Volume 10, Issue 1, January – 2025 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14716967

An Efficient Software Methodology with Structured Decision-Making Technique for the Software's Design

Jubayer Ahamed*
Dept. of Computer Science
American International University-Bangladesh
Dhaka, Bangladesh

Barno Biswas
Dept. of Computer Science
American International University-Bangladesh
Dhaka, Bangladesh

Sumshun Nahar Eity
Dept. of Computer Science
American International University-Bangladesh
Dhaka, Bangladesh

Farhana Afroz
Dept. of Social Science
American International University-Bangladesh
Dhaka, Bangladesh

Abstract:- Selecting an efficient methodology is crucial for ensuring project success in the various fields of software development. Software projects are often complex, requiring structured decision-making to address varying technical and non-technical challenges. The efficient project design requires a systematic and flexible methodology with structured decision-making technique in the fast-changing area of software development. The complicated nature of current software systems, coupled with diverse stakeholder demands and constant technical progress, requires a decision-making framework that effectively combines creativity and precision. Our study defines how effective software methodology is and tries to propose an efficient model with decision-making technique for the software's design. It identifies some issues from previous research and examines a sequential model that focusses on enhancing software design processes, improving decision-making across various project contexts and providing the solutions of the issues. The methodology highlights improving the efficiency of the software design process by distinctly defined phases, applying tools such as UML diagrams and stakeholder involvement to guarantee clarity and adaptability. Utilizing a systematic decision-making approach enables teams to adeptly manage restrictions, dependencies and resource limitations. Moreover, it enhances scalability and versatility, producing it appropriate for various sectors and project dimensions. This study focuses on the necessity of synchronizing new design ideas with practical implementation, providing an efficient model for producing flexible and effective software design solutions modified to satisfy evolving user requirements.

Keywords:- *Software Methodology; Decision-Making Approach; Software Design Solutions; Hybrid Framework; Decision Nature.*

I. INTRODUCTION

The rapid advancement of technology and the growing demand for excellent software have transformed the planning, development, and delivery of software projects. As businesses aspire for accelerated time-to-market and flexible user experiences, the selection of the optimal software development process has emerged as a critical determinant of project success. The design phase of a software project, which establishes the groundwork for the eventual product, demands a systematic process that corresponds with project goals, team skills and customer requirements. Nevertheless, selecting the appropriate approach is not simple, as each project contains distinct requirements regarding scope, resources, timeframes, and complexity. It is essential to provide a framework that integrates an effective software methodology with a strong decision-making approach to enhance the design and implementation of software projects.

Software methodologies like Agile, Scrum, Waterfall, and Lean present unique benefits and drawbacks. Agile prioritizes adaptability and ongoing cooperation, whereas Waterfall offers a systematic, linear methodology [1]. Conversely, Scrum emphasizes iterative development via time-constrained sprints. Despite the widespread use of these approaches, their effective application to specific projects needs careful consideration of several criteria, including project nature, team competence, and risk tolerance. The impracticality of a one-size-fits-all strategy has resulted in the development of hybrid models that integrate many approaches to address complicated project requirements. These hybrid methodologies necessitate organized decision-making frameworks to ensure their successful selection and application [2].

Decision-making techniques, including decision matrices, analytic hierarchy processes (AHP), and multi-criteria decision analysis (MCDA), provide structured approaches to evaluate and choose strategies based on established project objectives and limitations. These frameworks assist project managers in evaluating several variables, including time, cost, complexity, and stakeholder expectations, so assuring coherence between the selected approach and the intended project results. Effective decision-making not only reduces risks but also ensures improved resource utilization and project efficiency during the design phase, where essential architectural and functional decisions are established [4].

This study analyzes how effective software methodology, when integrated with systematic decision-making technique, can improve the design and development of software projects. It aims to discover the factors that influence methodology selection and analyze decision-making models which promote effective project design. This research seeks to offer insights into creating an integrated model that matches methodology selection with strategic project objectives, so ensuring efficient and high-quality software design.

In section 2, we discuss the previous works on software methodology with structured decision-making technique for the software's design and the design thinking (DT) using software methodologies. The proposed methodology section will be described in Section 3. In section 4, we set the conclusion part.

II. BACKGROUND STUDY

A. Software Architecture, Design Processes and Methodologies

M. Adil, I. Fronza et al. [3] demonstrated the process of software design and modeling as applied by students during the online software engineering course. Their research dealt with the distributed teams of students performing software design activities with the aid of Scrum and other collaborative software tools [3].

I. Lytra, C. Carrillo et al. [4] described the use of quality attributes in guiding software architecture design decisions. The building and rationale for high quality systems using such attributes were also discussed by the authors [4].

Another study described the huge troubles faced during the migration from Waterfall to Agile approaches [5]. M. Stoica, B. Ghilic-Micu, M. Mircea, and C. Uscatu et al. [6] focused on the explanation of how it is great to change from the Waterfall model to Agile development and from the usage of Scrumban helps to improve project versatility and teamwork.

T. Natarajan, S. Pichai et al. [7] outlined an action research study, on an extensive implementation analysis of the movement from Waterfall to Agile approaches in software development. The methodological limitations of the study consisted of a contextual nature of the collected data, which could be a limitation to the generalizable results [7].

Another research [8] analyzed how enforcement of standard prescriptive forms of documentation affects the creativity and open-endedness of the SW development process by opinionating that standardized forms may hamper necessary intellect and restrict the search for originality. J. Ahamed and D.Nandi et al. [9] described a new decision-making method to enhance the software architecture design. The paper highlighted decision-making in architecture because it outlines the structure and nature of software systems.

K. Kilova, V. Lazarova et al. [10] proposed a modern approach to designing software architecture for monitoring and quality assessment in higher education. Also, another study [11] addressed challenges in the software design process and proposes a sustainable procedural model aligned with Capability Maturity Model Integration (CMMI). This model maps out incremental stages from initial design to optimized processes which guide software teams to improve design reliability, efficiency, and scalability within the Software Development Life Cycle (SDLC) [11].

F. K. Y. Chan and J. Y. L. Thong et al. [12] focused on analyzing factors affecting the adoption of agile methodologies within organizations. It explored the existing frameworks, user acceptance challenges, and the benefits and limitations of agile methods in software development. One study [13] analyzed trends and characteristics within agile methodologies. It has provided a comprehensive review of agile frameworks, particularly in software development and focused on popular models like Scrum, Kanban, and Extreme Programming (XP).

K.N. Mohammed and S.C. Karri et al. [14] explored the Agile methodology for construction project management that has been associated with major problems, such as schedule delays and budget overruns. They also applied Agile practices that are normally used in IT projects to construction since construction projects experience some effects of design changes and process changes. It recommends sprints and a structured framework for managing changes within construction projects to ensure greater coordination and adaptability.

A.A.A. Adenowo and B.A. Adenowo et al. [15] analyzed a critical review of two prominent software engineering methodologies, named the Waterfall model and the Object-Oriented approach. It points out that the waterfall model is structural and sequential in nature; thus, it is ideal for projects whose requirements are already well-defined. The paper has been effective in comparing the Waterfall model and the OO approach, but it has failed to provide a detailed exploration on hybrid models, which possess positive features of both methodologies [15].

V. Chandra et al. [16] discussed and analyzed famous software development methodologies: the Waterfall, Agile, and Spiral models. The author has examined each methodology's framework and its corresponding phases and key characteristics with a focus on aspects such as flexibility, cost-efficiency, and risk management. Future studies might consider the use of actual case studies or quantitative methods for evaluating a success rate, costs, and time of delivery for each approach. It is also likely that hybrid models will yield other insights into adaptation to evolving project demands.

S. Balaji and Murugaiyan et al. [17] analyzed three major SDLC models: Waterfall, V-Model, and Agile. Each of the models is assessed for the most important parameters like project size, client requirements, and flexibility. The study provided only an overview of SDLC models but does not delve deeply into the hybrid models that are combinations of these approaches [17]. One study [18] has done a detailed analysis of the Agile and Waterfall models in SDLC and compared both according to their advantages, limitations, and specific applications in software quality engineering.

B. *Software Design thinking (DT) using Software Methodologies*

Software Design Thinking (DT) can be combined with software development methods like Agile, Scrum, and Extreme Programming (XP). This can help connect software solutions with what end users want. Several studies show that combining DT with standard development frameworks can be helpful.

Canedo et al. [19] analyzed how decision-makers felt about using Design thinking (DT) tools in Agile software projects. In the end, they found that DT improved teamwork and results that were focused on the user. The study only looked at short-term projects, though, which makes me wonder if DT can keep working well in large-scale, long-term Agile projects.

Steinke and Al-Deen et al. [20] investigated DT along with the Waterfall and Agile methods. However, their study was limited to theoretical frameworks and did not include any real-world validation in big enterprise settings. This suggests that there is a need for more real-world industry applications.

Parizi et al. [21] made a tool that helps people choose which Design thinking (DT) frameworks to use for gathering software requirements. This study gave information about which techniques work best, but it didn't investigate how well these tools can be used in large, complicated business processes.

Through iterative prototyping and involving users, Pereira and Russo et al. [22] highlighted the part that DT plays in making Agile software development better. The study showed that DT and Agile work well together, but it didn't look at any of the problems that might come up because DT is iterative and works so quickly in Agile settings.

Previous research, like [19], [21], mostly looked at small projects or certain types of organizations. But when the project is large, then it will be faced with some difficulties.

Sohaib et al. [23] examined how DT could be used with Extreme Programming (XP) and found that it increased creativity and satisfaction with customers. But their work was not directed at how to handle problems that come up when DT's focus on exploration and XP's focus on fast delivery clash.

Wangsa et al. [24] examined how Design thinking (DT), Agile, and Design Sprint work and compared them. The study focused on DT's strength in human-centered design but didn't give any useful advice on how decision-makers can choose between these approaches based on the needs of the project. That is why they will need useful decision models that can help teams choose the best methods. Gama et al. [25] investigated how DT is used in hackathons and found that DT tools help people be more creative when they must compete for time. But their study only looked at the short-term effects of hackathons [25]. This means that we cannot say much about the long-term effects of DT methods used in these settings.

C. *State-of-the-Art Innovations in Software Methodology with Structured Decision-Making Technique*

This section highlights ten of the most innovative and impactful recent contributions, summarizing their approaches, proposed systems, key findings, and observations.

Table 1: A Comparative Analysis of Different Research Works

Category	**Relevant Studies**	**Description**	**Identified Gaps**
Agile Adoption and Challenges	Chan et al. [12], Trihardianingsih et al. [13], Adenowo et al. [15]	Difficulties in implementing Agile in non-technical sectors and scaling practices in larger organizations.	Need for frameworks addressing non-technical contexts and large-scale Agile scaling challenges.
Hybrid Methodologies	Balaji et al. [17], Pargaonkar et al. [18]	Comparison of Waterfall, Agile, and hybrid methodologies for different project contexts.	Limited exploration of hybrid models' effectiveness and their application in industry-specific scenarios.
Methodological Efficiency	Balaji et al. [17], Pargaonkar et al. [18]	Challenges in optimizing SDLC methods for specific regulatory and risk-heavy industries.	Need for exploration of SDLC adaptations, especially for regulated fields.
Scalability in Large Projects	Canedo et al. [19], Parizi et al. [21]	Challenges in applying DT effectively in large-scale, long-term software projects.	Limited studies on how DT scales in enterprise-level and complex projects.
Lack of Real-World Validation	Steinke and Al-Deen [20]	Most studies use theoretical frameworks or case studies without real-world validation.	Need for empirical evidence and real-world industry applications in various sectors.
Decision-Making Support	Sohaib et al. [23], Wangsa et al. [24].	Difficulty in choosing appropriate methodologies based on project requirements.	Absence of specific decision methodological framework to help teams select suitable methods for specific needs.
Balancing Creativity and Delivery	Sohaib et al. [23]	Conflict between DT’s exploratory nature and fast delivery in XP or Agile settings.	Need for frameworks to balance creativity and speed in delivery-focused methods.

III. PROPOSED METHODOLOGY

The proposed model integrates iterative and flexible software development processes with structured decision-making techniques. It is designed to address scalability, adaptability, and decision-making challenges in software projects of varying complexity.

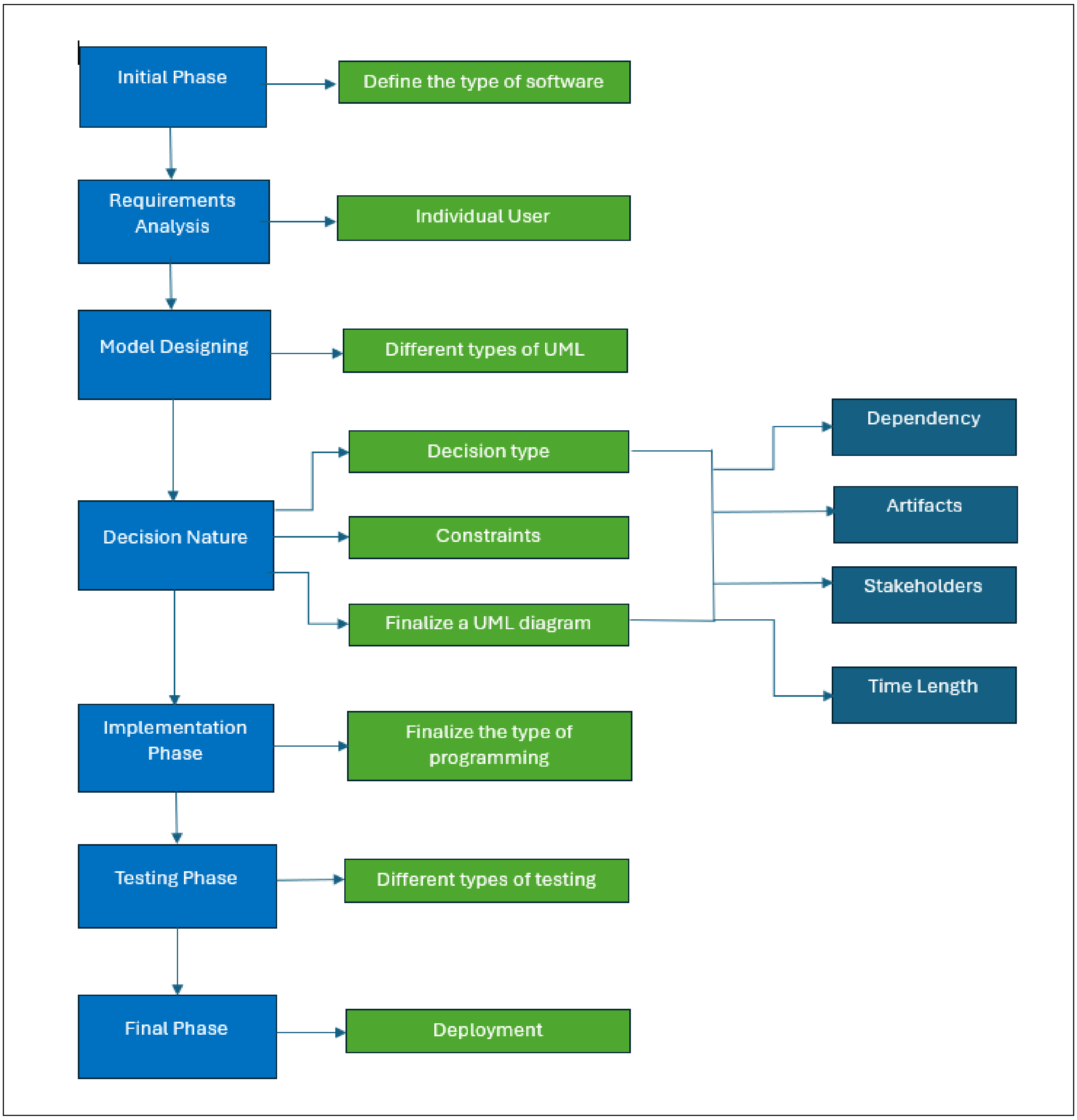

Fig 1: Proposed Software Model

A. Model Description

The diagram of our proposed methodology shows a systematic software development process, divided into continuous phases. Below is a comprehensive explication of each phase:

➢ *Initial Phase*

- *Define the Type of Software:*

This phase highlights the identification of the software's purpose, domain, and overarching needs for development (e.g., web application, mobile application, or enterprise solution).

➢ *Requirements Analysis*

- *Individual User:*

This phase collects precise requirements from end-users or stakeholders, highlighting user demands, functionality, and expected outcomes.

➢ *Model Designing*

- *Different types of UML (Unified Modeling Language) :*

This phase includes the development of visual representations of the software system through UML diagrams, including use case diagrams, sequence diagrams, class diagrams, and activity diagrams, to help with comprehension and communication of the system architecture.

➢ *Decision Nature*

- *Decision type:*

Specifies the different kinds of decisions that need to be taken, such as those addressing the architecture or the prioritization of features.

- *Constraints:*

Prioritizes determining constraints such as financial constraints, technological constraints, the availability of resources, or regulatory requirements.

- *Finalize a UML Diagram:*

The UML diagram(s) are refined and confirmed in order to confirm that they are aligned with the requirements and constraints.

These three terms have described some attributes and taken a final decision based on the attributes. Those attributes are given below:

✓ Dependency: Those decisions that have an impact on other parts or systems.
✓ Artifacts: Activities that result in actual outcomes, such as documentation, diagrams, or prototypes, are examples of actual outputs.
✓ Stakeholders: In the process of decision-making, it guarantees that every significant factor are taken into consideration.
✓ Time Length: Defines the project deadlines clearly.

➢ *Implementation Phase:*

- *Finalize the Type of Programming:*

It chooses the programming language(s) and development frameworks that are most appropriate for putting the software into action, considering the requirements and the design of the system.

➢ *Testing Phase*

- *Different Types of Testing:*

The application is tested using a variety of methods, including the following, to ensure that it satisfies quality standards. The testing types are given below:

✓ Unit Testing
✓ Integration Testing
✓ System Testing
✓ User Acceptance Testing (UAT)

➢ *Final Phase*

- *Deployment:*

The last phase involves delivering the software to the end-user or deploying it to the development environment, so providing it accessible for use.

B. Overview and Key Features of the Proposed Methodology

In this section, we explain how our proposed methodology will solve the issues that we found in our background study in this section. We are going to add a table to discuss this clearly. Also, we will try to discuss some key features of our proposed methodology.

Table 2: Overview of the Proposed Methodology

Identified issue from Section 2	How it will be solved using our proposed model
Need for frameworks addressing non-technical contexts and large-scale Agile scaling challenges [12] [13] [15].	Our proposed model has used UML diagrams in model designing phase to set the Agile processes to a scale. It focuses on the dependencies between teams and iterations. Our model considers the time length, stakeholder communication and industrial dependencies. Based on this, it defines constraints for non-technical and Agile scaling needs. After that, it will finalize the programming type in the implementation phase to integrate Agile-friendly frameworks. In the testing phase of our model, it will add scalability and collaboration testing to verify compatibility with large-scale teams.
i) Limited exploration of hybrid models' effectiveness and their application in industry-specific scenarios. ii)Need for exploration of SDLC adaptations, especially for regulated fields [17] [18].	Our proposed model follows the step-by-step procedure. It goes sequentially. After finalizing user's need, designing process and decision type, this model will finalize the project type based on the industry specific scenario and user's choice. So, it supports in-depth analysis based on the phases-to-phases analysis.
Limited studies on how DT scales in enterprise-level and complex projects [19] [21].	Our proposed model identifies the project's complexity and enterprise-level scaling in the initial phase. Then it will collaborate with individual users to identify specific design thinking (DT) scaling requirements in the next phase. In the model designing phase, our model focuses on the dependencies and constraints in scaling DT. After that, it tries to involve all relevant stakeholders to ensure the feasibility of DT scaling in the decision nature. Then it selects scalable and modular programming frameworks.
Need for empirical evidence and real-world industry applications in various sectors [20].	Our proposed model identifies and explains the type of software to focus on generating data for empirical studies and applications across sectors. In the requirements analysis phase, it will select cross-sector requirements through user interviews.
Absence of specific decision methodological framework to help teams select suitable methods for specific needs [23] [24].	Our proposed model defines the need for a decision-support framework in the scope of the software. Then it will identify team-specific requirements and challenges in method selection. It focuses on constraints and artifacts that inform decision-making. It specifies the decision types and finalizes a UML model that guides method selection. Then it creates a modular decision-support framework. Our model validates the framework by simulating various team scenarios in the testing phase of our proposed model.
Need for frameworks to balance creativity and speed in delivery-focused methods [23].	Our proposed model chooses lightweight, flexible programming frameworks to balance creativity and speed in delivery-focused methods. It will test for balance between delivery speed and creativity in diverse scenarios.

Now, some key features of our proposed methodology are explained below:

- Hybrid Framework: Integrates Agile adaptability with Design Thinking's emphasis on user-centered design, delivering both innovation and rapidity.
- Structured Decision-Making: Utilizes decision-making techniques during significant times to help teams in determining the most effective strategy of action.
- Scalability: The methodology is versatile for both individual efforts and large-scale organizational conditions.
- Continuous Feedback Loop: Engages stakeholders at each stage, facilitating immediate adjustments.
- Risk Management: Decision checkpoints reduce risks by insuring that only well tested and validated increments are implemented.

This proposed methodology addresses the weaknesses noted in the literature by integrating an effective development framework with systematic decision-making techniques. It guarantees that software design is consistent with corporate goals, reconciles innovation with expeditious delivery and is scalable proficiently for extensive projects.

IV. CONCLUSION AND FUTURE WORK

Choosing an effective methodology is essential for maintaining project success across diverse domains of software development. Software projects frequently exhibit complexity, enabling systematic decision-making to tackle diverse technical and non-technical difficulties. An effective project design implies a systematic and adaptable approach, accompanied by a structured decision-making technique, in the rapidly evolving field of software development. The intricate structure of today's software systems, together with numerous stakeholder requirements and ongoing technological advancements, demands a decision-making framework that adeptly integrates creativity and accuracy.

In this research, we explored a comprehensive methodology integrating structured decision-making with iterative software design processes to address the challenges of modern software development. The proposed model effectively balances flexibility and systematicity by combining established frameworks, such as UML diagrams and Agile methodologies, with robust decision-support systems tailored to project-specific requirements. By emphasizing phases like user-centered requirements analysis, model designing, and decision validation, the methodology enhances clarity, scalability, and adaptability across diverse project contexts. Moreover, the model's scalability makes it suitable for both small-scale and enterprise-level projects, addressing gaps identified in existing literature. By focusing on modular programming frameworks and continuous feedback loops, the methodology ensures alignment with dynamic user needs and technological advancements. This holistic approach also bridges the divide between theoretical constructs and practical application, offering a versatile solution to contemporary software development challenges.

Future studies can expand on this framework by integrating AI-driven decision-making tools to further enhance adaptability and predictive capabilities. Additionally, real-world validation across various industries would solidify the methodology's applicability and efficacy. This research contributes a flexible, scalable, and structured model to the software development domain, making the way for more efficient, user-centric, and resilient software solutions.

REFERENCES

[1]. S. Saeed, N. Jhanjhi, M. Naqvi and M. Humayun, “Analysis of Software Development Methodologies”, in International Journal of Computing and Digital Systems, Vol. 8, No.5, pp. 445-460, 2019.

[2]. M. Poppendieck and M. A. Cusumano, “Lean Software Development: A Tutorial”, in IEEE computer Society, pp. 26-32, 2012.

[3]. M. Adil, I. Fronza, and C. Pahl, "Software Design and Modeling Practices in an Online Software Engineering Course: The Learners’ Perspective," in *Proceedings of the 2021 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1-8, 2021. [Online]. Available: https://doi.org/10.1109/EDUCON45462.2021.9449093.

[4]. I. Lytra, C. Carrillo, R. Capilla, and U. Zdun, "Quality Attributes Use in Architecture Design Decision Methods: Research and Practice," in *Proceedings of the 2022 IEEE International Conference on Software Architecture (ICSA)*, pp. 1-10, 2022. [Online]. Available: https://doi.org/10.1109/ICSA54710.2022.00020.

[5]. F. Almeida, "Challenges in Migration from Waterfall to Agile Environments," ResearchGate, 2017. Available: https://www.researchgate.net/publication/Challenges_in_Migration_from_Waterfall_to_Agile_Environments. [Accessed: Nov. 02, 2024].

[6]. M. Stoica, B. Ghilic-Micu, M. Mircea, and C. Uscatu, "Analyzing Agile Development – from Waterfall Style to Scrumban," Informatica Economica, vol. 20, no. 4, pp. 5-14, 2016. Available: http://revistaie.ase.ro/content/80/01%20-%20Stoica,%20Ghilic,%20Mircea,%20Uscatu.pdf. [Accessed: Nov. 02, 2024].

[7]. T. Natarajan and S. Pichai, "Transition from Waterfall to Agile Methodology: An Action Research Study," IEEE Access, vol. 12, pp. 49341-49362, 2024. Available: https://ieeexplore.ieee.org/document/10488855. [Accessed: Nov. 02, 2024].

[8]. R. Mohanani, P. Ralph, B. Turhan, and V. Mandić, "How Templated Requirements Specifications Inhibit Creativity in Software Engineering," *IEEE Trans. Software Eng.*, vol. 48, no. 10, pp. 4074-4086, 2022. Available: https://researchr.org/publication/MohananiRTM22 [Accessed: Nov. 02, 2024].

[9]. J. Ahamed and D. Nandi, "A Decision-Making Technique for Software Architecture Design," International Journal of Mathematical Sciences and Computing, vol. 9, no. 4, pp. 44-49, Dec. 2023. Available: https://www.mecs-press.org/ijmsc/ijmsc-v9-n4/v9n4-5.html. [Accessed: Nov. 02, 2024].

[10]. Kilova, Kristina, et al. "Modern Models and Approaches for Design of Architecture of a Software Application for Monitoring and Quality Assessment in Higher Education." CBU International Conference Proceedings.... Vol. 5. Central Bohemia University, 2017.

[11]. Al-Sarayreh, Khalid T., et al. "A sustainable procedural method of software design process improvements." Indonesian Journal of Electrical Engineering and Computer Science 21.1 (2021): 440-449.

[12]. Chan, Frank KY, and James YL Thong. "Acceptance of agile methodologies: A critical review and conceptual framework." Decision support systems 46.4 (2009): 803-814.

[13]. Trihardianingsih, Liana, et al. "Systematic Literature Review of Trend and Characteristic Agile Model." Jurnal Teknik Informatika 16.1 (2023): 45-57.

[14]. Mohammed, Khaza Nawaz, and Karri Syam Chambrelin. "An analytical approach in usage of agile methodologies in construction industries–A case study." Materials Today: Proceedings 33 (2020): 475-479.

[15]. Adenowo, Adetokunbo AA, and Basirat A. Adenowo. "Software engineering methodologies: a review of the waterfall model and object-oriented approach." International Journal of Scientific & Engineering Research 4.7 (2013): 427-434.

[16]. Chandra, Vishal. "Comparison between various software development methodologies." International Journal of Computer Applications 131.9 (2015): 7-10.
[17]. Balaji, Sundramoorthy, and M. Sundararajan Murugaiyan. "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC." International Journal of Information Technology and Business Management 2.1 (2012): 26-30.
[18]. Pargaonkar, Shravan. "A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering." International Journal of Scientific and Research Publications (IJSRP) 13.08 (2023): 345-358.
[19]. E. D. Canedo and A. T. S. Calazans, "Design thinking use in agile software projects: Software developers' perception," in ICEIS (2), 2020. Available: https://www.scitepress.org/Papers/2020/93875/93875.pdf
[20]. G. H. Steinke and M. S. Al-Deen, "Innovating information system development methodologies with design thinking," Proceedings of the 5th International Conference on Applied Innovations in IT, 2018. Available: https://opendata.uni-halle.de/bitstream/1981185920/12695/1/ICAIIT%20V%202_02%20Steinke_Al-Deen_LaBrie.pdf
[21]. R. Parizi et al., "Design thinking in software requirements: What techniques to use? A proposal for a recommendation tool," Conference on Software Engineering-CIbSE, 2020. Available: https://repositorio.pucrs.br/dspace/bitstream/10923/20445/2/Design_Thinking_in_Software_Requirements_What_Techniques_to_Use_A_Proposal_for_a_Recommendation_Tool.pdf
[22]. J. C. Pereira and R. de F. S. M. Russo, "Design thinking integrated in agile software development: A systematic literature review," Procedia Computer Science, vol. 138, 2018. Available: https://www.sciencedirect.com/science/article/pii/S1877050918317484
[23]. O. Sohaib et al., "Integrating design thinking into extreme programming," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 3, 2019. Available: https://opus.lib.uts.edu.au/bitstream/10453/126025/4/OCC-121627_AM.pdf
[24]. K. Wangsa et al., "A comparative study between design thinking, agile, and design sprint methodologies," International Journal of Agile Systems and Management, vol. 15, no. 1, 2022.
[25]. K. Gama et al., "The developers' design thinking toolbox in hackathons: A study on the recurring design methods," International Journal of Human-Computer Interaction, 2023.