of BDD in addressing communication problems between business stakeholders during software development.

IC2: Studies that investigate the impact of BDD on the software development process.

IC3: Studies that are focused on the applicability of BDD in software industry.

EC1: Duplicate paper.

EC2: Studies that are not written in English language.

EC3: Studies that are not related to the domain of software development.

4) SELECTION OF RELEVANT PAPERS

The primary search approach produces a large number of research publications, not all of which are directly according to the set questions designed for the research, and there is also duplication. As a result, the searched articles must be re-evaluated and screened in order to obtain genuinely essential publications. The approach given in [23] and [24] was used to determine the relevance and screening of publications.

The first step is to classify the studies by title and remove duplicates. There were a number of articles available that were irrelevant to the specified topic, thus they were screened by the title and inappropriate research articles was excluded. Finally, by observing the specified approach, the selected articles are screened for following stage of the assessment.

5) ABSTRACT BASED KEYWORDING

The screening of articles was also performed by applying the two-stage abstract-based key wording approach given in [25]. The abstract was at first analyzed to decide the fundamental part of the paper, its commitment to the point, and the most significant keywords. The keywords found from numerous papers are then converged through which a conceivable comprehension of the commitment of the review has been created. At last, these keywords were utilized to identify the articles for review processing.

6) QUALITY ASSESSMENT CRITERIA

The quality assessment of shortlisted studies is an essential step in conducting an SLR. The main objective of carrying out an assessment of quality in this study aimed to ensure that the studies are relevant to the research area and produced through well-reputed research venues. A systematic assessment of the standard of the nominated research is a significant obligation of a review. As the strategies of these investigations varied, the worth of these studies was evaluated using the sequential review approach followed in [26]. The approach includes quality evaluation standards that consider all key elements involved in research [27], including concept, study design, method of data collection, data analysis, discussion, and outcomes. The marks are represented as internal scoring, external scoring and publication type as P. To identify the quality of the research articles, the authors developed the questionnaire presented in Table 3.

Both external and internal quality criteria, as adopted in [28], have been utilized to examine the quality of research included. The internal criteria (evaluated through C1, C2, C3, C4 and C5) are utilized to measure an article's internal quality, whereas the external quality (assessed through C6) depicted the constancy and reliability of the article's publication source. To score and assess the external quality, "Journal Citation Reports (JCR)" and also the "Computer Science Conference Rankings (CORE)" was utilized. The overall score is the sum of each criterion's single score. The final score can range from is considered as high ranked if it is 6 or higher than 6, the normal rate is counted when it is between 4 and less than 6, and low ranked if it is less than 4.

TABLE 3. Questionnaire to assess quality.

C#	Assessment Question	Expected Answers	Score
Internal Scoring			
C1	Study described well abstract?	a. Yes b. Intermediate c. No	a. 1 b. 0.5 c. 0
C2	Study described literature-review in detail?	a. Yes b. Intermediate c. No	a. 1 b. 0.5 c. 0
C3	Feature description / selection defined clearly?	a. Yes b. Intermediate c. No	a. 1 b. 0.5 c. 0
C4	Methodology section defined clearly?	a. Yes b. Intermediate c. No	a. 1 b. 0.5 c. 0
C5	Relevant conclusion and effectively based on results?	a. Yes b. Intermediate c. No	a. 1 b. 0.5 c. 0
External scoring			
C6	A study published in CORE ranked conference, proceedings and symposium.	a. CORE rank A b. CORE rank B c. CORE rank C d. Q1 e. Q2 f. Q3 g. Other	a. 1.5 b. 1 c. 0.5 d. 2 e. 1.5 f. 1 g. 0

D. DATA ANALYSIS

This section exhibits the screening process of the records acquired by applying the search string on the digital repositories. The obtained articles were shortlisted through a multi-phased shortlisting and screening processes. After this, the classification of the studies was carried out as presented through Table 7 which has been made on the basis of year of publication, type of publication, empirical type followed in the study, the technique of BDD observed in the research study, and the impact of applying BDD technique on software development. Then, the calculation of quality scoring is shown which is followed by a detailed analysis of the investigating areas of this study.

1) SEARCH RESULT

The primary search generated 1494 articles from the specified online research portals. The selection technique mentioned in the preceding section was used in this step. The steps of the selection process are likewise represented in Figure 2, and the results of each phase is presented in Table 4.

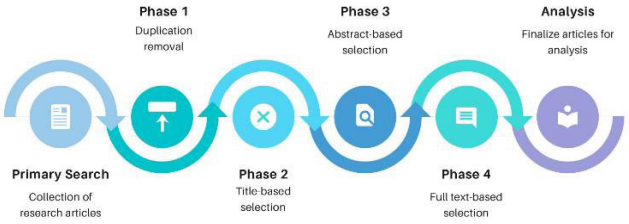

FIGURE 2. Selection procedure.

TABLE 4. Selection Process by Phase.

DATABASE	RECORDS OBTAINED THROUGH PRIMARY SEARCH	P-I	P-II	P-III	P-IV
Springer Link	363	31	23	14	8
Science Direct	665	15	10	5	5
IEEE XPLORE	330	45	25	11	10
ACM Digital Library	136	20	12	7	8
Total	**1494**	**111**	**70**	**37**	**31**

A screening process has been applied based on the inclusion and exclusion criteria applied on titles, abstracts, and full articles. In phase I (P-I), duplicate papers were removed, which resulted in acquiring 111 articles. In phase II (P-II), the title-based screening was carried out. In phase III (P-III), abstract-based screening was done on the 70 resultant articles found in the previous phase. Then, the full text-based analysis was made in phase IV (P-IV) on 37 articles, resulted in acquiring a total of 31 articles that are found most appropriate to be included in this SLR for information extraction and analysis. Table 4 shows the number of papers found from each database.

Advanced computer databases searches were made to find research papers from many different journals, conferences, and meetings. Furthermore, different online databases were used to find the research papers that are part of this review. Some of the most common ones were ACM digital library, which made up about 28% of the shortlisted papers. Another 35% were from IEEE Xplore and almost 21% of the selected studies are obtained through Springer Link while nearly 18% came from Science Direct. Figure 3 shows the Digital Library (DL)-wise distribution of the selected studies.

These years are outliers as no other publication type has been observed in the pool of selected papers for these years. The papers produced through symposium and workshops were fewer as compared to those published in journals and presented in conferences. The years 2012 and 2019 show the same trend where a total of 3 papers were produced through different publication channels, which include workshop, conference, and journal. Overall, the trend in publications across different publication types reveal that the publication types of conference and journal being more popular among the four categories.

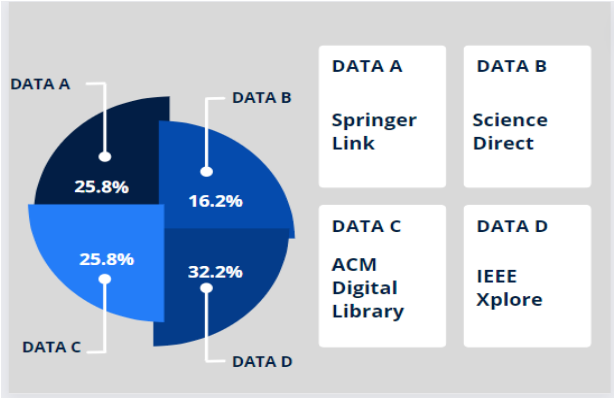

FIGURE 3. DL-wise ratio of selected studies.

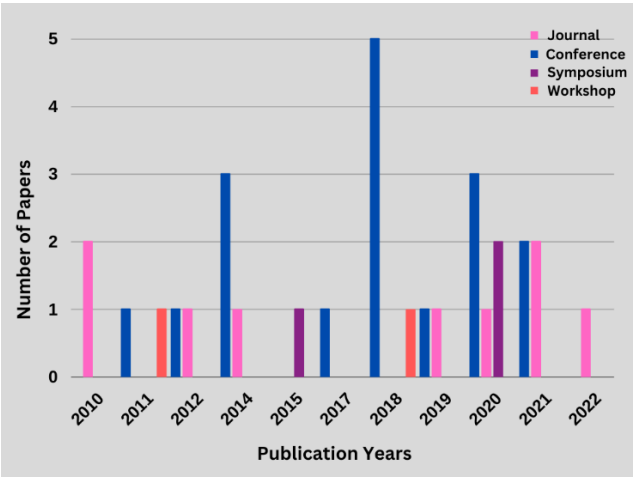

FIGURE 4. Selected studies distribution over the year.

Figure 4 shows the distribution of selected research papers published over a period of 12 years. The analysis of publication sources of the selected studies reveals that these articles were sourced from four primary publication channels, namely journals, conferences, symposiums, and workshops. Among the shortlisted studies, approximately 58% were published in conferences, demonstrating the significant role of conferences in identifying research findings. The Journal publications accounted for around 26% of the selected articles while each of the workshop and symposiums represented almost 16% of these studies. The data reveals that most of the shortlisted papers are presented in conferences where the highest number of conference papers are produced in 2018. Conversely, the years 2010 and 2022 show more journal papers as compared to any other publication type.

Figure 5 presents the distribution of research articles published between 2010 and 2022, classified by the research method employed in the study. The data shows that in 2010, one case study and two experiments were published. In 2011, one experiments and two reviews were reported. In 2012, the research consisted of two reviews and one experiment. In 2014, three experiments were reported. In 2017, two

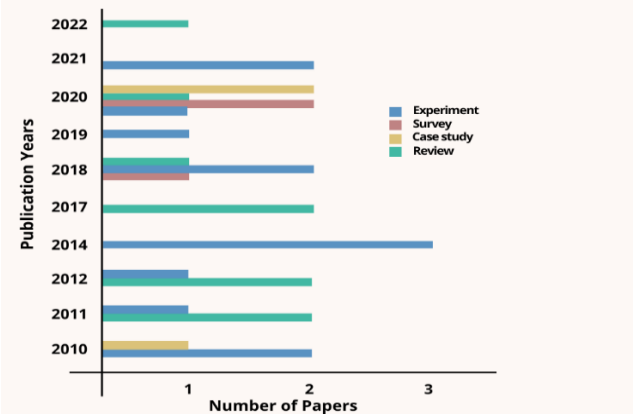

FIGURE 5. Types of research.

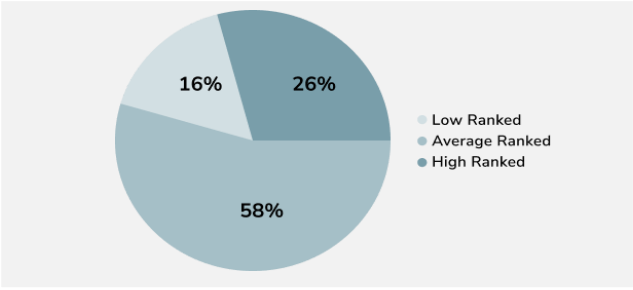

FIGURE 6. Score-based articles ranking.

reviews were published, while in 2018, one review paper, one survey, and two experiments were reported. In 2020, the research comprised of two case studies, two surveys, one review, and one experiment. On the other hand, one experiment was conducted in 2019 and one review paper was published in 2022. Finally, in 2021, two experiments were reported. Overall, the data in Figure 5 highlights the types of research methods used in the studies reported in the analyzed research papers. The data shows that review and experiments were the most commonly used methods, with case studies, and reviews being less frequent. The trends in research methods also varied across different years, with some years seeing a higher frequency of certain research methods than others.

A developed system was used for rating the studies that were found to help decide which ones were the most important for the research. The quality evaluation of the studies was made by authors according to the defined criteria and reviewed by two independent reviewers. Each part of the study is given a score based on criteria that have listed in Table 3. Out of all 31 selected articles, 26% got the score of 6 or higher than 6 and nominated as the high-graded articles, 58% of the selected articles get normal rank, having a range between 4 and less than 6 while 16% of these studies exhibit low rank with score less than 4. This information is represented in Figure 6.

2) ASSESSMENT OF RESEARCH QUESTIONS

The assessment of research questions are given after classifying and evaluating the results of each inquiry set at the start of the review.

a: ASSESSMENT OF QUESTION 1. WHAT TECHNIQUES AND METHODS ARE USED TO REDUCE INTRINSIC AMBIGUITY AND COMMUNICATION GAP BETWEEN THE MEMBERS OF AGILE TEAM?

Almost 22% of the selected studies provided details of the BDD techniques applied for software development. Table 5 shows BDD techniques used for software development. The majority of the research studies used BDD for the removal of ambiguity and communication gaps. A case study was discussed in the study [19] in which BDD was used in combination with Scrum to restructure various Educational and Academic Management Systems for Courses Based on Active Learning Methodologies (EAMS-CBALM) components, which demonstrates that the BDD process improved the communication between the agile team concerning the Scrum method employed in the EAMS-CBALM development to express system requirements clearly and allow communication throughout the development process. The techniques that are used in the selected studies for the removal of ambiguity in requirement are shown in Table 5.

A study [29] has established ScrumontoBDD approach which consolidates Scrum ontologies to improve the process of building a product. It uses special terms called ''ontologies'' to make sure that customer stories, which are written in plain language, are understood clearly and without confusion. One study [2] has suggested a new way to use BDD based stream that consolidates testing and verification in a consistent way that assists designers to compose properties by initializing from natural language. On the other hand, the studies [6] have been adopted in which customers and developers might interconnect in the same language due to a ubiquitous language based on the business domain. Developers will utilize the language to name classes and methods throughout the design and execution phases. Two case studies were discussed in which Project A settled on utilizing Scrum [30], [31] in addition to BDD, which benefits the development team's proper clarification and understanding of the requirement before moving to the code because it overcomes domain linguistic and cultural barriers by being clear and direct on the necessities. A research [32] involved a natural language to establish a supported stream for BDD in which the customer takes part in a discussion of the system just for the build step of the structure and also the skeletons of the code semi-automatically from a known scenario. BDD utilizes normal language as a method of correspondence to guarantee that all task individuals, including designers and partners, have a mutual understanding of the framework to be delivered [33].

The study [34] showed that the adoption of BDD might promote broader stakeholder engagement in the creation of ATS and ensure that both technical and non-technical INSPIRE stakeholders effectively know to INSPIRE implementation requirements and their effects. The BEAST Methodology [35] was developed, in which stakeholders are required to provide requirements that describe the behavior of the whole system. These BDD behaviors are automatically

TABLE 5. BDD techniques.

Techniques	Description	Ref No.
Test case generation	Construction of test cases using simple text language can be done by using BDD that helps the development team to simply identify the existing needs, while QA teams can thoroughly test the product.	[2], [36]
ScrumOntoBDD	A special term ontologies is used to make sure that the requirements are understood clearly without confusion	[29]
UML	UML profile is defined that allow to create executable Foundational UML (fUML) stories and scenarios.	[32], [37]
Conformance testing	BDD scenarios is used for the conformance and acceptance testing.	[34]
Beast methodology	Beast methodology is utilized in the improvement of a MAS (Multi-Agent System) for shortcoming conclusion in FTTH (Fiber to the Home) organizations.	[35]
T-BDD	The T-BDD technique is used for backend testing to perform and achieve a high degree of test inclusion.	[38]

transformed into JUnit test cases by the BEAST tool, which provides clear traceability from user requirements. As a result, stakeholders may be kept up to date on the project's progress. Moreover, a tool that behaves nicely is developed in the study [31] that is used for reducing the manual code maintenance need by translating natural language statements into executable step functions. Similarly, another study [37] presents a BDD model library with foundational UML (FUML) activities for evaluating equality and inclusion. Another study [38] describes T-BDD, a testing strategy that combines TDD with BDD methodologies. The T-BDD approach was used in the Vixio backend system, proving its capacity to produce a high percentage of test coverage. Furthermore, when the feature parameters were changed, the T-BDD technique demonstrated its flexibility. Also, an Eclipse-based development tool is used that gives developers a specific vocabulary for designing certain scenarios that are executable and also provides forward updates on the project state based on confirmed provided behavior.

Hence, different research studies analysis indicates that removing ambiguities of natural language is one of the most important point or major benefit of using BDD to reduce communication gap.

b: ASSESSMENT OF QUESTION 2. WHAT ARE THE BDD IMPACTS ON SOFTWARE DEVELOPMENT STAGES?

Approximately 43% of the chosen studies included information regarding the impacts of BDD. Around 26% of shortlisted studies focused on the benefits of BDD in improving communication. Furthermore, approximately 23% of the selected studies emphasized the impact of BDD on cost reduction. Around 17% of the research looked on the role of BDD in requirement verification and code quality maintenance. Furthermore, almost 13% of the selected studies focused on collaboration and the establishment of clear requirements. Two studies, accounting for 7% of the total, mainly assessed the ability of BDD to eliminate requirement duplication and improve the overall quality of software development. The remaining studies in the analysis focused on the impact of BDD on the overall performance and success of the software development process.

The BDD implementation helps the development team to improve communication between the local and remote employees, as well as arrangement with industry people and the speed of the supply, by high re-ease of use and increase of acceptance increases high of the tasks, defining BDD scenarios help to align requirements and expectation that improve product quality and stability [39]. A well-known culture of the proposed BDD advances the accuracy ratio and raises the self-confidence of all members [40], [41]. Similarly, the review conducted in the study [18] shows that BDD consumes more time and cost than TDD but achieves higher customer satisfaction specifically in the beginning of software development process.

The study [42] analyzed the effect of BDD on internal code quality as well as on the overall quality of the code, and identifies improvement in the quality of the product through BDD. Likewise, a few examinations were found in a research [15] in which BDD is tried for the improvement of computer arrangements while getting great outcomes. Similarly, a case study was led [43] on portable application growth utilizing agile software, a sum of 42 interviews was performed (before utilizing BDD and in the wake of utilizing BDD), and the positive effects are highlighted clarification, simplicity of improvement, project arrangement, and harmful impacts are pitiable execution, and trouble to change development mindset. However, the study [1] introduced a step accommodating four principles for evaluating the output of the proposed structure of BDD and its principles were acknowledged by somewhere around 75% of the specialists studied. Specialists featured the significance of reuse inside BDD, a more prominent spotlight on the meaningfulness and lucidity of the resultant details.

A study [12] identified that using BDD encourages team members to interact with each other. This might mean that the information-gathering process needs to be modified as a result of increased team member collaboration. Furthermore, participants have reported other good qualities that may be related to the usage of BDD, such as an improvement in feature knowledge. In [44], the work demonstrated how a leaner is helped and guided by BDD and towards adopting more agile operation and a detailed workflow is provided. It improves collaboration and delivers value for all stakeholders in the development of complex products

in large companies by splitting the client's include demand into more modest, well-distinct outcomes and connecting the structure of an organization's significance across the complete steps. On the other hand, a new approach is proposed in [34] which is Infrastructure for Spatial Information in the European Community (INSPIRE) conformance testing of web-based Geo-graphic Information (GI) facilities utilizing BDD, in which non-specialized stakeholders can take part in the requirement (ATS) Application Tracking System authoring process and get experiences in the consistence cycle.

The highest Focus of any software engineering project is Quality without measuring we cannot ensure the level of quality, many techniques or types are addressed in [45] to understand which model or technique can be applied to which type of Software development Life Cycle (SDLC) phase. Sometimes software deliverables do not perfectly meet the requirements, it is due to lack of testing and test cases for resolving. In this scenario, the BDD is used to generate test cases as well as execute reports [36], [46] described that the regression test selection (RTS) technique is used for BDD specifically to fast up the development process while maintaining software quality. However, the studies [47], [48], [49] show that the fundamental part of BDD is refactoring and is used for the improvement of maintainability of the artifacts. After the analysis of previous researches, it can be seen that BDD has many impacts on SDLC and quality of the product. Figure 7 represents the studies that described the BDD impact on y-axis and x-axis shows the amount of papers that discussed the mentioned BDD impact. The analysis of the BDD impact on software development reveals that 8 out of the 31 selected studies explored the benefits of BDD in enhancing communication, while 7 of these studies emphasized its effect on cost. Five of the shortlisted articles investigated the role of BDD in verifying requirements and maintaining code quality and 4 of the selected studies focused on the impact of BDD on collaboration and clear requirements. Two studies examined the ability of BDD to remove duplication in requirements, increase quality, and remaining studies focused on the performance and the overall success of software development process. Another analysis dimension for this RQ was made to identify the BDD impact that could relate to specific stages of software development. Figure 8 presents the distribution of the percentages of papers that are focused on different stages of software development.

Approximately half of the research studies examined the significance of the requirement gathering process within the context of software development. About 6% of these studies focused on design for the adoption of a designing strategy for integrating BDD in cooperation with Business stakeholders, while about 7% of these studies examined the use and implementation of user stories. The papers revolved around automated regression testing, with a notable percentage of 37%, which highlighted the importance of facilitating both manual and automated testing approaches and examined how

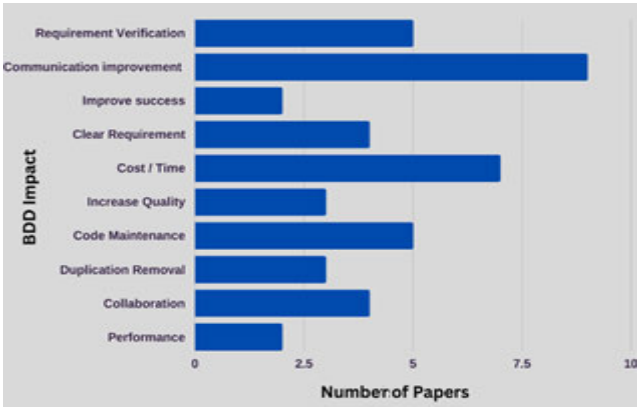

FIGURE 7. BDD impact on software development process.

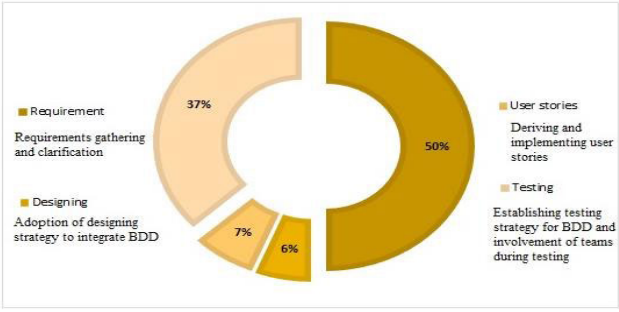

FIGURE 8. Distribution according to software development stages.

teams were involved in the process of choosing testing procedures.

c: ASSESSMENT OF QUESTION 3. WHAT CHALLENGES AND OPPORTUNITIES HAVE BEEN REPORTED TO USE BDD IN SOFTWARE DEVELOPMENT PROCESS?

Approximately 35% of the selected studies analyzed challenges and opportunities in BDD. Table 6 summarizes the various problems and opportunities related to BDD in software development. Experts believe that BDD might have a stronger positive influence during the requirements phase of software development [50]. In [38] the combination of BDD and TDD can produce T-BDD strategy that is utilized for backend testing, perform and accomplish a high level of test-inclusion and performed better when contrasted with TDD testing strategy. Similarly the study [51] shows that Large-scale programs were not planned for BDD also writing test case difficulties and adoption of new technologies are the challenges. Looking at the advantages of BDD can help with tackling the issues of enormous scope project (teamwork, correspondence, necessities elaboration, and confirmation).

On the other hand, in a study [52] analyze by reviewing software practitioners, Binamungu et al. found the troubles and risks of BDD. The main difficulties connected with BDD are questioning and stabling to participate as it affects individuals from various levels, absence of instruction,

format to compose BDD particulars, and support. However, enhanced requirement testability, enhanced documentation, better domain knowledge capture, improved software knowledge, and implementation are the benefits [17]. Duplication of test cases is also a big challenge [53]. End users may readily understand software specifications since they are stated in domain-specific words and also produce better APIs since it promotes developing testable programs [14]. Moreover, the study [54] shows that BDD bridges the gap between quality analysts and product owners, as well as between quality analysts and customers or development teams, and also guides the development process [6], the client stories endorsement, and the automation of testing cases. However, the fundamental test is to persuade a client to pay for the expense of learning BDD and inefficiently composed situations due to the absence of involvement. In [55], Nascimento et al. reported the positive and negative impact of BDD on the agile development team by doing a case study. Some of the positive impacts are clear implementation, reduction in rework, custom execution, improvement in task division, and quality improvement. However, lack of team commitment (to BDD), difficulty for inexperienced developers, and difficulty in terms of UI are some negative impacts on BDD implementation [56]. The study [35] proposed a BEAST methodology in which stakeholders should produce a bunch of behavior specification that depicts the entire framework this instrument is fundamentally utilized in the improvement of a MAS for shortcoming conclusion in FTTH (Fiber to the Home) organizations. The challenges and opportunities of using BDD are mentioned in Table 6.

TABLE 6. Challenges and opportunities of BDD.

Opportunities and challenges of BDD	
Opportunities	Keeping an image of understanding the main features of the research.
	Ensuring correct Execution
	Alignment of the members of the collaborators.
	Decreasing the changes that happen unexpectedly.
	Keeping an image in mind of decreasing the problems in implementation.
	Improvement in the technical aspect of the process
	Minimal effort of reducing the repeating the cycle.
	Breaking down most part of the user stories and making them as minimal as possible.
	Reduce the gap of the medium towards communication.
	Focus on User Needs
Challenges	Certain scenarios can be easier to understand for some whereas difficult for others
	It is interlinked with each other so need of previous planning is mandatory.
	Difficult for unexperienced developers
	Problems understanding in UI
	Test case Duplication
	poorly written scenarios
	Convincing the stakeholders and the customer to pay for the research for the domain of BDD.
	template to write BDD specifications
	adoption of new technologies
	BDD is incompatible with the waterfall approach

IV. DISCUSSION

This section discusses the main findings and future prospects of BDD. It presents techniques to organize the software development process through BDD and gives an idea of the steps involved while using BDD during the process of making software.

A. PROPOSED TAXONOMY

This study presents a taxonomy of BDD, which classifies its impact into eight major categories that were established after a thorough review of earlier research papers in the area of BDD. These categories have been recognized as important areas where BDD has shown its impact and effectiveness through an extensive review and synthesis of the existing literature. The first category, Development, outlines the advantages of implementing BDD, including cost savings, enhanced cooperation, and a development process that is more rapid with timely outcomes. The second category, Requirement, focuses on the advantages of BDD in terms of enhancing clarity, readability, and verification of requirements. The importance of BDD in aiding regression testing, acceptance testing, and obtaining thorough test coverage is highlighted in the third area, automation. The fourth category, Collaboration, underlines the importance of BDD in improving collaboration and good communication between the development team and stakeholders. The sixth category, demonstrates how BDD helps create high-quality products and successful project results. Duplication Removal, the sixth category, highlights the importance of BDD in reducing failures, increasing refactoring, and getting rid of redundant code. The seventh area, Communication, deals with bridging stakeholder communication gaps through the use of BDD methodologies. Also the taxonomy includes certain methods and approaches that have been studied in the literature, like ScrumONTOBDD, UML, BEAST methodology, and test case creation. Additionally, the automation of test cases using the Cucumber framework is highlighted as a relevant approach in the context of BDD. This taxonomy offers a thorough framework for understanding and evaluating the numerous effects and uses of BDD in software development facilitating future research and practical implementation in the field.

BDD is a way of making software that makes it easier to understand by turning the code into simple language. It allows customers to define the behavior of the programs and goals in a form that everyone can understand. Every member who wants to give a thought to something or define a goal also defines a behavior first. From the analysis of the previous research, it was identified that in software development generally opt for quick development, this only

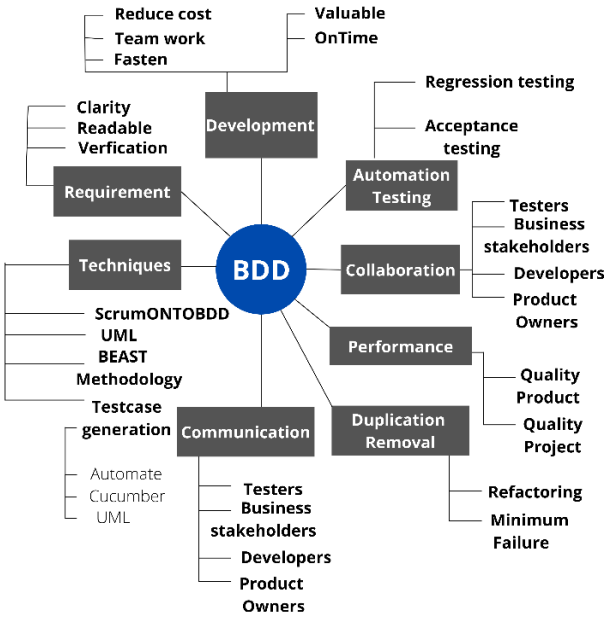

FIGURE 9. A taxonomy of BDD.

happens with self-directed and efficient team with good collaboration [54], [55] and communication [39], [50] among themselves as well as among the product owner and business stakeholders. BDD helps to encourage the developers to focus only on the requirements that are given or the requested behavior of an application. It helps to avoid the focus of developers on unnecessary features. It combines, augments and refines the practice that is used in TDD and acceptance testing. Teamwork shows the overall performance of what is being done. However, if all the team members are not on the same page, then it would be difficult to deliver the required software to the customer. Behavior-driven development also allows for automating certain processes. Regressing testing [36], [57], [46] can be carried out after the completion of the build by running the feature file of BDD that are formed with the usage of some appropriate tool such as cucumber tool. It is also useful for test case generation [2]. It shows that the developed build is according to the required behavior. By using this quality product and project developed in a short time and cost [17]. Teamwork or collaboration is one of the important steps. BDD encourages all the team members convinced or on the same page [12] that helps to increase the performance quality and reduce the cost and delivery valuable work to the customer on time. Test cases are automatically generated by writing a BDD script using the cucumber tool. It is used to write acceptance tests. Figure 9 shows the knowledge areas that are covered in the selected papers using BDD techniques.

B. PROPOSED FRAMEWORK

The framework presented in this study provides a comprehensive and structured approach for applying BDD in software development. It outlines the process stage by stage, offering a detailed understanding of how BDD can be incorporated into the development lifecycle. Each stage of the framework focuses on specific aspects, such as requirements clarification, collaboration, automation, Performance, duplication removal, and communication improvement.

A framework for BDD has been designed to define the process of applying these techniques in the software development process and stages. In this framework, stage A is the requirement gathering and analysis; after that, user stories are created by the business analyst. User stories are one of the important steps. Now all the team members, business stakeholders, and also the client have the option to decide about the testing strategy in the initial stage as shown in figure 10. This framework is introduced by the analysis of the previous research and work. In this framework, the gaps or challenges identified from the previous research are somehow covered. The difficulty of getting a whole team on the same page is solved by adding a step of test requirement analysis or selection strategy. In the development process, they are very helpful for the developers for development, the testers for the creation of the test strategy, and the designers to think and design according to the user stories. The next stage, B, is the selection of the test strategy, in which all the members of the development team, business stakeholders, and the client are involved and decide on the testing strategy in an initial stage so that the gap that was mentioned in the previous work of getting the whole team on the same page is solved. After the selection of the test strategy, the behavior of the system is defined and written in Gherkin. After that in the third stage (C), the development and testing take place. The developers develop the system according to the requirement or the defined behavior. After the development phase, the developed build moves into the testing phase, in which manual or automated testing is conducted. In automation, the results are evaluated by running the BDD script file using automation tools that support BDD. The script file consists of different possible scenarios, and each scenario has several possible test cases. If the result is positive, it means the developed build or the system behavior is according to the requirement.

The proposed framework can be applied by involving the specified components as identified through reviewing the state-of-the-art BDD. Most of the studies support BDD as it minimizes the chances of misinterpretation and subsequent rework [6], [14]. Due to the involvement of business stakeholders and assessment of behaviors of application along with the development process, this technique could help improve customer satisfaction as it serves as live documentation, making it simpler for developers to understand the current functionality and make modifications without any problem. It could lead to clarify requirements at early stages of software development and could also help in improving the outcomes of a software development projects as well as the quality of end-product.

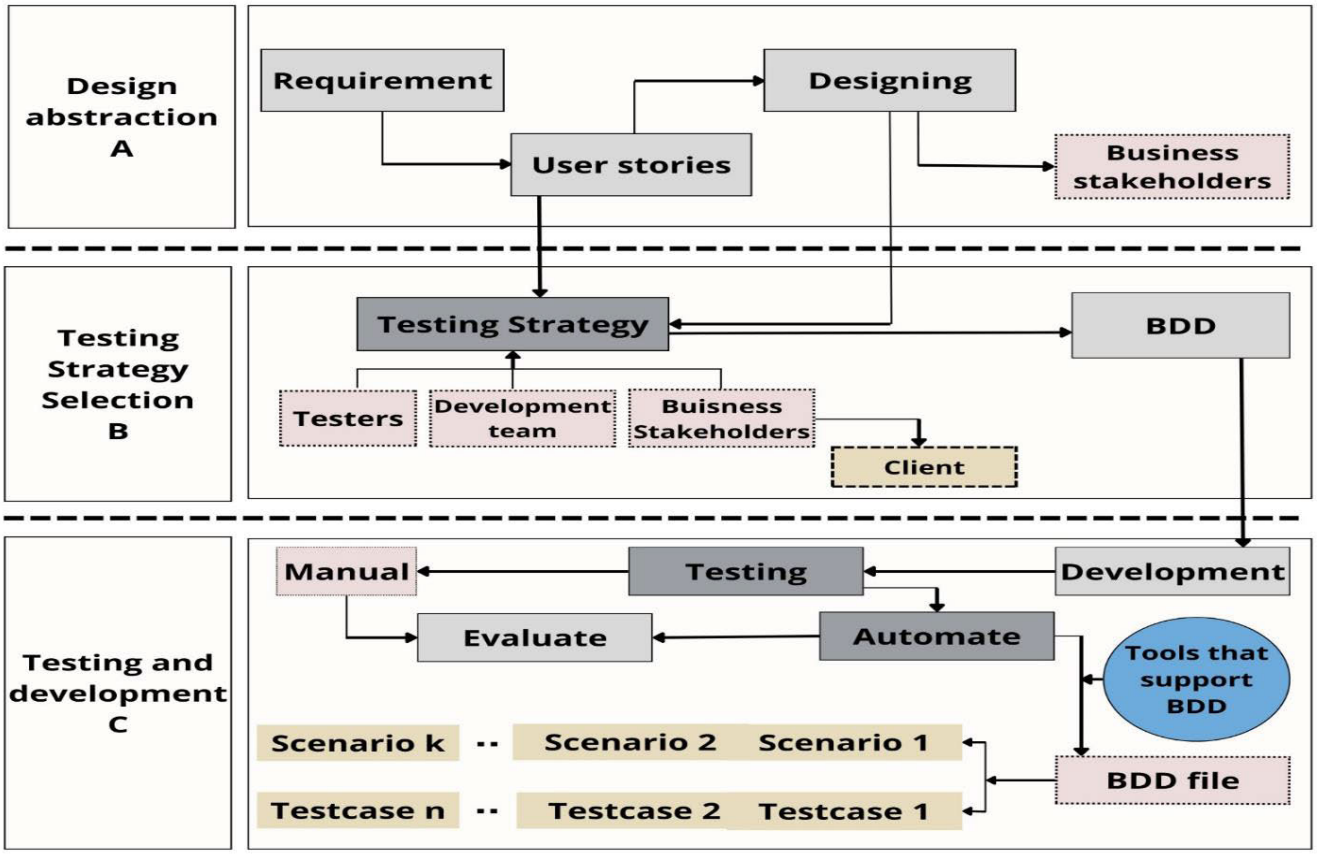

FIGURE 10. Framework of BDD.

C. WORK FLOW

A workflow is established to exhibit incorporation of BDD in different stages of software development. It is an illustration to guide how BDD can be applied in a software development project and what activities are required to be performed while linking the software development components presented through framework. It covers the complete process, from gathering the initial requirements to development, testing, and delivery. The workflow highlights the sequential steps and interactions between stakeholders, including business analysts, developers, testers, and other project members. It gives a high-level overview of how BDD may be included into the larger software development process while highlighting the techniques' collaborative and iterative aspects. The workflow presents an in-depth understanding of the whole implementation process, demonstrating the connections and relationships between various phases, while the framework offers a detailed breakdown of the individual stages and activities involved in BDD. This workflow helps to bridge the gap between the people making the software and the people who will use it by clearly defining how behavior-driven development is used throughout the process. This flow consists of several steps in which BDD plays an important role in software development. This flow is created by the analysis of the previous research. According to our analysis, a common issue that was faced when documenting the requirement is describing what is needed by the user. For solving this issue user stories [58] were introduced by Agile which tackles this problem. Gathering the requirement is one of the most important parts of the software development process. Firstly, the product owner and the user have a conversation about what they need. After that quality assurance engineers, product owner, and the Developers elaborate on the requirement using User stories. User stories or use case is a brief explanation of the requirement or system behavior or what exactly the client demand or user need. User stories are written in the form of

As a ————
I want to ————
So that I can ————

These user stories are helpful for the designer to design the system according to the scenarios and the given requirements in which all single steps are covered. BDD can apply to user stories by adding scenarios that reflect the acceptance of requirements. Development stage starts after the completion of BDD scenarios that define the behavior of the system. The main problems to use BDD are to convince a client for the BDD approach or getting a development team on the same page. To address these issues, the testing strategy is

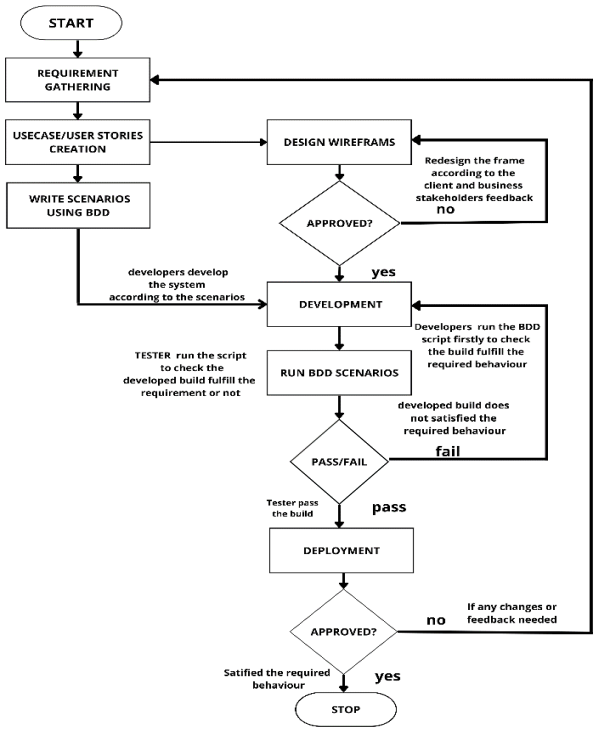

FIGURE 11. Work Flow of using BDD in the process of developing soft.

required to be initially decided and the cost and benefits of the specific strategy are to be discussed with the client or the development team instead of deciding in the middle of the development. After that, these requirements or user stories can be transformed into scenarios using BDD techniques. Scenarios are very significant because they can be used to make sure all the cases are covered or not. The scenarios are very helpful for the designer to contrast the design and for the developers to develop the system and act as automated tests. The developer used that BDD file for development that helps to only focus on the user need and eliminate the waste means less rework due to misinterpreted requirements and acceptance criteria. The tester uses these scenarios as the basis for their tests. Automation and regression testing are also done by using this flow. Figure 11 shows the proper working flow of using this technique because they can be used to identify whether all the bases are covered or not. These scenarios are written in the form of Given, When and Then [3]. After creating all the scenarios and test cases these files are shared with the business stakeholders, developers, and designers. If business stakeholders or the customer wants any changes or some requirement is missing, then it tackles it in the early stages. From the previous research, it was determined that BDD plays an important role for fasten up the development process and providing a quality product and customer satisfaction. Natural language usage bridges the gap between the business stakeholders and the developing team and allows customer interaction in some development phases. BDD is upheld by various toolkits including JBehave [7], Cucumber [8], etc. The cucumber is one of the most famous and useful tools in BDD. It is used to execute automated acceptance tests in BDD style. It uses the language that is widely used by all the members and helps in connecting with the technical and non-technical teams. While using BDD with cucumber automation tests are being created and that can be understood by all the software development team and members because they are written in business domain language. By using the flow defined in Figure 10 or the analysis of the previous research small experiment was run on a small project to check how to fasten the development process or customer satisfaction by using this technique with cucumber. It was an application that helps people to donate online. As a result of using our proposed approach, it was identified how fast the development process occur and how fast the system can be delivered to the customer with good quality and customer satisfaction in a low time and suitable cost.

D. PROBLEMS AND GAPS

Some of the notable problems that could be faced by the software development teams while incorporating BDD in software development projects indicate the gaps in BDD literature which could be examined for future research. This has been evaluated through our observation and analysis of the previous research. BDD is used for the implementation of an application that describes the behavior in the form of stakeholder's perspective. As this approach have a lot of advantages side by side but also has some challenges. It is known that BDD involves a lot of details for specifications of requirements which sometimes are very hard to comprehend. It is very difficult to get a whole team on the same page because some of the team members are unwilling to replace their current working flow with BDD and also require some sort of learning. Many organizations and developers still do not understand the differences between the BDD concepts or where they BDD and TDD overlap but for BDD, prior exposure to TDD ideas is essential. In paper [59] an experiment was carried out that showed TDD and BDD techniques increased the external quality of the delivered product. However, a decrease in productivity and internal quality were noted in BDD which might be due to the additional steps involved in BDD. On the other hand, implementation of BDD requires money or time [42]. If it has an insufficient budget, then it's hard to implement BDD. It is a high-automation methodology, and some basic coding skills are required for QA engineers. It is not an issue if you have the knowledge, skills, and practice of automating tests (TDD and acceptance TDD). BDD may not be effective if it is written in the wrong format or lack of knowledge and experience about the terms and conditions of BDD, also not share the vision with stakeholders in advance. Duplication in suites can also be one of the most important challenges in BDD [53]. It is very hard to find faults in large BDD suites also it is a very time taking process to continuously look for customer's satisfaction as it is considered as one of the main goals of software

TABLE 7. Classification table and quality scoring.

Ref. No.	Year	Publication Type	Empirical Type	Technique	BDD Impact	Quality Score						Total Score
						C1	C2	C3	C4	C5	C6	
[1]	2020	Conference	Survey	No	Readable and Clear requirements	1	1	1	1	1	1	6
[2]	2014	Conference	Experiment	BDD for test case generation	Verification	1	0	0.5	1	1	0.5	4
[6]	2011	Conference	Review	No	Collaboration improvement	1	1	1	1	1	1	6
[12]	2014	Conference	Experiment	No	Improve Success	1	0.5	1	1	1	1.5	6
[14]	2018	Conference	Survey	No	Communication improvement	1	1	0.5	0.5	1	1.5	5.5
[15]	2015	Symposium	No	No	maintainability	0.5	1	1	1	1	0	4.5
[17]	2020	Symposium	Review	No	Communication improvement, quality, cost, time	1	1	0.5	0.5	1	0	4
[18]	2020	Conference	Review	No	Cost, time and quality	1	0.5	1	1	1	0.5	5
[29]	2021	Journal	Experiment	Scrumonto BDD	Communication improvement	1	1	1	1	1	2	7
[31]	2012	Conference	No	No	Improve Success	1	1	0.5	0.5	1	1	5
[32]	2012	Conference	Experiment	UML	Readable and clear requirements	1	0.5	1	1	0.5	0.5	4.5
[34]	2014	Conference	Experiment	Conformance testing through BDD	Team Collaboration	1	1	0.5	1	1	0.5	5
[35]	2014	Journal	Experiment	Beast methodology	Time, Communication	0.5	1	1	1	1	2	6.5
[37]	2010	Journal	Experiment	UML	verification	1	0.5	1	1	0	1.5	5
[38]	2019	Journal	No	T-BDD	Performance	1	0.5	1	1	1	0	4.5
[39]	2019	Conference	Experiment	No	Reduce manual code maintenance	0.5	0	1	1	1	1	4.5
[42]	2018	Conference	Experiment	No	Quality and productivity improvement	0.5	0.5	1	1	0.5	0.5	4
[43]	2010	Journal	Case study	No	Communication, maintainability	1	0.5	1	0	1	0.5	5
[44]	2018	Conference	No	No	Communication improvement	0.5	1	1	1	1	0	4.5
[45]	2012	Journal	Review	No	Requirement Clarification	1	0	1	0	1	0.5	4.5
[36]	2018	Conference	Review	BDD for test case generation	Requirement clarification	1	1	1	1	1	0	6
[46]	2021	Conference	Experiment	No	Verification, maintainability, Time	1	0.5	1	0.5	1	1	5.5
[47]	2011	Workshop	Experiment	No	Performance and maintenance	1	0.5	0.5	1	0.5	0	3.5
[48]	2017	Conference	Review	No	Communication improvement	0.5	1	1	0.5	1	0.5	5.5
[49]	2022	Journal	Review	No	Duplication removal	1	1	1	1	1	2	7
[51]	2021	Journal	No	No	Communication improvement, quality, cost, time	1	1	1	1	1	2	7
[52]	2018	Workshop	Experiment	No	Duplication removal	1	1	1	1	1	0.5	5.5
[53]	2021	Conference	No	No	Duplication removal	1	1	0.5	0.5	0.5	0.5	4
[54]	2018	Conference	Review	No	Collaboration	1	1	1	0.5	1	1	5.5
[55]	2020	Conference	Case study	No	Collaboration	1	1	1	0.5	1	1	5.5
[56]	2020	Symposium	Survey	No	Communication improvement, quality, cost, time	1	1	0.5	1	1	0.5	5

development when using BDD. According to our analysis, in BDD automation, if any of the test case lines fail, then the whole process of running the script is stopped. In BDD, maintenance does not come easily [14]. It is appropriate only with the collaboration of developers, business analysts, and testers. It is an overhead if only testers can read BDD tests. In-depth knowledge of the scripting language is not required in BDD. However, organizations now require it every day. One of the most challenging phases in BDD is to convince a customer of the new technique's learning cost and to convince a team to switch to it. And the gap that has been identified is to overcome the time-consuming criteria of running BDD scenarios. Also, if BDD test suites are running and any of the BDD script lines fail, then the entire BDD suite would stop and show an error without running further. This is one of the major gaps that makes BDD slow because, for further evaluation, the script has to start again after fixing that error. Also, BDD management can also be challenging if they grow over a handful of features and multiple members of team are involved in it for writing and updating them over time.

V. LIMITATIONS

The limitations of this review are highlighted below:

The search string has been developed to retrieve the most relevant research papers from multiple online repositories with an effort to reduce the risk of omitting important studies by adding a number of keywords. However, there may be alternate keywords or synonyms that might alter the result.

Papers from various sources are carefully examined and selected by authors. The research portals through university's subscription were used to search articles. However, it is important to acknowledge that the limited access to certain sources may have mistakably led to the oversight of certain papers. This potential limitation could affect the comprehensiveness of the literature review and may have excluded related articles from sources that were not retrieved due to the accessibility limitation of the university's subscription.

The shortlisting and classification process in this research study was precisely conducted by the authors and subsequently reviewed by two independent reviewers. Consensus thorough discussions were done to settle any differences in the classification results. The high degree of agreement, as evidenced by the Kappa coefficient value of 0.92, demonstrates the reviewers' strong consensus. This rigorous technique improves the shortlisting and classification process's reliability and credibility, contributing to the overall robustness of the research findings.

The framework of BDD (presented as Figure 10) is established as one of the outcomes of reviewing the state-of-the-art BDD; however, the its validity is not the part of this study. It can be considered as one of the future research dimensions of the applications of BDD in software development.

VI. CONCLUSION

This articles presents a systematic literature review which provides the contribution to the body of knowledge in the field of software engineering and development by providing a comprehensive analysis of BDD, its potential applications, and practical guidelines for its implementation. The 31 published studies were carefully selected for the review by following a systematic and multi-stage shortlisting process. This review of the selected articles was carried out by evaluating the aspects related to the investigating areas of this work. After having an in-depth review of the previous researches, it has been concluded that BDD plays an important role to cover the communication gap between the business stakeholders and the development team members because it is written in a simple Gherkin Language that is understandable by technical as well as non-technical personas. BDD plays an important role in speeding up the development process and gaining a large amount of customer satisfaction. Considering this, the framework and the workflow for using BDD in the software development process is proposed. Furthermore, a taxonomy of BDD has been developed to give direction for future work. Although, BDD has many significant future implications yet its use also exhibits many challenges. This study will be useful for the practitioners and researchers of the software engineering field in terms of applying BDD and identifying its applicability during the process of software development.

REFERENCES

[1] L. P. Binamungu, S. M. Embury, and N. Konstantinou, "Characterising the quality of behaviour driven development specifications," in *Proc. Agile Processes Softw. Eng. Extreme Program., 21st Int. Conf. Agile Softw. Develop. XP*, Copenhagen, Denmark. Cham, Switzerland: Springer, Jun. 2020, pp. 87–102.

[2] M. Diepenbeck, U. Kühne, M. Soeken, and R. Drechsler, "Behaviour driven development for tests and verification," in *Proc. 8th Int. Conf. Tests Proofs (TAP)*, York, U.K. Cham, Switzerland: Springer, Jul. 2014, pp. 61–77.

[3] A. Mishra and A. Mishra, "Introduction to behavior-driven development," in *IOS Code Testing: Test-Driven Development and Behavior-Driven Development With Swift*. New York, NY, USA: Apress, 2017, pp. 317–327.

[4] A. Aitken and V. Ilango, "A comparative analysis of traditional software engineering and agile software development," in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 2013, pp. 4751–4760.

[5] R. A. de Carvalho, F. L. de Carvalho e Silva, and R. S. Manhaes, "Mapping business process modeling constructs to behavior driven development ubiquitous language," 2010, *arXiv:1006.4892*.

[6] C. Solis and X. Wang, "A study of the characteristics of behaviour driven development," in *Proc. 37th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, Aug. 2011, pp. 383–387.

[7] A. Okolnychyi and K. Fögen, "A study of tools for behavior-driven development," *Full-Scale Softw. Eng. Trends Release Eng.*, vol. 7, no. 6, Feb. 2016.

[8] W. Ye, *Instant Cucumber BDD How-To*. Birmingham, U.K.: Packt, 2013.

[9] E. Alameda, "Introduction to testing with rspec," in *Foundation Rails 2*. New York, NY, USA: Apress, 2009, pp. 253–289.

[10] M. Senapathi and M. L. Drury-Grogan, "Refining a model for sustained usage of agile methodologies," *J. Syst. Softw.*, vol. 132, pp. 298–316, Oct. 2017.

[11] L. Vijayasarathy and D. Turk, "Agile software development: A survey of early adopters," *J. Inf. Technol. Manag.*, vol. 19, no. 2, pp. 1–8, 2008.

[12] M. Senapathi and A. Srinivasan, "An empirical investigation of the factors affecting agile usage," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.*, May 2014, pp. 1–10.

[13] H. Alaidaros, M. Omar, and R. Romli, "The key factors of evaluating Agile approaches: A systematic literature review," *Int. J. Supply Chain Manag.*, vol. 8, no. 2, pp. 1–11, 2019.

[14] L. P. Binamungu, S. M. Embury, and N. Konstantinou, "Maintaining behaviour driven development specifications: Challenges and opportunities," in *Proc. IEEE 25th Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, Mar. 2018, pp. 175–184.

[15] M. Rahman and J. Gao, "A reusable automated acceptance testing architecture for microservices in behavior-driven development," in *Proc. IEEE Symp. Service-Oriented Syst. Eng.*, Mar. 2015, pp. 321–325.

[16] P. Rai, "Extending automated testing to high-level software requirements: A study on the feasibility of automated acceptance-testing," Tech. Rep., 2016.

[17] T. Couto, S. Marczak, and F. Gomes, "On the understanding of how to measure the benefits of behavior-driven development adoption: Preliminary literature results from a grey literature study," in *Proc. 19th Brazilian Symp. Softw. Qual.*, Dec. 2020, pp. 1–7.

[18] H. M. Abushama, H. A. Alassam, and F. A. Elhaj, "The effect of test-driven development and behavior-driven development on project success factors: A systematic literature review based study," in *Proc. Int. Conf. Comput., Control, Electr., Electron. Eng. (ICCCEEE)*, Feb. 2021, pp. 1–9.

[19] P. L. de Souza, A. F. do Prado, W. L. de Souza, S. M. dos S. F. Pereira, and L. F. Pires, "Combining behaviour-driven development with scrum for software development in the education domain," in *Proc. ICEIS*, vol. 2, 2017, pp. 449–458.

[20] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," School Comput. Sci. Math., Dept. Comput. Sci., Keele Univ., Univ. Durham, Durham, U.K., Tech. Rep. EBSE-2007-01, 2007.

[21] U. Omer, R. Tehseen, M. S. Farooq, and A. Abid, "Learning analytics in programming courses: Review and implications," *Educ. Inf. Technol.*, vol. 28, pp. 1–48, Feb. 2023.

[22] D. Rosenstreich and B. Wooliscroft, "Measuring the impact of accounting journals using Google scholar and the g-index," *Brit. Accounting Rev.*, vol. 41, no. 4, pp. 227–239, Dec. 2009.

[23] K. Adnan, R. Akbar, and K. S. Wang, "Development of usability enhancement model for unstructured big data using SLR," *IEEE Access*, vol. 9, pp. 87391–87409, 2021.

[24] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, nos. 9–10, pp. 833–859, Aug. 2008.

[25] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. 12th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*, vol. 12, 2008, pp. 1–10.

[26] M. S. Farooq, A. Hamid, A. Alvi, and U. Omer, "Blended learning models, curricula, and gamification in project management education," *IEEE Access*, vol. 10, pp. 60341–60361, 2022.

[27] U. Omer, M. S. Farooq, and A. Abid, "Introductory programming course: Review and future implications," *PeerJ Comput. Sci.*, vol. 7, p. e647, Jul. 2021.

[28] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval, "Requirements engineering education: A systematic mapping study," *Requirements Eng.*, vol. 20, no. 2, pp. 119–138, Jun. 2015.

[29] P. L. de Souza, W. L. de Souza, and L. F. Pires, "ScrumOntoBDD: Agile software development based on scrum, ontologies and behaviour-driven development," *J. Brazilian Comput. Soc.*, vol. 27, no. 1, p. 10, Dec. 2021.

[30] K. Schwaber, *Agile Project Management With Scrum*. Microsoft Press, 2004.

[31] M. S. Murugaiyan and S. Balaji, "Succeeding with agile software development," in *Proc. IEEE-Int. Conf. Adv. Eng., Sci. Manage. (ICAESM)*, Mar. 2012, pp. 162–165.

[32] M. Soeken, R. Wille, and R. Drechsler, "Assisted behavior driven development using natural language processing," in *Proc. 50th Int. Conf. Objects, Models, Compon., Patterns (TOOLS)*, Prague, Czech Republic. Berlin, Germany: Springer, May 2012, pp. 269–287.

[33] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Reading, MA, USA: Addison-Wesley, 2004.

[34] F. J. Lopez-Pellicer, M. Á. Latre, J. Nogueras-Iso, F. J. Zarazaga-Soria, and J. Barrera, "Behaviour-driven development applied to the conformance testing of INSPIRE web services," in *Connecting a Digital Europe Through Location and Place*. Cham, Switzerland: Springer, 2014, pp. 325–339.

[35] Á. Carrera, C. A. Iglesias, and M. Garijo, "Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development," *Inf. Syst. Frontiers*, vol. 16, no. 2, pp. 169–182, Apr. 2014.

[36] A. Sheshasaayee and P. Banumathi, "Impacts of behavioral driven development in the improvement of quality software deliverables," in *Proc. 3rd Int. Conf. Inventive Comput. Technol. (ICICT)*, Nov. 2018, pp. 228–230.

[37] I. Lazăr, S. Motogna, and B. Pârv, "Behaviour-driven development of foundational UML components," *Electron. Notes Theor. Comput. Sci.*, vol. 264, no. 1, pp. 91–105, Aug. 2010.

[38] I. B. K. Manuaba, "Combination of test-driven development and behavior-driven development for improving backend testing performance," *Proc. Comput. Sci.*, vol. 157, pp. 79–86, Jan. 2019.

[39] A. Scandaroli, R. Leite, A. H. Kiosia, and S. A. Coelho, "Behavior-driven development as an approach to improve software quality and communication across remote business stakeholders, developers and QA: Two case studies," in *Proc. ACM/IEEE 14th Int. Conf. Global Softw. Eng. (ICGSE)*, USA, May 2019, pp. 105–110.

[40] J. Smart and J. Molak, *BDD in Action*. New York, NY, USA: Simon and Schuster, 2023.

[41] B.-Y. Wang, Y.-C. Yen, and Y. C. Cheng, "Specifying Internet of Things behaviors in behavior-driven development: Concurrency enhancement and tool support," *Appl. Sci.*, vol. 13, no. 2, p. 787, Jan. 2023.

[42] L. A. Cisneros, C. I. Reis, M. Maximiano, and J. A. Quiña, "An experimental evaluation of ITL, TDD and BDD," in *Proc. 13th Int. Conf. Softw. Eng. Adv. (ICSEA)*, 2018, pp. 20–24.

[43] R. Green, T. Mazzuchi, and S. Sarkani, "Communication and quality in distributed agile development: An empirical case study," *Int. J. Comput. Inf. Eng.*, vol. 4, no. 1, pp. 38–44, 2010.

[44] G. Munkácsi, "Applying behaviour driven development to enhance agile software development processes," Univ. Appl. Sci., Finland, Tech. Rep., 2018.

[45] M. Tuteja and G. Dubey, "A research study on importance of testing and quality assurance in software development life cycle (SDLC) models," *Int. J. Soft Comput. Eng.*, vol. 2, no. 3, pp. 251–257, 2012.

[46] J. Xu, Q. Du, and X. Li, "A requirement-based regression test selection technique in behavior-driven development," in *Proc. IEEE 45th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2021, pp. 1303–1308.

[47] R. Borg and M. Kropp, "Automated acceptance test refactoring," in *Proc. 4th Workshop Refactoring Tools*, May 2011, pp. 15–21.

[48] A. Egbreghts, "A literature review of behavior driven development using grounded theory," in *Proc. 27th Twente Student Conf. IT*, 2017.

[49] M. Irshad, J. Börstler, and K. Petersen, "Supporting refactoring of BDD specifications—An empirical study," *Inf. Softw. Technol.*, vol. 141, Jan. 2022, Art. no. 106717.

[50] J. F. Smart, *BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle*, 1st ed. Shelter Island, NY, USA: Manning Publications, 2014.

[51] M. Irshad, R. Britto, and K. Petersen, "Adapting behavior driven development (BDD) for large-scale software systems," *J. Syst. Softw.*, vol. 177, Jul. 2021, Art. no. 110944.

[52] L. P. Binamungu, S. M. Embury, and N. Konstantinou, "Detecting duplicate examples in behaviour driven development specifications," in *Proc. IEEE Workshop Validation, Anal. Evol. Softw. Tests (VST)*, Mar. 2018, pp. 6–10.

[53] I. Romero-Pena, G. Padilla-Zárate, and K. Cortés-Verdín, "Identification of test cases duplication: Systematic literature review," in *Proc. 9th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, 2021, pp. 104–111.

[54] L. Pereira, H. Sharp, C. de Souza, G. Oliveira, S. Marczak, and R. Bastos, "Behavior-driven development benefits and challenges: Reports from an industrial study," in *Proc. 19th Int. Conf. Agile Softw. Develop., Companion*, May 2018, pp. 1–4.

[55] N. Nascimento, A. R. Santos, A. Sales, and R. Chanin, "Behavior-driven development: A case study on its impacts on agile development teams," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops*, Jun. 2020, pp. 109–116.

[56] N. Nascimento, A. R. Santos, A. Sales, and R. Chanin, "Behavior-driven development: An expert panel to evaluate benefits and challenges," in *Proc. 34th Brazilian Symp. Softw. Eng.*, Oct. 2020, pp. 41–46.

[57] R. F. P. Barbosa, "Feature-trace: An approach to generate operational profile and to support regression testing from BDD features," Universidade de Brasília, Brasília, Brasil, Tech. Rep., 2020.

[58] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Improving agile requirements: The quality user story framework and tool," *Requirements Eng.*, vol. 21, no. 3, pp. 383–403, Sep. 2016.

[59] A. S. Dookhun and L. Nagowah, "Assessing the effectiveness of test-driven development and behavior-driven development in an industry setting," in *Proc. Int. Conf. Comput. Intell. Knowl. Economy (ICCIKE)*, Dec. 2019, pp. 365–370.

MUHAMMAD SHOAIB FAROOQ is currently a Professor in artificial intelligence with the University of Management and Technology, Lahore. He was the affiliate member of George Mason University, USA. He possesses more than 28 years of teaching experience in the field of computer science. He has published many peer-reviewed international journals and conference papers. His research interests include the theory of programming languages, big data, the IoT, internet of vehicles, machine learning, blockchain, and education.

UZMA OMER received the Ph.D. degree in computer science from the University of Management and Technology, Lahore, Pakistan. She is currently an Assistant Professor with the Department of Information Sciences, University of Education, Lahore. Her research interests include learning analytics, computer science education, machine learning, the IoT, e-learning systems, and educational technology.

AMNA RAMZAN received the B.S. degree in computing science and the M.S. degree in information technology from the University of Management and Technology, in 2021 and 2023, respectively. Her research interests include information security and cryptography.

MANSOOR AHMAD RASHEED was born in Lahore, Pakistan. He received the B.S. and M.S. degrees in software engineering from the University of Management and Technology, Pakistan. His research interests include blockchain, machine learning, deep learning, requirement engineering, game design and development, and software engineering.

ZABIHULLAH ATAL received the master's degree in information technology from the Virtual University of Pakistan. He is currently an Assistant Professor with the Computer Science Department, Kardan University, Afghanistan. His research interests include computer networks and information security, the IoT, machine, neural networks, smart grid applications and technologies, cloud computing, distributed systems, and blockchain.