European Journal of Social Sciences Studies
ISSN: 2501-8590
ISSN-L: 2501-8590
Available on-line at: www.oapub.org/soc

DOI: 10.46827/ejsss.v9i2.1575 Volume 9 | Issue 2 | 2023

RECONSIDERATION AND PROPOSAL OF DEVELOPMENT MODELS IN PROJECTS - "QUASI" DEVELOPMENT MODELS: QUASI-WATERFALL AND QUASI-AGILE

Takaaki Fujita[i]
Independent Researcher,
Japan

Abstract:
Diverse development models, including waterfall development, iterative development, and agile development, have been put forth and implemented across real-world contexts. When engaging in discussions on project management, the examination and exploration of development models assume paramount importance and are integral. This paper embarks upon an investigation and scrutiny of these development models, culminating in the proposition of "Quasi" Development Models: Quasi-Waterfall and Quasi-Agile.

Keywords: waterfall development, agile development, hybrid development, development models

1. Introduction

1.1 Development Models

Development models, which encompass systematic procedures for managing and executing projects, play a pivotal role in project management. These models guide the tailoring of development processes to match a project's specific needs, propelling the project forward in a coherent and strategic manner. Over time, numerous development models have emerged, each contributing unique perspectives to the body of academic literature on this subject (ex. [1-4, 12-15, 26-35]). Given their importance, a comprehensive discussion and reassessment of these models is of immense significance.

This paper seeks to reevaluate prevailing development models, focusing on significant paradigms such as the waterfall and agile methods. The selection of a development model often varies based on regional and organizational factors. For instance, in Japan, the widely accepted waterfall model reflects cultural characteristics such as a preference for risk aversion and methodical planning. In contrast, the agile model, which promotes adaptability and iterative development, has gained global recognition, demonstrating varying cultural approaches to project management. The

[i] Correspondence email: t171d603@gunma-u.ac.jp

debate between these two models has become a topic of widespread discussion, both in Japan and globally, in recent years (ex. [21-25, 36-47]).

Given this context, it is essential to critically examine and propose alternate development models that cater to the diverse needs of different projects and cultural contexts. This paper introduces the concept of "Quasi" Development Models: Quasi-Waterfall and Quasi-Agile. These models aim to enhance the effectiveness and efficiency of development processes, while aligning with the unique requirements of various regions and organizations. It's crucial to note that this paper represents a theoretical proposal and no empirical case studies have been conducted to test these models. Moreover, the definitions provided herein are conceptual and not definitive.

1.2 Structure of This Discussion Paper

This paper is designed to investigate and evaluate the principles and applications of the waterfall and agile development models. Additionally, it explores the concept of hybrid development, which merges aspects of both agile and waterfall methodologies. Within this hybrid development framework, we introduce the ideas of quasi-waterfall and quasi-agile development. These concepts address the critical question:

- Which development model should form the foundation of a hybrid approach?

The structure of this paper will be explained. In section 2, we will examine waterfall development, agile development, and the integration of these approaches in hybrid development. Section 3 will further explore hybrid development by distinguishing quasi-waterfall development and quasi-agile development, followed by a reexamination. In section 4, we will discuss the conclusion of this paper and future challenges.

2. Re-investigation and Re-examination of Prior Literature on Development Models

In this section, we will conduct an extensive investigation and analysis of the extant literature concerning development models. Our aim is to undertake a meticulous evaluation of the merits and drawbacks associated with each development model. While it is crucial to acknowledge that this overview may encompass broad observations and established facts, we appreciate your understanding.

2.1 Waterfall Development Model

The waterfall development model, initially introduced in scholarly literature in 1988 [5], has since become a prominent fixture in the software development industry, particularly in Japan, where it reigns as the predominant development paradigm.

This development model is distinguished by its linear and sequential methodology, where each project phase follows a strict and predetermined order, reminiscent of water gracefully descending a waterfall. These phases typically encompass requirements definition, system and software design, implementation and unit testing, integration and system testing, and finally, operation and maintenance. Each

phase must be fully completed before proceeding to the next, without the possibility of revisiting previous stages.

Notably, waterfall development finds extensive utilization in large-scale projects that demand meticulous and systematic approaches. This model facilitates a structured trajectory, enabling comprehensive planning, meticulous design, and precise documentation - factors of utmost importance for the triumphant execution of such significant endeavors.

2.1.1 Advantages of Waterfall Development

The advantages of waterfall development are as follows:

- Scale and Duration: Waterfall development is particularly apt for large-scale, long-duration projects. Its linear, phase-by-phase approach lends itself well to complex endeavors that require structured progression.
- Documentation and Planning: This model is beneficial in situations that necessitate comprehensive documentation for traceability, detailed security considerations, and upfront budget allocation. Clear documentation facilitates auditing and accountability.
- Knowledge Transfer: The structured nature of waterfall development allows easy transfer of knowledge when changes in project members occur or the handover of responsibilities is necessary.
- Cost Management: As waterfall development requires thorough planning and design upfront, it enables precise cost estimations and effective budget management.
- Schedule Creation: The model supports the establishment of comprehensive project schedules, providing clarity and assurance to stakeholders.
- Skills Clarity: Given its organized structure, it clearly defines the required skills for each phase, facilitating efficient recruitment during personnel needs.

2.1.2 Disadvantages of Waterfall Development

The problems associated with waterfall development are as follows:

- Uncertainty and Rigidity: Waterfall development can be problematic when detailed requirement gathering and design in the early stages are challenging. This rigidity can impose burdens on later-stage team members, lead to changes in customer requirements, and result in inaccurate cost estimations and financial losses.
- Team Understanding: The strict division between phases might lead to a lack of comprehensive understanding and collaboration among team members, potentially causing disconnects within the project team.
- Change Management: Incorporating new requirements or changes from customers or end-users during the development process is difficult due to the model's rigid structure.

- Rework and Scope Creep: If customer requirements increase beyond the initial plan, it can lead to significant rework. Particularly, changes demanded in the later stages can severely impact project timelines and budgets.

2.2 Agile Development Model

Agile development has emerged as a prominent development paradigm within the software engineering domain in recent years. The principles of this model gained widespread recognition upon the release of the Agile Manifesto for Software Development in 2001 [6]. Currently, a variety of agile methodologies, including Extreme Programming, Scrum, and User-Centered Design, have been extensively embraced and continue to evolve.

Agile development demonstrates particular effectiveness in smaller-scale projects characterized by evolving requirements. In contrast to traditional development approaches that prioritize comprehensive documentation and upfront planning, agile development emphasizes minimal initial design [10]. This shift enables frequent reassessment and iteration throughout the development lifecycle, empowering teams to swiftly adapt to evolving requirements. Moreover, agile projects commonly adopt the practice of releasing incremental deliverables on a weekly or even daily basis, fostering a culture of continuous feedback and improvement. Consequently, agile development endeavors to minimize inefficiencies and enhance productivity by fostering an environment of perpetual learning and iterative development.

2.2.1 Advantages of Agile Development

The advantages of agile development are as follows:

- Scalability and Timeline: Agile methodologies are ideal for small-scale projects with brief development periods due to their iterative nature and fast-paced development cycles.
- Prioritization of Features: Agile development allows prioritization of high-impact features. These features, often driven by customer requirements, can be developed and delivered first, maximizing value early on [7].
- Issue Detection: Agile's iterative approach encourages regular reflection and adaptation, making it easier to surface and address project issues promptly.
- Customer Involvement: Agile emphasizes customer collaboration, enabling the incorporation of real-time feedback from customers and end-users. This active participation leads to a product closely aligned with user needs.
- Error Detection and Adaptation: Agile practices such as continuous integration and testing allow early detection of errors or missing requirements, making it easier to adapt and reduce the cost of change.
- Increased Communication: Agile fosters more frequent and effective communication with customers during the development process. This results in robust information dissemination methods and enables a more accurate incorporation of user perspectives.

- Flexibility: Agile methodologies allow development teams to adapt quickly to changes. For example, they can start coding even before the detailed design is complete, making it suitable for today's fast-paced business environments.

2.2.2 Disadvantages of Agile Development

The disadvantages of agile development are as follows:

- Skill Requirement: Agile development relies heavily on a team of highly skilled technical members. This can create difficulties with resource allocation, especially in environments with limited skill availability, potentially leading to unforeseen increases in personnel costs.
- Documentation: Agile practices emphasize working software over comprehensive documentation. However, inadequate creation of documents and artifacts can sometimes lead to an uncontrolled expansion of requirements, making the project difficult to manage.
- Project Scale: Agile methodology is often employed for small to medium-scale projects due to its flexible and adaptive nature. However, this might constrain the method's applicability to larger, more complex projects, potentially limiting revenue opportunities.
- Contractual Challenges: Agile's iterative approach may be incompatible with fixed-cost or fixed-scope contracts, requiring careful consideration of contract terms and conditions to ensure both parties' expectations align.
- Duplicate Development: Agile's decentralized decision-making could potentially lead to duplicate development of similar functionalities if there isn't effective communication and coordination within the team [11].
- Schedule and Effort Estimation: Agile prioritizes adaptability and customer satisfaction over timeline adherence, which can lead to ambiguity in development schedules and effort estimations [8].
- Integration Efforts: The constant addition of incremental functionalities can make integration efforts time-consuming. However, advancements in development technologies are increasing the automation of integration processes, potentially mitigating this challenge [11].
- Transition Challenges: Companies that predominantly follow waterfall development may face resistance when transitioning to Agile Development. This can be due to the cultural shift required, the perceived risks involved, or a lack of understanding of the potential benefits of Agile practices [9].

2.3 Hybrid Development Model: Integration of Waterfall and Agile Approaches

In recent times, there has been a surge of interest in the hybrid development model, which amalgamates elements from both agile and waterfall development methodologies. This hybrid model represents a pragmatic and field-centric approach, offering promising prospects for further advancements.

For instance, within the scope of this paper, hybrid development refers to the assimilation of agile practices into each phase of the waterfall model, encompassing requirements definition, external design, and internal design. By conducting iterative cycles of agile-based requirements definition, the potential for ensuring quality assurance is augmented. Furthermore, it is also feasible to merge requirements definition and external design phases, followed by iterative cycles of agile-based requirements definition and design. The crux of hybrid development lies in tailoring the development approach to suit the specific requirements of the project. Research on hybrid development is currently thriving [16-20, 48-50].

Below, we briefly mention the advantages and disadvantages. However, it is important to note that the advantages and disadvantages can vary depending on whether the field is more adept at agile or waterfall development, so not all aspects will be covered in detail.

2.3.1 Advantages of Hybrid Development

By combining waterfall and agile approaches, hybrid development can create a development model that is more closely aligned with the actual project requirements. It allows for easier tailoring to suit specific project needs.

2.3.2 Disadvantages of Hybrid Development

Unlike waterfall and agile approaches, hybrid development lacks a well-established theoretical foundation. As a result, there is a possibility of a chaotic development process without clear guidelines.

3. "Quasi" Development Models: Quasi-Waterfall and Quasi-Agile

3.1 About "Quasi"

When discussing project management, it is essential to consider the concept of the critical path. Alongside the critical path, there exists a concept known as the quasi-critical path. The quasi-critical path refers to the path that should be chosen when the actual critical path cannot be determined. It represents the path that closely aligns with the critical path but is slightly less time-constrained.

In this paper, we introduce the concept of "quasi" development methods, which are development approaches that encompass elements from both traditional waterfall and agile methods. For instance, implementing agile development within each phase of the waterfall development process (e.g., requirements definition, external design, internal design) can be regarded as a quasi-waterfall approach. Conversely, in scenarios such as student projects, agile-oriented development methods are often employed. A development method that integrates rigorous schedule management and meticulous requirements definition into agile-oriented practices can be referred to as quasi-agile development.

3.2 The Necessity of Quasi-Waterfall and Quasi-Agile Development

The concept of "quasi" development approaches emerges as a response to the evolving needs of different project environments. These environments may require a shift in the development methodology employed, making a strictly waterfall or agile approach inadequate.

For instance, large corporations often handle substantial projects, which may naturally align with the structure and stability offered by the waterfall model. However, transitioning abruptly from a rigid waterfall structure to a flexible agile one can be a daunting task. Instead, it may be more practical to gradually incorporate agile practices into specific phases of a waterfall model. This approach could allow organizations to harness the benefits of both methodologies, thereby optimizing their development process.

Conversely, for smaller-scale projects, such as those typically conducted in research settings or by university students, a full-scale waterfall development model may be excessive. Instead, a more flexible approach that draws from agile principles might be preferable. In these contexts, integrating methodologies from waterfall development like Work Breakdown Structures (WBS), Gantt charts, and meticulous task and schedule management, along with elements of agile development such as iterative requirement definitions and design phases, could help mitigate the occurrence of rework and schedule delays.

By critically evaluating the project environment and carefully considering which development model to base the project on, it becomes feasible to tailor the development process accordingly. This flexibility enhances our ability to provide clear direction and guidelines that best suit the project's unique needs.

3.3 Quasi-Waterfall Development Model

As previously mentioned, the quasi-waterfall development model combines the systematic and structured approach of the traditional waterfall model with the adaptability inherent in agile techniques. The agile principles of user stories, iterative development, regular feedback, and the ability to adapt to change are seamlessly integrated into the rigid structure of the waterfall model. This harmonious integration gives the model its distinctive "quasi" nature, providing a versatile methodology that leverages the benefits of both waterfall and agile practices.

3.3.1 Requirements Definition

Agile's user story technique is used here. User stories help in gathering and refining requirements from the user's perspective. Instead of a one-time requirement-gathering process, agile's iterative requirement refinement is adopted. This iterative process allows ongoing input from stakeholders, creating opportunities to adapt and respond to changes in requirements.

3.3.2 System and Software Design

Design principles from agile methodologies are introduced. Instead of finalizing the entire system's design in one phase, collaborative and emergent design practices are employed. This means the design evolves over time, informed by team collaboration and lessons learned during development, promoting adaptability and flexibility.

3.3.3 Implementation and Unit Testing

The traditional waterfall model would have this as a separate phase, strictly after the completion of the design phase. However, here we introduce agile's iterative development, and continuous integration. Regular, short development cycles (sprints) are conducted, at the end of which pieces of functionality are delivered. Regular feedback during these sprints helps in early detection and rectification of errors.

3.3.4 Integration and System Testing

In traditional waterfall, this phase happens after all the development is done, but in quasi-waterfall, agile's continuous integration and testing are utilized. This approach ensures all the developed pieces work together efficiently, and problems are identified and fixed in an ongoing manner, reducing the risks and efforts of a post-development testing phase.

3.3.5 Operation and Maintenance

This phase sees the incorporation of the agile principle of 'responding to change'. Even in the operation and maintenance phase, changes to the system are addressed in an iterative manner, ensuring the system continuously evolves and remains up-to-date.

By incorporating agile practices within each phase of the waterfall model, the quasi-waterfall development model can cater to projects needing a clear, linear structure while benefiting from the flexibility and adaptability of agile development.

3.4 Quasi-Agile Development Model

As previously mentioned, the Quasi-Agile Development Model encompasses the iterative nature of agile methodologies while integrating comprehensive planning and meticulous schedule management from waterfall approaches. This hybrid model mitigates potential pitfalls encountered in pure agile practices, such as scope creep and schedule overruns, thereby providing a more controlled and controlled agile experience.

3.4.1 Pre-iteration

In traditional agile methodologies, requirements, and planning are typically addressed in a "just-in-time" and flexible manner. However, the quasi-agile model introduces an upfront phase dedicated to detailed requirements analysis and schedule planning. Requirements are gathered meticulously, with a strong emphasis on documentation, akin to the waterfall model. A comprehensive project schedule is also formulated, delineating the sequence, duration, and interdependencies of iterations.

3.4.2 Iterative Development

This phase mirrors the essence of traditional agile development, featuring iterative cycles (sprints) where design, development, and testing occur concurrently. Nevertheless, each sprint is guided by the detailed roadmap derived from the pre-iteration phase. This approach provides a clear vision and minimizes deviation from the project scope.

3.4.3 Continuous Integration

Continuous integration, a pivotal agile practice involving frequent code integration and testing, remains integral to the quasi-agile model. This approach facilitates early defect detection and mitigates integration risks. While not exclusive to the quasi-agile model, the combination of continuous integration with the detailed upfront planning from the waterfall model sets it apart.

3.4.6 Release Planning

Unlike pure agile approaches, wherein releases are often flexible and driven by the product owner's priorities, quasi-agile adheres to a meticulously defined release plan formulated during the pre-iteration phase. Each iteration aligns with specific planned releases, ensuring the timely development and delivery of critical features.

3.4.7 Post-iteration

In traditional agile practices, retrospectives predominantly focus on process improvement. In the quasi-agile model, retrospectives are accompanied by meticulous updates to the overall project schedule. Changes in the project plan, potential risks, and lessons learned from previous iterations are documented and utilized to refine the project roadmap.

The quasi-agile model represents a harmonious amalgamation of the flexibility and customer collaboration offered by agile methodologies with the comprehensive planning and control associated with the waterfall model. This balanced approach enhances efficiency and effectiveness in projects that necessitate a higher degree of control and predictability without sacrificing the inherent benefits of agile practices.

4. Conclusion

4.1 Summary of This Short Paper

This short paper investigated and explored waterfall development and agile development. Furthermore, the paper discussed hybrid development and proposed the concept of quasi-waterfall and quasi-agile development, albeit in a broad manner.

4.2 Future Directions

As this paper primarily serves as a proposal, future plans entail the implementation of concrete case studies and quantitative analyses. It is also being contemplated to establish precise definitions and undertake empirical investigations to distinguish Quasi-Waterfall

and Quasi-Agile Development from other hybrid development approaches. Moreover, it is crucial to tackle the following challenges: scrutinizing techniques for customizing quasi-waterfall and quasi-agile development, presenting theoretical frameworks, and exploring the incorporation of the "quasi" concept within other project management methodologies.

Acknowledgements
I humbly express my sincere gratitude to all those who have extended their invaluable support, enabling me to successfully accomplish this paper.

Conflict of Interest Statement
The author declares no conflicts of interest.

About the Author
Takaaki Fujita finds great enjoyment in his work as a system engineer/IT service manager, as he is constantly involved in diverse system development projects. Additionally, he delights in exploring various fields of mathematics, including discrete mathematics, combinatorics, algebra, and graph theory, alongside his keen interest in ICT education and project management (cf. [51-52]).

References

[1] Morimoto, S. (2007). Support research and challenges towards practicalization in software development processes. Journal of the Graduate School for Advanced Studies in Industrial Technology, 1, 105-110.
[2] Sakamoto, N. (2010). Agile development that brings happiness: Differentiating from waterfall development. Nikkei Computer, 749, 120-123.
[3] Homma, M. (2014). Limitations of waterfall development and expectations for evolutionary prototype development. System/Control/Information, 58(6), 227-232.
[4] Dohi, R. (2012). Why is prototype development not adopted in government IT procurement? Abstracts of the Spring National Conference of the Japan Society of Information and Management, 262.
[5] DOD-STD-2167A, Military Standard: Defense System Software Development. United States Department of Defense. 29 Feb 1988.
[6] Agile Manifesto. (nd.). Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/iso/ja/manifesto.html
[7] Yatsumi, T., & Kitachuu, H. (2019). Consideration on the structural changes brought about by "design thinking" in IT system development. Abstracts of the Spring National Conference of the Japan Society of Information and Management, 185-188.

[8] Furuta, M. (2015). Application of agile development methods in the development of core business systems in the finance and insurance industries. Fujitsu, 66(3), 63-68.
[9] Yoshida, C. (2015). Study on estimation and contract models for agile development. PhD Paper, University of Tsukuba.
[10] Yamane, S. (2011). Theory and practice of game development in higher education: Using Global Game Jam as an example. Research Reports on Computers and Education, 2011(5), 1-6.
[11] Oba, M. (2018). Globalization of the economy and the necessity of agile development. Economic Science Studies, 21(1-2), 129-156.
[12] Petersen, Kai, Claes Wohlin, and Dejan Baca (2009). The waterfall model in large-scale development. Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15-17, 2009. Proceedings 10. Springer Berlin Heidelberg, 2009.
[13] McCormick, Mike (2012). Waterfall vs. Agile methodology, MPCS, N/A 3
[14] Balaji, Sundramoorthy, and M. Sundararajan Murugaiyan (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. International Journal of Information Technology and Business Management 2.1: 26-30.
[15] Dima, Alina Mihaela, and Maria Alexandra Maassen (2018). From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. Journal of International Studies (2071-8330) 11.2 (2018).
[16] Belling, Shawn (2020). Succeeding with Agile Hybrids: Project Delivery Using Hybrid Methodologies. Apress.
[17] Mahadevan, Lakshman, William J. Kettinger, and Thomas O. Meservy (2015). Running on hybrid: Control changes when introducing an agile methodology in a traditional "waterfall" system development environment. Communications of the Association for Information Systems 36.15.
[18] Schuh, Günther, et al. (2017). Agile-waterfall hybrid product development in the manufacturing industry—Introducing guidelines for implementation of parallel use of the two models. 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE.
[19] Reiff, Janine, and Dennis Schlegel (2022). Hybrid project management–a systematic literature review. International journal of information systems and project management 10.2 45-63.
[20] Yahya, Norzariyah, and Siti Sarah Maidin (2022). The Waterfall Model with Agile Scrum as the Hybrid Agile Model for the Software Engineering Team. 2022 10th International Conference on Cyber and IT Service Management (CITSM). IEEE, 2022.
[21] Stoica, Marian, et al. (2016). Analyzing agile development-from waterfall style to Scrumban. Informatica Economica 20.4 (2016): 5.
[22] Mishra, Alok, and Yehia Ibrahim Alzoubi (2023). Structured software development versus agile software development: a comparative analysis. International Journal of System Assurance Engineering and Management (2023): 1-19.

[23] Jangra, Keshav, Simran Yadav, and Vimmi Malhotra (2023). Optimizing The Product Development Process for Speed to Market, Quality, And Customer Satisfaction: A Comparative Study of Agile and Waterfall Methodologies. International Research Journal of Modernization in Engineering Technology and Science, 5(4), Retrieved from https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2023/36854/final/fin_irjmets1682413432.pdf

[24] Dursun, Mehtap, and Nazli Goker (2022). Evaluation of project management methodologies success factors using fuzzy cognitive map method: waterfall, agile, and lean six sigma cases. International Journal of Intelligent Systems and Applications in Engineering 10.1: 35-43.

[25] Kodmelwar, Manohar K., et al. (2022). A comparative study of software development waterfall, spiral, and agile methodology. Journal of Positive School Psychology 6.3 (2022): 7013-7017.

[26] Clear, Tony (2003). The waterfall is dead... long live the waterfall!!." ACM SIGCSE Bulletin 35.4: 13-14.

[27] Yang, Fangkun (2013). How to Migrate from Waterfall. Development Approach to Agile Approach.

[28] Van Casteren, Wilfred (2017). The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. Research Gate 2: 1-6.

[29] Stoica, Marian, Marinela Mircea, and Bogdan Ghilic-Micu (2013). Software development: agile vs. traditional. Informatica Economica 17.4.

[30] Fitzgerald, Brian, Nancy L. Russo, and Tom O'Kane (2003). Software development method tailoring at Motorola. Communications of the ACM 46.4: 64-70.

[31] Sharma, Sheetal, Darothi Sarkar, and Divya Gupta (2012). Agile processes and methodologies: A conceptual study. International journal on computer science and Engineering 4.5: 892.

[32] Mohanarangam, Karthik (2020). Transitioning to agile—in a large organization. IT Professional 22.2: 67-72.

[33] Salnikov, Nikita (2021). How software development methodologies affect dynamic capabilities under extreme contexts: a COVID-19 study on agile and waterfall methodologies. MS thesis

[34] Gheorghe, Alina-Mădălina, Ileana Daniela Gheorghe, and Ioana Laura Iatan (2020). Agile Software Development. Informatica Economica, 24.2

[35] Liu, Tong (2022). Application of Agile Project Management in Software R&D Management of M Enterprise. Frontiers in Computing and Intelligent Systems 1.3: 85-87.

[36] Khoza, Lucas T., and Carl Marnewick (2020). Waterfall and agile information system project success rates-a South African perspective. *South African Computer Journal* 32.1: 43-73.

[37] Kettunen, Janne, and Miguel A. Lejeune (2020). Waterfall and agile product development approaches: Disjunctive stochastic programming formulations. *Operations Research* 68.5: 1356-1363.

[38] Kettunen, Janne, and Miguel A. Lejeune (2020). Waterfall and agile product development approaches: Disjunctive stochastic programming formulations. *Operations Research* 68.5: 1356-1363.

[39] Mousaei, T. M. (2020). Review on Role of Quality Assurance in Waterfall and Agile Software Development. *Journal of Software Engineering & Intelligent Systems* 5 (20.

[40] Farahat, Abdallah Magdy, and Domenico Defina (2022). Novel Adaptive Approach for Applying and Combining Traditional Waterfall and Agile Project Management Methodologies. *Abu Dhabi International Petroleum Exhibition and Conference*. SPE.

[41] Ben-zahia, Mehemed Abdusalam Omer, Ali Aburas, and Miloud Ghawar (2022). The Challenges of Software Development: Waterfall and Agile. Libyan International Conference for Applied Science and Engineering.

[42] Maslyuk, O. O., and K. A. Alekseieva (2019). Differences between Waterfall and Agile Project Management. *Збірник Матеріалів*: 81.

[43] Chan, Lok Shan (2020). Selection of Waterfall and Agile Methodologies in Software Testing.

[44] Fagarasan, C., et al. (2021). Agile, waterfall, and iterative approach in information technology projects. *IOP Conference Series: Materials Science and Engineering*. Vol. 1169. No. 1. IOP Publishing.

[45] Mishra, Alok, and Yehia Ibrahim Alzoubi (2023). Structured software development versus agile software development: a comparative analysis. International Journal of System Assurance Engineering and Management: 1-19.

[46] Giavarina, Giacomo (2021). *Analysis of a hybrid project management approach between Waterfall and Agile. Case study in the European automotive sector*. Diss. Politecnico di Torino.

[47] Kikui, Takahiro (2023). Sequential decision making for specification changes during system development Comparing Waterfall and Agile Method." *Abstracts of Annual Conference of Japan Society for Management Information Annual Conference of Japan Society for Management Information 2022*. The Japan Society for Management Information (JASMIN), 2023.

[48] Bengdara, Hamdy (2021). Hybrid Approaches of Project Management.

[49] Wankhede, Rashmi (2016). Hybrid Agile Approach: Efficiently Blending Traditional and Agile Methodologies.

[50] Tip, Plaky Pro. "What is hybrid project management?."

[51]Fujita, T. (2023). Revitalizing Education Through ICT: A Short Overview of Japan's Current Landscape. European Journal of Social Sciences Studies, 8(5).

[52] Fujita, T. (2023). Breaking Down Barriers: Proposals for Overcoming Challenges in Student Project Management. European Journal of Management and Marketing Studies, 8(2).