A Hybrid Model for The Software Development Life Cycle

Dr. Manish Kumar, Dr. Ashish Kumar Saha

Vinoba Bhave University, Hazaribag, Jharkhand-825301

DOI: https://doi.org/10.51583/IJLTEMAS.2025.140400124

Received: 29 April 2025; Accepted: 08 May 2025; Published: 24 May 2025

Abstract: This study addresses an important and critical problem in the area of information technology. The software development life cycle, also known as development models, is the subject of software management practices that analyze software development. The main purpose of this research is to craft a development model that meet the requirements of various system and eliminate the defect of earlier model. This study suggests a hybrid model that incorporates elements from the five most popular development models: waterfall, iteration, spiral, V-shaped, and extreme programming. Through some modifications, the model suggested in this study maintains some of the characteristics and advantages of previous models. As a result, it avoids and resolves many of the software issues that plagued previous models. Therefore, the model we have just suggested is an integrated model that applies to most software applications and systems.

Keywords: Software Management Processes, Software Development, Development Models, Software Development Life Cycle, Hybrid Model.

I. Introduction

The being of various system development models in software field has created a series of problems due to their different applications and usage patterns. System engineers were the only ones who could recognize and address these problems while creating software. The cost and effort required to develop these systems could increase due to these problems. Some of these system models lack risk analysis and plans, and this is one of these problems. Limiting to large projects is another problem. To address the cost and effort factors, especially in small and medium-sized projects, this study attempts to create a model that can be used for all projects regardless of their size.

Objectives

Create a model proposal that mimics the benefits of earlier software process management models.

Implement the suggested model in a variety of projects to demonstrate its functionality and ensure its relevance.

The Rationale behind Selecting This Research Topic.

Some of the factors that influenced the choice of the research topic are detailed below:

Software engineering is considered a fundamental part of design and plays an vital role in the development and creation of programs.

The relationship of this model with the development of different software and its notable importance in the face of numerous software engineering projects.

Examine various models to determine their advantages, disadvantages, strong point and flaw.

Build a model that integrates or combines different aspects of the strength of the previous software engineering models.

The Proposal for the Intended Model: "hybrid model".

A software process is simplify and presented from a particular point of view in a software process model [1]. Reuse-based development, formal systems development, evolutionary development, and waterfall model are just some of the general software process models. Software development models are compiled into a single model as shown in Figure (1). These models are:

1. Waterfall model
2. Iteration model
3. V-shaped model
4. Spiral model
5. Extreme model

The Fundamental Hybrid Model in the context of the System Development Life Cycle (SDLC).

The main components of the systems development processes are shown in Figure 1, along with the logical order of these components or phases, ensuring that this model is appropriate for many phases of systems development, whether small, medium, or large. These procedures constitute the main form of the hybrid model:

Planning: Includes organizing the activities needed to specify resources and schedules, as well as creating plans for the system development process and other project-related information. [6]

Requirements: This phase allows us to identify three main types of requirements:

Abstract functional requirements: which articulate the functions of the system in an idealized framework?

System properties: which describe the non-functional requirements that the system must satisfy.

Undesirable characteristics: which specify behaviours that are not acceptable within the system. It is essential that these requirements effectively define and support the organization's overall goals for the system. [7]

Design: This is a multifaceted process with several phases, which includes four essential elements:

Data structuring

Software development

Data representation

Detailed processes (algorithms)

The specifications of the design process are converted into a format suitable for the software, allowing its quality to be assessed before the next phase, called "Implementation", begins. During this phase, the design is carefully documented for integration into the software repository**.**

Implementation: This phase involves converting the project into a computer-comprehensible format. It also organizes the system components using one of the programming languages according to the established plan.

Integration Development: This phase aims to connect the different components of a system into a coherent unit, thus creating a formal system. During this phase the different parts must be integrated with each other without damaging the rest of the system components.

Deployment: This is the process of delivering the complete system to the client for operational use, helping to identify problems that may arise during its initial application.

Testing: The testing phase begins with the initial phase of any system, namely "planning". It includes test planning, requirements gathering, design evaluation, representation and integration.

Maintenance: Once the software is delivered to the client, it may undergo various modifications. These changes are usually motivated by difficulties encountered, the need to adapt to changes in the external environment or requests from clients to improve performance or functionality.

Risk Analysis: This phase includes all development phases, from planning to maintenance. It identifies all anticipated risks and recommends the actions necessary to mitigate them.

The development of medium and large systems begins with a planning phase, during which the essential requirements of the system such as personnel, time, materials and costs are established. In contrast, small projects typically begin with the design or programming phase, during which the appropriateness of the proposed plan for the project is assessed. This phase also serves to identify and assess project risks associated with schedule, human resources and materials considerations. Next comes the requirements gathering and analysis phase, during which requirements are gathered through customer participation and are then analyzed to determine their conformity with the system objectives, taking into account existing risks. The requirements phase is followed by the design phase, which focuses on creating preliminary designs through algorithms and diagrams that illustrate the development of the system. This design is then tested to ensure that it meets the stated requirements. Next comes the system representation phase, which involves implementing the project for the computer devices according to the specifications described in the project. In addition, the programming components are tested to confirm their proper functioning. The integration phase follows the system representation, in which all implemented programming components are integrated and tested for two fundamental purposes: to verify the absence of defects and to ensure compliance with all system requirements. This phase must be completely completed before deploying the system for customer use. The final phase involves the deployment and installation of the system on customer devices, which is critical and must be carried out in accordance with the established plan and risk analysis. [1][7]

The hybrid model is distinguished by its suitability for small, medium and large installations, as well as its inherent flexibility. Within this framework, the process can be initiated at any stage, including planning, requirements gathering and analysis, design or programming representation, depending on the specific type and complexity of the system.

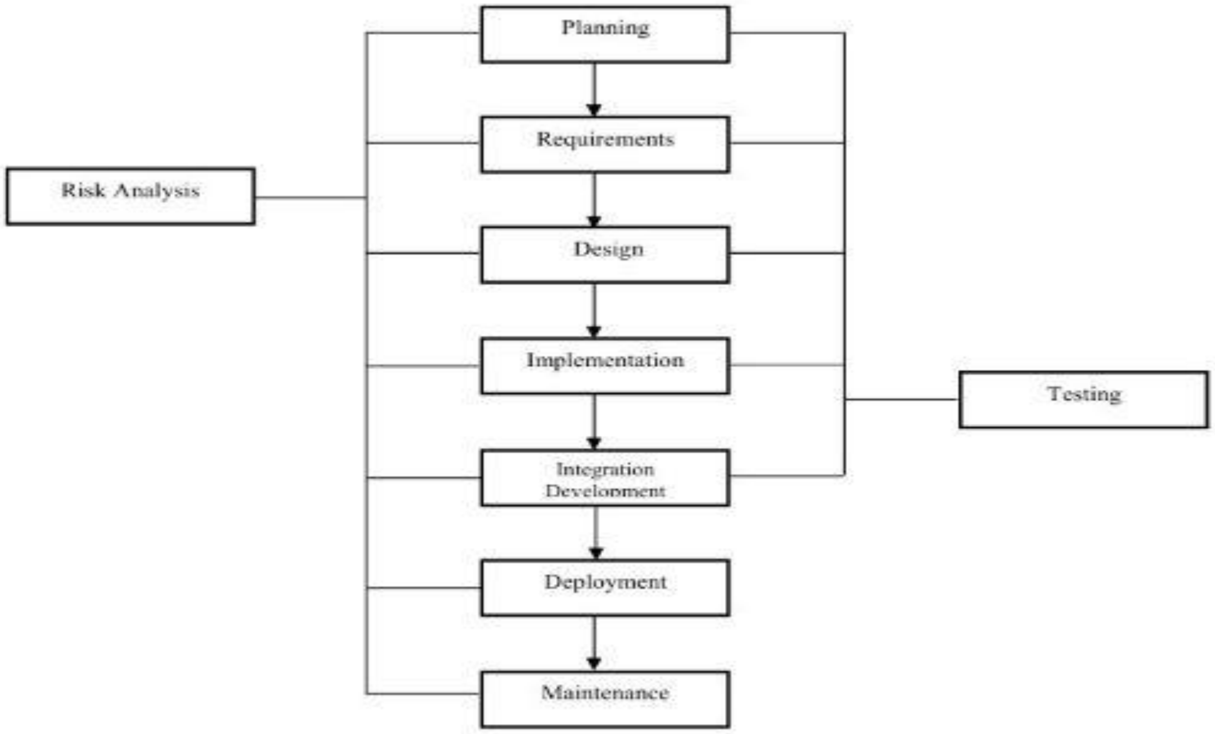

Fig. (1): The Primary of Hybrid Model in SDLC

An analysis of the Hybrid Model in the context of the System Development Life Cycle.

The complete illustration of the hybrid model sheds light on its operational framework for various system development methodologies, as shown in Figure 2. This illustration describes the relationship between the planning phases and the phase dedicated to assessing potential risks associated with the project. Further evaluate the plan based on the identified risks. Furthermore, the figure highlights the interconnections between the planning and risk analysis phases and those involved in requirements gathering, design, implementation and finally integration of programming components, as well as implementation and maintenance of the system. The proposed hybrid model offers several advantages:

Advantages:

Ease of understanding and implementation.

Encourage good practices: define before design, design before coding.

Clearly define goals and deliverables.

This methodology is particularly effective for existing products and teams that may not be strong enough.

It is simple and easy to use.

Each stage of the process is associated with a different outcome.

The probability of success is higher than the waterfall model and other methodologies, mainly due to the timely formulation of test plans throughout the development lifecycle.

It is suitable for both small projects with clearly defined requirements and more complex projects. Considerable emphasis is placed on risk analysis.

This approach is beneficial for large-scale critical initiatives.

Promotes strong team cohesion and prioritizes the final product.

The process is iterative in nature and adopts a test-based strategy to manage requirements and ensure quality assurance.

Figure 2 illustrates the relationship between the different testing processes and the different development phases, with the exception of the implementation and maintenance phases. These latter phases are linked to the previous development phases and risk analysis processes. In addition, the design phase is categorized into two segments: high-level design, which focuses on the fundamental aspects of system design, such as graphics and algorithms based on the type of design used, and low-level design, which focuses on programming design and technical details that support the overall system design. Additionally, this model allows to launch any phase of the system development process, adapting to a wide range of project types.

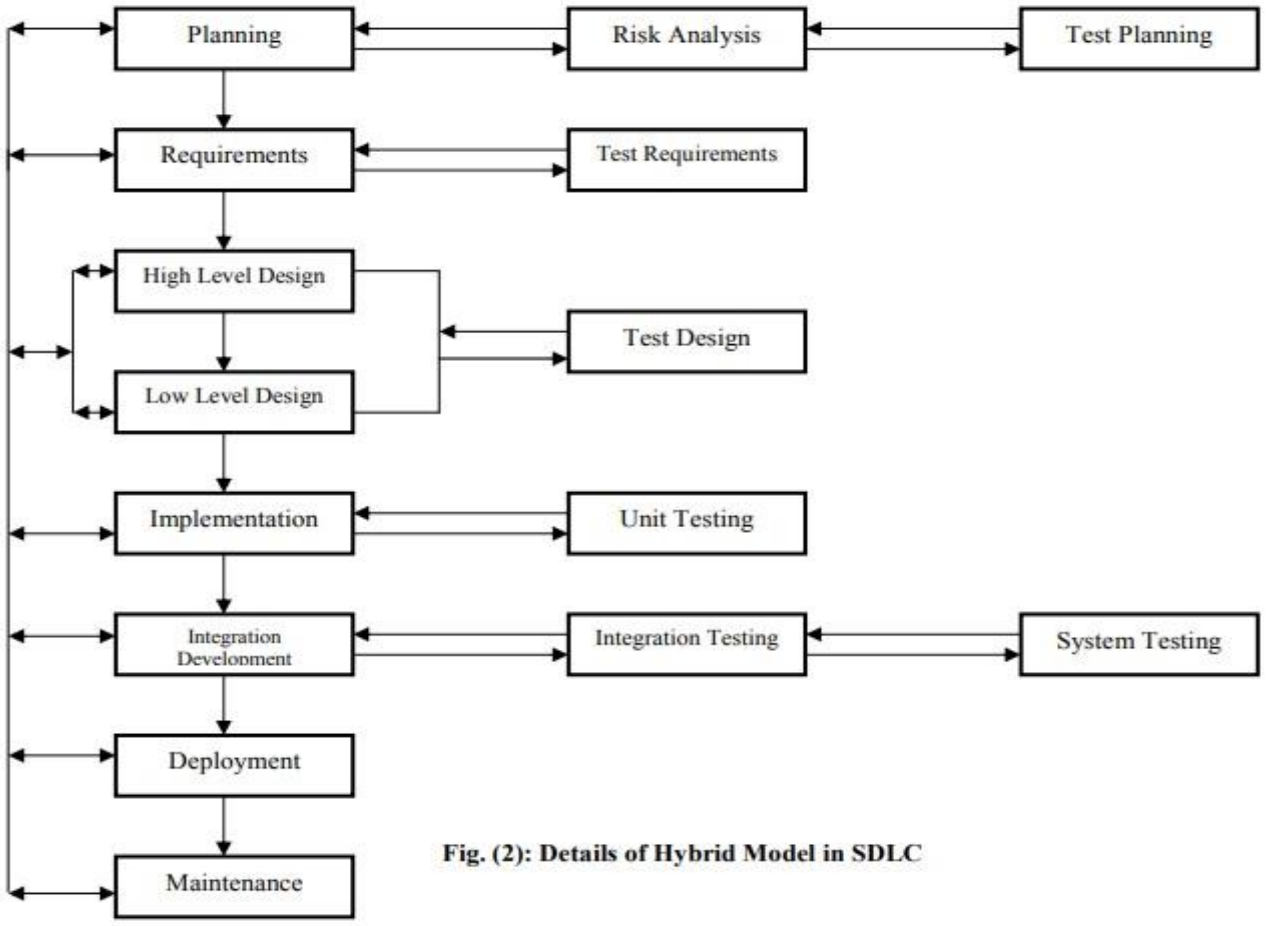

Fig. (2): Details of Hybrid Model in SDLC

An analysis of the Hybrid Model in relation to the Spiral Model.

The models can be compared as shown in Table (1) below. [8]

Hybrid Model	Spiral Model
This approach is designed to accommodate projects of all sizes, from small to large.	This framework is ideal for larger and more significant projects.
It specifies the endpoint for each distinct phase of the project.	It involves phases that are both frequent and overlapping.
The model prioritizes the planning phase alongside risk management strategies.	The focus remains on effective risk management.
It is user-friendly and can be easily applied, especially in small to medium projects.	Implementation requires a certain level of experience.
The success of the project is dependent on the stages of risk analysis and testing.	It is based on the concept of repetition to create outcomes that exceed a simple prototype.

Conclusion and Suggestions for Future Work

II. Conclusion

Based on the established model, the following conclusions are drawn:

The hybrid model is based on five development methodologies: waterfall, spiral, iteration, V, and extreme programming models.

The hybrid model integrates the advantages of these five methodologies while being adaptable to projects of different sizes, including small, medium, and large-scale efforts.

The hybrid model is divided into two parts: a main part that deals with the basic processes and a detailed part that focuses on the operational aspects of these processes.

Recommendations for Future Research

Conduct a comparative analysis of the hybrid model with five other development methodologies in terms of profitability and risk management, based on specifically defined criteria.

Apply the hybrid model to a wide range of projects, taking into account different requirements and needs.

Enhance the hybrid model to integrate traditional methodologies with advanced software engineering techniques such as object-oriented systems development.

References

1. Lan Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.
2. Karlm, "Software Lifecycle Models", KTH, 2006.
3. Lifecycle Models, National Instruments Corporation, 2006.
4. Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.
5. Rlewallen, "Software Development Life Cycle Models", 2005.
6. Barry Boehm edited by Wilfred J. Hansen, "Spiral Development: Experience, Principles, and Refinements", 2000.
7. Roger S Pressman, "Software Engineering: A Practitioner's Approach", 7th edition, McGrawHill, 2009.
8. B. W. Boehm, "Classics in Software Engineering", Yourdon Press -Upper Saddle River, NJ, USA, Pages: 323 – 361, ISBN: 0-917072-14-6, 1979.