Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Unveiling the Benefits and Challenges of Test-Driven Development in Agile: A Systematic Literature Review

Sabar Tampubolon[1], Teguh Raharjo[1]
sabartampubolon@gmail.com, teguhr2000@gmail.com
[1]Faculty of Computer Science, Universitas Indonesia, Jakarta, Indonesia

Article Information

Submitted :27 Mar 2024
Reviewed: 30 Mar 2024
Accepted : 8 Apr 2024

Keywords

Agile, Agile software development, test-driven development, Systematic Literature Review

Abstract

The adoption of Test-Driven Development (TDD) in Agile software development prompts extensive discussion. Advocates highlight its benefits, while skeptics question empirical evidence. This study investigates TDD in Agile settings, examining its merits and challenges. Conducting a systematic literature review, it synthesizes insights from scholarly and industry sources. Results indicate TDD aids development, aligns with Agile practices, and enhances product delivery. Yet, challenges include procedural complexity and skill requirements. Proficiency in Agile practices like refactoring and unit testing is essential. TDD's impact on productivity is moderate and can be counterproductive. This research contributes new perspectives on TDD and Agile development, benefiting academia and informing practitioners for informed decision-making.

A. Introduction

Since Kent Beck's paradigm shift in software development emerged in 2013, there has been extensive research, experiments, and case studies exploring the effectiveness of Test-Driven Development (TDD) in Agile software development [1]. Notably, a prominent study posits that the frequent utilization of TDD in incremental testing not only enhances the quality of the delivered code but also engenders more refined designs [2].

According to Forrest Shull et al., TDD serves as a facilitative tool in uncovering code deficiencies and expediting the development of high-quality remedial measures. Nonetheless, the maintenance and management of TDD-related test cases necessitate greater exertions compared to conventional methodologies. Moreover, TDD engenders code that is inherently comprehensible, methodically structured, and amenable to streamlined maintenance [2].

In contrast, the investigations conducted by Karac I and Turhan B disclose a dissimilar perspective. They reveal that developers often exhibit an inclination to produce an excessive amount of production code, disregarding the imperatives of refactoring and neglecting to align test cases with the evolutionary trajectory of the production code. Additionally, TDD experts exhibit a markedly abbreviated development cycle in contrast to their novice counterparts, and TDD's efficacy thrives when deployed in relation to smaller developmental components [1].

These contradictory outcomes from the studies have stimulated debates concerning the practical deployment of TDD within the Agile software development milieu. Consequently, queries arise pertaining to the underlying rationale behind the adoption of TDD, the accrued benefits thereof, as well as the attendant challenges that demand circumspection during the implementation of TDD.

This study endeavors to expound upon the benefits and challenges associated with the integration of TDD into Agile software development through a systematic scrutiny of the extant literature. By assimilating and evaluating diverse scholarly publications spanning the past decade, this research aspires to furnish nuanced insights into the benefits and impediments intrinsic to TDD practices, particularly within the ambit of the Agile methodology. It is anticipated that the findings derived from this study will furnish invaluable guidance to practitioners, researchers, and software development teams when navigating the terrain of Agile software development practices and their resultant outcomes.

B. Literature Review

Test-Driven Development (TDD)

Test-Driven Development (TDD) represents a groundbreaking approach to software development, encompassing the integration of test-first development (TFD) and refactoring techniques [3], [4]. TDD methodology places utmost emphasis on the creation of test code prior to the production code [3]. This process follows a concise cycle, comprising the following stages [5]:

- Red: The formulation of unit tests for functions or features yet to be implemented.

- Green: The construction of minimalistic production code that successfully passes the unit tests.
- Refactoring: The undertaking of necessary revisions to the unit tests and ensuring the compliance of the production code with the test suite.

Such iterative iterations facilitate improvements to the unit test code, whilst preserving the core functionality. By embracing these phases recurrently, TDD guarantees the development of code of superior quality and comprehensive test coverage [3] (Figure1).

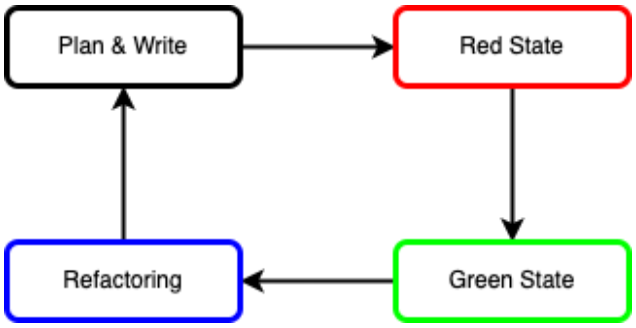

Figure1. Illustration of the Phases of Test-Driven Development

Within the realm of software industry, TDD commonly assumes the role of a best practice in pair programming scenarios. Pair programming entails collaborative coding between two individuals, utilizing a single computer, monitor, keyboard, and mouse. One participant assumes the role of the "driver," operating the computer, while the other takes on the role of the "navigator." These roles interchange at regular intervals [6]. This methodology is adopted to ensure the efficacy of the unit tests employed within TDD.

Furthermore, [3] elaborates on the manifold advantages of TDD, encompassing enhanced code design, increased test coverage, and improved maintainability. Thus, TDD assumes paramount significance in facilitating code refactoring endeavors and managing intricate software complexities.

Agile Software Development

According to the agilealliance.org website, agility refers to the capability of creating and responding to change. It represents an approach for effectively dealing with unpredictable changes in a software development context. It should be noted that agility encompasses more than just frameworks such as scrum, extreme programming, or feature-driven development (FDD); it also encompasses various practices such as sprints, planning sessions, stand-ups, pair programming, and test-driven development (TDD) [7].

Schwaber and Sutherland further emphasize that agile software development is characterized by fundamental principles, namely customer collaboration, responsiveness to change, and iterative development. Agile teams collaborate closely with stakeholders, including customers, users, and business representatives, throughout the development process, fostering effective communication and active participation. This facilitates the gathering of early

feedback, enabling agile teams to adapt and make necessary adjustments based on customer requirements and needs [8].

Ultimately, the adoption of Agile methodologies empowers organizations to enhance their ability to deliver high-quality software products that effectively cater to customer needs and readily adapt to dynamic changes within the market.

Research Questions

Two previously utilized literature sources examine the existing knowledge concerning Test-Driven Development (TDD) in software development. These two studies center their focus on TDD implementation and draw conclusions based on its application.

One of the literature sources concentrates on substantiating the quality of products, internal product quality, and the resulting productivity derived from TDD implementation. This research employs empirical data and expert opinions as evidential support [2].

Conversely, the other literature source revolves around inquiries regarding TDD, encompassing its adoption within the industry, fundamental factors contributing to TDD's success, and deviations from the "test-first" approach of TDD, utilizing evidence spanning a period of 15 years (2003-2018) in TDD's usage [1].

Apart from these two literature sources, a plethora of publications delve into topics such as TDD implementation [9], the learning process of TDD [10], [11], and the significant role TDD plays in software development [5], [12]. Nevertheless, there are also instances where concerns are voiced, asserting that TDD may burden developers without delivering noteworthy effects on productivity [13], [14] and product quality [15].

These circumstances give rise to inquiries regarding the accrued advantages and challenges encountered when utilizing or implementing TDD in Agile software development. The research questions addressed in this study are as follows:

RQ1: What are the benefits of Test-Driven Development in Agile software development?

RQ2: What challenges emerge in conjunction with the benefits of Test-Driven Development in Agile software development?

C. Research Method

The research methodology employed in this study entails a systematic literature review (SLR) based on the guidelines established by Kitchenham [16]. It encompasses a series of seven distinct steps, classified into three principal categories, namely, planning the review, conducting the review, and disseminating the findings (Figure2).

Prior investigations have predominantly concentrated on exploring the effects stemming from the adoption of TDD within the industry and elucidating how TDD engenders advancements in software development practices. This study affords a perspective that the implementation of TDD within diverse industrial contexts yields heterogeneous benefits and challenges [1], [2].

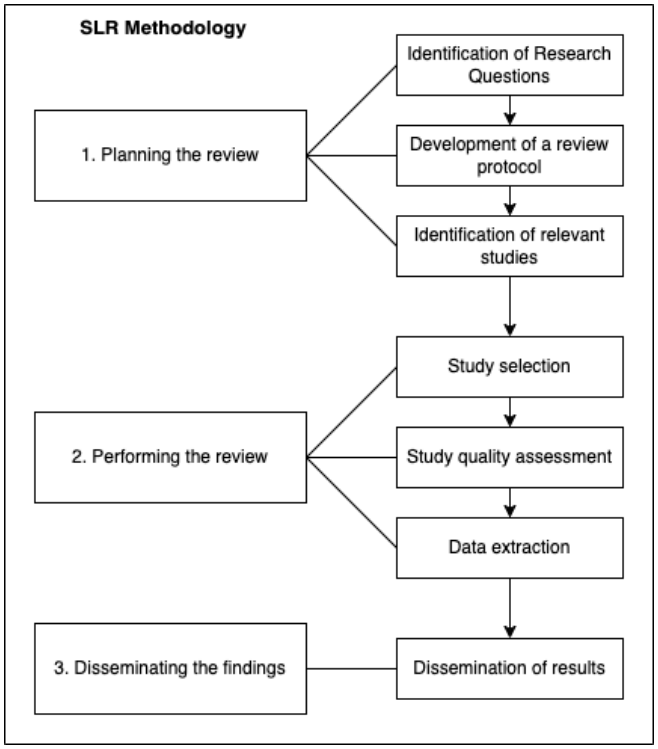

Figure2. SLR Methodology [16]

Identification of Research Questions

The research questions employed in this study are consistent with the previously delineated research inquiries. These two research questions can be amalgamated into the overarching query of "What are the benefits and challenges presented by Test-Driven Development within the Agile paradigm?" The keywords encapsulating this research question encompass the notions of "benefits" (KW1), "challenges" (KW2), "Test-Driven Development" (KW3), and "Agile" (KW4). By harnessing these four keywords, the study restricts its focus solely to the exploration of the benefits and challenges engendered by the application of TDD within the realm of Agile software development methodology.

Development of a review protocol

Based on the extracted research keywords, it is possible to formulate the search keywords. The initial keyword pertains to the benefits of Test-Driven Development (TDD), which can be represented by utilizing keywords such as ("BENEFIT" OR "ADVANTAGE" OR "STRENGTH" OR "USEFULNESS" OR "HELPFULNESS"). The second keyword focuses on the challenges, obstacles, and limitations associated with TDD, and it can be represented by keywords such as ("CHALLENGE" OR "PROBLEM" OR "TROUBLE" OR "ISSUE" OR "OBSTACLE" OR "LIMITATION" OR "BARRIER"). The third keyword relates specifically to TDD and can be represented by using terms like ("TDD" OR "TEST DRIVEN DEVELOPMENT" OR "TEST-DRIVEN DEVELOPMENT"). Finally, the last keyword is "AGILE".

By combining these four keywords, the search can be conducted using keywords such as ("BENEFIT" OR "ADVANTAGE" OR "STRENGTH" OR "USEFULNESS" OR "HELPFULNESS") AND ("CHALLENGE" OR "PROBLEM" OR

"TROUBLE" OR "ISSUE" OR "OBSTACLE" OR "LIMITATION" OR "BARRIER") AND ("TDD" OR "TEST DRIVEN DEVELOPMENT" OR "TEST-DRIVEN DEVELOPMENT") AND "AGILE".

After establishing the search keywords, it is essential to apply inclusion (IN) and exclusion (EX) criteria to the search results. These criteria are crucial for determining the selection of relevant literatures and defining the scope of the chosen literatures. The IN criteria encompass the following: (IN1) studies published between 2013 and 2022, covering the past ten years; (IN2) publications in international journals, proceedings, or conferences; (IN3) availability of full-text access; (IN4) studies written in the English language; and (IN5) studies that focus on Agile methodology and TDD practices or techniques. Conversely, the EX criteria entail: (EX1) excluding opinions, feedback, discussions, and presentations, and (EX2) excluding broader discussions on the topic of TDD and Agile methodology.

Identification of relevant studies

Following the identification of keywords and the establishment of inclusion and exclusion criteria, the subsequent stage entails the selection of appropriate literature databases. A literature search will be executed employing the predefined keywords while constraining the search outcomes based on IN1 and IN2. The study will utilize a range of literature databases, namely ACM Digital Library, Emerald Insight, IEEE Xplore, ScienceDirect, Sage Journals, and Scopus. The comprehensive exploration of these six databases is anticipated to yield a substantial corpus of literatures that meets the requirements of the study.

Study selection

After determining the keywords, inclusion and exclusion criteria, and selecting the literature databases to be utilized, the subsequent step entails conducting a comprehensive literature search based on the provided information. The search will be executed within the chosen literature databases using the predefined keywords. The keyword search will be restricted to the previous decade (2013-2022) and will primarily target international journals, proceedings, and conferences.

Following the conducted search process, a total of 500 international literature sources were identified from six distinct databases, encompassing the publication years between 2013 and 2022. Consequently, the identified literature will undergo a meticulous manual screening procedure based on the inclusion criteria IN3, IN4, and IN5. Each individual literature source will be obtained and scrutinized. Throughout the reading and comprehension phase, the exclusion criteria EX1 and EX2 will also be applied. After the application of these criteria and the subsequent elimination of literature, adhering to IN3, IN4, IN5, EX1, and EX2, a final compilation comprising 103 literature sources will be acquired (Figure3).

Study quality assessment

After the selection of 103 literature sources for this study, the subsequent step entails conducting a study quality assessment to ensure the reliability of the utilized literature. Several crucial points that can be employed in the assessment process include:

- Was the literature search conducted comprehensively?
- Were there any duplications of studies or literature?
- Does the study specifically focus on Agile?
- Does the study discuss or address the usage of TDD in Agile methodology?
- Have endeavors been made to avoid bias?
- Do the conclusions provided rely on the presented data?

Following the completion of the study quality assessment, a total of 37 literature sources have been chosen as the primary literature to be employed in the systematic literature review. The process of literature implementation and elimination can be observed in the accompanying diagram (Figure3).

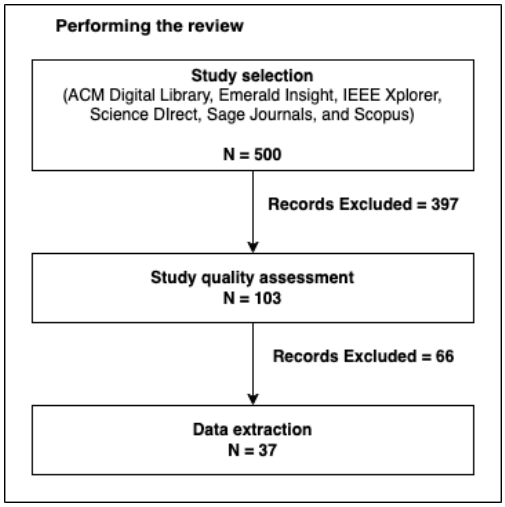

Figure3. Performing the Review

Data Extraction

Based on the search results, a corpus of 500 literature sources was identified across six databases. Through meticulous screening, a total of 397 sources that did not satisfy the predefined inclusion and exclusion criteria were successfully eliminated, resulting in a refined set of 103 literature sources. Subsequently, a thorough study quality assessment was conducted, leading to the exclusion of 66 sources and yielding a final selection of 37 literature sources. These definitive sources were further analyzed based on their publishers and publication years to ascertain their distribution within the study.

In terms of publisher distribution, Information and Software Technology contributed five literature sources, as did the International Conference on Software Engineering and The Journal of Systems & Software, while the International Journal on Empirical Software Engineering and Measurement provided three sources. The remaining publishers each contributed one literature source to the study (Table 1).

Table 1. Distribution of Related Studies

No	Publisher	Quantity
1	Astronomy and Computing	1
2	Brazilian Symposium on Software Component, Architecture, and Reuse	1
3	Business Process Management Journal	1
4	Computer and Information Science	1
5	Conference on Pattern Languages of Programs	1
6	Environmental Modelling & Software	1
7	European Conference on Software Architecture	1
8	Fairness, Accountability and Transparency in socio-technical systems	1
9	Information and Software Technology	5
10	Innovation and Technology in Computer Science Education Conference	1
11	International Computing Education Research Conference	1
12	International Conference on AI Engineering	1
13	International Conference on Software and Systems Process	1
14	International Conference on Software Engineering	5
15	International Conference on System, Control, and Automation	1
16	International Conference on Technical Debt	1
17	International Federation of Automatic Control	1
18	International Journal of Operation & Production Management	1
19	International Journal on Empirical Software Engineering and Measurement	3
20	Software Engineering for Computational Science and Engineering	1
21	Special Interest Group Computer Science Education	1
22	Symposium on Applied Computing	1
23	The Journal of Systems & Software	5

Regarding the distribution by publication year, the majority of the selected literature sources originated from 2013 and 2019, with six sources each (Table 2).

Table 2. Distibution of Studies Publication Year

No.	Study Publication Year	Quantity
1	2013	6
2	2014	3
3	2015	5
4	2016	3
5	2017	2
6	2018	2
7	2019	6
8	2020	3
9	2021	3
10	2022	4

D. Result and Discussion

Upon reviewing the 37 selected final literature, a range of benefits and challenges in implementing TDD in Agile software development were identified. These benefits have the potential to directly impact developers, contributing to process improvement and enhancing the quality of the resulting software. Nevertheless, the encountered challenges often serve as deterrents for developers considering the adoption of TDD in their development practices.

Benefits of TDD in Agile

Based on the analysis of 37 selected literature sources, 28 of them provide insights into the benefits of employing Test-Driven Development (TDD) in Agile

software development. The identified benefits, as categorized in the presented table, encompass various aspects (Table 3).

Table 3. Benefits of TDD Implementation

No.	Category	Related Studies	Total Studies
1	Simplifying works and tasks	[5], [10], [11], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]	20
2	Bring advantages to other development practices and phases	[10], [17], [18], [20], [23], [31], [34], [35], [36], [37]	10
3	Worth pursuing	[5], [10], [11], [18], [24], [28], [33], [38]	8
4	Easy to maintain	[10], [11], [20], [21], [22], [23], [24]	7
5	Simply to learn and apply	[5], [17], [18], [19], [38]	5
6	Works well with clear or unclear requirements	[17], [39], [40]	3
7	Delivering good products on time	[23], [39], [41]	3

The review indicates that TDD can be effectively applied in software development projects, regardless of whether the initial requirements are well-defined or ambiguous [17], [39], [40]. This capability empowers developers in making informed decisions regarding the development process throughout the software implementation phase, thereby ensuring the timely delivery of high-quality products [23], [39], [41].

Additionally, TDD serves as a complementary practice within Agile methodologies and other development phases. Ten literature sources validate this claim, highlighting the value of TDD in facilitating rapid defect identification [10], [18], [23], promoting developers' learning of the development process [18], [20], [23], [31], serving as a strategy for managing Technical Debt [35], [37], streamlining continuous integration and supporting key Agile practices such as unit testing and user stories [10], [17], [20], [23], [31], [34].

In terms of implementation, TDD emerges as an easily acquired and utilized practice [10], [11], [20], [21], [22], [23], [24], suitable for both novice and experienced programmers across various professional levels [5], [17]. Its versatility also extends to managing changes in project requirements and functionalities, thereby fostering adaptability and simplicity in software development tasks [5], [10], [11], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33].

Moreover, TDD represents a valuable practice that warrants further consideration and exploration. Beyond enhancing programming capabilities [5], [10], [11], [18], [24], [28], [38], it offers a vibrant community that can contribute to

enriching developers' knowledge and insights in the realm of software development guided by TDD principles [33].

Challenges of TDD in Agile

Based on the analysis of 37 selected literature sources, 21 of them elucidate the challenges encountered by developers when adopting Test-Driven Development (TDD) in Agile software development. These challenges can be classified according to the categories presented in the following table (Table 4).

Table 4. Challenges of TDD Implementation

No.	Category	Related Studies	Total Studies
1	It requires a larger effort to implement	[5], [17], [21], [32], [36], [42], [43], [44], [45]	9
2	It requires more effort from the developer	[5], [11], [12], [18], [21], [29]	6
3	Good skills are necessary	[10], [11], [12], [38]	4
4	It has a minimal or negligible impact on productivity	[13], [14], [17], [42]	4
5	Willingness to implement is important	[14], [20], [46]	3
6	It requires more repetition	[5], [12], [21]	3
7	It doesn't provide significant help with design	[12], [18], [42]	3
8	Advanced unit testing is essential	[17], [21]	2
9	The quality of results is not up to our expectations	[15], [42]	2
10	Refactoring practice is necessary	[33]	1
11	Pair programming is necessary	[17]	1

The findings of this review draw the conclusion that there are several challenges that demand careful consideration during the implementation of TDD in Agile software development processes. These challenges directly impact developers, as well as influencing the development process and overall productivity. Furthermore, the impact on the quality of the final software product is not significantly substantial.

From a developer's perspective, the integration of TDD into software development necessitates a greater exertion, particularly for novice developers [5], [11], [12], [17]. Its implementation also affects developer performance by necessitating significant skills to maximize the utilization of TDD [10], [21], [32], [38], [43], [44]. Moreover, TDD compels developers to proactively address potential flaws prior to the commencement of the development process [18], [21],

[36]. It is not uncommon for the volume of test code to exceed that of production code, thereby rendering its implementation intricate and time-consuming [5]. Under certain circumstances, managing collective code developed by multiple programmers becomes more arduous when each programmer possesses editing privileges [29]. Consequently, the test code employed may not be promptly updated to reflect comprehensive code changes [42], [45]. These factors generally diminish developer enthusiasm for TDD adoption during software development processes [14], [20], [46].

Within the development process, TDD is frequently coupled with practices such as refactoring [33], unit testing [17], [21], and pair programming [17], all of which are integral to the Agile software development methodology. To optimize the efficacy of TDD, all three practices necessitate meticulous implementation and proficiency. Furthermore, the iterative nature of the Agile methodology mandates the continuous execution of TDD [5], [12], [21].

The ultimate outcomes resulting from TDD implementation do not deviate significantly from the expected standards [12], [18], [42]. The discernible quality of these outcomes is primarily limited to internal processes rather than external manifestations [15], [42], thereby failing to yield a pronounced disparity between TDD and non-TDD software products. Moreover, the review findings indicate that the impact of TDD implementation on productivity is negligibly minimal and, in some instances, may even impede the progress of software development endeavors [13], [14], [17], [42].

E. Conclusion

This study aims to investigate the advantages gained and challenges encountered in the implementation and application of Test-Driven Development (TDD) within Agile software development processes. The benefits and challenges are elucidated through succinct descriptions based on the examination of relevant literature sources.

The findings of this study reveal that out of the 20 literature sources examined, the integration of TDD into Agile software development practices enhances work efficiency and simplifies task execution, as reported by many of the sources. Moreover, a subset of 10 literature sources emphasizes the advantageous impact of TDD on various Agile practices and phases. Furthermore, the literature indicates that TDD exhibits versatility, accommodating both well-defined and ambiguous requirements, exhibiting user-friendliness, facilitating learning and adoption, promoting effective project management, and enabling timely product delivery. Consequently, TDD emerges as a promising practice warranting further in-depth exploration.

Nevertheless, alongside its benefits, the implementation of TDD in Agile software development is not without challenges. Prominent among these challenges is the heightened level of effort demanded, not only in terms of the procedural aspects but also from the participating developers. This demand arises from the requisite proficiency in Agile practices such as refactoring, unit testing, and pair programming. Furthermore, a subset of literature sources highlights that the implementation of TDD may yield only modest improvements in productivity,

and in some cases, it may even have an adverse impact on overall productivity levels within the software development context.

Implications of Study

This study is anticipated to yield valuable implications for both the academic community and practitioners alike. From an academic standpoint, this research aims to provide novel insights and a comprehensive understanding of the utilization of Test-Driven Development (TDD) in the Agile software development industry. By elucidating the associated benefits and challenges, this study contributes to the knowledge base on TDD practices within the industrial domain, warranting further investigation into this approach. For practitioners, this investigation aspires to furnish a thorough overview of TDD in Agile software development, enabling informed decision-making regarding its adoption within their respective organizational development processes.

The findings of this research offer significant contributions to the field of TDD, benefiting scholars and professionals alike. By shedding light on the implementation of TDD in the context of Agile software development, the study explores the benefits and challenges associated with its adoption. Through the provision of pertinent and insightful outcomes, this investigation serves as an invaluable resource for practitioners, empowering them to make well-informed choices concerning the incorporation of TDD.

In sum, this research holds substantial implications for the advancement of knowledge regarding the implementation of TDD in the realm of Agile software development. The resulting implications contribute to the extant literature and expand our comprehension of the utilization of TDD in the dynamic landscape of the software development industry.

Limitation of Study

This study employed literature sources from six literature databases. However, only literature sourced from three databases successfully underwent the rigorous literature selection process, ultimately reaching the final stage. Conversely, literature obtained from the remaining three databases failed to meet the established criteria for literature elimination. Additionally, the initial assumption positing that research conducted within the past decade would sufficiently encompass all relevant aspects for this study proved to be partly inadequate, as there remain diverse facets of TDD warranting deeper investigation through a broader range of case studies.

Future Work

This study centers on the implementation of Test-Driven Development (TDD) in the context of Agile software development, aiming to elucidate the advantages and challenges associated with its adoption. The literature review undertaken sheds light on the disadvantages arising from TDD implementation, shifts in developers' attitudes toward TDD, and the organizational perspectives and acceptance of TDD in contrast to prior non-TDD practices. Moreover, the research scope could be broadened by incorporating literature from additional databases.

F. References

[1] I. Karac and B. Turhan, "What Do We (Really) Know about Test-Driven Development?," *IEEE Softw.*, vol. 35, no. 4, pp. 81–85, Jul. 2018, doi: 10.1109/MS.2018.2801554.

[2] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, and H. Erdogmus, "What Do We Know about Test-Driven Development?," *IEEE Softw.*, vol. 27, no. 6, pp. 16–19, Nov. 2010, doi: 10.1109/MS.2010.152.

[3] K. Beck, *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Professional, 2002.

[4] D. Astels, *Test Driven Development: A Practical Guide*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

[5] S. Romano, D. Fucci, G. Scanniello, B. Turhan, and N. Juristo, "Findings from a multi-method study on test-driven development," *Inf. Softw. Technol.*, vol. 89, pp. 64 – 77, 2017, doi: 10.1016/j.infsof.2017.03.010.

[6] Agile Alliance, "Pair Programming." Accessed: Jun. 16, 2023. [Online]. Available: https://www.agilealliance.org/glossary/pairing/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'pair*20programming))~searchTerm~'~sort~false~sortDirection~'asc~page~1

[7] Agile Alliance, "Agile Alliance." Accessed: Jun. 16, 2023. [Online]. Available: https://www.agilealliance.org/agile101/

[8] K. Schwaber and J. Sutherland, "The Scrum Guide." 2017.

[9] I. B. K. Manuaba, "Combination of test-driven development and behavior-driven development for improving backend testing performance," *Procedia Comput. Sci.*, vol. 157, pp. 79–86, 2019, doi: 10.1016/j.procs.2019.08.144.

[10] M. T. Baldassarre *et al.*, "Studying test-driven development and its retainment over a six-month time span," *J. Syst. Softw.*, vol. 176, p. 110937, 2021, doi: 10.1016/j.jss.2021.110937.

[11] I. Blasquez and H. Leblanc, "Experience in learning test-driven development: Space invaders project-driven," *Annu. Conf. Innov. Technol. Comput. Sci. Educ. ITiCSE*, pp. 111–116, 2018, doi: 10.1145/3197091.3197132.

[12] M. Ghafari, T. Gross, D. Fucci, and M. Felderer, "Why research on test-driven development is inconclusive?," *Int. Symp. Empir. Softw. Eng. Meas.*, pp. 1–10, 2020, doi: 10.1145/3382494.3410687.

[13] C. Matthies, J. Huegle, T. Dürschmid, and R. Teusner, "Attitudes, beliefs, and development data concerning agile software development practices," *Proc. - 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Educ. Training, ICSE-SEET 2019*, pp. 158–169, 2019, doi: 10.1109/ICSE-SEET.2019.00025.

[14] C. H. Duarte, "The quest for productivity in software engineering: A practitioners systematic literature review," *Proc. - 2019 IEEE/ACM Int. Conf. Softw. Syst. Process. ICSSP 2019*, pp. 145–154, 2019, doi: 10.1109/ICSSP.2019.00027.

[15] D. Fucci, "Understanding the dynamics of test-driven development," *36th Int. Conf. Softw. Eng. ICSE Companion 2014 - Proc.*, pp. 690–693, 2014, doi: 10.1145/2591062.2591086.

[16] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S.

Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009, doi: https://doi.org/10.1016/j.infsof.2008.09.009.

[17] G. Scanniello, S. Romano, D. Fucci, B. Turhan, and N. Juristo, "Students' and professionals' perceptions of test-driven development: A focus group study," *Proc. ACM Symp. Appl. Comput.*, vol. 04-08-April-2016, pp. 1422–1427, 2016, doi: 10.1145/2851613.2851778.

[18] F. Besson, P. Moura, F. Kon, and D. Milojicic, "Bringing Test-Driven Development to web service choreographies," *J. Syst. Softw.*, vol. 99, pp. 135–154, 2015, doi: 10.1016/j.jss.2014.09.034.

[19] M. T. Baldassarre, D. Caivano, D. Fucci, S. Romano, and G. Scanniello, "Affective reactions and test-driven development: Results from three experiments and a survey," *J. Syst. Softw.*, vol. 185, p. 111154, 2022, doi: 10.1016/j.jss.2021.111154.

[20] K. Buffardi and S. H. Edwards, "Impacts of adaptive feedback on teaching test-driven development," *SIGCSE 2013 - Proc. 44th ACM Tech. Symp. Comput. Sci. Educ.*, no. 0106, pp. 293–298, 2013, doi: 10.1145/2445196.2445287.

[21] A. Nanthaamornphong, K. Morris, D. W. I. Rouson, and H. A. Michelsen, "A case study: Agile development in the community laser-induced incandescence modeling environment (CLiiME)," *2013 5th Int. Work. Softw. Eng. Comput. Sci. Eng. SE-CSE 2013 - Proc.*, pp. 9–18, 2013, doi: 10.1109/SECSE.2013.6615094.

[22] D. Heaton and J. C. Carver, "Claims about the use of software engineering practices in science: A systematic literature review," *Inf. Softw. Technol.*, vol. 67, pp. 207–219, 2015, doi: 10.1016/j.infsof.2015.07.011.

[23] C. T. Schmidt, S. Ganesha, and J. Heymann, "Empirical insights into the perceived benefits of agile software engineering practices: A case study from SAP," *36th Int. Conf. Softw. Eng. ICSE Companion 2014 - Proc.*, pp. 84–92, 2014, doi: 10.1145/2591062.2591189.

[24] V. Martinez, M. Zhao, C. Blujdea, X. Han, A. Neely, and P. Albores, "Blockchain-driven customer order management," *Int. J. Oper. Prod. Manag.*, vol. 39, no. 6, pp. 993–1022, 2019, doi: 10.1108/IJOPM-01-2019-0100.

[25] L. G. Azevedo, R. Da Silva Ferreira, V. T. Da Silva, M. De Bayser, E. F. De Soares, and R. M. Thiago, "Geological data access on a polyglot database using a service architecture," *ACM Int. Conf. Proceeding Ser.*, pp. 103–112, 2019, doi: 10.1145/3357141.3357603.

[26] R. Mukherjee and K. S. Patnaik, "A survey on different approaches for software test case prioritization," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 33, no. 9, pp. 1041–1054, 2021, doi: 10.1016/j.jksuci.2018.09.005.

[27] E. Guerra, J. Yoder, M. F. Aniche, and M. A. Gerosa, "Test-Driven Development Step Patterns For Designing Objects Dependencies," pp. 1–15, 2013, doi: 10.5555/2725669.2725686.

[28] D. P. Holzworth *et al.*, "Agricultural production systems modelling and software: Current status and future prospects," *Environ. Model. Softw.*, vol. 72, pp. 276–286, 2015, doi: 10.1016/j.envsoft.2014.12.013.

[29] M. Müller, W. Vorraber, M. Herold, C. Schindler, W. Slany, and K. Tanaka, "Streamlining value in a FOSS project," *ACM Int. Conf. Proceeding Ser.*, vol. 2,

pp. 231–234, 2019, doi: 10.1145/3344948.3344976.
[30] B. Hutchinson, N. Rostamzadeh, C. Greer, K. Heller, and V. Prabhakaran, "Evaluation Gaps in Machine Learning Practice," *ACM Int. Conf. Proceeding Ser.*, pp. 1859–1876, 2022, doi: 10.1145/3531146.3533233.
[31] K. Buffardi and S. H. Edwards, "Effective and ineffective software testing behaviors by novice programmers," *ICER 2013 - Proc. 2013 ACM Conf. Int. Comput. Educ. Res.*, pp. 83–90, 2013, doi: 10.1145/2493394.2493406.
[32] R. A. de Carvalho, M. S. de Azevedo, S. C. M. de Souza, G. V. S. Arueira, and C. S. Cordeiro, "Developing and Testing Software for the 14-BISat Nanosatellite," *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 71–74, 2016, doi: 10.1016/j.ifacol.2016.11.128.
[33] S. Romano, F. Zampetti, M. T. Baldassarre, M. Di Penta, and G. Scanniello, "Do Static Analysis Tools Affect Software Quality when Using Test-driven Development?," *Int. Symp. Empir. Softw. Eng. Meas.*, pp. 80–91, 2022, doi: 10.1145/3544902.3546233.
[34] I. F. Da Silva, P. A. Da Mota Silveira Neto, P. O'Leary, E. S. De Almeida, and S. R. D. L. Meira, "Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study," *J. Syst. Softw.*, vol. 88, no. 1, pp. 189–206, 2014, doi: 10.1016/j.jss.2013.10.040.
[35] L. Waltersdorfer, F. Rinker, L. Kathrein, and S. Biffl, "Experiences with technical debt and management strategies in production systems engineering," *Proc. - 2020 IEEE/ACM Int. Conf. Tech. Debt, TechDebt 2020*, pp. 41–50, 2020, doi: 10.1145/3387906.3388627.
[36] A. W. Brown, S. Ambler, and W. Royce, "Agility at scale: Economic governance, measured improvement, and disciplined delivery," *Proc. - Int. Conf. Softw. Eng.*, pp. 873–881, 2013, doi: 10.1109/ICSE.2013.6606636.
[37] W. N. Behutiye, P. Rodríguez, M. Oivo, and A. Tosun, "Analyzing the concept of technical debt in the context of agile software development: A systematic literature review," *Inf. Softw. Technol.*, vol. 82, pp. 139–158, 2017, doi: 10.1016/j.infsof.2016.10.004.
[38] D. Fucci, B. Turhan, N. Juristo, O. Dieste, A. Tosun-Misirli, and M. Oivo, "Towards an operationalization of test-driven development skills: An industrial empirical study," *Inf. Softw. Technol.*, vol. 68, pp. 82–97, 2015, doi: 10.1016/j.infsof.2015.08.004.
[39] P. C. Broekema *et al.*, "Cobalt: A GPU-based correlator and beamformer for LOFAR," *Astron. Comput.*, vol. 23, pp. 180–192, 2018, doi: 10.1016/j.ascom.2018.04.006.
[40] F. J. Behmer and R. Jochem, "Organizational planning for quality management in the digital age," *Bus. Process Manag. J.*, vol. 26, no. 3, pp. 679–693, 2020, doi: 10.1108/BPMJ-12-2018-0365.
[41] C. J. Torrecilla-Salinas, J. Sedeño, M. J. Escalona, and M. Mejías, "Estimating, planning and managing Agile Web development projects under a value-based perspective," *Inf. Softw. Technol.*, vol. 61, pp. 124–144, 2015, doi: 10.1016/j.infsof.2015.01.006.
[42] D. Fucci *et al.*, "An External Replication on the Effects of Test-driven Development Using a Multi-site Blind Analysis Approach," *Int. Symp. Empir. Softw. Eng. Meas.*, vol. 08-09-Sept, 2016, doi: 10.1145/2961111.2962592.

[43] V. Mezhuyev *et al.*, "Acceptance of the methods of decision-making: A case study from software development companies in Ukraine and Malaysia," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1479, pp. 199–204, 2019, doi: 10.1145/3316615.3316677.

[44] M. Irshad, R. Britto, and K. Petersen, "Adapting Behavior Driven Development (BDD) for large-scale software systems," *J. Syst. Softw.*, vol. 177, p. 110944, 2021, doi: 10.1016/j.jss.2021.110944.

[45] V. Golendukhina, V. Lenarduzzi, and M. Felderer, "What is Software Quality for AI Engineers? Towards a Thinning of the Fog," *Proc. - 1st Int. Conf. AI Eng. - Softw. Eng. AI, CAIN 2022*, pp. 1–9, 2022, doi: 10.1145/3522664.3528599.

[46] S. Bellomo, R. L. Nord, and I. Ozkaya, "A study of enabling factors for rapid fielding combined practices to balance speed and stability," *Proc. - Int. Conf. Softw. Eng.*, pp. 982–991, 2013, doi: 10.1109/ICSE.2013.6606648.