Iraqi Journal for Computer Science and Mathematics

Journal Homepage: http://journal.esj.edu.iq/index.php/IJCM
e-ISSN: 2788-7421 p-ISSN: 2958-0544

A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions

Abdulbasit ALazzawi[1], Qahtan M.Yas[2,*], Bahbibi Rahmatullah[3]

[1]Diyala University, College of Science, Diyala, IRAQ.

[2]Diyala University, College of Veterinary Medicine, Diyala, IRAQ.

[3]Faculty of Arts, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Tanjung Malim, Perak, MALAYSIA.

*Corresponding Author: Qahtan M.Yas

DOI: https://doi.org/10.52866/ ijcsm.2023.04.04.014
Receive July 2023; Accepted October 2023; Available online November 2023

ABSTRACT: The software development process needs specific and studied steps within a reliable plan to achieve the requirements for the success of any project. Software development life cycle (SDLC) methodologies have provided several models that meet the needs of the various proposed projects. These methodologies present various scenarios that can be applied in the process of developing systems to make them more efficient and predictive. The paper aims to illuminate the paramount Software Development Life Cycle (SDLC) methodologies by conducting a comprehensive review of the pros and cons of the various models widely used for software design. Furthermore, the paper discusses fundamental trajectories that are shaping the future landscape of SDLC methodologies. This review included two main types of software development life cycle approaches such as traditional SDLC (heavy-weight) and agile SDLC (light-weight) approaches. The traditional approach included several models such as the Waterfall model, Iterative model, Spiral model, V-Model, and Big Bang Model. Whereas, the agile approach included various models such as the eXtreme Programming (XP) Model, scrum Model, Feature Driven Development (FDD) Model, and Kanban Model. A comprehensive analytical study of all software development life cycle models was achieved and highlighted their most prominent strengths and weaknesses of them. SDLC methodologies wield substantial ramifications across a multitude of sectors, contingent upon several models tailored to individual developmental and research contexts. In culmination, the paper furnishes an all-encompassing perspective on paramount SDLC models, encompassing two principal paradigms: the traditional and the agile approaches. These encompass fundamental sub-models that encapsulate pioneering models poised for application in system development, thus facilitating their refinement more efficiently and predictably.

Keywords: SDLC methodologies, Traditional SDLC, Heavy-weight approach, Agile SDLC, Light-weight approach

1. INTRODUCTION

Software development life cycle methodologies are promising to be adapted in the development of all sectors and projects. Rapid technological development has led to the emergence of frameworks for the development of projects that can be applied in all fields. Therefore, all developers and consultants in organizations should apply the SDLC methodology and choose the appropriate model for each project [1].

The SDLC methodology is meticulously crafted within the contours of an organizational structure, delineating a constellation of indispensable stages arranged within a systematic and cogent framework.. The methodology structure is implemented from top to bottom in the form of sequential steps such as requirements analysis and plans, product design, implementation, testing, and maintenance [2]. SDLC methodology includes two main types: the traditional heavy-weight approach and the agile lightweight approach to software development [3]. The software development team depends on the nature of the project according to the feature of each model within the SDLC methodology to be applied [4]. Each model in the SDLC methodology has strengths and weaknesses [5].

Usually, the outline of a software development process consists of several stages that describe how to create, replace, maintain, modify, or improve the new software. The life cycle methodology focuses on the quality of the

software and the overall development stages of the new project. SDLC aspires to create a high-quality program to satisfy customers and their expectations that are completed within a specific duration and reasonable cost [6]. The software development process consists of several sequential steps. Typically, these steps refer to all basic processes such as requirements analysis, design, testing, and maintenance [7].

The SDLC methodology suffers from some limitations and security challenges in its various models when developing software [8]. These security issues form a major challenge to developers due to the increasing security vulnerabilities in software. Since the security characteristics of systems are a primary goal, it is a major issue throughout the software development lifecycle [9]. In addition, the great spread of internet applications, cloud computing systems, social media, and the internet of things has led to the expansion of these limitations [10]–[12].

Conversely, numerous studies and literary works have delved into the realm of SDLC methodologies, employing diverse models to craft software products. In a concise survey, S. Shaikh and S. Abro [13] undertook a comparative analysis of the traditional and agile approaches for software product selection. Their study underscored distinct features, merits, and drawbacks of software products within the SDM. M. H. Miraz and M. Ali [14] addressed the burgeoning challenges posed by six traditional SDLC models in the realm of blockchain-enabled smart contracts. These models proved inadequate, particularly in the realms of testing, verification, and validation phases. Consequently, addressing this predicament necessitated the formulation of novel standards to surmount the incongruities stemming from traditional models within the blockchain domain. H. J. Christanto and Y. A. Singgalen [15] advocated a novel approach to dissect and blueprint an Information System for the Thesis Consultation Process (SIBIMA), thereby ameliorating real-world quandaries using the SDLC methodology and Waterfall model at Atma Jaya Catholic University of Indonesia. This approach buttressed educational services by aligning with the university's ethos, facilitating a symbiotic enhancement of educational quality among students, lecturers, and academic personnel. N. Rachma and I. Muhlas [16] explored the design of Android-based learning applications through a comparative lens, contrasting the waterfall models and prototypes during the research and development phase. Their study concluded that the waterfall model best suits generic systems or programs, whereas prototype models are better suited for tailored systems and software. In light of the analytical examination, both methods manifest distinct advantages and disadvantages, thereby furnishing developers with the requisite insights to select the most apropos method for their software development needs. J. de V. Mohino et al. [17] unveiled an innovative S-SDLC methodology engineered to seamlessly integrate security factors, thereby expediting the identification of vulnerabilities within stipulated project timelines without incurring supplementary costs or time investments. This approach harmoniously merges agile traits into software development environments, harnessing the agility to pinpoint the most pivotal elements and artifacts for software projects.

This study discusses software development lifecycle methodologies and provides a comprehensive review of the pros, cons, and new research directions. Section 1, discusses an introduction and general background to software lifecycle methodologies. Section 2, is an overview and comparative analysis of the strengths and weaknesses of the various SDLC models. Section 3, highlights new research directions. Section 4, limitations of the Research. Section 5, conclusion and future work.

Various SDLC models have been defined and designed to be used during the software development process. These are also known as software development process models [3]. The most important and popular SDLC models applied in the industry are as follows:

1.1 TYPES OF TRADITIONAL SDLC APPROACH

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

1.2 TYPES OF AGILE SDLD APPROACH

- eXtreme Programming (XP) Model
- Scrum Model
- Feature-Driven Development (FDD) Model
- Kanban Model

To ensure the success of the software development process, each model must include a series of unique sets of steps. SDLC models relied upon essential steps such as requirements analysis, designing, implementation, testing, and

maintenance of high-quality software [18],[17]. See figure 1, illustrates the essential steps in the development process of the SDLC approach.

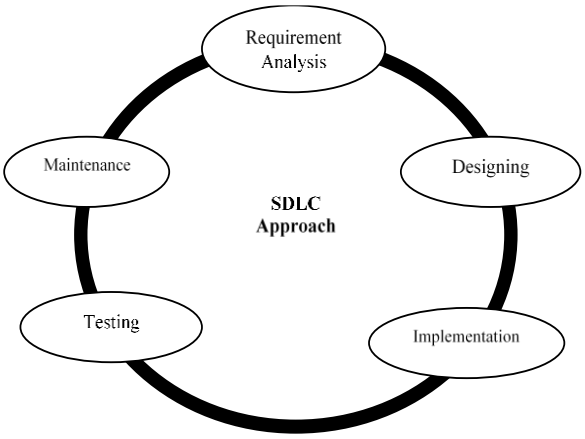

FIGURE 1. Essential steps for the development process of the SDLC approach

1.3 ESSENTIAL STEPS APPLIED FOR SDLC MODELS

Several steps must be applied to ensure the success develop of SDLC models [18],[19]. These steps will be discussed in detail and their impact on the development process of various models.

- Step 1: Requirements analysis

This stage is very important as the customer's requirements are to be analyzed and documentation in a clear and natural language. A specialized team starts collecting information about the proposed project and saved it in the document. This document contains a full description of the project development process. The document containing the customer's requirements is named the Software Requirements Specification (SRS). Finally, the developers analyze the information, draw up a project development plan and conduct a feasibility study according to the economic and technical roles of the new product [19], [20].

- Step 2: Designing

Design is the next stage after analyzing the information for the project that depicts the structural form of the requirements within a clear programming language. This stage includes the design of the project form and how it will be implemented. Thus, this design is documented in a document called Software Design Description (SDD). Thus, the SDD document is reviewed by experts under strict criteria to assess risk, product quality, design, cost, and timetable to get the best product [19], [20].

- Step 3: Implementation and unit testing

This stage includes the coding part that was precisely documented in SDD in the design stage earlier. The testing process is conducted for all small and working units in the project. In the product inspection stage, modifications can be made in the coding if necessary [20]–[22].

- Step 4: Testing

The testing phase of the program is important to identify errors and how address them to improve product quality. Sometimes the unit tester cannot properly check the program interface between the units. Therefore, a full system test must be performed and then the software that is part of the system should also be tested. The maximum of the software development budget is used in the software testing phase of the SDLC. This stage is important to build trust in the developers before the product is delivered to the customer and or published in the market [20]–[22].

- Step 5: Maintenance

The program maintenance stage is the last stage of program development to be delivering the product to the customer, operating it, and publishing it in the market. The new product is deployed through a specific segment to be tested in a real business environment called User Acceptance Testing (UAT). At this stage, the errors in the coding part of the product are corrected gradually to be improved and then deleted the old parts of the product are during the period of its use [19], [20].

2. COMPARATIVE ANALYSIS OF SDLC MODELS

The software development life cycle (SDLC) models are designed that are applied in the development of new projects. Usually, these models are also called Software Development Process Models (SDPM) [23]. Each model has specific steps that are followed to ensure the success of the model development process. The development models include two main types, the traditional model, and the agile model, which will be discussed in detail in this section [24].

2.1 TRADITIONAL SDLC TYPES

The traditional model is one of the oldest approaches to the software development process [24]. One of the most important features of this model is its sequential approach, meaning that all its steps are carried out sequentially, step by step. The traditional approach is mainly based on planning, analysis, and detailed discussion of the project. This model documents all steps of the product such as gathering requirements, designing, coding, and testing all steps, and then deploying the product [25]. This approach included several models that will be discussed in detail. See figure 2, shows the design of the traditional approach architecture.

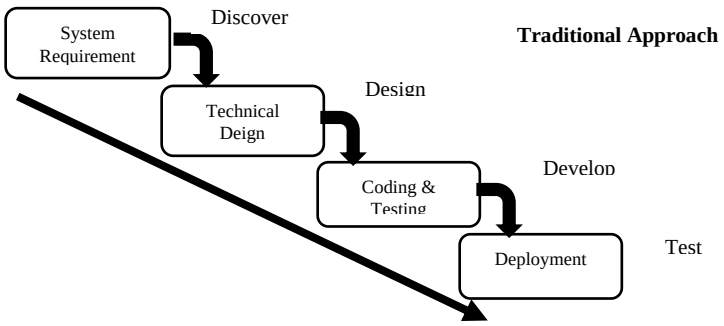

FIGURE 2. Sequence Model

2.1.1 WATERFALL MODEL DESIGN

The waterfall model is the first model within the SDLC approach that was proposed by Royce in the year 1970 [25]. This model is easy to understand and can be used successfully to develop new projects. In addition, this model is known as the sequential linear life cycle model. In this model, the software development process is applied in separate stages [26]. The waterfall model is successfully applied if the current stage is completed to complement the next stage sequentially [2]. See figure 3, depicts the stages of the waterfall model sequentially.

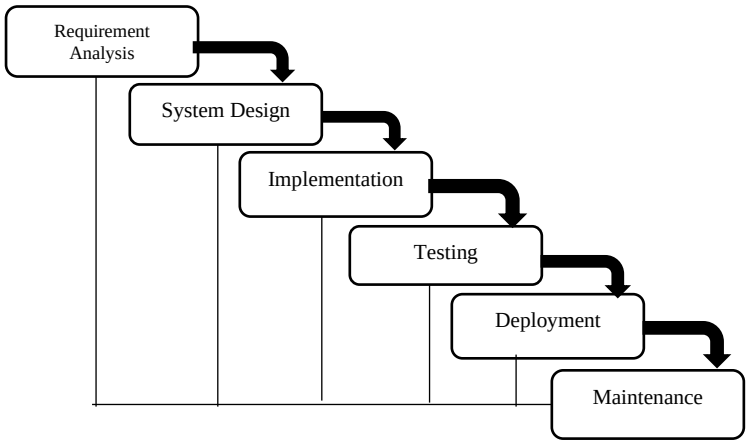

FIGURE 3. Waterfall model based on sequence model

- **PROS AND CONS OF THE WATERFALL MODEL**
  For each model there are strengths and weaknesses which are discussed in detail as follows:

Table 1. - Pros and Cons of the waterfall model

Strengths	Weaknesses
•The program does not run until the model lifecycle is completed •The amount of risk and ambiguity in this model is high •In complex and object-oriented projects is bad •Long-term and ongoing projects are weak •Change in project requirements is causing a problem •Measuring the progress of the model stages is difficult •Adjusting the scope often shut down the model •Difficulty identifying technical and commercial obstacles and challenges early	•Simple and easy to understand and use •Easy to manage due to the strength of the model. •Each stage has a specific output and review process. •Ease of processing all stages and completing them sequentially •Suitable for small projects because the requirements are understandable •The model has specific and clear stages •Model landmarks are very understandable •Distribute and arranging tasks is easy •Ease of documenting results and model stages

2.1.2 ITERATIVE MODEL DESIGN

The iterative model is one of the SDLC models that rely on implementing the requirements of software development through small units repeatedly to reach a completed system. The iterative or incremental model is divided into several architectures units to meet the requirements of the software development process [27]. Each architecture unit is comprised of design, development, testing, and implementation. In each version, the development process continues to obtain the final version to be ready to publish according to the requirements of the proposed model. To ensure the success of the iterative version of software development, all requirements must be checked and tested at each cycle of development for the new model [28]. See figure 4, shows the iterative model design.

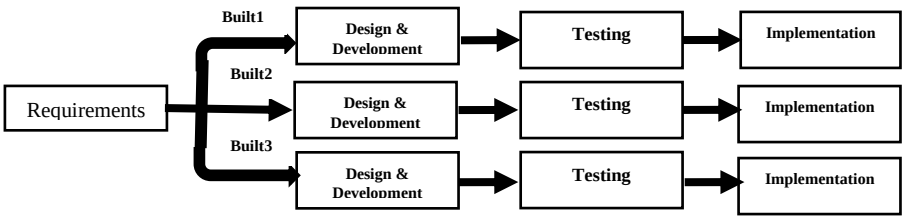

FIGURE 4. Iterative processing

- **PROS AND CONS OF THE ITERATIVE MODEL**

For each model there are strengths and weaknesses which are discussed in detail as follows:

Table 2. - Pros and Cons of the Iterative model

Strength	Weaknesses
•This model is scalable quickly and early •Results can be obtained quickly and periodically •The model is scalable •Easy to measure progress •Cost is limited to change requirements •Ease of conducting tests and handling errors through iterations •Ease of identifying and resolving potential risks during iterations	•The need for more resources •Often the low cost does not match the size of the changing requirements •System architecture design problems often arise because not all requirements are collected at the beginning of the model lifecycle •Defining the system needs to specify the number of iterations/increments •Not suitable for small-scale projects.

•Ease of managing and analyzing risks provided that the highest risk step is taken •A new product can be delivered for each iteration •Possibility of applying the identified challenges and risks to the next iteration •Supports changing requirements. •Less time spent on the initial run •Suitable for large and bulky projects •The speed of completion of the program facilitates the evaluation process and feedback from customers	•Model management is complicated •There is a high risk if the end of the project is unknown •Significant experience in analyzing potential risks is required to ensure project progress

2.1.3 SPIRAL MODEL

The spiral model consists of four stages based on the integration of two approaches, the iterative development model with the sequential development model (waterfall) which is focused on risk analysis that is developed by scientist Barry Boehm [29]. Essentially, the model progressively improves the product through an iterative process around the coil. The model starts with the stage of collecting business requirements in the baseline spiral [13]. In addition, in the product maturity stage in the next spirals, the rest of the main system, sub-system, and unit requirements are identified. While the conceptual design stage of the spiral consists of the architectural and logical design of the units as well as the design of the final product in the later stages of the spiral [30]. Thus, in the last stage of the spiral, the product is deployed in the labor market. See figure 5, shows the spiral model design.

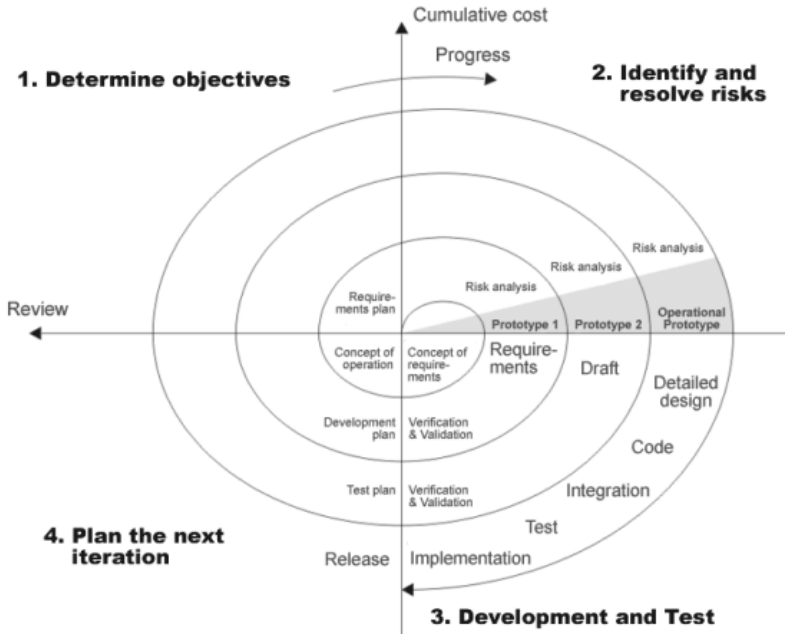

FIGURE 5. Loop Processing steps diagram [31]

- **PROS AND CONS OF SPIRAL MODEL**

For each model there are advantages and disadvantages which are discussed in detail as follows:

Table 3. - Pros and cons of the spiral model

Strengths	Weaknesses
•Able to deal with changes in project requirements •It handles a large number of prototypes •More accurate in dealing with requirements •Quick to display the system to users •Ability to manage risks well	•Model management is too complicated •An early end to the project is uncertain •His steps are complicated •Often the spiral continues indefinitely •It needs documentation and intermediate stages

2.1.4 V. MODEL

The V-model shares a resemblance with the waterfall model, as both are executed through a sequence of stages cascading from the uppermost tier downwards. [32]. In this model, the testing process is associated with each stage of the development cycle. This model is strictly disciplined as no stage begins until the completion of the one before it [25]. Typically, the coding design of the model is linked on both sides of the model, in addition to verification on one side is done with validation on the other side of this model [33]. Thus, the testing process takes place parallel for each side of the model. See figure 6, shows the V model design.

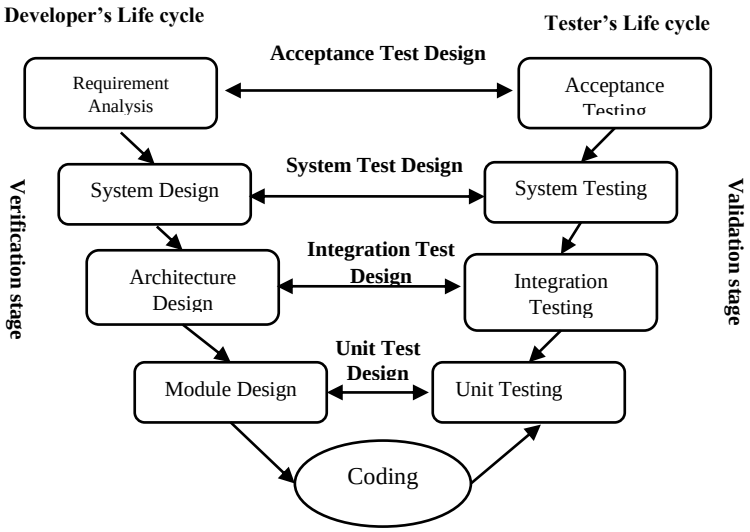

FIGURE 6. V model diagram

- **PROS AND CONS OF SPIRAL MODEL**

For each model there are advantages and disadvantages which are discussed in detail as follows:

Table 4. Pros and cons of the V. model

Strengths	Weaknesses
•This model is very disciplined •Its stages are sequential •Perfect for small projects •Easy to understand and use •Model management is very easy	•The model is ambiguous and highly risky •Not suitable for complex and object-oriented projects •Not suitable for long-term and ongoing projects •Not useful for projects with moderate and high changes •It is not possible to return and change its functions when the test stage is reached

2.1.5 BIG BANG MODEL

The Big Bang model is one of the SDLC models assigned for small projects and does not follow specific steps [34]. This model depends on the inputs (money and efforts) and the outputs (the product), it often does not correspond to the customer's requirements due to its exact purpose being ambiguous. This model has characterized its speed and less analysis, as well development team is very small. The Big Bang model design focuses on all possible project development resources with less or no planning [35]. Usually, changes in the model requirements do not affect the entire project renewal process. Thus, this model is ideal for small projects that do not need a full understanding of the requirements with a small development team, in addition, it is also useful for practical and academic projects [36]. See figure 7, shows the Big Bang model design.

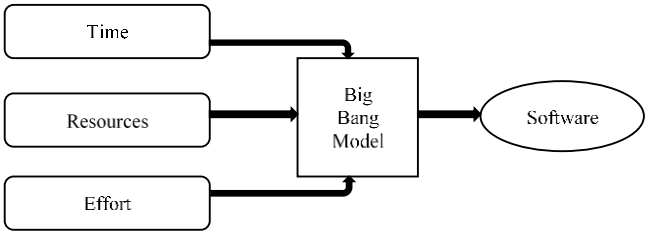

FIGURE 7 Big Bang model architecture

- **PROS AND CONS OF THE BIN BANG MODEL**

For each model there are advantages and disadvantages which are discussed in detail as follows:

Table 4. Pros and cons of the Big Bang model

Strengths	Weaknesses
•Simple and easy model	•Ambiguous and the risk is high
•It doesn't require planning or non-existent	•Not suitable for object-oriented projects and complex
•Model management is easy	•Not useful for ongoing and long-term projects
•It doesn't require large resources	•Less understanding of the requirements leads to high costs
•The model is very flexible	
•It is a suitable educational platform for new students	

2.2 AGILE SDLC TYPES

The agile approach is a lightweight type of software development and it is often called the 'moving quickly' approach. This approach relies upon the software development on small parts called 'iterations' or 'increments', where each part represents a stage of development for the project. [37],[24]. The agile approach focuses primarily on customer collaboration and interaction with the product development process rather than traditional methods such as requirements analysis, plans, and tools [13]. This approach consists of various models that will be discussed in detail. See figure 8, illustrates the agile approach architecture.

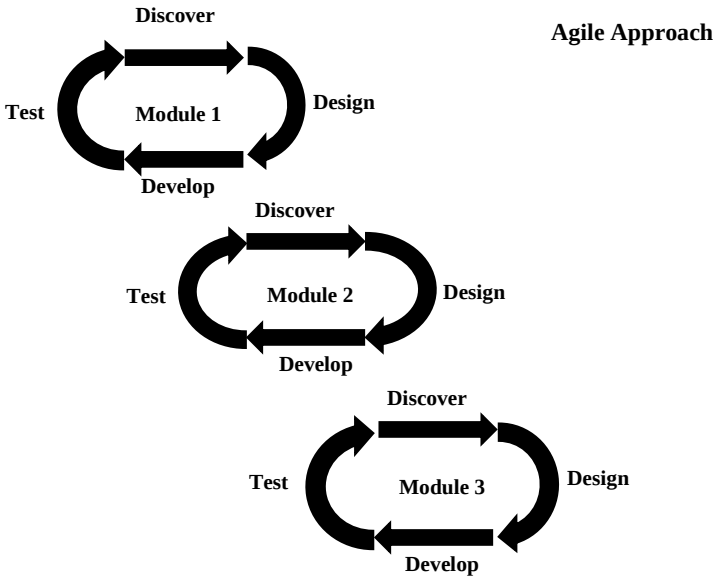

FIGURE 8. Agile approach architecture

2.2.1 EXTREME PROGRAMMING (XP) MODEL

The Extreme Programming (XP) model, a subdivision of the agile methodology, was conceptualized in the 1990s and crystallized into its definitive structure after several subsequent years. Rooted in the collaborative efforts of the development team, this model centers on creating the necessary program. The process involves delivering the product to the customer and systematically repeating iterations. Following each iteration, rigorous testing is conducted, culminating in the collection of valuable feedback [34].. The

extreme programming model is one of the most common models of the agile approach that has been developed in the current era. This model has weaknesses, such as a lack of documentation and poor design, which makes it not ideal for medium and large projects. In addition, the structure of the model is so complex that no specific design activity can be adopted [38]. Whereas, this model is suitable for small enterprises to develop new high-quality products. The XP model includes a set of simple and specific principles applied in the software development process based on four main stages: planning, coding, design, and testing [39]. See figure 9, illustrates the eXtreme Programming (XP) model architecture.

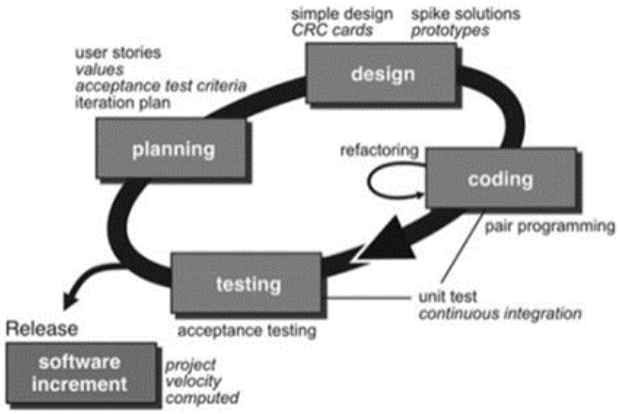

FIGURE 8. eXtreme Programming (XP) model architecture [39]

- **PROS AND CONS OF THE EXTREME PROGRAMMING (XP) MODEL**

For each model there are Strengths and Weaknesses which are discussed in detail as follows:

Table 4. Pros and cons of the extreme programming (xp) model

Strengths	Weaknesses
• Fulfill customer needs with product quality • Ease of adapting to the constant changes in requirements • Adopt pair programming to improve product quality • Ideal for small projects	•No documentation •The structure of the model is too weak •Little attention to design •An urgent need for integration among programmers for a mutual understanding of programming and advanced skills

2.2.2 SCRUM MODEL

The Scrum model is the most important and modern among the agile models. It was developed by Ken Schwaber and Jeff Sutherland developers in 1993. According to them, the scrum model is defined as a framework that allows developers to solve complex project problems and deliver a high-quality, innovative product. This model is simple and easy to understand to manage and control each step of product development to meet the requirements of the customer in a gradual and practical way [40]. In this model, all activities are carried out within a standard framework consisting of several sprints. These sprints organize all activities by size and complexity of design [41]. Scrum is continuously adaptable to all changes in requirements. This model organizes the work of the teams to get the work done more productively to reach a high-quality product [42]. See figure 10, shows Scrum model architecture.

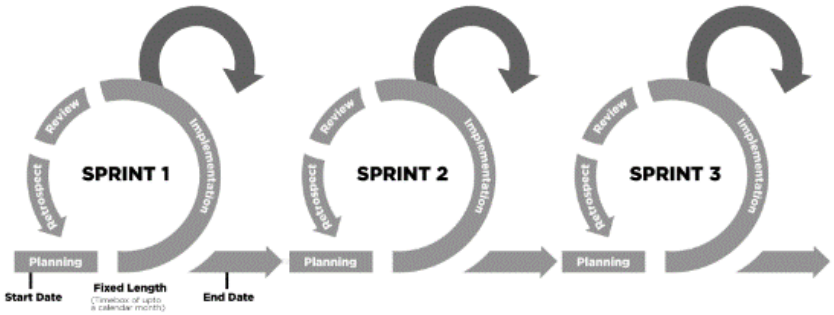

FIGURE 10. Scrum model architecture [43]

- **PROS AND CONS OF SCRUM MODEL**

For each model there are Strengths and Weaknesses which are discussed in detail as follows:

Table 7. Pros and cons of Scrum model

Strengths	Weaknesses
•Team members communicate effectively and efficiently •A good management •Receive customer comments about the product on a regular basis •Its products are of high quality to meet the needs of the customer •Daily scrum team meetings provide significant growth and productivity	•Staff lack full knowledge of scrum •Poor technical practice in the scrum model •Difficulty mastering the model •Not suitable for medium and large projects

2.2.3 FEATURE DRIVEN DEVELOPMENT (FDD) MODEL

A Feature Driven Development (FDD) model was developed by researchers Jeff DeLuca and Peter Coad in late 1999. This module mainly focuses on the design and construction phases, as well as the quality aspects of the product during the development process. In addition, in this model, the project progress steps are monitored during the product delivery process [44]. FDD model works incrementally and iteratively with a slight difference in some factors, and it is similar to another model. One weakness of this model is that it is not suitable for large and long-term projects. Conversely, it is an ideal model for small and medium projects only. The size of the team is directly proportional to the size of the project, usually [45]. This model is effective in new projects that require updating and upgrading the code used to reach the second version of the current application. It is a flexible and effective approach that adapts to the different types of systems to be developed [46]. See figure 11, shows Feature Driven Development (FDD) model architecture.

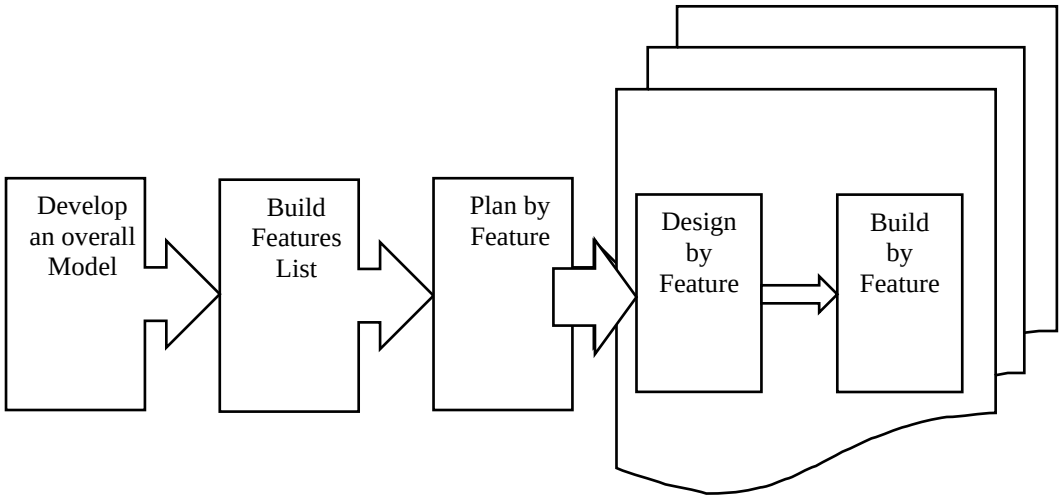

FIGURE 11. Feature Driven Development (FDD) model architecture

- **PROS AND CONS OF FEATURE DRIVEN DEVELOPMENT (FDD) MODEL**

For each model there are Strengths and Weaknesses which are discussed in detail as follows:

Table 8. Pros and Cons of Feature Driven Development (FDD) Model

Strengths	Weaknesses
•Recursive ascending pattern	•Cannot validate the system
•Focuses on design, construction and product quality	•Lacks the necessary training to write the requirements
•Suitable for small and medium enterprises	•Weak to handle changing requirements
•The size of the team is proportional to the size of the project	•Lack of experienced and trained staff
	•Not suitable for large projects
•Flexible model and adapts to the development of systems	•It involves a lot of risks

2.2.4 KANBAN MODEL

The Kanban model was developed by Toyota Company for tracking their manufacturing operations. It is considered one of the most approved methods by systems development organizations around the world. The Kanban approach is based on the concept of just in time (JIT). The Kanban approach provides information about what is needed and how much is needed for delivery within a specified time schedule. It has the ability to easily identify bottlenecks during development stages that may affect the workflow [47]. In addition, it maintains transparency among the work team within the Kanban board management. It also has the ability to know each task at any time, which helps to monitor the workflow. It is also characterized by following the limit option, which helps to organize tasks at the maximum [48]. The main objective of the Kanban model is to reduce the time period possible to complete the project. Briefly, three basic principles can be identified, which are that each worker should know his current task, the computing developer must make developmental changes to the system, and if the Kanban model is applied, the employees must be prepared to be leaders within the program development [49]. See figure 12, shows the Kanban model architecture.

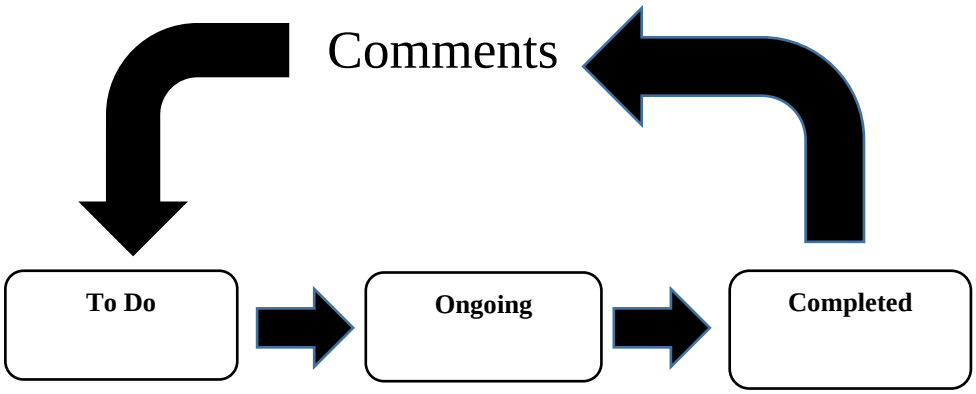

FIGURE 12. Kanban model architecture

- **PROS AND CONS OF KANBAN MODEL**
  For each model there are Strengths and Weaknesses which are discussed in detail as follows:

Table 9. Pros and cons Kanban model

Strengths	Weaknesses
•It is an easy-to-use model •It can be applied in various industries •Customizable •Supports cooperation •Minimal overheads	•Does not work independently •Cannot be used in a dynamic environment •It does not support repetition •lack of time •There are no timings for each stage

3. COMPARATIVE ANALYSIS FOR THE TRADITIONAL AND AGILE APPROACHES

In the comparative analysis section, the most important differences, weaknesses and strengths between the traditional and agile approaches will be discussed. Table 10, shows the main differences between the traditional and agile approaches according to several criteria.

Table 10. A major differences between of the Traditional and Agile approaches [13],[24]

Key criteria	Traditional approach	Agile Approach
Requirements development	Fixed changing	Frequently changing
Customer role	limited	Large
Developer skills	Sufficient skills	Knowledgeable
Development type	Stable	Easily changeable
Project size	Large	Small and medium
Potential risks	Limited effect	Great effect
Coding value	Expensive	Cheap
Team members	Large team	Small team
Changes cost	High	Low

Table 11. Comparative analysis of the Traditional and Agile approaches [50]

Approaches	Strengths	Weaknesses
Traditional approach	Predictive approach Depends on a detailed development plan More documentation in orientation Strictly complete all changes in requirements Suitable to the large projects Emphasize documentation in orientation It has a specific requirement model before implementation and coding Coding is documented by business analysts	Weekly meetings are not suitable for developers Time frame for each iteration is short Lack of communication between developers and customers

Agile Approach	Adaptive approach It adapted to the dynamic requirements changes The product is frequently tested Minimum risks Interaction with the clients Open communication and minimal documentation Suitable to small and medium projects More comments in source code Focuses on the importance of communicating with the customer Teams collaborate closely often in same environment	It does not depend on a detailed development plan It is difficult for beginner developers or new members Delay completing iterations Increase development costs Frequent change in requirements causes design problems It is unable to include reusable components that may lead to more risks

4. FUTURE DIRECTIONS

In this section, a new view on the topic of our study will be presented based on previous studies relevant to the review. Furthermore, this review also offers a novel perspective by outlining potential future directions in the field software development life cycle (SDLC) methodologies are promising to be adapted in the development of all sectors and projects [1]. Briefly defined, the concept of the software development life cycle methodology is a structured analysis consisting of specific logical steps. The SDLC methodology is represented in a systematical and logical framework consisting of a set of defined steps. The methodology is implemented from top to bottom in the form of sequential steps [2]. This methodology depends on several criteria that represent all the basic steps for building the required model. Typically, these steps refer to all the basic operations required to implement the model such as requirements analysis, design, testing, and maintenance [18]. Basically, these criteria should be taken into account for future studies on this topic. It is known the SDLC methodologies that have been applied to develop various projects within industrial sectors that depend on multiple criteria to determine the important elements in these sectors. Multi-criteria decision-making (MCDM) techniques represent a promising research area to solve various problems facing researchers [51]–[54]. This vital field includes many important techniques in decision-making. Proper techniques were used as AHP and ANP methods to calculate the criteria weights for any case study, especially for developing software in the industrial sectors. This technique relies on the opinions of experts in the evaluation process based on their preferences [54]–[56]. This technique also has the ability to calculate the order of priority and importance of each criterion. In addition, many other decision-making techniques can compute the ranking of alternatives based on criteria weights taken from other methods, such as SAW, HAW, WPM, WSM, MEW, TOPSIS, VIKOR, GRA, MOORA, and others[57]–[61]. On the other hand, machine learning algorithms represent the ideal direction for the classification and prediction of the SDLC model's performance [62]. Therefore, these techniques represent a promising direction for many researchers to obtain reliable results based on multiple criteria for each case study.

5. LIMITATIONS

Several limitations are outlined in this review. The most important limitations related to SDLC methodologies in this research will be reviewed as follows [8]:

- **LIMITATIONS OF THE TRADITIONAL APPROACH**

- The regression method applied in this approach does not allow going back, so the step must be completed.
- The inaccuracy of the timetable required to complete the project sometimes leads to the failure of the project within its specified period.
- This approach is inflexible because it does not allow changes to be made according to customer requirements.
- It is suitable for large term projects.
- Any change of the approach structure leads to the start again of the project and an additional cost to the project.
- Increased software development time and expenses as the customer keeps adding requirements to the list.
- Only the member in the current stage is working and the rest of the team remains in the idle stage.

- **LIMITATIONS OF THE AGILE APPROACH**

- Stakeholders are often used in project management if there is no specific timetable for the end of the project.
- Project costs are clearly affected if the tasks are not specified accurately.
- The failure of the project is certain if the developer team does not fulfill their required tasks.
- Ideal for small and medium enterprises
- Requires a highly experienced team to complete the project within the specified period.
- The strict control of the team is frustrating to work for them that affects the completion of the project.
- Leaving a team member during the development period negatively affects the project.
- Failure to conduct the test, failure affects the quality of the project.

6. CONCLUSION

Software Development Life Cycle (SDLC) methodologies represent the best solution to the software development process in all areas of life. This methodology included various traditional and agile models that can be applied to conduct various software development processes to make them predictable and more efficient. The aim of this study is to conduct a comprehensive review of SDLC methodologies and identify their strengths and weaknesses. SDLC methodologies included the most important approaches are the traditional approach which consists of the waterfall model, the spiral model, the iterative model, the V model, and BANG. In addition to the agile approach, which consists of the XP model, the scrum model, the FDD model, and the Kanban model. Several strengths, weaknesses, and limitations of this approach have been identified. Research limitations represented the study doesn't cover all models due to the similarity in their characteristics and the lack of sufficient studies on them. Future works emphasize the need to conduct more surveys and analytical studies on these models because they are closely related to various aspects of life. It can also focus on new research directions using methodologies such as decision-making techniques and machine learning in software development by adopting SDLC methodologies**.**

Funding

None.

ACKNOWLEDGEMENT

These authors would like to thank the University of Diyala\Scientific Research Committee (SRC) for supporting this major research project and also to some friends who gave me good advice during working on this research.

CONFLICT OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] H. F. Rahmani and E. Himawati, “Combining SDLC Method And ITIL Framework by Involving Auditors,” *J. AKSI (Akuntansi dan Sist. Informasi)*, vol. 5, no. 1, pp. 6–12, 2020, doi: 10.32486/aksi.v5i1.431.

[2] H. S. Modi, N. K. Singh, and H. P. Chauhan, “Comprehensive Analysis of Software Development Life Cycle Models,” *Int. Res. J. Eng. Technol.*, vol. 4, no. 6, pp. 117–122, 2017, [Online]. Available: https://irjet.net/archives/V4/i6/IRJET-V4I618.pdf

[3] A. Alzayed and A. Khalfan, “Understanding Top Management Involvement in SDLC Phases,” *Int. J. Comput. Appl.*, vol. 183, no. 37, pp. 30–49, 2021, doi: 10.5120/ijca2021921759.

[4] M. Kumar, “A Comparative Study of Universally Accepted SDLC Models for Software Development,” *Int. J. Sci. Res. Sci. Technol.*, vol. 4, no. 5, pp. 1084–1092, 2018, [Online]. Available: www.ijsrst.com

[5] A. Adel and B. Abdullah, “A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model,” *IJCSI Int. J. Comput. Sci. Issues*, vol. 12, no. 1, pp. 106–111, 2015, [Online]. Available: https://www.academia.edu/10793943/A_Comparison_Between_Three_SDLC_Models_Waterfall_Model_Spira

l_Model_and_Incremental_Iterative_Model

[6] S. Kumar Dora and P. Dubey, "Software Development Life Cycle (Sdlc) Analytical Comparison and Survey on Traditional and Agile Methodology," *J. Res. Sci. Technol.*, vol. 2, no. 8, pp. 22–30, 2013, [Online]. Available: www.abhinavjournal.com

[7] R. N. Salsabila and P. I. Listyorini, "Patient Clinical Data Integration in Integrated Electronic Medical Record System using System Development Life Cycle (SDLC)," *2nd Int. Conf. Heal. Sci. Technol. 2021*, no. 269, pp. 21–26, 2021, [Online]. Available: http://ojs.udb.ac.id/index.php/icohetech/article/view/1073%0Ahttp://ojs.udb.ac.id/index.php/icohetech/article/download/1073/916

[8] R. D. Amlani, "Advantages and limitations of different SDLC models," *Int. J. Comput. Appl. Inf. Technol.*, vol. 1, no. 3, pp. 6–11, 2012.

[9] S. Z. Hlaing and K. Ochimizu, "An integrated cost-effective security requirement engineering process in SDLC using FRAM," in *Proceedings - 2018 International Conference on Computational Science and Computational Intelligence, CSCI 2018*, IEEE, 2018, pp. 852–857. doi: 10.1109/CSCI46756.2018.00170.

[10] A. M. Fernandes, A. Pai, and L. M. M. Colaco, "Secure SDLC for IoT Based Health Monitor," *Proc. 2nd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2018*, no. Iceca 2018, pp. 1236–1241, 2018, doi: 10.1109/ICECA.2018.8474668.

[11] G. K. Ouda and Q. M. Yas, "Design of Cloud Computing for Educational Centers Using Private Cloud Computing : A Case Study," in *Journal of Physics: Conference Series*, 2021, pp. 1–8. doi: 10.1088/1742-6596/1804/1/012119.

[12] D. S. Ibrahim, A. F. Mahdi, and Q. M. Yas, "Challenges and Issues for Wireless Sensor Networks : A Survey," *J. Glob. Sci. Res.*, vol. 6, no. 1, pp. 1079–1097, 2021.

[13] S. Shaikh and S. Abro, "Comparison of Traditional and Agile Software Development Methodology: a Short Survey," *Int. J. Softw. Eng. Comput. Syst.*, vol. 5, no. 2, pp. 1–14, 2019, doi: 10.15282/ijsecs.5.2.2019.1.0057.

[14] M. H. Miraz and M. Ali, "Blockchain Enabled Smart Contract Based Applications: Deficiencies with the Software Development Life Cycle Models," vol. 33, no. 1, pp. 101–116, 2020, [Online]. Available: http://arxiv.org/abs/2001.10589

[15] H. J. Christanto and Y. A. Singgalen, "Analysis and Design of Student Guidance Information System through Software Development Life Cycle (SDLC) dan Waterfall Model," *J. Inf. Syst. Informatics*, vol. 5, no. 1, pp. 259–270, 2023, doi: 10.51519/journalisi.v5i1.443.

[16] N. Rachma and I. Muhlas, "Comparison Of Waterfall And Prototyping Models In Research And Development (R&D) Methods For Android-Based Learning Application Design," *J. Inov. Inov. Teknol. Inf. dan Inform.*, vol. 5, no. 1, p. 36, 2022, doi: 10.32832/inova-tif.v5i1.7927.

[17] J. de V. Mohino, J. B. Higuera, J. R. B. Higuera, and J. A. S. Montalvo, "The application of a new secure software development life cycle (S-SDLC) with agile methodologies," *Electron.*, vol. 8, no. 11, 2019, doi: 10.3390/electronics8111218.

[18] S. Malik, "Software testing: Essential phase of SDLC and a comparative study of software testing techniques," *Int. J. Syst. Softw. Eng.*, vol. 5, no. 2, pp. 38–45, 2017, [Online]. Available: https://www.academia.edu/40510082/Software_Testing_Essential_Phase_of_SDLC_and_a_Comparative_Study_of_Software_Testing_Techniques

[19] A. Coronato, "Agile software development life cycles," *Eng. High Qual. Med. Softw. Regul. Stand. Methodol. tools Certif.*, pp. 161–172, 2018, doi: 10.1049/pbhe012e_ch13.

[20] "SDLC Tutorial/ tuyorialspoint." https://www.tutorialspoint.com/sdlc/index.htm

[21] T. T and M. Prasanna, "Research and Development on Software Testing Techniques and Tools," vol. 4, no. 4, pp. 1479–1493, 2018, doi: 10.4018/978-1-5225-7598-6.ch109.

[22] T. Jindal, "Importance of Testing in SDLC," *Int. J. Eng. Appl. Comput. Sci.*, vol. 01, no. 02, pp. 54–56, 2016, doi: 10.24032/ijeacs/0102/05.

[23] I. H. Sarker, F. Faruque, U. Hossen, and A. Rahman, "A survey of software development process models in

software engineering," *Int. J. Softw. Eng. its Appl.*, vol. 9, no. 11, pp. 55–70, 2015, doi: 10.14257/ijseia.2015.9.11.05.

[24] Y. Leau, W. K. Loo, W. Y. Tham, and S. F. Tan, "Software Development Life Cycle AGILE vs Traditional Approaches," vol. 37, no. Icint, pp. 162–167, 2012.

[25] T. Chittagong and T. Islam, "Introducing a New Sdlc Trigon Model for," in *Proceedings of the International Conference on Sustainable Development in Technology for 4th Industrial Revolution 2021 (ICSDTIR-2021)*, 2021, pp. 1–7.

[26] P. Agarwal, A. Singhal, and A. Garg, "SDLC Model Selection Tool and Risk Incorporation," *Int. J. Comput. Appl.*, vol. 172, no. 10, pp. 6–10, 2017, doi: 10.5120/ijca2017915143.

[27] S. S. Kute and S. D. Thorat, "A Review on Various Software Development Life Cycle (SDLC) Models," *Int. J. Res. Comput. Commun. Technol.*, vol. 3, no. 7, pp. 776–781, 2014.

[28] O. J. Okesola, A. A. Adebiyi, A. A. Owoade, O. Adeaga, O. Adeyemi, and I. Odun-Ayo, "Software Requirement in Iterative SDLC Model," *Adv. Intell. Syst. Comput.*, vol. 1224 AISC, pp. 26–34, 2020, doi: 10.1007/978-3-030-51965-0_2.

[29] S. Shylesh, "A study of software development life cycle process models," *SSRN Electron. J.*, pp. 1–7, 2017.

[30] A. K. . Z. Islam and D. A. Ferworn, "A Comparison between Agile and Traditional Software Development Methodologies," *Glob. J. Comput. Sci. Technol.*, vol. 20, no. 2, pp. 7–42, 2020, doi: 10.34257/gjcstcvol20is2pg7.

[31] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 3, pp. 8–13, 2010, doi: 10.1145/1764810.1764814.

[32] I. Journal, V. Kumari, and S. Kulkarni2, "IRJET-Use of Artificial Intelligence in Soware Development Life Cycle Requirements and its Model Use of Artificial Intelligence in Software Development Life Cycle Requirements and its Model," *Int. Res. J. Eng. Technol.*, p. 1857, 2018, [Online]. Available: www.irjet.net

[33] N. Dwivedi, D. Katiyar, and G. Goel, "A Comparative Study of Various Software Development Life Cycle (SDLC) Models Neha," *Int. J. Res. Eng. Sci. Manag.*, vol. 5, no. 3, pp. 141–144, 2022, [Online]. Available: https://madhavuniversity.edu.in/software-development-life-cycle.html

[34] M. H. Miraz and M. Ali, "Blockchain Enabled Smart Contract Based Applications: Deficiencies with the Software Development Life Cycle Models," *Balt. J.*, vol. 33, no. 1, pp. 101–116, 2020, [Online]. Available: http://arxiv.org/abs/2001.10589

[35] A. Singh and P. J. Kaur, "Analysis of software development life cycle models," *Lect. Notes Electr. Eng.*, vol. 476, pp. 689–699, 2019, doi: 10.1007/978-981-10-8234-4_55.

[36] A. Dubey and L. C. McInnes, "Position paper - Proposal for a scientific software lifecycle model," in *Proceedings of SE-CoDeSE 2017: 1st International Workshop on Software Engineering for High Performance Computing in Computational and Data-Enabled Science and Engineering - Held in conjunction with SC 2017: The International Conference for High Performanc*, 2017, pp. 22–26. doi: 10.1145/3144763.3144767.

[37] R. P. Pawar, "A Comparative study of Agile Software Development Methodology and traditional waterfall model," *IOSR J. Comput. Eng.*, pp. 1–8, 2015.

[38] B. G. Sudarsono, "Using an Extreme Programming Method for Hotel Reservation System Development," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 6, pp. 2223--2228, 2020, doi: 10.30534/ijeter/2020/01862020.

[39] I. G. N. Suryantara and J. F. Andry, "Development of Medical Record With Extreme Programming SDLC," *Int. J. New Media Technol.*, vol. 5, no. 1, pp. 47–53, 2018, doi: 10.31937/ijnmt.v5i1.706.

[40] V. Hema, S. Thota, S. Naresh Kumar, C. Padmaja, C. B. Rama Krishna, and K. Mahender, "Scrum: An Effective Software Development Agile Tool," in *IOP Conference Series: Materials Science and Engineering*, 2020, pp. 1–11. doi: 10.1088/1757-899X/981/2/022060.

[41] J. Andry and W. Rakkha, "Development Point of Sales Using SCRUM Framework," *J. Syst. Integr.*, vol. 10, no. 1, pp. 36–48, 2019, doi: 10.20470/jsi.v10i1.359.

[42] M. A. Subih *et al.*, "Comparison of agile method and scrum method with software quality affecting factors,"

Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, pp. 531–535, 2019, doi: 10.14569/ijacsa.2019.0100569.

[43] "TOOLS TUTORIALS," 2022. https://www.toolsqa.com/agile/scrum/sprint

[44] H. K. Flora and S. V. Chande, "A Systematic Study on Agile Software Development Methodologies and Practices," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 3, pp. 3626–3637, 2014, [Online]. Available: http://www.ijcsit.com/docs/Volume 5/vol5issue03/ijcsit20140503214.pdf

[45] A. Jerom, R. Sp, and P. G. Scholar, "A Survey on Comparative Analysis of Agile Software Development Methodologies," *Recent Trends Comput. Sci. Softw. Technol.*, vol. 5, no. 1, pp. 36–48, 2019, [Online]. Available: https://www.academia.edu/43902409/A_Survey_on_Comparative_Analysis_of_Agile_Software_Development_Methodologies?from=cover_page

[46] V. Upadrista, "Agile Methodology," *Art Consult. Sell. IT*, pp. 99–106, 2015, doi: 10.1201/b18065-15.

[47] W. Zayat and O. Senvar, "Framework Study for Agile Software Development Via Scrum and Kanban," *Int. J. Innov. Technol. Manag.*, vol. 17, no. 4, pp. 1–25, 2020, doi: 10.1142/S0219877020300025.

[48] P. Kumar, P. Sahithi, M. Pradeep Kumar, and B. Tech Student, "Implementing Scrum and Kanban Approaches for E-Commerce Web Application: An Agile Framework," *IJSRD-International J. Sci. Res. Dev.*, vol. 9, no. April, pp. 2321–0613, 2021, [Online]. Available: https://www.researchgate.net/publication/353014043

[49] K. Risener, "A Study of Software Development Methodologies," *Comput. Sci. Comput. Eng.*, pp. 1–31, 2022, [Online]. Available: https://scholarworks.uark.edu/csceuht/103

[50] M. STOICA, M. MIRCEA, and B. GHILIC-MICU, "Software Development: Agile vs. Traditional," *Inform. Econ.*, vol. 17, no. 4/2013, pp. 64–76, 2013, doi: 10.12948/issn14531305/17.4.2013.06.

[51] Q.M. Yas, Mahdi, A.F., AL-Shamary, A.K.J. and Radam,N.S, "A Multi Criteria Analysis in Ranking Composite Material Using Gray Relational Analysis: A Case Study," in *In 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2020, pp. 1–7.

[52] Q. M. Yas, B. N. Adday, and A. S. Abed, "Evaluation Multi Diabetes Mellitus Symptoms by Integrated Fuzzy-based MCDM Approach," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 13, pp. 4069–4082, 2021.

[53] Q. M.Yas, E. M. Hameed, A. Badr, and B. Al-bander, "Multi Risk Factors Evaluation for Lung Cancer Incidence Based Decision Support Systems," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 13, pp. 3409–3419, 2021.

[54] A. A. Zaidan, B. B. Zaidan, M. A. Alsalem, O. S. Albahri, A. S. Albahri, and M. Y. Qahtan, "Multi-agent learning neural network and Bayesian model for real-time IoT skin detectors: a new evaluation and benchmarking methodology," in *Neural Computing and Applications*, Springer London, 2020, pp. 8315–8366. doi: 10.1007/s00521-019-04325-3.

[55] Q. M. Yas and G. K. Ouda, "Toward on Develop a Framework for Diagnosing Novel- COVID-19 Symptoms Using Decision Support Methods," in *Communications in Computer and Information Science*, 2022, pp. 1–18.

[56] A. K. Jassim Al-Shamary, Q. M. Yas, A. M. Badr, R. Al Shalabi, and S. H. Aldulaimi, "Multi Criteria Decision Making Technique for Evaluation and Selection performance Large Scale Data of Composite Materials," in *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems, ICETSIS 2022*, IEEE, 2022, pp. 96–103. doi: 10.1109/ICETSIS55481.2022.9888862.

[57] Q. M. Yas and F. K. Zaidan, "A NEW HYBRID MULTI-CRITERIA DECISION APPROACH FOR EVALUATING AND BENCHMARKING VACCINES," *Int. J. Tech. Phys. Probl. Eng.*, vol. 15, no. 54, pp. 33–38, 2023.

[58] A. A. M. Al-Azzawi, M. L. Talal, B. J. Khadhim, and Q. M. Yas, "Selecting Optimal Educational Boards Based on a Decision Support Approach," *Int. J. Tech. Phys. Probl. Eng.*, vol. 15, no. 1, pp. 345–351, 2023.

[59] Q. M. Yas and A. A. Zadain, "Towards on Develop a Framework for the Evaluation and Benchmarking of Skin Detectors Based on Artificial Intelligent Models Using Multi-Criteria Decision-Making Techniques," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 31, no. 03, p. 1759002, 2017, doi: 10.1142/S0218001417590029.

[60] F. M. Jumaah, A. A. Zaidan, B. B. Zaidan, R. Bahbibi, M. Y. Qahtan, and A. Sali, "Technique for order performance by similarity to ideal solution for solving complex situations in multi-criteria optimization of the

tracking channels of GPS baseband telecommunication receivers," *Telecommun. Syst.*, pp. 1–19, 2017, doi: 10.1007/s11235-017-0401-5.

[61] S. H. Aldulaimi, "The Evaluation of Virtual Training and Employee Effectiveness : A Case Study from Babco," *Glob. Sci. J.*, vol. 10, no. 5, pp. 2309–2324, 2022.

[62] B. Al-bander, Q. M. Yas, H. Mahdi, and R. K. H. S. Al-hamd, "Benchmarking of deep learning algorithms for skin cancer detection based on a hybrid framework of entropy and VIKOR techniques," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 29, pp. 2634–2648, 2021, doi: 10.3906/elk-2103-65.