A comparative study of Software Development Life Cycle (SDLC) models

Buen Bajrami[1], Ilija Jolevski[1], Kostandina Veljanovska[1]

[1] *University St. Kliment Ohridski – Bitola, Faculty of Information and Communication Technologies, 1 Maj bb., 7000 Bitola, North Macedonia University, Address, City, Country*

bajrami.buen@uklo.edu.mk, ilija.jolevski@uklo.edu.mk, kostandina.veljanovska@uklo.edu.mk

Abstract:
Software Development life cycle (SDLC) is a framework model used in project management that defines the stages included in an information system and a software development project, from an initial feasibility study to the maintenance of the completed application. There are different software development life cycle models that has to be specified for each software development project. These models are also called "Software Development Process Models". Each process model follows a series of phases unique to its type to ensure success in the steps of its development. We analyze the models one by one, review the latest research and provide comparisons between the Spiral Model, Waterfall Model, Iterative Model, Big Bang Model, V-Model and RAD Model. Then, from the obtained analysis we give an interpretation about their use to customers. And finally, we present the appropriate model for our proposed development project, which according to analysis and research is the most appropriate.

Keywords:
Spiral Model, Waterfall Model, Iterative Model, Big Bang Model, V-Model, RAD Model

1. Introduction

Organizations involved in software development utilize specialized systems that are structured into distinct phases, which outline the processes for developing, maintaining, and improving software. These processes are integral to the SDLC, which plays a crucial role in software project development. Various models exist, each differing in their approach, and organizations select the model that best aligns with their specific requirements. Over the past decades, significant progress has been made in this area, with the development of these models and the continuous emergence of new models that greatly assist engineers in software development. In this paper, we analyze six SDLC models, presenting how they function, their significance, and their practical applications. There are many more SDLC models that can be used for software development, but our research focus only on six of them. We discuss the categories of software development where these models are most applicable and advantageous. Additionally, we explore the future challenges each model may face, highlight areas of strength, and provide recommendations for further improvements.

2. Selection of the appropriate model

In order to arrive at the appropriate selection of an SDLC model, we must take into account several factors that will facilitate the comparison and making the decision to continue the web application development process. Therefore, we must base ourselves on some questions that give us a more realistic assessment of the client's requirements and the project as a whole.

a) How realistic and sustainable are the customer's requirements?
b) What is the size of the project?
c) What are the access levels of the project?
d) What is the dynamic development plan of the project?

The purpose of this paper is to build a scenario taking into account the client's requirements, about the development of a web-based application. The client wants a web application which will function as an online shop, where people will be able to order food online in a restaurant. Also, the customer requests that it be possible for a restaurant customer to make a reservation for a table, at most two weeks in advance. Where all orders and reservations will be stored in the web application database. Below we have described the functions of our project through a use case diagram.

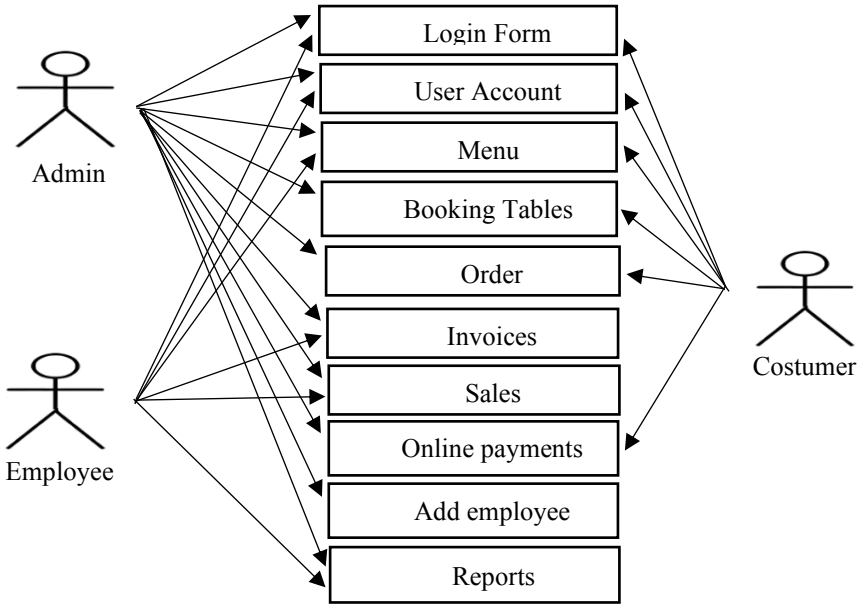

Figure 1.Restaurant online platform - Use Case Diagram

In this paper, we have taken as a basis a total of six models of software development. They are: Spiral Model, Waterfall Model, Iterative Model, Big Bang Model, V-Model, RAD Model.

3. Literature review

In the literature review, we have analyzed all the models included in the research, presenting the orientation of the development and use of these models. Software engineering involves the analysis and design of a software system that handles one or several specific tasks. According to [1] analysis describes the “what” of a software system, which means what happens in the current and what will be required in the new software system. This refers to requirement analysis or gathering. In recent time, the most popular methodological approaches for developing software for a computer-based information system are the popular traditional Waterfall Model [2]. For example [3] the author created a simulation system of the Waterfall model, which was able to help project managers in determining how to achieve maximum productivity with the minimum number of hours, costs and workers. The authors [4] have conducted a comparative study between the iterative waterfall life cycle model and the incremental software development cycle model for resource optimization using the Simphony.NET simulation tool. Spiral model [5] is similar to the incremental model, with more emphasis placed on risk analysis. As for the Big Bang model, the authors [6] show that the Big Bang Model is one in which a large amount of people or money is gathered, a lot of energy is expended, and the perfect software product comes out or not. No mater which model we choose there is possibility to include generative artificial intelligence (GenAI) which can bring unparalleled enhancements in various phases of SDLC. Enterprises could embrace GenAI in their software development process in order to stay competitive in the huge software market. By integrating Gen AI quicker development cycles, superior code quality, and increased innovation could be realized [7].

4. Comparative study of SDLC models

In this paper, we research each model and addressed them through three main factors. The first factor examines the general data of a model, including details of its functioning and the reasons for selecting a particular model. The second factor focuses on the application of an SDLC model in different areas of software engineering. Finally, the third factor addresses the challenges of the models by analyzing their respective strengths and weaknesses.

4.1. Spiral Model

The Spiral model is a software development life cycle (SDLC) model used for risk management, combining elements of the iterative and Waterfall models. This model is particularly suitable for large, complex, and extensive projects. A notable feature of this model is that it releases a prototype after each phase of the spiral, which is then refined. It is effective in managing risks by continuously creating and testing prototypes, allowing for an analysis of their strengths and weaknesses. Below is an illustration depicting the main stages of the Spiral model [8].

The Iterative model consists of four stages: planning of objectives, risk analysis, engineering or development, and review. A project repeatedly passes through all these stages, with the phases forming a spiral within the model [9].

1. Determine objectives and find alternate solutions. This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.
2. Risk Analysis and resolving – In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.
3. Develop and test: This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.
4. Review and planning of the next phase – In this phase, the software is evaluated by the customer.

4.1.1. Spiral Model Risk Analysis and Model Application

The Spiral model analyzes the proposed results and systematically identifies, analyzes, and addresses all possible risks. Following this, methods such as prototyping, simulation, benchmark testing, analytical models, and user research are employed to develop the lowest-risk and most cost-effective strategy [10]. At this stage of the project, the product manager refers to the risk register to identify all potential risks associated with the spiral's objectives and requirements. These risks may be technical, financial, market-related, operational, or environmental. Consequently, risk registers are created with the aim of investigating or addressing these risks in both the short and long term. A risk register is an essential component of various product management documents and templates, including discovery documents and written epics [11].

The spiral model is one of the most widely recognized Software Development Life Cycle (SDLC) models and is applied in a broad range of projects. This model is particularly well-suited for projects that require frequent releases and those where changes may be requested at any stage of the process. Additionally, it is ideal for projects with medium to high risk levels and for those where cost and risk analysis plays a critical role. The spiral model is also highly beneficial in projects with unclear or complex requirements, where a more flexible and adaptive approach is necessary to address uncertainties and challenges effectively.

4.2. Waterfall Model

The Waterfall model is a classical approach used in the system development life cycle, characterized by a linear and sequential process. It is termed "Waterfall" because the model progresses systematically from one phase to the next in a downward manner. This model is divided into distinct phases, where

the output of one phase serves as the input for the next. Each phase must be completed before the subsequent phase begins, with no overlap between phases [12].

All the stages of project development following the model Waterfall are [13]:

1. Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
2. System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared.
3. Implementation − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
4. Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
5. Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
6. Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released.

4.2.1. Waterfall Model Risk Analysis and Model Application

The risks associated with the Waterfall methodology can be mitigated or eliminated through effective risk management. In Waterfall, risks are managed using charts that track the risk from the moment it is detected. Managers monitor its progression, and if the risk intensifies as the project advances, its line on the chart rises. In such cases, developers and managers work to identify and replace the product features contributing to the risk. Conversely, if the risk line decreases as the project progresses, the risk is considered to have low priority. Risk charts in Waterfall are divided into three sections: the first begins when the risk is detected and ends when developers start seeking a solution; the second phase continues until a solution is found; and the third phase, the risk removal phase, concludes on the risk's expiry date. [14].

Every software project is unique and requires a tailored Software Development Life Cycle (SDLC) approach based on various internal and external factors. The Waterfall model is particularly suitable in certain situations. It is most effective when the project requirements are well-documented, clear, and fixed from the outset. The model is also ideal when the product definition remains stable throughout the development process and when there are no ambiguous or unclear requirements. Furthermore, the Waterfall model works well when there are sufficient resources with the necessary expertise to support the development. Finally, it is best suited for short-term projects where a structured, linear approach is more efficient.

4.3. Iterative Model

This model does not start with a complete set of requirements. Instead, the design and development process begins with the basic requirements. Once the product is developed according to these initial requirements, it undergoes a review to identify or define additional requirements. Upon completion of the review phase, the current iteration concludes. Based on this review, the next set of requirements is determined, and the design and development of the second iteration begin. Once the product is developed according to the requirements of the second iteration, it is reviewed again to identify or define further requirements. These iterations continue until the final product is developed [15].

1. Requirement Gathering and Analysis - Business requirements are gathered during this phase of the Iterative model. Then, an analyst determines whether they can be met within financial constraints.
2. Design - During this phase of the iterative model, the project team receives the complete list of criteria to begin work in a specific direction. They then utilize various diagrams, such as data flow

diagrams, class diagrams, activity diagrams, state transition diagrams, and others, to gain a clear understanding of the program design and to assist them in progressing with development.

3. Implementation - At this point in the project, according to the iterative model, the actual coding of the system begins. This phase is influenced by the analysis and design from the Design phase. All requirements, plans, and design strategies have been executed.
4. Testing - This phase involves comparing the current build iteration against a set of rules and standards to determine whether it meets them. This type of testing includes performance testing, stress testing, security testing, requirements testing, usability testing, multi-site testing, disaster recovery testing, and others.
5. Deployment - After all phases are completed, the software is deployed to the working environment.
6. Review - In this phase, after the product has been deployed, the behavior and validity of the deployed product are checked. If any errors are found, the process begins again from the requirement gathering stage.
7. Maintenance - In the maintenance phase, after the software has been deployed in the working environment, bug fixes or new updates may be required.

4.3.1. Iterative Model Risk Analysis and Model Application

In the Iterative model, risk management is not just a component of project management; it is the essence of project management. It focuses on minimizing negative outcomes while balancing the achievement of desirable ones: preventing certain adverse events while also ensuring that the project's positive risks are realized [16]. The process of managing risk in the Iterative Model involves several key steps to ensure that risks are identified, analyzed, and controlled effectively throughout the development lifecycle. The first step is risk management planning, where we determine who needs to be involved and establish the timing and frequency of risk management activities. Next, during risk identification, we analyze potential risks and opportunities across various business categories. In the quantitative risk analysis phase, we subjectively assess the probability and impact of these risks, identifying those that are critical. Then, in the qualitative risk analysis stage, we prioritize which areas require time and resources for risk management. Following this, risk response planning is conducted to determine the best strategies to reduce the likelihood and impact of identified risks. Finally, in risk monitoring and control, the risk response plan is implemented, with ongoing management to ensure progress and compliance.

The Iterative model finds several productive applications across various fields, where its flexibility and adaptability offer significant advantages. In digital marketing, for instance, the model is used to create more engaging advertisements. By conducting thorough analysis to understand user requirements, iterative cycles allow for the continuous improvement of advertisement designs. The model is also applicable in the development of advanced technologies, such as smartphones, cars, and electronic devices, where ongoing iterations enable gradual enhancements and updates. Furthermore, engineering teams often rely on the Iterative model when developing new features or implementing problem-fixing techniques [2]. These iterations, which may not be visible to end users, allow teams to refine their products internally before release, ensuring higher quality and functionality.

4.4. BIG BANG model

In the Big Bang model, the developers of this model do not follow any specific process. Only the necessary funds are enough to start the project. The result of this model may not meet the client's requirements because the client's requirements are not concrete and complete either. This model is suitable for small projects where a very small number of developers can work. [17]

The Big Bang model in software development consists of three key phases [17]:

1. Minimal Planning: Little to no planning is done. Developers begin coding with only a general idea of the final goal, without detailed requirements.

2. Chaotic Development: Components are developed and integrated in an ad-hoc manner, with minimal testing and uncontrolled changes during the process.
3. Delivery or Reorganization: The product is either completed or delivered or fails to meet requirements, leading to a full reorganization or restart of the project.

4.4.1. BIG BANG Model Risk analysis and Model Application

The Big Bang model is a software development life cycle (SDLC) approach where the development process begins with little or no planning and all the modules or components are developed simultaneously. Risk analysis in the Big Bang model involves identifying potential risks and assessing their impact on the project. The Big Bang model, while simple in its approach, presents several risks that must be carefully analyzed to avoid potential pitfalls [24]. One major concern is the lack of requirements definition, where unclear or insufficient requirements can lead to misunderstandings, scope creep, and difficulties in delivering a product that meets user expectations. Another challenge is integration, as in the Big Bang model, modules or components are developed independently and only integrated at a later stage, which can lead to unexpected issues. Additionally, insufficient testing coverage is a common risk, as the simultaneous development approach may overlook thorough testing. This also increases the effort required for debugging, as the simultaneous development of modules raises the likelihood of defects and bugs. Moreover, the model offers limited scope for change control, making it difficult to accommodate changes to requirements or design once development is in progress. Lastly, scalability and performance issues can arise, as these considerations may not be adequately addressed during the initial stages of development in the Big Bang model.

In this model, minimal time is spent on detailed planning, and development begins directly with the required funds and efforts as inputs. This approach is well-suited for short-term projects, such as academic or practical endeavors. It is particularly effective for small teams, usually consisting of two or three developers, working collaboratively. However, this paradigm is not suitable for large-scale or complex software development due to its high risk, though it may be used for temporary, experimental, or very small software projects. It is typically ideal for smaller projects with small development teams focusing on constructing a simple application or prototype. Additionally, the model works well for academic, learning, or practice projects, offering a flexible and low-cost framework. Finally, it can be useful for software products with unclear or poorly understood requirements, particularly when no specific release date is established, allowing for ongoing experimentation and development flexibility.

4.5. V-Model

V-model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. V-Model is one of the many software development models.

The V Model contains Verification phases on one side and Validation phases on the other. The implementation/coding phase joins the verification and validation phases in V-shape. Thus it is called a V Model. We present below both phases: [18]

Verification phases:

1. Business requirement analysis: This is the first step where product requirements understood from the customer's side. This phase contains detailed communication to understand customer's expectations and exact requirements.
2. System Design: In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.
3. Architecture Design: The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology detail, etc.
4. Module Design: In the module design phase, the system breaks down into small modules. The detailed design of the modules is specified, which is known as Low-Level Design

5. Coding Phase: After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding.

Validation phases:

1. Unit Testing: In the V-Model, Unit Test Plans (UTPs) are developed during the module design phase. These UTPs are executed to eliminate errors at code level or unit level.
2. Integration Testing: Integration Test Plans are developed during the Architectural Design Phase.
3. System Testing: System Tests Plans are developed during System Design Phase.
4. Acceptance Testing: Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere [18].

4.5.1. V-Model Risk Analysis and Model Application

The V-Model is a highly structured approach to software development that involves a series of phases, each of which is associated with a specific set of deliverables. The V-Model is often used in safety-critical industries, such as aerospace and defense, where the consequences of software failure can be catastrophic. Risk management is an essential component of the V-Model methodology. It involves identifying potential risks and taking steps to mitigate or eliminate them. The V-Model provides a framework for risk management by breaking the development process down into a series of phases, each of which is associated with a specific set of deliverables. It provides a structured approach to identifying and mitigating potential risks, which can help to ensure that the project is completed on time, within budget, and to the required quality standards [19].

It is advisable to use the V Model on small-to-medium-sized software projects where the requirements are clear without any ambiguity. For projects where acceptance criteria are proper, V Model is the preferable choice. The V Model is useful when ample technical resources are available with technical expertise, and tech stacks and tools are not dynamic [20].

4.6. RAD Model

Rapid Application Development or RAD means an adaptive software development model based on prototyping and quick feedback with less emphasis on specific planning. In general, the RAD approach prioritizes development and building a prototype, rather than planning. With rapid application development, developers can quickly make multiple iterations and updates to the software without starting from scratch. This helps ensure that the final outcome is more quality-focused and aligns with the end users' requirements [22].

Below we have presented the development stages of the RAD model [22]:

1. Business Modeling: On basis of the flow of information and distribution between various business channels, the product is designed
2. Data Modeling: The information collected from business modeling is refined into a set of data objects that are significant for the business
3. Process Modeling: The data object that is declared in the data modeling phase is transformed to achieve the information flow necessary to implement a business function
4. Application Generation: Automated tools are used for the construction of the software, to convert process and data models into prototypes
5. Testing and Turnover: As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.

4.6.1. RAD Model Risk Analysis and Model Application

Risk analysis in the Rapid Application Development (RAD) Model of the Software Development Life Cycle (SDLC) focuses on identifying potential risks and assessing their impact on the project's success. One of the key risks is inadequate requirements gathering, as the fast-paced nature of RAD can lead to incomplete or inaccurate understanding of the project's needs. Additionally, resource availability is critical, as this model demands a dedicated and highly skilled team to meet the rapid development timelines. Technical complexity is another concern, especially in projects that involve complex integrations or advanced technical components. Customer involvement is essential in this model, as continuous feedback and active decision-making are crucial for success. A lack of customer participation can lead to misunderstandings, delays, and misaligned project goals. Integration and compatibility issues often arise in RAD projects, especially when integrating new systems or components with existing infrastructure. Furthermore, testing and quality assurance can be compromised due to the limited time available for comprehensive testing, increasing the risk of defects or performance issues. Lastly, security and data privacy are potential risks, as rapid development may overlook critical security measures, leading to vulnerabilities and the possibility of data breaches. These risks must be carefully managed to ensure the success of a RAD-based project [4, 22].

When deciding to use the Rapid Application Development (RAD) model, it is important to evaluate the specific project needs, as this model may not be productive for unsuitable demands. This model is best applied in cases where the system requirements are well-defined and a short development timeline is required. It is also effective for projects that can be modularized, with components that are reusable or readily available for development. Additionally, the model is suitable when existing system components can be leveraged to build a new system with minimal changes. However, this model should only be used when teams consist of domain experts, as extensive knowledge and the ability to apply advanced techniques are crucial for successful implementation. Finally, RAD is appropriate when the project budget allows for the use of automated tools and specialized techniques necessary for the rapid development process.

5. Discussion

Taking into account the model characteristics, risk analysis, possibilities of application we can summarize what are the challenges for each model.

Spiral model challenges

The spiral model presents several challenges that need to be considered before its implementation. One major drawback is the high cost, making it unsuitable for small projects due to the extensive resources required. Additionally, the model relies heavily on risk analysis, necessitating the involvement of experts in risk management at every stage of the project. Its complexity is another significant challenge [5], as it is the most intricate SDLC model, involving extensive documentation and multiple intermediate phases. Furthermore, managing time can be difficult, as the number of phases is not predetermined at the start of the project. This uncertainty complicates time planning and may result in an overall increase in the project budget.

Waterfall model challenges

The Waterfall model is one of the earliest software development models, but it comes with several challenges that should be carefully evaluated when selecting the appropriate approach for a project. One of the main drawbacks is that no working software is produced until the later stages of the development lifecycle. This leads to high levels of risk and uncertainty, making it unsuitable for complex or object-oriented projects. Additionally, it is a poor fit for long or ongoing projects, as it does not accommodate changes in requirements well, particularly when there is a moderate to high risk of such changes [12]. Measuring progress during the various stages can be difficult, as each phase must be completed before moving to the next. Furthermore, integration is performed as a "big-bang" at the end of the project, which delays the identification of potential technological or business challenges until the final stages, limiting the ability to address them early in the process.

Iterative model challenges

The Iterative model, while highly flexible and adaptive, comes with several challenges that need to be considered. First, it may not be suitable for smaller projects, as the overhead associated with iterative cycles can outweigh the potential benefits. Additionally, more resources in terms of time and money may be required due to the repeated cycles of development, testing, and feedback, making it resource-intensive. Changes in requirements during the development process can also lead to budget overruns, as accommodating these changes often requires additional time and effort. Furthermore, the iterative approach may not be ideal for projects with strict or inflexible requirements, as the model's continuous changes and improvements may not be feasible in such environments. Lastly, effective communication between the development team and users is crucial for the success of the Iterative model, as ongoing feedback and collaboration are essential to its process.

BIG BANG model challenges

The Big Bang model faces several future challenges that need to be addressed to enhance its effectiveness. First, the client's requirements must be very clear from the outset, as the lack of thorough planning can lead to misunderstandings and deviations from the desired outcome. Second, it is not suitable for large projects, where the complexity and scope exceed the capabilities of this simplistic approach. Lastly, the model does not offer high security, making it a risky choice for projects where data protection and security are critical. These limitations highlight the need for further refinement and adaptation to make the model more robust and reliable in a broader range of applications.

V-Model challenges

Similar to other models discussed, the V model also presents several challenges that should be carefully considered when deciding to use it for project development. One significant drawback is that the model is very rigid and offers minimal flexibility, which can be problematic for projects that require adaptability. Additionally, the V model is not ideal for complex projects, as its linear structure may not accommodate the dynamic needs of more intricate systems. Another challenge is that software is only developed during the implementation stage, meaning there are no early prototypes available for review or testing. Lastly, if changes occur mid-project, both the test and requirement documents must be updated, adding to the complexity and potential delays in the project timeline. These challenges highlight the importance of thorough analysis before choosing the V model for a development project.

RAD Model challenges

Like all SDLC models, the RAD model has its own set of challenges that must be considered before choosing it for a project. One significant challenge is that powerful and efficient tools used in RAD require highly skilled professionals, making it essential to have a team with the right expertise. Another issue is that the absence of reusable components can jeopardize the project’s success, as RAD relies heavily on reusing existing elements. Additionally, the team leader must collaborate closely with both developers and customers to ensure the project is completed on time. This model is unsuitable for systems that cannot be modularized, as the model depends on breaking down the system into modules. Moreover, customer involvement is required throughout the entire development life cycle, which can be challenging to maintain. Lastly, RAD is not appropriate for small-scale projects, as the costs of automated tools and advanced techniques may exceed the overall project budget, making it impractical for smaller projects.

Discussion of models comparison

The SDLC models comparison in this paper was done for a small-scale project that does not require significant costs or a large number of developers. Other models are better suited for larger projects, which demand higher costs, more developers, additional tests or prototypes, and more time. Consequently, we believe that the Big Bang model is the most suitable for developing our web application. This model requires a small team of developers and a relatively short time frame for project completion, provided that our requirements for the web application are clear and precise, leaving no room for confusion or challenges during the process. Therefore, we recommend using the Big Bang model for small projects, with a focus on improving security as a future challenge. This will enhance the project’s practicality and seriousness, ensuring no deficiencies in any aspect. Below, we present all the SDLC models based on Table 1, where we evaluated and compared each model using six performance factors.

Table.1.The comparison of SDLC Models

Features	Spiral Model	Waterfall Model	Iterative Model	Big Bang Model	V-Model	RAD Model
Understanding Requirements	Medium	High	Medium	Medium	High	High
Cost	High	Low	High	Low	High	High
Risk Involvement	High	High	Low	High	High	High
User Involvement	Medium	Low	High	Medium	High	Medium
Guaranty of Success	High	Low	Low	Low	Medium	High
Flexibility	High	Low	Medium	Medium	Low	High

6. Conclusion

In conclusion, Software Development Life Cycle (SDLC) models serve as essential frameworks that guide the development process of software applications. After conducting a comprehensive review of the literature and analyzing each model individually, we have been able to assess which model is most suitable for web application development. While each SDLC model may be appropriate and functional for specific projects, the primary objective of this research was to identify the most appropriate model for our project by comparing four key parameters across each model. These parameters have enabled us to determine which model offers the optimal solution for our specific case. It is imperative that the choice of an SDLC model is carefully aligned with project goals, stakeholder needs, and the dynamic nature of the software development landscape. As technology continues to evolve, the adaptability and flexibility of SDLC models remain critical for ensuring the success of software projects in meeting user expectations and achieving business objectives. By engaging in continuous evaluation, refinement, and the adoption of best practices, organizations can effectively leverage SDLC models to innovate, collaborate, and deliver value in an ever-changing digital environment. Nevertheless, it is crucial to consider the limitations and challenges inherent in each model and to focus on their ongoing improvement and development. Additionally, developers should explore the potential integration of generative artificial intelligence within the SDLC. This inclusion could significantly transform the SDLC process, particularly in how software is planned, developed, delivered, and maintained. Generative AI holds the potential to automate repetitive tasks performed by developers, testers, and machines, thereby streamlining code generation and other related processes.

References:

[1] Bennett, S., McRobb, S. & Farmer, R., Object- Oriented Systems Analysis and Design Using UML, McGraw-Hill Higher Education, 2002.
[2] Pfleeeger, S.L. et al, Software Engineering: Theory and Practice, 3rd Ed. US:Prentice Hall., 2006.
[3] Y. Bassil, ""A Simulation Model for the Waterfall Software Development Life cycle"," International Journal of Engineering&Technology, pp. Vol.2, no.5, pp. 2049-3444, 2012.
[4] Prakriti Trivedi et al., ""A Comparative Study between Iterative Waterfall and Incremental Software Development Life Cycle Model for Optimizing the Resources using Computer Simulation"," Information Management in the Knowledge Economy (IMKE), pp. Vol. 7,No.5,pp. 188-194. , 2013.
[5] B. B, ""A Spiral Model of Software Development and Enhancement"," ACM SIGSOFT Software Engineering Notes", "ACM", 1986.
[6] Wollack, Edward J., ""Cosmology: The Study of the Universe"," Universe 101: Big Bang Theory. NASA, 2010.

[7] H. Padmanaban, et al., "Implication of Artificial Intelligence in Software Development Life Cycle: A state of the art review", International Journal of Recent Research Aspects, vol. Vol. 6, no. Issue 2, pp. 93-928, 2019.
[8] TechTarget, 2019. URL:www.techtarget.com/searchsoftwarequality/definition/spiral-model.
[9] Artoftesting, "www.artoftesting.com," 2023. URL: https://artoftesting.com/spiral-model.
[10] Shiksha, "www.shiksha.com," 2023. [Online]. Available: https://www.shiksha.com/online-courses/articles/spiral-model-in-software-engineering/.
[11] Logrocket, "www.blog.logrocket.com," 2023.: https://blog.logrocket.com/product-management/how-to-create-an-effective-risk-register-like-a-product-manager/.
[12] Economictimes, 2024. URL: https://economictimes.indiatimes.com/definition/waterfall-model.
[13] Tutorialspoint, 2023. URL: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.
[14] Hygger, "www.hygger.io," 2023.: https://hygger.io/blog/the-risks-of-waterfall-methodology/.
[15] Tc, "www.tutorialscampus.com," 2023. [Online]. Available: https://www.tutorialscampus.com/sdlc/iterative-model.htm.
[16] Netlogx, "www.netlogx.com," 2023. [Online]. Available: https://netlogx.com/blog/2022/05/17/iterative-risk-management-methodology/.
[17] JavaTpoint, "www.javatpoint.com," 2023. [Online]. Available: https://www.javatpoint.com/software-engineering-big-bang-model.
[18] Javatpoint, 2024. [Online]. Available: www.javatpoint.com/software-engineering-v-model.
[19] Medium, 2024.:medium.com/smart-project-kit/risk-management-in-v-model-methodology-62c65ab2154b#:~:text=The%20V%2DModel%20and%20Risk%20Management&text=V%2DModel%20methodology.,It%20involves%20identifying%20potential%20risks%20and%20taking%20steps%20to%20mitigat
[20] Testsigma, 2024. URL: https://testsigma.com/blog/v-model-in-software-development-life-cycle/.
[21] Javatpoint, 2024. URL: https://www.javatpoint.com/software-engineering-incremental-model.
[22] Kissflow, 2024. URL: https://kissflow.com/application-development/rad/rapid-application-development/.
[23] tutorialspoint, 2023. URL: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.
[24] geeksforgeeks, 2024. URL: https://www.geeksforgeeks.org/big-bang-integration-testing/.