

Decoding Software Success: A Study of Five Engineering Models

Miral D. Dave^{1,*}, Parvin Bloch²

Abstract

Choosing the right software development life cycle (SDLC) model plays a vital role in determining the overall success of a software development project, as it directly affects the planning, execution, quality, and timely delivery of the final product. Different software engineering models provide structured approaches for planning, designing, developing, testing, deploying, and maintaining software systems, with each model offering distinct advantages depending on project characteristics. This study presents a comparative evaluation of five widely used software engineering models: Waterfall, V-model, incremental, Spiral, and Agile. The comparison is conducted using key evaluation criteria, including development methodology, requirement flexibility, stakeholder participation, risk management, testing strategy, documentation, development cost, delivery time, and suitability for various project environments. The study reviews the strengths, limitations, and practical applications of each model by synthesizing findings from existing literature and established software engineering practices. Traditional approaches such as the Waterfall and V-Model are shown to be effective for projects with stable and well-defined requirements, where structured planning and comprehensive documentation are essential. In contrast, incremental, Spiral, and Agile models provide greater flexibility, support continuous improvement, encourage customer collaboration, and adapt more effectively to changing requirements, making them suitable for dynamic and complex software projects. The comparative analysis demonstrates that no single SDLC model is universally applicable to every software development scenario. Instead, the selection of an appropriate model should be based on factors such as project size, complexity, risk level, regulatory requirements, customer involvement, and requirement stability. The findings highlight that Agile and other iterative development approaches have become increasingly popular because of their adaptability, rapid delivery cycles, and emphasis on continuous customer feedback. This study provides practical guidance for software engineers, project managers, researchers, and organizations in selecting the most suitable software development model to improve software quality, project efficiency, and overall project success.

Keywords: Software development life cycle (SDLC), software engineering, Waterfall Model, V-model, incremental model, Spiral model, Agile methodology, comparative analysis, software process models, project management

*Author for Correspondence

Miral D. Dave
E-mail: miral.dave@nobleuniversity.ac.in

^{1,2}Assistant Professor, Department of Computer Engineering,
Noble University, Junagadh, Gujarat, India

Received Date: March 14, 2026
Accepted Date: July 10, 2026
Published Date: September 05, 2026

Citation: Miral D. Dave, Parvin Bloch. Decoding Software Success: A Study of Five Engineering Models. Journal of Software Engineering Tools & Technology Trends. 2026; 13(2): 27–35p.

INTRODUCTION

Software has become a fundamental component of modern society, supporting critical operations in sectors such as healthcare, education, banking, manufacturing, transportation, communication, and e-commerce. As organizations increasingly rely on software-driven solutions, the demand for reliable, scalable, secure, and high-quality software systems is growing. Delivering such systems requires not only technical expertise but also the adoption of an appropriate software development life cycle

(SDLC) model that provides a structured framework for managing the entire software development process [1].

An SDLC model defines the sequence of activities involved in software development, beginning with requirement analysis and continuing through system design, implementation, testing, deployment, and maintenance. These models help development teams organize project activities, allocate resources efficiently, manage risks, maintain product quality, and ensure that software is delivered within the planned schedule and budget. Therefore, the effectiveness of a software project is influenced not only by the skills of the development team but also by selecting a development methodology that matches the project's objectives and constraints [2].

Over the years, numerous software engineering models have been introduced to address various development challenges. Traditional models, including the Waterfall and V-Model, emphasize sequential development, comprehensive documentation, systematic verification, and validation. These methods work best for projects where the requirements are clearly defined from the beginning, regulatory standards must be carefully followed, and modifications are expected to be minimal throughout the development process [3]. In contrast, modern methodologies, such as the incremental, Iterative, Spiral, and Agile models, support progressive software development, continuous customer involvement, regular feedback, and rapid adaptation to changing business and technical requirements. Their adaptability makes them an excellent choice for software projects that must respond quickly to changing requirements, evolving market conditions, and highly competitive business environments [4].

Selecting an unsuitable SDLC model may lead to schedule overruns, increased development costs, communication problems, reduced software quality, and, in some cases, project failure. Conversely, choosing a methodology that aligns with the project size, complexity, requirement stability, risk level, customer participation, and organizational objectives can significantly improve the development efficiency, product quality, and stakeholder satisfaction. Consequently, understanding the characteristics, strengths, and limitations of different software engineering models has become essential for effective software project management [5].

This study provides a detailed comparison of five commonly used software development models: Waterfall, V-model, incremental, Spiral, and Agile. It examines their key features, strengths, limitations, and practical applications to better understand their suitability for various software projects. The comparison is based on several important evaluation parameters, including the development approach, flexibility in handling requirement changes, customer involvement, risk management, testing practices, documentation, development cost, delivery time, and overall suitability for different categories of software projects [6]. By examining these models from both theoretical and practical perspectives, this study aims to assist researchers, software engineers, project managers, and organizations in selecting the most appropriate software development methodology to achieve successful, efficient, and high-quality software systems.

WATERFALL MODEL

The Waterfall Model is among the oldest and most commonly used approaches in the SDLC. Introduced by Winston W. Royce in 1970, this model follows a structured, step-by-step process in which every stage must be completed before moving to the next. The development process generally consists of requirements gathering and analysis, system design, coding and implementation, testing, deployment, and ongoing maintenance. Because of its organized and sequential nature, the Waterfall Model is particularly suitable for projects with clearly defined requirements and minimally expected changes during development [7]. Because the output of one phase serves as the input for the next, changes introduced after the development has started are generally difficult and expensive to implement.

This approach works best for projects with well-defined and consistent requirements, where careful planning and detailed documentation play vital roles throughout the development process. Its structured workflow makes project progress easy to monitor and manage. However, the Waterfall Model provides limited flexibility for accommodating requirement changes and often delays customer feedback until the later stages of development [8]. Consequently, it is less appropriate for projects that involve evolving business needs or uncertain requirements (Figure 1).

V-MODEL (VERIFICATION AND VALIDATION MODEL)

The V-model is an extension of the Waterfall Model that emphasizes quality assurance by integrating verification and validation activities throughout the software development process. The development phases on the left side of the "V" are directly associated with the corresponding testing activities on the right side, ensuring that each development stage is systematically verified and validated before deployment [9].

This structured relationship between development and testing helps identify defects at an early stage, thereby reducing the cost of error correction and improving software reliability. V-model is widely adopted in industries where software quality and safety are critical, including healthcare, aerospace, automotive systems, and defense applications [10]. Although it provides excellent traceability and documentation, the model is less effective when project requirements change frequently because modifications often require revisiting multiple development and testing phases (Figure 2).

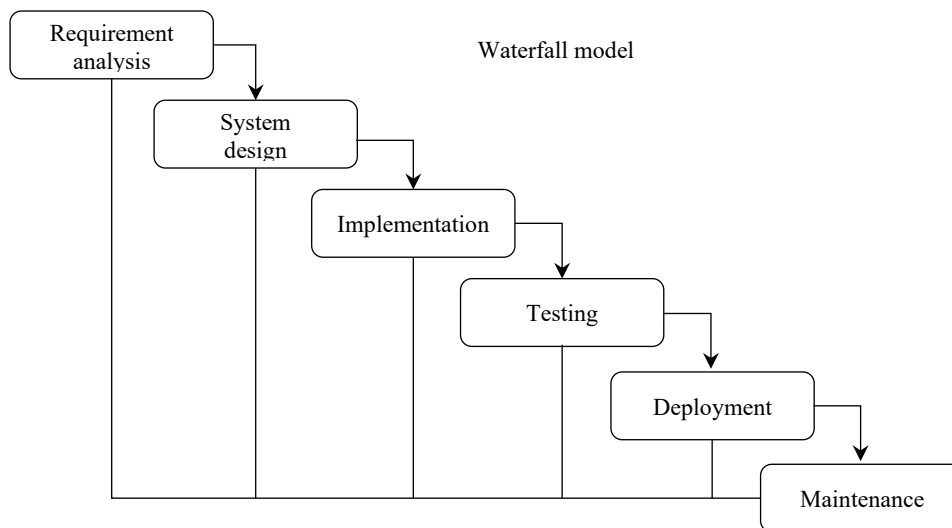


Figure 1. Waterfall software development model.

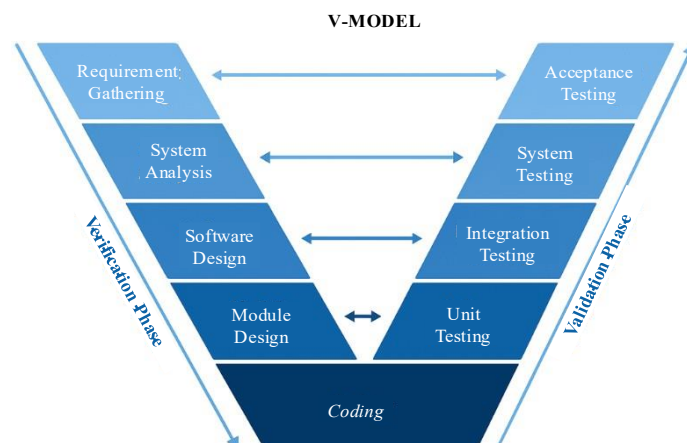


Figure 2. Verification and validation (V-model).

INCREMENTAL MODEL

The incremental model develops software through a series of smaller functional releases instead of delivering the complete system at once. Each increment represents a working portion of the final product and introduces additional features based on the previously completed components. This gradual development strategy allows users to access usable software early, while enabling developers to incorporate feedback into subsequent releases [11].

The model improves project flexibility by reducing development risks and simplifying the testing and debugging activities. It is particularly effective for medium- and large-scale projects, where requirements can be prioritized and implemented in stages [12]. Nevertheless, successful implementation requires careful planning, effective module integration, and a well-defined overall system architecture to ensure that individual increments function together as a complete software solution (Figure 3).

SPIRAL MODEL

The Spiral model, introduced by Boehm in 1988, combines iterative software development with continuous risk management. Software is developed through repeated cycles, known as spirals, each consisting of planning, risk assessment, engineering, implementation, and customer evaluation. This iterative process enables project teams to identify and mitigate technical, financial, and operational risks before they become significant issues [13].

The Spiral model is particularly appropriate for large, complex, and high-risk software projects, where requirements may evolve throughout development. Regular customer involvement and continuous evaluation contribute to improved software quality and better alignment with the expectations of users. Despite these advantages, the model requires experienced project teams, comprehensive risk analysis, and greater financial investment, making it less suitable for small projects with limited budgets and straightforward requirements, as shown in Figure 4.

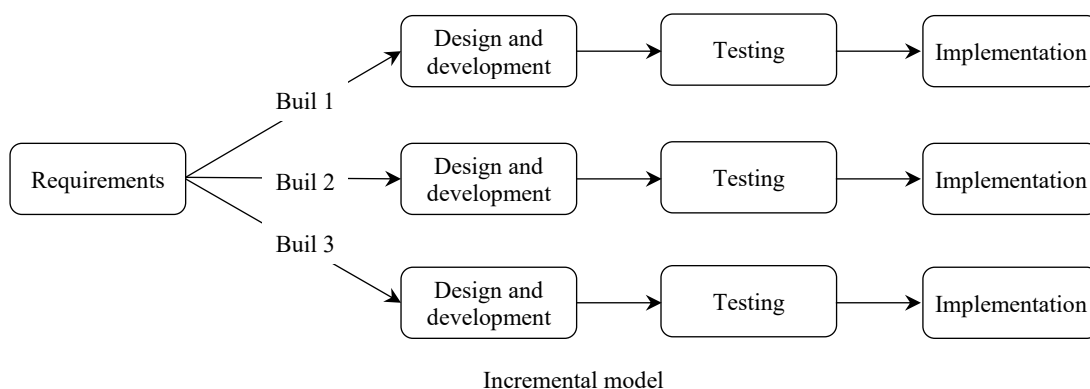


Figure 3. Incremental software development model.

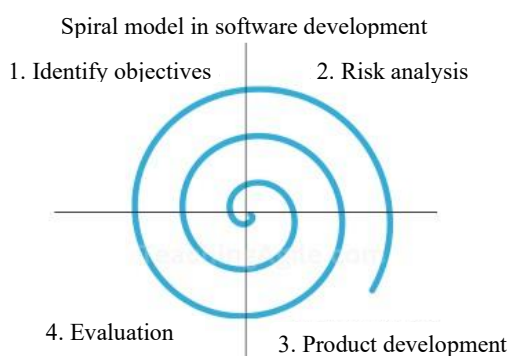


Figure 4. Spiral model with risk analysis.

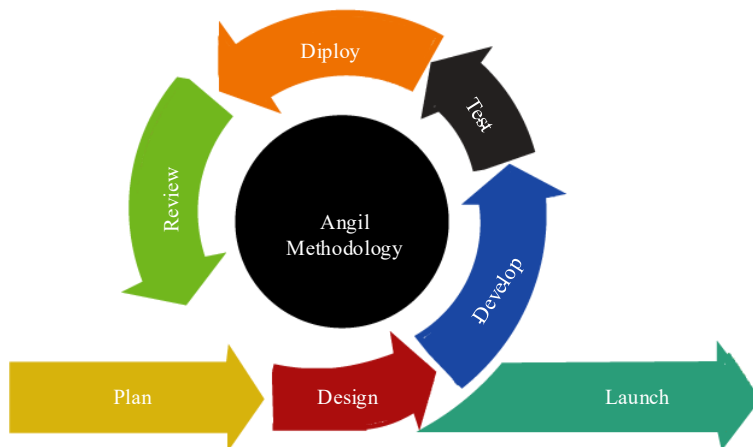


Figure 5. Agile development life cycle.

AGILE MODEL

The Agile Model is a contemporary software development methodology that emphasizes adaptability, teamwork, and continuous delivery of valuable software throughout the development process. Formalized through the Agile Manifesto in 2001, Agile promotes short development iterations, regular customer interaction, rapid feedback, and continuous improvement throughout the project life cycle. Instead of attempting to deliver the entire system at the end of development, Agile teams release functional software in small increments, allowing requirements to evolve based on stakeholder feedback [14].

Agile methodologies, such as Scrum, Kanban, and Extreme Programming (XP), have been widely adopted because they improve adaptability, accelerate software delivery, and enhance customer satisfaction. Regular testing and continuous integration are vital for improving software quality and minimizing potential risks throughout the development process. Simultaneously, the success of Agile projects depends on strong collaboration among stakeholders, skilled development teams, and clear and consistent communication. Nevertheless, applying Agile methodologies in industries with strict regulatory requirements can be difficult, as these environments often demand detailed documentation, rigorous compliance procedures, and formal approval processes (Figure 5).

Table 1 presents a comparative analysis of the five software engineering models based on the development approach, flexibility, customer involvement, risk management, documentation, testing, cost, development time, and project suitability.

LITERATURE REVIEW

1. *Title:* A Comparison between Three SDLC Models: Waterfall Model, spiral model, and incremental/Iterative Model
2. *Title:* Comparison between Various Software Development Methodologies
3. *Title:* Which model is best for the Software Project? “A Comparative Analysis of Software Engineering Models”
4. *Title:* A Comparison between Five Models of Software Engineering
5. *Title:* Explaining software developer acceptance of methodologies: a comparison of five theoretical models
6. *Title:* Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies

OBJECTIVES

The primary objective of this study is to examine and compare five widely used SDLC models—Waterfall, V-model, incremental, Spiral, and Agile—to understand their characteristics, strengths, limitations, and suitability for different software development projects.

Table 1. Comparison of five software engineering models.

Feature	Waterfall model	Incremental model	Iterative model	Spiral model	Agile model
Approach	Linear and sequential	Developed in small increments	Repeated refinement through iterations	Risk-driven iterative development	Adaptive and iterative development
Requirement stability	Must be fixed before development	Mostly stable	Can change gradually	Can change throughout the project	Frequently changing requirements are expected
Customer involvement	Low	Moderate	High	High	Very High
Risk management	Poor	Moderate	Good	Excellent	Good
Flexibility	Very Low	Moderate	High	High	Very High
Testing	Performed after development	Performed after each increment	Conducted in every iteration	Conducted in every spiral cycle	Continuous testing throughout development
Documentation	Extensive	Moderate	Moderate	Extensive	Minimal to Moderate
Delivery	Single final delivery	Multiple incremental releases	Multiple improved versions	Releases after each spiral cycle	Frequent working software releases
Development cost	Low	Moderate	Moderate	High	Moderate to High
Development time	Long	Medium	Medium	Long	Short iterations with frequent releases
Best suited	Small projects with fixed requirements	Medium-sized projects with partial requirement changes	Projects with evolving requirements	Large, complex, high-risk projects	Dynamic projects with rapidly changing requirements
Advantages	Simple, easy to manage, well-documented	Early delivery, easier testing, reduced risk	Continuous improvement, accommodates changes	Strong risk analysis, suitable for complex systems	Fast delivery, customer satisfaction, high adaptability
Disadvantages	Difficult to accommodate changes, late testing	Requires good planning, integration challenges	Scope may expand, require repeated testing	Expensive, complex, requires risk management expertise	Requires experienced teams, less documentation, and difficult to predict timelines

The specific objectives of the study are:

- To examine the fundamental principles, workflow, and key features of the waterfall, V-model, incremental, Spiral, and Agile software development models.
- To compare these SDLC models, important evaluation criteria, including the development approach, requirement flexibility, customer involvement, risk management, testing methodology, documentation, development cost, and delivery time, were considered.
- The advantages and limitations of each software engineering model were evaluated in relation to different project environments and organizational requirements.
- To investigate the factors that influence the selection of an appropriate software development model, such as project size, complexity, requirement stability, budget, timeline, and risk level.
- To identify the most suitable SDLC model for various categories of software projects based on practical application and project-specific needs.

METHODOLOGY

This study adopts a *comparative and qualitative research methodology* to examine the characteristics and performance of five widely used SDLC models: *Waterfall, V-model, Incremental, Spiral, and Agile*. This study focuses on identifying the strengths, limitations, and practical applications of each model to determine their suitability for different software development environments.

Research Design

A comparative research design was used to analyze the strengths, limitations, and applicability of the five software engineering models.

Data Collection

The study is based on secondary data collected from:

- Peer-reviewed journal articles
- Conference papers
- Books on software engineering
- IEEE Xplore, ACM Digital Library, SpringerLink, and ScienceDirect
- Reputable online publications and technical reports

Evaluation Criteria

The five models are compared using the following parameters:

- Development approach
- Requirement flexibility
- Customer involvement
- Risk management
- Cost of development
- Development time
- Testing strategy
- Documentation
- Ease of implementation
- Project suitability
- Advantages and limitations

Data Analysis

A qualitative comparative analysis was conducted by reviewing the collected literature and evaluating each model against the selected criteria. The findings are summarized in comparative tables to highlight the similarities and differences.

Scope of the Study

The study focuses on five commonly used software engineering models:

- Waterfall model
- Incremental model
- Iterative model
- Spiral model
- Agile model

The analysis emphasizes their characteristics, benefits, drawbacks, and suitability for various software development projects.

Outcome

Based on comparative analysis, this study identifies the most appropriate software engineering model for different project scenarios, considering factors such as project size, complexity, requirement stability, customer involvement, and risk level.

DISCUSSION

A comparative analysis of five software engineering models—Waterfall, incremental, Iterative, Spiral, and Agile—shows that no single model is universally suitable for all software development projects. Each model has unique strengths and limitations, and its effectiveness depends on factors such as project size, complexity, requirement stability, customer involvement, budget, and risk factors.

The Waterfall Model is a linear and systematic software development approach in which each phase is completed before moving on to the next. It performs well on projects with clearly defined and stable requirements, where extensive documentation and strict planning are necessary. Although this approach provides a structured process, it is not very adaptable, making it challenging to incorporate changes after the development process has begun. This limitation can result in higher costs and delays if the requirements evolve during the project.

The incremental model addresses some of the limitations of the Waterfall Model by dividing the software into smaller, manageable increments. Each increment delivers functional software, allowing users to receive working features earlier and to provide feedback throughout the development. This approach reduces project risk and improves customer satisfaction, although it requires careful planning and the integration of successive releases.

The Iterative model emphasizes continuous refinement through repeated development cycles. Instead of attempting to deliver a complete product in a single phase, the software is progressively improved based on testing and user feedback. This approach is particularly suitable for projects in which requirements are likely to change and develop over time. However, multiple iterations can increase development time and costs if changes are not effectively managed.

The Spiral model combines iterative development with systematic risk analysis. Each cycle includes planning, risk assessment, engineering, and customer evaluation, making it particularly effective for large, complex, and high-risk projects. Although it provides excellent risk management, its implementation requires experienced teams and greater financial investment, making it less practical for smaller projects.

The Agile Model has become one of the most widely adopted software development approaches because of its flexibility, customer collaboration, and ability to respond quickly to changing requirements. Agile promotes the continuous delivery of working software through short development iterations and frequent customer feedback. This results in improved product quality and increased customer satisfaction. However, Agile requires close stakeholder involvement, effective communication, and experienced development teams. In projects with strict regulatory requirements or where comprehensive documentation is essential, Agile may present implementation challenges that hinder its effectiveness.

Overall, the comparison demonstrates that selecting an appropriate software engineering model is a strategic decision that should be based on project characteristics rather than a preference for a particular methodology. Traditional models, such as Waterfall, remain effective for projects with fixed requirements and predictable environments, whereas modern approaches, such as Agile, are better suited for dynamic projects requiring frequent changes and rapid delivery. Similarly, the Spiral model is preferable for high-risk systems, whereas Incremental and Iterative models provide balanced solutions for projects requiring gradual development and improvement.

CONCLUSION

The study "Decoding Software Success: A Study of Five Engineering Models" demonstrates that the choice of software engineering model plays a significant role in the success of a software project. Through a comparative analysis of the Waterfall, incremental, Iterative, Spiral, and Agile models, it is evident that each model has distinct characteristics, advantages, and limitations that make it suitable for specific project environments.

The Waterfall Model is most appropriate for projects with well-defined and stable requirements, whereas the Incremental and Iterative models provide greater flexibility by enabling gradual development and continuous improvement. The Spiral model is particularly effective for large and high-risk projects because of its emphasis on risk analysis and management. In contrast, the Agile Model

offers the highest level of adaptability, customer collaboration, and rapid delivery, making it the preferred choice for projects with frequently changing requirements.

The results suggest that no single software engineering model can be considered the best choice for every project or situation. The success of a software project depends on selecting a model that aligns with the project's objectives, complexity, budget, timeline, risk level, and customer involvement. Organizations should carefully evaluate these factors before choosing a development methodology to ensure efficient resource utilization, improved software quality, and higher project success rates.

In conclusion, understanding the strengths and weaknesses of different software engineering models enables project managers and software developers to make informed decisions when planning and executing software projects. As software development continues to evolve, combining the best practices of traditional and modern methodologies may further enhance project performance and support the successful delivery of high-quality software systems.

REFERENCES

1. Alshamrani A, Bahattab A. A comparison between three SDLC models: Waterfall model, Spiral model, and incremental/iterative model. *Int J Comput Sci Issues*. 2015;12:106.
2. Chandra V. Comparison between various software development methodologies. *Int J Comput Appl*. 2015;131:7–10. doi:10.5120/ijca2015907294.
3. Shukla Kr, Saxena A. Which model is best for the software project? *Int J Comput Appl*. 2013;76:18–22. doi:10.5120/13290-0806.
4. Munassar NM, Govardhan A. A comparison between five models of software engineering. *Int J Comput Sci Issues*. 2010;7:94.
5. Riemenschneider CK, Hardgrave BC, Davis FD. Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE Trans Softw Eng*. 2002;28:1135–45. doi:10.1109/TSE.2002.1158287.
6. Kumar G, Bhatia PK. Comparative analysis of software engineering models from traditional to modern methodologies. In: *Proceedings of the Fourth International Conference on Advanced Computing & Communication Technologies*. IEEE; 2014. p. 189–96. doi:10.1109/ACCT.2014.73.
7. Royce W. Managing the development of large software systems [Internet]. 1970. p. 1–9. Available from: <https://www.praxisframework.org/files/royce1970.pdf>
8. Boehm BW. A Spiral model of software development and enhancement. *Computer*. 1988;21:61–72. doi:10.1109/2.59.
9. Rodríguez-Martínez L, Mora M, Álvarez F, Garza L, Durán H, Muñoz J. Review of relevant system development life cycles (SDLCs) in service-oriented software engineering (SoSE). *J Appl Res Technol*. 2012;10:94–113. doi:10.22201/icat.16656423.2012.10.2.396.
10. Nugroho S, Hadi SH, Hakim L. Comparative analysis of software development methods between parallel, V-shaped and iterative. *Int J Comput Appl*. 2017;169:7–11. doi:10.5120/ijca2017914605.
11. Oluyomi AO, Olorunshola OE, Martins EI. Comparative analysis of structured system development life cycle methodologies: rapid application development, prototyping, and scrum. *i-Manag J Softw Eng*. 2020;15:1.
12. Akinsola JE, Ogunbanwo AS, Okesola OJ, Odun-Ayo IJ, Ayegbusi FD, Adebisi AA. Comparative analysis of software development life cycle models (SDLC). In: *Computer Science On-line Conference*. Cham: Springer International Publishing; 2020. p. 310–22.
13. Rodríguez LC, Mora M, Martin MV, O'Connor R, Alvarez F. Process models of SDLCs: comparison and evolution. In: Syed MR, Syed SN, editors. *Handbook of research on modern systems analysis and design technologies and applications*. Hershey (PA): IGI Global Scientific; 2009. p. 76–89. doi:10.4018/978-1-59904-887-1.ch005.
14. Sharma M. A survey of project scenario impact in SDLC models selection process. *Int J Sci Eng Res*. 2011;2:1–4.