Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Navigating Software Development Methodologies: A Comparative Analysis and Recommendation of Software Process Models

Purvi Sankhe[1], Dr. Mukesh Dixit[2]
[1].*Research Scholar, Sanjeev Agrawal Global Educational University, Bhopal, Madhya Pradesh, India.*
Email:purvisankhe@gmail.com
[2]. *Associate Professor, Sanjeev Agrawal Global Educational University, Bhopal, Madhya Pradesh, India.*
Email: mukesh.d@sageuniversity.edu.in

Abstract
As software moves through the phases of system design, development, testing, and deployment, the Software Development Life Cycle (SDLC) is crucial for guaranteeing its operation and quality. This study explores the critical elements that establish the best software development process for various software applications. Within the two main paradigms of agile development and traditional development, it assesses six development process models. It also suggests a new model that makes use of machine learning to suggest the best course of action for a certain project. Prototyping, Spiral, Scrum, Extreme Programming (XP), Rapid Application Development (RAD), and Feature-Driven Development (FDD) are just a few of the software development approaches that are analyzed and contrasted in this study. The suggested model uses a two-step procedure for making recommendations: Choosing a Holistic method: The first step includes deciding on a suitable software development method in accordance with project data parameters, including project kind, complexity, and team makeup. Enhanced technique Selection: In the second phase, the model enhances the technique selection within the selected development approach by taking into account the particular project scope, schedule limitations, stakeholder preferences, and required flexibility. The goal of the paper is to give developers a methodical framework for choosing software development approaches that are in line with project specifications and limitations all along the It provides a data-driven approach to method selection by incorporating machine learning into the recommendation model, increasing the probability of successful project execution and desired results. The ultimate goal of this study's findings is to help software development projects succeed by promoting well-informed decision-making and efficient resource allocation. Selecting a software development technique has advanced significantly with the help of the suggested recommendation model. Through the use of machine learning algorithms, it helps developers make decisions that improve the effectiveness and success of their projects.
Keywords*: Development, Project, Recommend, Process, Methodology*

1. Introduction

Choosing the right software development technique is essential to the success of an IT project in the field of project management. Systems analysts are responsible for determining the best option to follow for each project while managing it and comprehending the needs of the business. This research examines six different approaches within the main software development paradigms and suggests a machine learning model to suggest the best software development methodology for certain projects. Through methodical planning, development, testing, and system rollout, the software development life cycle (SDLC) is

essential to guaranteeing software functionality and fulfilling user requirements [1][2]. Within the SDLC spectrum, there are several strategies that range from the iterative and adaptive agile approaches to the sequential, plan-driven waterfall method [3]. The spiral paradigm, for example, combines the iterative flexibility of agile with structured phases similar to waterfall. The best SDLC method depends on a number of variables, including team preferences, resource availability, project complexity, and project size. The SDLC is essential to efficient project management, which includes adhering to budgets, schedules, and quality standards. In addition, the SDLC guarantees that user and stakeholder needs are satisfied and assists in managing project risks [4]. Notably, many software development methodologies— such as Extreme Programming (XP), Scrum, Rapid Application Development (RAD), and prototyping— offer unique benefits and limitations that make them appropriate for projects with differing time constraints and communication needs [5][6].

The purpose of this study is to present an in-depth analysis of software development methodologies in relation to important data factors. Prior studies demonstrate the existence of recommendation systems designed to assist in choosing technologies and techniques for software requirement elicitation for the software development life cycle (SDLC) [7]. Still, there are gaps in creating a coherent recommender model that incorporates these insights for choosing a software development process.

Motivated by these gaps, this research suggests a two-stage recommender model in response to these deficiencies. IT professionals can choose the best software development technique in the initial step by considering factors like project complexity, scope, resource availability, and team dynamics. Subsequently, the model takes stakeholder preferences, project scale, timetable, and desired flexibility into account when fine-tuning methodology choices within the selected approach.

The proposed system aims at assisting IT developers in choosing software development methodologies that correspond with project requirements and limitations, guaranteeing efficient project management and intended results. This paper makes a significant contribution by providing a methodical approach to methodologically sound decision-making when choosing a software development technique.The remainder of this paper is structured as follows: The relevant literature on software development methodologies and choosing a software development technique is reviewed in Section II. The methods used to create the recommender model is presented in Section III. Section IV describes the model's results obtained in depth. Section V highlights discussion on comparison of proposed system and existing system whereas section VI concludes by discussing the findings, consequences, and potential future directions of this study. IT developers will acquire the knowledge required to properly manage the software development lifecycle through the use of this methodical approach.

2. Literature Review

This article [8] covers Waterfall, V-Model, Iterative, Agile, and Hybrid techniques in detail, as well as industry approaches for managing information system projects throughout their lifecycles. It explores how decision support matrices help choose the optimal course of action and contrasts Agile and conventional methods. The intention is to help project managers and information system practitioners make well-informed decisions specific to their initiatives. The article demonstrates the distinct benefits of several SDLC approaches, including Waterfall, Agile, Scrum, Kanban, and Iterative models. By contrasting Agile with conventional techniques—conventional methods stressing structure, while Agile methods emphasize flexibility and client involvement—one can see how adaptability and systematic planning may coexist in harmony. The best project management approach is chosen depending on the particulars of the project with the use of decision support matrices. The correct development approach is essential for effective IS project management since it entails choosing and modifying strategies that use innovation, teamwork, and technology to accomplish organizational objectives.

Kendra E. Risener (2022) [9] discusses six methodologies in software development: waterfall, agile, spiral model, extreme programming, SCRUM, and Kanban. The paper explores the benefits and drawbacks of each methodology and contrasts them to determine the most suitable for IT companies. The research identifies a gap in understanding methodologies in software engineering to enhance development efficiency, avoid financial and time losses, and reduce developer frustration. By focusing on common

characteristics of development frameworks, this study aims to promote interest in effective development approaches that can save crucial resources: money and time.

3. In order to improve software security, a wide range of concepts and software development methodologies have been examined in this paper by authors [10] Any gaps or weaknesses in these approaches have also been noted. A brief analysis of various software development methodologies is provided to highlight their unique contributions to the development process as a whole. The rationale behind giving software security top priority is discussed, including a look at common threats. Furthermore, the theory of software security is examined in terms of its practical applications and basic principles. Additionally, approaches to detecting and resolving security vulnerabilities are discussed. The analysis shows that developing secure software requires a comprehensive and proactive approach. Establishing secure design principles and incorporating security requirements from the outset of development marks the beginning of this process. By implementing security measures from the outset of the design process, implementing secure coding practices, carrying out regular security testing, and following secure configuration and patching procedures, organizations can significantly lower the risk of security vulnerabilities. Gap identified is: To ensure that security is incorporated into the software development process from the beginning to the end, more investigation is required. While there are tools available for detecting errors in the code, their precision and efficiency could be improved. Our main goal should be to improve these tools' automatic code-checking capabilities. Furthermore, in order to improve software's defense against contemporary threats and issues, we must investigate novel approaches to its design.

The goal of this study [11] is to examine how well Agile approaches may be implemented in software projects as a more efficient substitute for the conventional waterfall methodology. Finding the critical elements that guarantee software is continuously delivered to the market at the predetermined quality level and timetable is the main goal. Scrum and Kanban, two well-known Agile implementation techniques, are assessed in this study. These techniques are evaluated according to variables including complexity, practicability, and predictability. Through this assessment, the study hopes to draw attention to the benefits and drawbacks of each methodology and offer insights into project characteristics that might help choose the best methodology for a particular project. Of all the Agile implementation techniques examined, Scrum turned out to be the most basic and reliable. This structure makes it easier to release software products on a regular basis, which empowers teams to handle challenging issues and quickly adjust to shifting market conditions. The study's limitation is that, despite thoroughly analyzing the characteristics and methods of both agile and traditional methodologies, data parameters for the selection of development methodologies have not been provided. Additionally, specific recommendations regarding the methodology to be chosen for software product development under different conditions have not been made.

In accordance with project management industry standards, this study [12] use a six-pointed star framework to carefully examine several software development process models (PMBOK). The study comprises: choosing well-known software development methodologies such as RAD, Spiral, Agile, Z, and AZ, or waterfall, iterative, and Z.

creating a survey to assess these models using the six-pointed star framework's requirements. giving the survey to seasoned people in the software sector in order to get their input. use statistical techniques to analyse the survey data and evaluate each model's performance with varying project sizes and variables. assessing and contrasting lightweight and heavyweight approaches to ascertain which models are best suited for particular project objectives and scopes. Research gaps stated: Limited participant diversity and possible prejudice towards more seasoned users of software industry. Survey replies that are subjective affect objectivity. Concentrate on well-established models, sometimes ignoring cutting-edge approaches. Restricted applicability of the results. absence of qualitative understanding in addition to quantitative analysis. Filling in these gaps can improve the study's relevance for scholars and practitioners of software development.

This paper [13] has discussed 11 methodologies namely – waterfall, agile, prototype method, pragmatic programming, spiral model, RAD model, dynamic system development model, extreme programming, SCRUM, v-model and iterative and incremental model. Methodology in this paper focuses on The two practices that make up software development methodologies are heavyweight and lightweight. Applying heavyweight methodologies is worthwhile for projects with well-defined requirements that don't require modifications. One of the heavyweight approaches. The project owner is clearly involved in the phases of planning and research. Lightweight approaches are suitable for projects with changing requirements and unstable specifications as a result of internal or external development project factors. gaps identified in this research states: It claims that it lacks a methodology to meet all the needs and produce an ideal finished product. With the assistance of seasoned team members and project managers, the optimal and appropriate methodology should be chosen after taking into account all the variables. For the development process to produce better results, a combination of software methodologies could be chosen as opposed to just one.

This study [14] explores into a comprehensive analysis of several software development processes and tools, exploring their individual merits and downsides in detail. The purpose of this study is to provide project owners with useful documentation that will help them choose the best software development approach for their needs. In actuality, no single methodology is ideal in every situation involving a project. Skilled project teams use a mix of approaches according to the particular requirements of the project. Through the effective application of the best-fitting approaches, this adaptive approach optimizes software development outcomes for a variety of project situations. research gap identified is that this study provides document based on study of methodologies no any experimental analysis is done to suggest development methodology.

In order to investigate the fundamental justifications for software developers' adoption and application of Software Development Methods (SDMs), this study [15] reviewed the literature on the subject. Studies that analyzed the rational, irrational, and/or non-rational reasoning behind SDM behaviors were categorized using an analytical framework derived on Weber's work on social acts [16]. 28 of the 111 empirical studies on SDM acceptance and use that were found explicitly looked at the reasoning of software engineers. Of them, 14 studies dealt with the adoption of SDM and 14 with its use. The results show that a significant portion of the body of knowledge already in existence involves study on the reasoning of software developers. Furthermore, the analysis indicates that the majority of research currently in circulation favours adopting and using SDM primarily from the perspective of logical decision-making.

This paper [17] Includes guidelines to teach students about various process models. Paper is focusing mainly on Waterfall Model, Incremental Model, Agile Mode (Extreme Programming). also includes explanation of the process models and how artifacts are created in different process models to the students using the game SimSE. One of the main reasons SimSE is used in the classroom is to explain how different software engineering tasks are completed in various process models. Because most tasks in all models are similar., Gap identified states, it can occasionally be very challenging to theoretically explain about selection of appropriate software process model for the development of project.

This study [18] examines a number of well-liked software development approaches, with a focus on project-characteristic-based selection. It talks about the difficulties in implementing new techniques and suggests ways to get beyond them. The document also describes best practices for developing software products. Nevertheless, only extreme values are taken into consideration for the project characteristics, and not all accessible approaches are included in the assessment and selection procedure in this work. A broad range of software development approaches would provide more reliable outcomes, giving developers more choices and improving the process of selection accuracy. The study's identified research gap calls for a review, the incorporation of new approaches, and the provision of a more thorough chart to help with the selection process.

With a focus on their function in software development management, this study [19] examines software development life cycle models and the related approaches. It highlights the distinctions between approaches, how they affect project success, and how to compare them. The article examines current two-

and single-criteria classification techniques, pointing out their drawbacks and suggesting fresh single-criteria classification strategies to overcome them. A unique four-criteria multi-criteria categorization system is also presented, offering a thorough hierarchical foundation for software development processes and life cycle models. Along with discussing current approaches, the study presents a novel strategy based on machine learning and retrospective analysis techniques for choosing a software development style. In order to be in line with the Sustainable Development Goals (SDGs). One of the gap identified here is that it doesn't offer a recommendation system to assist developers in choosing a particular methodology for creating software projects of better quality.

3. Methods

The traditional software development methodology and the agile software development approach are the two primary methods available for the development of software products [20] [21]. Additionally, the Agile software development technique is classified into Scrum, Extreme Programming (XP), and Feature Driven Development models, while the traditional software development approach is divided into Rapid Application Development (RAD), Prototyping, and Spiral Model [22].

3.1 Traditional software development approach

The conventional software development technique is a popular approach that is recognized for its methodical and structured nature. It progresses in distinct stages, each of which depends on the success of the preceding one [23]. This approach works well for projects whose requirements are well-defined and stable and where frequent changes are not expected. Its rigid structure, however, can make it difficult to adjust when a phase is over [24][25].Methodologies like Rapid Application Development (RAD), Prototyping Model, and Spiral Model are frequently used under the Waterfall methodology.

3.1.1. RAD Model

With short cycles and element-based building, the Rapid Application Development (RAD) methodology focuses on producing software quickly. When project scope is constrained and needs are well-defined, its goal is to swiftly construct working systems [26].

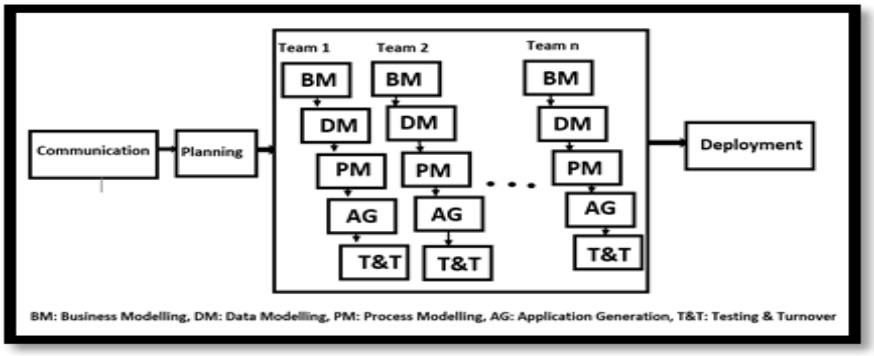

Figure 1 Rapid Application Development Model

RAD uses focus groups or workshops to collect requirements, iterative prototyping with user input, component reuse, and rigorous deadlines for design improvements. As seen in figure 1, through iterative development and component reusability, the RAD paradigm comprises phases such as Business Modeling, Data Modeling, Process Modeling, Application Generation, and Testing to enable quick and effective software delivery. The Rapid Application Development (RAD) methodology was created to address the shortcomings of conventional system development techniques, including the waterfall model and its variations [27].

3.1.2 Prototyping Model

A software development methodology called the Prototyping Model shown in figure 2 places emphasis on building a working prototype first in order to get user feedback and fine-tune requirements before

releasing the finished product. It includes making incremental changes to the prototype in response to user feedback until the intended functionality and design are realized. Ensuring that the software closely aligns with user expectations and needs is the goal of this model. It is particularly helpful in situations when needs are ambiguous or prone to change, and when early user participation and validation are crucial. The Prototyping Model facilitates communication between development teams and stakeholders, which leads to a more successful and user-friendly final product. Before creating the complete product, prototyping allows for the affordable testing of a solution concept with actual users [28] [29].

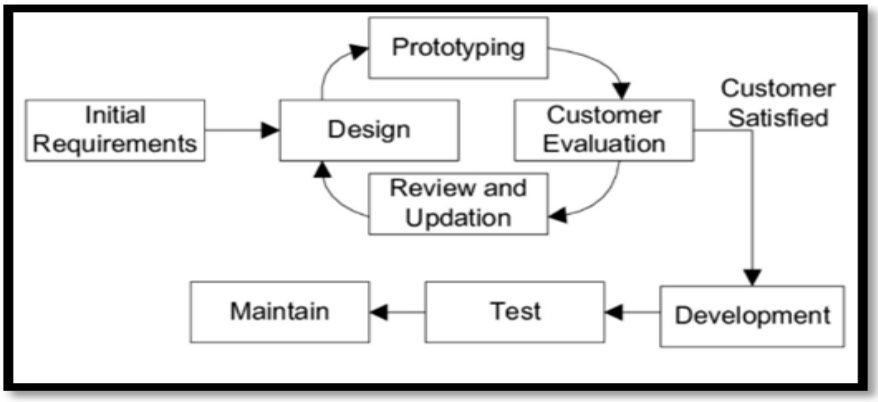

Figure 2 Prototyping Model

3.1.3 Spiral Model

A software development methodology called the Spiral Model combines aspects of the flexible iterative approaches and the structured Waterfall Model. As seen in figure 3 it divides a project into cycles that are iterative and comprise phases for planning, risk analysis, engineering, and evaluation. Its emphasis on proactive risk management, which attempts to recognize and resolve possible problems at every iteration, is one of its defining characteristics. Large, complicated projects with plenty of unknowns and serious implications from failure are best suited for this strategy [30]. It makes it possible for customers to be involved in the process continuously, adjusts easily to changing needs, and guarantees that well-documented, high-quality software is produced through constant improvement. Owing to these benefits, the Spiral Model is frequently employed in sectors where safety, dependability, and risk management are crucial, such as aerospace, defense, and critical systems development. Through developing prototypes, spiral development reduces development risks during the iterative and incremental process of building a system [31] [32].

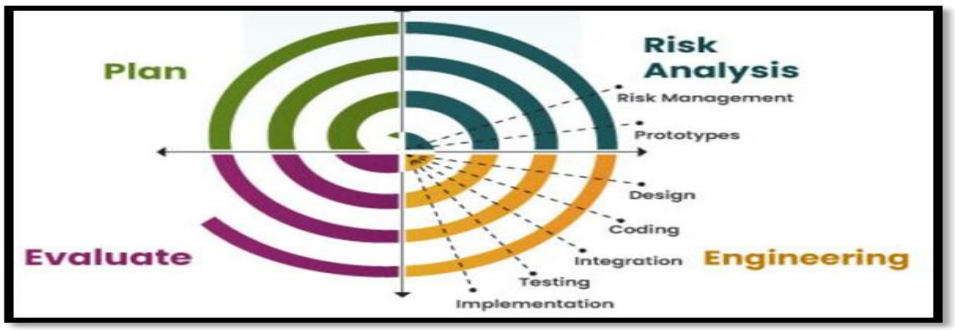

Figure 3 Spiral Model

3.2 Agile software development approach

Agile software development is a process that emphasizes iterative progress and involves autonomous collaboration amongst cross-functional teams to revise requirements and create solutions [33] [34]. Its

ability to deliver value quickly, improve the level of reliability and quality, and increase adaptation to changing needs a revolutionary change in development efficiency and responsiveness is its main advantage [35] [36].

3.2.1 Extreme Programming

Extreme Programming (XP), Scrum, and Feature-Driven Development (FDD) are all agile software development methodologies that focus on iterative, flexible, and collaborative approaches to software development. Figure 4 highlights process of extreme programming. XP is a software development methodology that emphasizes rapid feedback, continuous testing, and continuous integration. XP places a strong emphasis on customer involvement and team communication, with the goal of delivering high-quality software that meets the customer's needs. XP also advocates for pair programming, where two programmers work together at a single computer, to increase code quality and knowledge sharing. Extreme Programming is a collection of principles, guidelines, and procedures that are applied methodically [37].

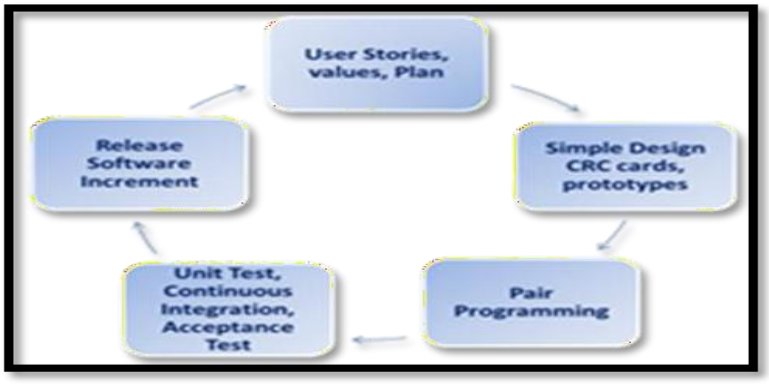

Figure 4 Extreme Programming

3.2.2 Scrum

Scrum is another popular agile methodology that emphasizes iterative development, collaboration, and self-organization. In Scrum, development work is divided into short iterations called sprints, each typically lasting between one and four weeks [38] [39]. During each sprint, the development team works to deliver a working product increment that meets the customer's requirements. Scrum also includes a set of roles, ceremonies, and artifacts, such as the Product Backlog, Sprint Review, and Daily Scrum, to help ensure effective communication and collaboration within the development team. Figure 5 shows working of scrum framework for the development of software product [40].

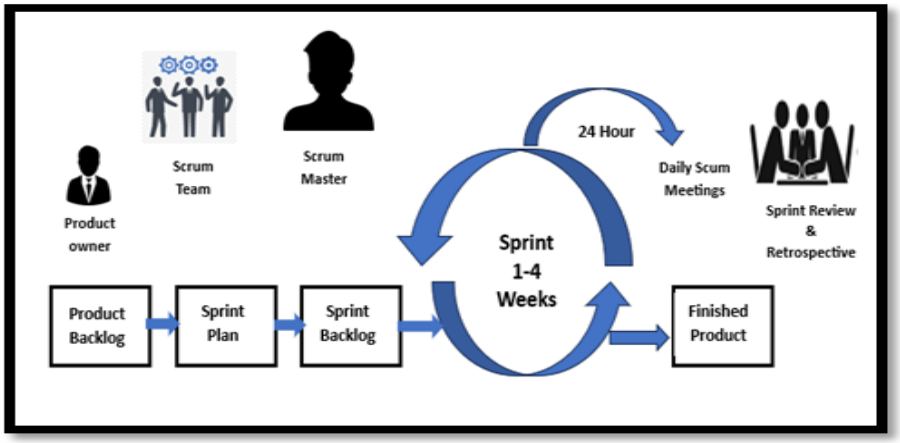

Figure 5 Scrum Framework

3.2.3 FDD

Feature-Driven Development (FDD) is an iterative and incremental software development process that focuses on delivering small, working software features. FDD starts with a high-level view of the project, identifying its overall scope and objectives, and then proceeds to break the project down into a set of small, manageable feature sets. Each feature set is then further broken down into individual features, which are implemented and tested iteratively. FDD also emphasizes the importance of good design and modeling practices, such as using UML diagrams and feature- centric design. Due of the large number of software features accessible today, the FDD model is frequently used in software development. The cluster approach offers software development problem-solving techniques, one of which is the FDD model, which concentrates on development projects with several features [41].FDD suffers with inadequate documentation and a lack of team member and customer communication controls. Additionally, there is a lot of modeling work and little iteration across the stages and stakeholders in FDD [42].

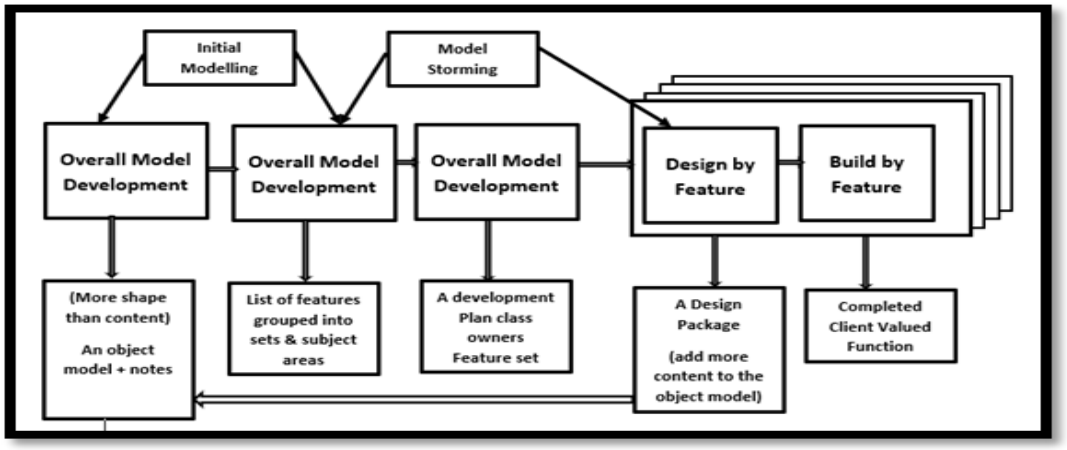

Figure 6 Feature Driven Development Model

Each of these methodologies has its own unique strengths and weaknesses, and choosing the right one for a particular project will depend on a variety of factors, such as the project's size, complexity, and team structure. However, all three methodologies share a common focus on collaboration, flexibility, and iterative development, which are key principles of agile software development. Comparison between different software development methodologies is shown in the table 1. It considers various factors like requirements, user involvement, development team, type of project & risks associated with it.

Table 1 Comparison between software developments models/methodologies

Sr. No.	Parameters	RAD	Prototyping	Spiral Model	Scrum	XP	FDD
Requirements of the Project							
1	Requirements are Partially defined early in the SDLC	Partially defined requirements	Partially defined requirements	Partially defined requirements	Partially defined requirements	Partially defined requirements	Partially defined requirements
2	Requirements are easily defined and understandable	Requirements are not clear at the start	Requirements are not clear at the start	Requirements are not clear at the start	Requirements are not clear at the start	Requirements are not clear at the start	Requirements are not clear at the start

3	Requirements are changed frequently	Few changes in requirements	frequent changes in requirements	frequent changes in requirements	frequent changes in requirements	frequent changes in requirements	Few changes in requirements
4	Change Management (in the early process)	change management possible during early stage	change management possible during each stage	change management possible during each stage	Easily adaptable to changes	Easily adaptable to changes	Easily adaptable to changes
User Involvement							
5	User Involvement	More involvement of user	More involvement of user	Average involvement of user	More involvement of user	More involvement of user	More involvement of user
Development Team							
6	Skilled developers/ Team	High Need of Skilled developers/ Team	Moderate Need of Skilled developers/ Team	High Need Of Skilled developers/ Team	Moderate Need of Skilled developers/ Team	High Need Of Skilled developers/ Team	High Need Of Skilled developers/ Team
7	Availability of Tester / Testing	Tester requires After Coding	Tester requires After Iteration	Tester requires After Iteration	Continuous testing	Tester requires After Coding	Continuous testing
8	Technical leadership skilled professionals	High requirement of leadership skilled professionals	Moderate requirement of leadership skilled professionals	High requirement of leadership skilled professionals	Moderate requirement of leadership skilled professionals	High requirement of leadership skilled professionals	High requirement of leadership skilled professionals
9	Team Size	Large Team size required	Average Team size required	Large Team size required	Average Team size required	Average Team size required	Large Team size required
Type of Project							
10	Project Size	Small Project Size	Average Project Size	Large Project Size	Average to large Project Size	Large Project Size	Large Project Size
11	Improvement of an Old System	Less applicable for improvement of old system	Less applicable for improvement of old system	Highly applicable for improvement of old system	Less applicable for improvement of old system	Less applicable for improvement of old system	Less applicable for improvement of old system
12	Cost	High Cost	Low Cost	High Cost	Low Cost	Reduced development Cost	High Cost
13	Duration of project	Short Duration of Project	Short Duration of Project	Long Duration of Project	Short Duration of Project	Short Duration of Project	Long term Projects

14	Deployment time	Faster Deployment time	Faster Deployment time	Moderate Deployment time	Faster Deployment time	Faster Deployment time	Faster Deployment time
15	Reusable Components	No	No	No	Yes	Yes	Reusable Features
16	Resource availability	High need of resource availability	Moderate need of resource availability	Moderate Risk Management	Moderate need of resource availability	Moderate need of resource availability	High need of resource availability
17	Flexibility	High Flexibility	High Flexibility	High Flexibility	High Flexibility	High Flexibility	High Flexibility
18	Adaptive to Customer needs	Moderate	Yes	Yes	Yes	Yes	Yes
19	Continuous Feedback	Yes	Yes	Moderate	Yes	Moderate	Yes
20	Maintenance	Average Maintenance	Average Maintenance	Complex Maintenance	Average Maintenance	Average Maintenance	Average Maintenance
Risk Associated							
21	Risk Analysis & Management	Less Risk Management	Less Risk Management	High Risk Management	High Risk Management	Moderate Risk Management	Moderate Risk Management
22	Possibility of risk association	Very less Possibility	Less Possibility	Less Possibility	Less Possibility	Less Possibility	Less Possibility
23	Security	vigorous Security	Weak Security	High Security	High Security	High Security	High Security
24	Guarantee of success	Good	Good	High	High	High	High

3.3. Analyzing the usage of software development methodologies using data parameters

3.3.1. RAD Model

To address problems early, the RAD technique makes use of stakeholder feedback and iterative prototyping. It works well for projects whose specifications vary over time, requiring early prototypes to accommodate modifications. RAD prioritizes user interaction and works well for inexperienced teams. Although RAD awarded projects have consistent funding, strict deadlines, and well-defined specifications, their significant stakeholder participation may make them unsuitable for initiatives with little resources 3.1 Rapid Application Development [43].

3.3.2 Prototyping

Projects with changing needs are best suited for the prototyping model, which enables the rapid construction of prototypes that may be improved upon as they are developed. Depending on the needs, user involvement might vary, which is advantageous for teams with less topic expertise or experience. It works well for fulfilling deadlines, facilitating resource allocation, and enhancing current systems with steady requirements. Other approaches, meanwhile, might be more appropriate for projects with a high degree of risk. The decision ultimately comes down to the needs of the project and the risks involved [44].

3.3.3 Spiral

Projects with changing or ambiguous objectives that require periodic refinement benefit from the use of the spiral model. It encourages segmenting requirements into doable chunks and smoothly implementing modifications. To make sure their requirements are met, users can participate in restricted or active ways. Because it accommodates a variety of tools and approaches and allows for ongoing input, the Spiral model is appropriate for teams with limited expertise. It helps with budgeting and resource allocation, especially for projects with tight deadlines or limited resources, and is effective for enhancing current systems with steady funding and essential requirements. The iterative structure of the paradigm encourages development's efficiency and adaptability.

3.3.4 Extreme Programming

In order to produce high-quality software, Extreme Programming (XP), an Agile methodology, places a strong value on close client involvement and iterative development. It places a strong emphasis on refactoring, pair programming, acceptance testing, continuous integration, release planning, simple design, and proactive client involvement. With techniques like acceptance test-driven requirements, user story-driven development, customer-participated pair programming, and on-site customer feedback, user involvement is crucial. The XP team prioritizes continual refactoring and operates as a small, autonomous, multidisciplinary team with shared code ownership. Although XP works well for complicated projects, it has drawbacks that must be addressed if it is to be maintained. These drawbacks include scheduling uncertainty, customer relations problems, technology difficulties, and team dynamics [45].

3.3.5 Scrum

Agile framework Scrum places a strong emphasis on iterative sprints, cross-functional teamwork, keeping a product backlog up to date, and important meetings including sprint planning, daily scrum, sprint review, and retrospective [46]. In order to make sure that customer demands are satisfied, user participation activities include creating user stories, adding to the product backlog, and offering feedback. The Scrum development team prioritizes communication and teamwork and is small, self-organizing, cross-functional, and dedicated to working on projects for sprint durations. They have all the required abilities. Risks associated with Scrum projects include those related to technology, requirements, scheduling, people, and the environment. To guarantee project success and stakeholder satisfaction, risk management strategies like code reviews, exact requirement specification, Agile planning, team development, and continuous risk assessment are employed [47].

3.3.6 Feature Driven Development model

Teamwork, comprehension of project requirements, flexibility, quality, disciplined development, and effective project management are all stressed in feature-driven development (FDD). In order to ensure end-user happiness, iterative development, user acceptance testing, feedback integration, and continual input are all crucial. Small, cooperative, and goal-oriented, FDD teams work closely with clients to produce high-calibre software. While regular releases, stakeholder input, and high-quality features are the goals of projects, there are hazards such as unclear requirements and technological difficulties. To maximize success, risks are minimized through stakeholder participation, communication, and client- centric delivery.

Table 2 contains a list of the primary questions and training data for the model using various machine learning algorithms.

Table 2 Analysis of data parameters for the selection of software development methodology

Sr. No.	Questions	WATERFALL (TRADITIONAL) METHODOLOGY			AGILE METHODOLOGY		
		RAD	**Spiral**	**Prototyping**	**Scrum**	**XP**	**FDD**
1	All the requirements are clearly defined at the start of project only.	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	Not easily adaptable to any changes after the start of the project	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
3	User is involved only during starting phase of the project.	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	Extra Skilled professionals are not required in team	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
5	No need of tester from the start of project. Tester is required only during testing phase	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	Works only on small size projects	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
7	Time taken for development of product is more	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
8	cost required is more.	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE
9	Risk analysis and management is done at moderate level.	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
10	Risk is highly focused factor	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
11	Documentation is very important or created at each phase pf project.	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
12	Preferable for improvement of an old systems	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE
13	reusable components are Developed	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
14	Flexibility in the process	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE
15	Good security provided	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
16	Deployment time is less	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE

3.4 Implementation Methodology

The machine learning methods Kneighbors Classifier, Gaussian NB, DecisionTree Classifier, Random Forest Classifier, and Logistic Regression are applied in the development of recommendation systems. The flow of the implementation technique employed to create the software development methodology recommendation system is depicted in the figure.

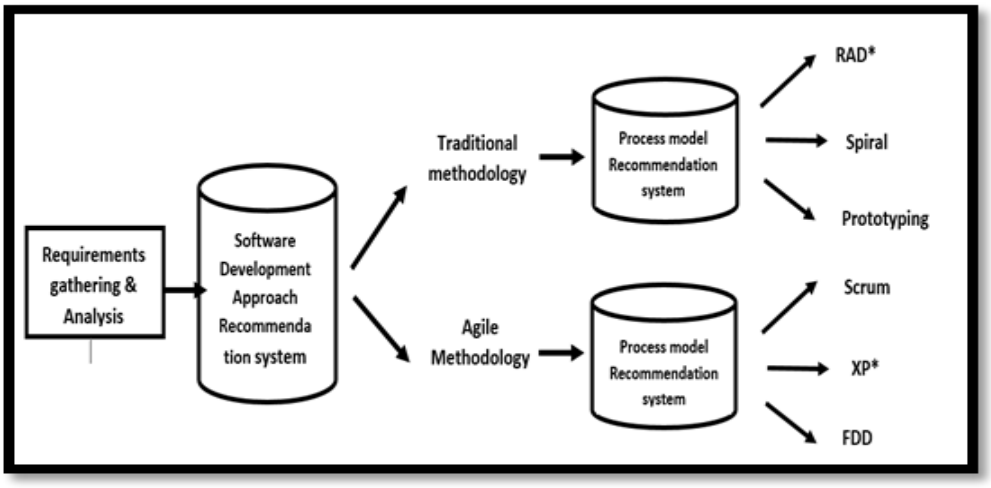

Figure 7 Recommender model block diagram

The recommendation system is divided into two stages, as shown in figure 8.
Phase I: It offers recommendations for the best software development process, including whether to use an agile or traditional software development approach.
Phase II: After deciding on a development approach, the next step is to choose between traditional and agile development methodologies. If the traditional technique is advised, phase II offers suggestions regarding spiral modeling, rapid application development, or prototyping. If the agile methodology is advised, phase II offers suggestions regarding scrum, extreme programming or Feature Driven Development.
Figure 9 shows steps used for the development of recommendation model. The following procedure makes up the methodology that was used to create the recommendation system in both phases.
I: Understand the Problem: We must first clearly define the issue that we are trying to solve. This includes identifying the tasks that must be completed and deriving certain inferences from the problem description.
II: Gather Data: Subsequently, data is collected from diverse sources. This data may originate from files, databases, or other sources, depending on the project. The amount and quality of data collected directly affect proposed system's accuracy.
III: Prepare the Data: The data must be cleaned up after it is collected. This includes removing any contradicting, missing or redundant data. Since raw data cannot be used right away, it is transformed into a clean dataset.
IV: Train the Model: To improve functionality, the model is trained. The dataset is divided into training and testing sets, with 80% of the data used for training and 20% for testing. The training set helps the model learn from the data, while the testing set allows to evaluate the model's performance.
V: Evaluate the Model: Testing is used to evaluate the model. On newly gathered data, its performance is assessed using a testing set. To assess its effectiveness, metrics including as recall, accuracy, and precision have been used. If the model doesn't work well, you might need modify a few of the parameters or settings.

VI: Use the Model: In the end, once we are satisfied with the model's performance, we may begin applying it to problems in the real world. This may mean using the knowledge the model has learned to address new issues, provide predictions, or classify data.

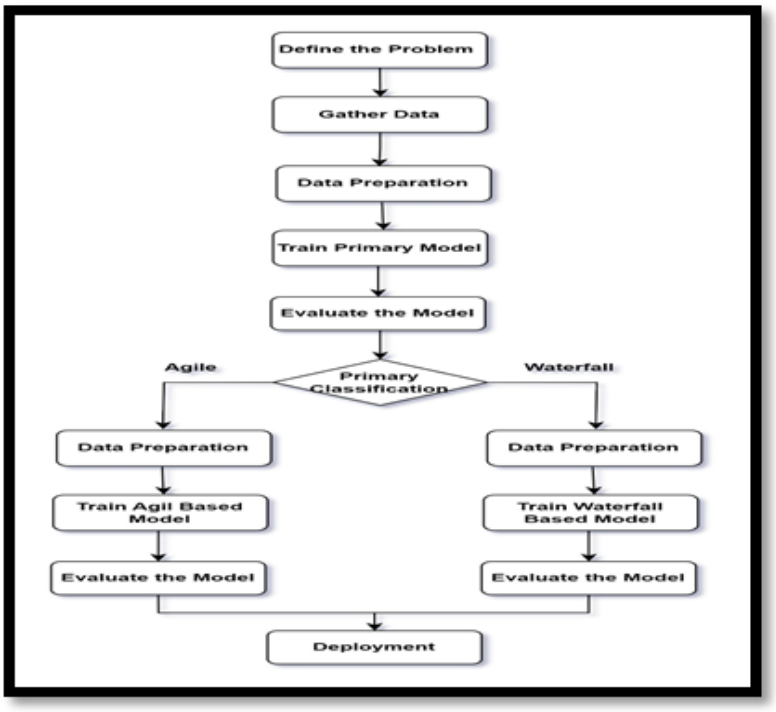

Figure 8 Flow diagram of proposed system

Briefing about the machine learning algorithms used for the development of recommendation system is as follows:

Logistic Regression: When considering scenarios where there are just two options and we wish to know how specific factors influence those options, logistic regression is a useful tool. It assists in our understanding of how numerous factors affect the results in diverse circumstances [48].

Regression modeling achieves two main goals. Firstly, it makes predictions about the outcome based on the most recent information available about the contributing elements. Second, it helps in gaining an understanding of the problem at hand by showing the proportionate contributions of each component to the outcome. We can therefore determine which aspects are actually significant and how they affect the result, while taking other factors into consideration [49].

Gaussian NB: Naive Bayes is a simple algorithm for learning and decision making. It makes the assumption that when determining which class object belongs to, individual features (such as color or size) don't really depend on one another. This presumption makes it function well in many contexts, but in real-world scenarios when features aren't totally independent, it occasionally causes problems [50].

Kneighbors Classifier: The K-NN algorithm is a simple machine learning method that falls under the category of supervised learning. It's comparable to receiving assistance from an instructor while learning under observation [51]. You give the machine many of labeled examples to work with, such pictures of dogs and cats, and tell it which is which. It can then identify whether a new picture it sees is of a dog or a cat based on its previous learning.

DecisionTree Classifier: Classification systems function identically to organizers, dividing large amounts of data into distinct categories. They are quite beneficial for data mining. These systems assist in categorizing newly received data, classifying it according to prior learning, and even making educated guesses about its classification. Here, we'll focus on the decision tree algorithm, which is a widely used

method for this. The process is visually represented using decision trees, which facilitates understanding. These are strong instruments that are employed in many fields, such as machine learning and image processing.

RandomForest Classifier: In machine learning and data science, the Random Forest classifier is a popular method used for sorting things into different groups. It's like having a bunch of teams working together. Each team uses its own decision-making process, called decision trees, to sort data into groups. Then, the final decision is made by combining all these different team's results. This could mean averaging their choices or going with the most common decision among them [52].

3.5 The performance differences among machine learning algorithms:

The differences in performance among various machine learning algorithms can be attributed to several factors, including the nature of the data, the characteristics of the algorithms themselves, and the specific context in which they are applied. Here are some key reasons that can help explain the accuracy results of different algorithms used in the proposed system:

Algorithm Complexity and Structure:

Decision Trees and Random Forests: These algorithms tend to be highly accurate because they can capture complex relationships within the data. Decision Trees build a model based on feature splits, effectively managing non-linear relationships. Random Forests, which consist of multiple decision trees, mitigate overfitting and enhance generalization by averaging the predictions from several trees.

K-Neighbors Classifier: The performance of this algorithm can vary greatly depending on the choice of 'k' (the number of neighbors) and the distance metric employed. A small 'k' can lead to overfitting, while a larger 'k' may obscure important patterns in the data. Its effectiveness is also influenced by the local structure of the dataset.

Data Quality and Preprocessing: The accuracy of machine learning models is significantly impacted by the quality of the input data. Problems such as missing values, noise, and irrelevant features can diminish model performance. For example, if the data is not adequately cleaned or pre-processed, algorithms like Gaussian Naive Bayes may struggle due to their assumptions about feature independence. The success of the algorithms can also hinge on how well the features are engineered. Algorithms like Logistic Regression may need careful feature selection and transformation to perform optimally, while tree-based methods can handle raw features more effectively.

Model Assumptions:

Gaussian Naive Bayes: This algorithm is based on the assumption that the features are normally distributed and independent when conditioned on the class label. If these assumptions are not met in the dataset, the model's effectiveness may decline, resulting in lower accuracy compared to other algorithms that do not rely on such strict assumptions.

Logistic Regression: This algorithm presumes a linear relationship between the input features and the log-odds of the outcome. If the actual relationship is non-linear, Logistic Regression might not perform as well as more adaptable models like Decision Trees or Random Forests.

Overfitting vs. Underfitting: Complex models such as Random Forests can achieve impressive accuracy on training data but risk overfitting if not carefully adjusted (for instance, by controlling the depth of the trees). This can result in diminished accuracy on new, unseen data. On the other hand, simpler models like Logistic Regression may underfit if the underlying relationships in the data are intricate, leading to lower accuracy.

The variations in accuracy among different machine learning algorithms can be linked to their structural properties, the quality and characteristics of the data, the assumptions they operate under, and how effectively they are tuned and applied to the specific problem. Grasping these elements is essential for choosing the right algorithm and enhancing model performance in real-world scenarios.

3.6 Factors contribute to variations in accuracy of ML algorithms:

Machine learning accuracy and recall varies across algorithms because of certain factors:
Algorithm Complexity: Sophisticated models such as Random Forests and Neural Networks ascribe ability to capture a variety of patterns which in return, tends to improve the accuracy and the recall. Such models may overfit and therefore not perform well on new data. On the other hand, weak models tend to have a lesser number of patterns but are relative in their tendency of overfitting.
Data Quality: Data quality which incorporates noise, outliers and missing values may lead to decrease in performance. Decision Trees, for example, tend to be adversely impacted by outliers, whereas other models such as tree-based models tend to perform better with missing values.
Feature Engineering: It is important to choose the right features to use as well as appropriate scaling of the features where applicable. Brooks suggests that irrelevant features usually deceive models and correct ones enhance the performance of algorithms such as the K-Nearest neighbors and Logistic regression.
Class Imbalance: In the case of class imbalance, models trained on such datasets will tend to focus on the majority class despite the minority class being vital, which leads to high accuracy with very little recall. In such cases resampling strategies or metrics that encourage recall can help.
Hyperparameter Tuning: Hyperparameter optimization is a very important step in modeling tasks as it includes selection of some important and effective parameters, like the depth of trees in Random Forests for improving the performance of the model. Advanced metric techniques like grid search improve performance metrics of the model by returning both accuracy and recall.
Metrics Used: Accuracy provides a measure of the level of the model's correct prediction while purpose of recall is to provide a measure of the true positive instances that have been detected by the model. Accuracy might be high together with precision but recall may be painfully low as true positives will be feebly found from the saturation of true negatives, mostly in datasets biased or gene enriched to one label. Data Split: The way data is organized into training data and testing data set is very crucial. Misalignment of dividing this data set can misrepresent precision and recall results. Cross-validation can provide a more accurate and realistic approximation of the estimation of prescription.
Model Interpretability: Decision Trees is one of such techniques which is used and enhances recall since they are easier to interpret however, more sophisticated algorithms for example Neural Networks are hard to explain.
Domain Knowledge: Every area of specialization has its unique features and that knowledge can aid in designing efficient algorithms, subsequently improving recall and precision.
so the disparities in accuracy and recall across different algorithms are accounted for by the consideration of a number of factors including the combination of algorithm characteristics, data quality and feature engineering, class balance, hyperparameter settings, and evaluation. Therefore, these parameters are necessary for determining the algorithm to use and consequently improving its performance in a particular task.

4. Results

It is clear from figure 8 that software development methodology recommendation system is divided into two phases. First phase recommends software development methodology and second phase recommends models under traditional development approach and agile development approach. Figure 9,10 & 11 shows results obtained for traditional development approach whereas Figure 13,14 & 15 shows results obtained for agile development approach.

4.1. Results of model trained under Traditional (Waterfall Methodology)

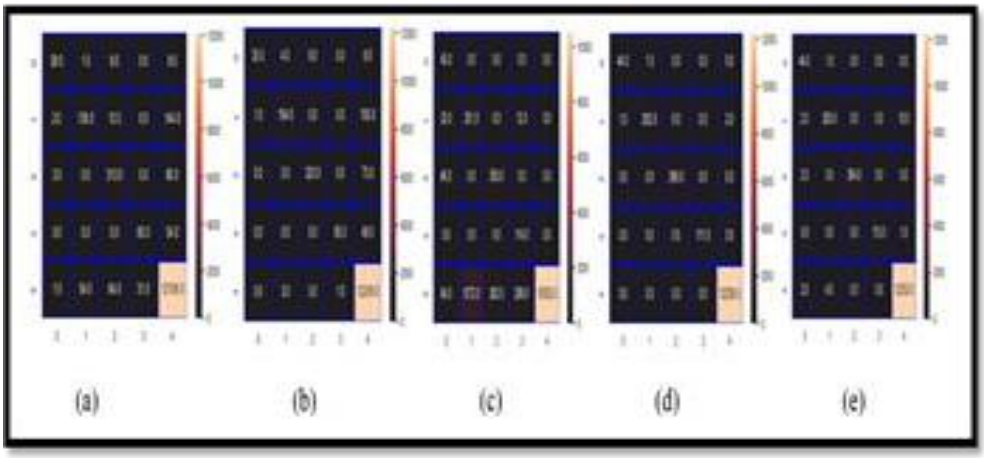

Figure 9 Confusion Matrix of models trained under traditional methodology (a)Logical regression (b) Kneighbors Classifier(c) Gaussian NB (d) DecisionTree Classifier (e) RandomForest Classifier

Figure 10 shows the comparative analysis of all five algorithms w.r.t. Recall, Precision, F1-score.

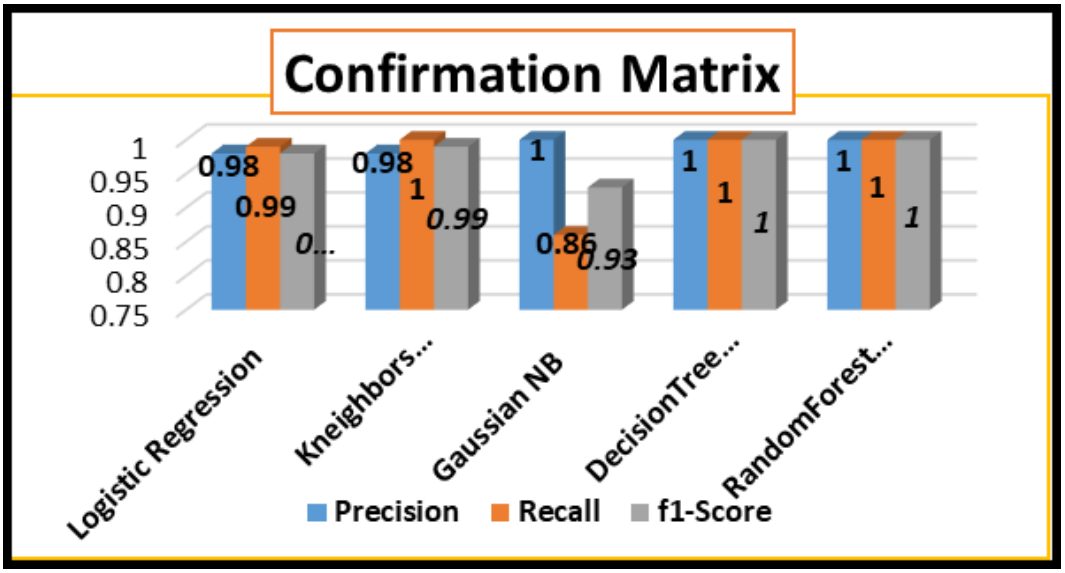

Figure 10 Confirmation Matrix of Machine learning algorithms applied for traditional methodology

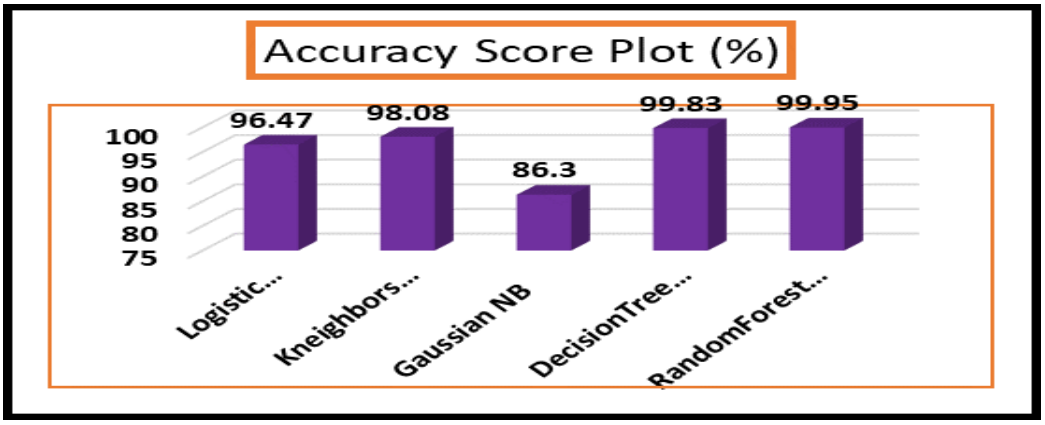

Figure 11 Accuracy score plot of machine learning algorithms applied to train traditional methodology recommendation model

Pre-processing and data cleaning using different dataset classifications have been shown to have a considerable impact on the accuracy of machine learning models, as illustrated in figures 10 and 11. Kneighbors classifier gives 98% accuracy and the Gaussian NB approach gives 86% accuracy, logistic regression yields 96% accuracy & the DecisionTree and Random Forest classifiers yields 100% accuracy.

4.2. Results of model trained under Agile Methodology

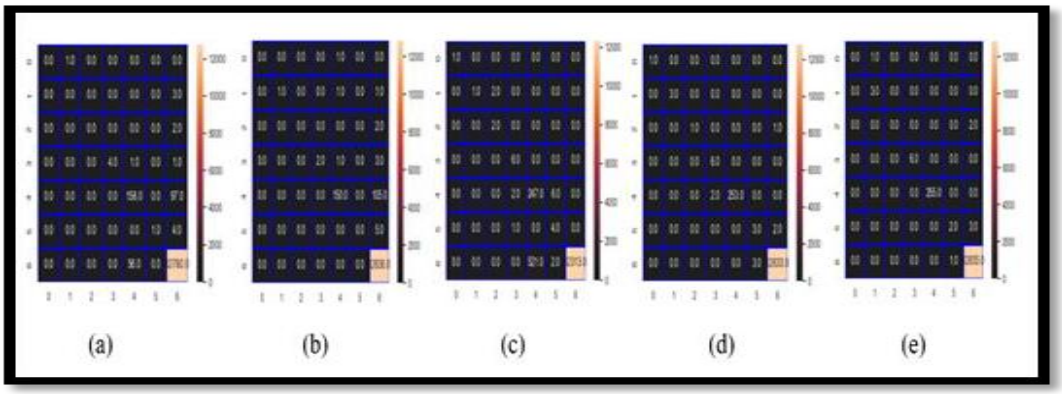

Figure 12 Confusion Matrix of models trained under agile methodology (a)Logical regression (b) Kneighbors Classifier(c) Gaussian NB (d) DecisionTree Classifier (e) RandomForest Classifier

Figure 13 shows the comparative analysis of all five algorithms w.r.t. Recall, Precision, F1-score.

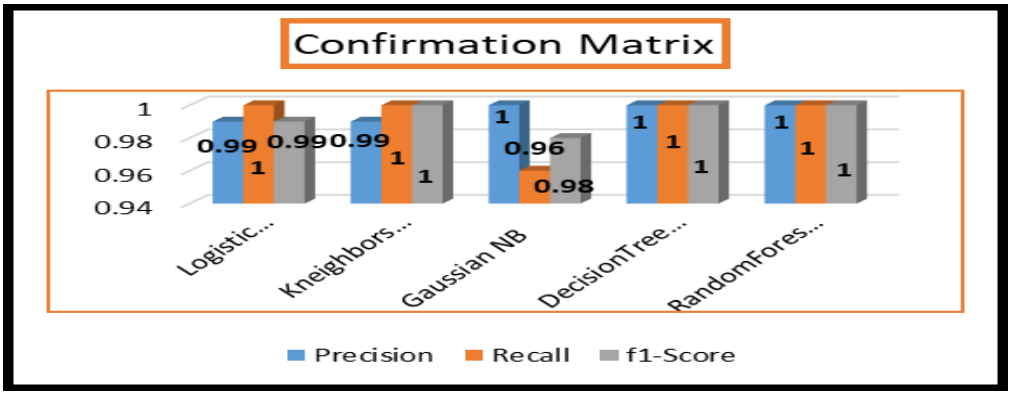

Figure 13 Confirmation Matrix of Machine learning algorithms applied for Agile methodology

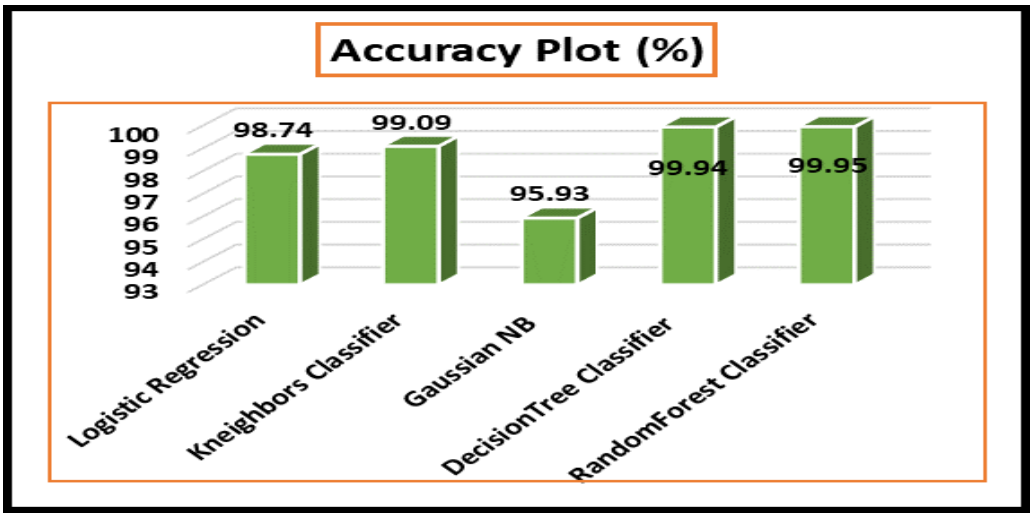

Figure 14 Accuracy score plot of machine learning algorithms applied to train Agile methodology recommendation model.

Pre-processing and data cleaning using different dataset classifications have been shown to have a considerable impact on the accuracy of machine learning models, as illustrated in figures 13 and 14. Kneighbors classifier & logistic regression yields 99% accuracy, the Gaussian NB approach gives 95% accuracy & the DecisionTree and Random Forest classifiers gives100% accuracy.

4.3 Performance of ML algorithms

Following is the insights into why certain models, such as Decision Trees or Random Forest, performed better compared to other ML algorithms used for the development of proposed recommendation model.Discrepancies in the performance of certain models, such as Decision Trees or Random Forests, as compared to simpler import models of Gaussian Naive Bayes (NB) seem to be connected with some underlying reasons:

Model Structure and Degree of Freedom: Decision trees are complex non-linear models characterized by their ability to discover intricate relationships in data. They do so by a series of decisions, where the data is partitioned on the basis of value of the feature variables enhancing the modeling of a more complex architecture and interaction of the features. Such flexibility usually results in a higher accuracy on various kinds of datasets. Random Forests is an ensemble technique that aggregates many decision trees to enhance accuracy and preservation of the structures. Predicting outcomes from a number of trees and then averaging the results, helps to avoid overfitting of any single tree, enhancing how the model performs on new unseen data.

Interaction of Features: Feature Interactions: Intrinsically, decision trees and Random Forests manage and even exploit interactions between features without being told to literally do so. Whereas, Gaussian NB, for instance, over deals with the assumption of independence of features relative to the class label, which is extreme and sometimes impractical. This assumption of independence can more often than not lead to poor outcomes in cases where there are feature dependence relationships.

Distribution Assumptions: Gaussian NB assumes that features follow a normal distribution. Nevertheless, if the actual distribution of the data is remarkably different from this assumption, then the model may not perform as well.

The reason Decision Trees and Random Forests are so powerful is because they do not make this assumption about the distribution of data, and therefore can work well with a greater variety of datasets.
The capacity to understand non-linear relationships: Decision Trees and Random Forests are good at modeling non-linear relationships between features and the target variable, because they have many decision nodes. GaussianNB is a Linear classifier and therefore will not fit very well to data sets that do not have a linear relationship between features which in turn will cause it to have lower accuracy and recall.
Performance on Imbalanced Datasets: Decision Trees and Random Forests are better at dealing with imbalanced datasets, but only if used in conjunction with techniques such as class weighting or resampling. Gaussian Naive Bayes could have some problems in those cases, because it is probabilistic in nature and tends to favor the majority class.

5. Discussion

5.1 Proposed System

Techniques and tools designed to give developers additional capabilities have multiplied over the past few decades in an attempt to improve development processes, reduce expenses, and increase productivity. But in addition to these cutting-edge tools, appropriate software development processes are also essential. For a software project or application to be successful, selecting the appropriate technique is essential. How the development and testing procedures go is determined by the approach that is chosen. A development technique is often used by IT organizations in accordance with the nature of the project or product being created. This choice isn't always easy to make, though. Frequently, the selected technique proves to be inappropriate, resulting in misunderstandings and inefficiencies throughout the process of development. Time, money, and resources are worthless as a result of this. In order to overcome this challenge, software developers must be given direction on which methodology to choose based on a variety of project parameters. As a result, a recommendation system that makes recommendations regarding the best methodology for a particular project or product is required. The suggested methodology includes developing a recommender model that will gather input from the teams of developers and customers/users and then use that information to determine which software development methodology is best for the project or product development process.

The recommendation system that we propose for the selection of the SDE is easy to use, can be easily implemented in the existing environment, and provides API compatibility option, tunable parameters, and training options. Limitations in deployment include people’s resistance to change, data quality problems, technical integration problems, resource shortages, scaling problems, and the need for continuous maintenance and users’ training. These are some of the limitations that if dealt with using changemanagement and adequate resource deployment will aid the implementation of the system and make the most of the available opportunities.

5.2 Existing System

Recommendation systems can work well with requirements elicitation, as demonstrated by Faiz Akram's research. These systems provide predictions about the needs of stakeholders based on their preferences for various requirements kinds through collaborative filtering. That's kind of like when you purchase online and the website makes product recommendations based on your past preferences. This system assists in resolving issues such as determining the needs of stakeholders during the requirements collecting phase and automating the process of selecting methods for gathering those needs [53].

Liang Wei developed four recommendation algorithms in 2021 that are freely accessible to all users. These systems assist users in selecting the appropriate software tools by taking into account factors such as previous tool usage and type of work. These systems can recommend which tools to use, for instance, if you're managing a project and need to keep track of tasks, communicate, manage code, and write papers. Liang Wei categorized these tools according to their capabilities and then used that information to provide recommendations through a rule-based framework [54].

Software developers are able to benefit from recommendation algorithms developed by Juri Di's work on CROSSMINER in 2021. At different stages of development, these algorithms provide recommendations on what code functions are required, how to use third-party tools, and which ones to use. They had to deal with issues such a lack of data, insufficient beginning points, defining what constitutes a good idea, and how to evaluate if the suggestions are effective in order to come up with these recommendations [55].

Michael B. and his group built a system to suggest project management techniques in 2022. This system determines if an algorithm can select the optimal course of action in a given circumstance. They also established protocols to increase the flexibility and agility of development. Imagine, then, having a tool that not only helps the development process become more flexible and agile overall, but also recommends the best approaches to manage a project [56].

5.3 comparison between the proposed system and existing system

All Features in one system: The new system brings together functions from different recommendation systems into one platform. Existing systems often focus on specific areas like tool selection or requirements gathering. In contrast, this model tackles both the choice of software development methods and the specific process models within those methods. This complete approach makes sure users get customized advice that takes into account many aspects of their projects [16].

User Data and Flexibility: The new system lets users or IT teams enter key details about their projects. It then uses this information to suggest the best software development method and suitable process model. This flexibility is key because it matches the advice to each project's unique factors and needs, which many current systems can't do.[16]

Smart Learning: The new system uses smart learning techniques to make better decisions. It looks at different project details and what users like to give more accurate and useful advice. This is better than old systems that might use fixed rules or past data without the ability to learn and change over time.[16]

overcoming Research Gap: Current methods often fail to offer a full recommendation system to help developers pick a specific method for software projects. The suggested system bridges this gap. It provides a two-step recommender model. This model guides users from first choosing an approach to picking a specific model. This improves the overall output and success of the software development process [4][16].

Focusing on Project Success: The suggested system stresses how important it is to pick the right method for project success. It knows that the choice of development technique affects the development and testing processes. It aims to cut down on mix-ups and waste that can happen from picking the wrong method [15][16].

Proposed system offers a full flexible, and machine learning-based approach to select best appropriate software development methods. It challenges the limits of current systems and enhancements the overall project management experience.

5.4 Limitations of existing system over proposed system

After researching existing systems, it's clear that there isn't one that can guide users on both choosing the right software development method and then picking the specific process model under that method. Some systems help with selecting techniques for gathering software requirements, while others aid in various stages of development like choosing tools, documenting, figuring out necessary code functions, and selecting third-party tools. Another system assists in picking the right software tools based on past usage and the type of work.

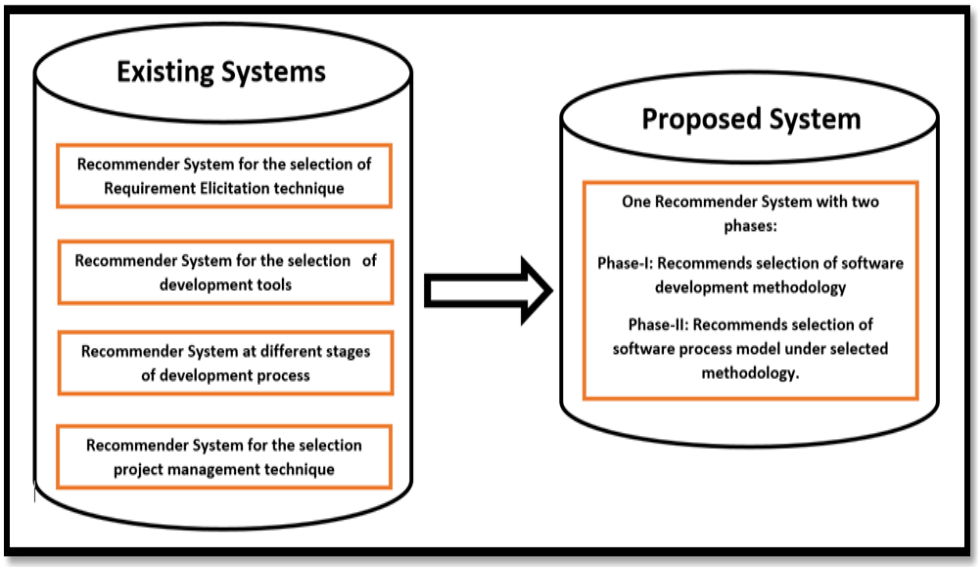

Figure 15: Comparison of Exiting System & Proposed System

Figure 15 shows the novelty of proposed system. It combines all the functionalities of existing recommendation system into one recommendation system. Users or IT teams just need to provide some key information, and the system will recommend the best software development method and the suitable process model for their project.

5.4.1 shortcomings of current systems and how the proposed model addresses these issues:

The Proposed System tackles several specific weaknesses of existing systems in the area of software development methodology selection. Here are the main weaknesses and how the proposed model overcomes them:

1. Limited scope of recommendation:

Weakness: Many current systems zero in on particular aspects of the software development process, like tool selection or requirements gathering. They fail to give a full picture that includes methodology selection. Proposed Model solution: The suggested model combines both methodology and process model suggestions into one system. This all-encompassing approach lets users get custom guidance that covers all key aspects of their projects helping them make better-informed choices [15].

2. Static Decision-Making:

Weakness: Current systems often depend on fixed rules or old data, which can make it hard for them to adjust to new project factors or changing needs. This can result in out-of-date or irrelevant suggestions.

Proposed Model solution: By using machine learning algorithms, the suggested model can look at up-to-the-minute data and user inputs to give flexible suggestions. This ability to adapt makes the suggestions more relevant and accurate so they fit current project needs better [16][15].

3. Lack of Proper Guidance:

Weakness: Current systems do not fully facilitate the user in their selection process, with most users choosing methodologies based on very little background or support information.

Proposed Model Solution: The proposed two-phase recommender model supports the user from the selection of an appropriate methodology for a given project based on initial project parameters through to finer methodology selection. This structured guidance of methodology selection enables the user to better deal with the intricacies of this process [16].

4. Absence of Adequate Consideration of Project-Specific Variables:

Weakness: All the parameters influencing methodology selection, like team dynamics, stakeholder preference, and project complexity, which are project-specific variables, are not appropriately considered in many of the available systems.

Propose Model Solution The proposed model takes into consideration a wide scale of project-specific factors including type, complexity, scope, available resources, and preferences of the stakeholders. With this consideration, recommendations are closely aligned with the actual needs and constraints of the project.

5.5 scenarios demonstrating the advantages of Proposed Model:

Following are the few scenarios that clearly demonstrate the advantages of integrating multiple functionalities into one recommendation system.

Scenario 1: Startup Developing an iPhone Application

Context: A client has approached a startup to design an iPhone mobile application with specific requirements and wanted it to be executed within the imposed short time frame.

Limitation in Current System: the team uses some of these tools separately to select methodologies and another tool to select appropriate development tools.

The advantage of the proposed model is that the recommendation system will allow inputs, such as project parameters, e.g., complexity of the mobile app, number of people in the team, and deadline. A well- rounded recommendation from such input would consider both the Agile methodology, as well as application-specific tools like JIRA as the project management tool and Git as the source control tool, among others, which support Agile practices. This reduces time in making decisions and will ensure all recommendations made are integrated and relevant to the project at hand.

Scenario 2: Large Organization with Heterogeneous Projects

Background: A large organization is undertaking software projects in different departments with different prerequisites and constraints.

Current System Limitation: Different methodologies with differences in requirements gathering among various systems of every department have caused inconsistencies and deficiencies in standardization across projects.

Proposed Model Advantage: The integrated system will provide a common platform where all departments can input their project specifics. Example: A department implementing a complicated enterprise application may receive recommendations of an appropriate methodology, such as Spiral, and the specific process models involving risk management strategies by the needs of the department. This standardization will thereby help improve collaboration and knowledge sharing within departments and enhance the overall efficiency of project management.

Scenario 3: Transitioning Team to Agile

Context: The traditional software development team is transitioning towards Agile methodologies to better project outcomes.

Current System Limitation: The team has separate learning resources to understand Agile methodologies and tool selection resources which supports Agile practices; it makes steep learning curve and probability of aligned tool usage.

Integrated Recommendation System Proposed Model Advantage: This system not only recommends Agile methodologies like Scrum but also trains the team about some best practices in addition to recommending a few highly suggested tools like Trello for task management, Slack for communication, all of them in one place. This integral support makes it easy for the team to adjust while ensuring they actually know what methodology to use and which tools to implement.

Scenario 4: A Project with Evolving Requirements

Context: A software project is a dynamic environment where changes in requirements continually emerge because of stakeholder feedbacks.

Current System Limitation: The team relies on a static recommendation system that does not adapt to changing project conditions, leading to potential mismatches between the chosen methodology and the evolving project needs.

The integrated system continued to analyze the data of a project and some feedback from a stakeholder, thereby adjusting real-time recommendations that were consistent with the analysis. For example, if the scope of the project was expanded, the system might recommend adjusting from a waterfall approach to more agile methodologies with certain tools that enhance iteration-based development. This allows the project to operate within the expectations of the stakeholders and respond appropriately to change.

Integrating multiple functions into a single recommendation system increases efficiency, consistency, and adaptability, ultimately improving project outcomes and collaboration among team members.

5.6 how developers or IT teams would benefit from the proposed recommendation system

Here are the use cases that show how developers or IT teams will benefit from the proposed recommendation system on the selection of software development methodologies:

Use Case 1: Development of mobile applications in a start-up.

Scenario: A start-up company plans to build a mobile application integrated with a new service. The workforce comprises a few junior developers who have no prior-experience in mobile app development.

How the system works:

Input Parameters: The group inputs input fields into the system such as type of the project (mobile application), number of the team (five developers), duration of the project (three months) and financial limits.

Recommendation: The system proposes Agile in the Scrum fashion considering the size and need for development iteration involving end user.

Outcome: The team embraces Scrum which enables them to work in time boxed sprints, receive feedback regularly and incorporate changes with ease. This results to the app being launched to the market within the set time frame and budget thus improving the morale of the team members as well as the project stakeholders.

Use Case 2: Revamping Enterprise Software Situation:

Scenario: A large organization with a great many departments needs a number of modifications to the current ERP system. This undertaking is cumbersome since it has an implication on many departments and requires involvement from all parties.

How the System Assists:

Input Parameters: The project manager enters several parameters: type of work (considered as complex), number of participants (high) and a reasonable time limit (12 months).

Recommendation: Regarding the system, it is recommended that one uses the Spiral model because it promotes a sequential mode of developing and managing risk.

Outcome: With the help of the risks controlling in Spiral model, it becomes possible to minimize risks and, at the same time, many departments can share their opinion and the team provide the product that

meets the expectations of different stakeholders. This in its turn leads to some reduction of complex operations, and the general result obtained will be more acceptable.

These use cases illustrate how the proposed recommendation system can provide tailored guidance to developers and IT teams based on specific project parameters. By leveraging the system, teams can make informed decisions about the most suitable methodologies, leading to improved project outcomes, enhanced collaboration, and more efficient use of resources. The ability to adapt recommendations based on real-time data and project needs ensures that the system remains relevant and valuable in diverse development scenarios.

5.7 potential limitations of the proposed recommendation system.

Possible Limitations of the Proposed Recommendation System

Dependence on Input Quality: The system depends on the nearby and correct project parameters. Writing recommendations can be affected by the quality of the input in a way that does not necessarily help the project succeed.

Complexity of Project Variables: There are many aspects in running a software project such as the team on board, the stakeholders, and many more conditions and preferences. It may not be able to identify all these complexities; this means that the recommendations generated in the project may not be as precise.

Limited Methodologies: The system may favor a set of SDM that receive most attention and adoption today, or which are widely used, and which might not necessarily be the best for a given project. This limitation may help minimize users' flexibility.

Static Recommendations: The results in the system might also be based on the first inputs that were given and might not refresh with continual change in project circumstances. This may have given out dated advice in case the whole project changes during its life cycle.

User Expertise: For the second limitation, the recommendations provided by the system may be quite abstract and for the new users of the system, such recommendations may be more difficult to interpret, which leads to the improper usage of methodologies. As is often the case with applications, the most benefit is probably going to be derived from it by those who have used it before on a regular basis.

Integration Challenges: Despite the fact that these tools were intended to be integrated with other software, minor problems like data incompatibility or different formats could hamper integration.

Resource Intensity: Such a system will need to be created and sustained, both of which take time and require staff and money, the latter of which may not be available in small design teams or less well- funded companies.

Over-Reliance: Some of the negative aspects that can be associated with teams might include the fact that over reliance of the team to the system might lead to a reduction in the overall critical thinking and decision-making processes that might be subscribed with the different teams. While using the system, the users need to apply their knowledge and at the same time, not blindly follow the recommendation of the system.

Ethical and Bias Considerations: A range of prejudicial programming biases could occur as the algorithms of the system tended to encode summaries consistent with the historical data fed into the system, or with the inputs provided by users, wherein the end solutions tend to favor certain methodologies than others.

Scalability Issues: Depending on the general number of organizations and specific project volumes the system has to grow. Depending on its design, the system might fail to give indication of performance problems that arise from the increased amount of data or more users reducing its reliability.

5.8 Challenges in the Proposed Recommendation System

challenges in implementing a recommended system are as below:

Data Availability: It is important for the system to use accurate and detail information about the projects. Lack of structured, or at best, fragmented historical data diminishes the accuracy of recommendations. Indicators such as project scope and team characteristics are important; however, if not

consistently gathered within the system, they may produce fewer valuable outputs. It is challenging to take a snapshot-in-time data and provide recommendations as they change as a project progresses over time.

Handling Increased Data Volume: As the numbers of projects increase, a similar increase must be reflected by the system while serving the expected quality.

Algorithm Complexity: While complex models may extend the recommendation range and precision, they are portrayed as weak points due to high computational demand, degradation of system performance. User Load Management: It should support many users and ideally do so effectively irrespective of the size of a company.

Generalizability to Different Projects: Diversity of Methodologies, the system should be able to contain different methodologies including Waterfall, Agile and provide recommendations in relation to different projects.

Varied Project Contexts: It is important for recommendations to consider industry requirements besides organizational cultures to identify if projects, meeting such needs should be considered.

Adaptability to Emerging Trends: The system has to be consistent with the contemporary paradigm shifts of software development.

User Expertise: The system should be intelligent in such a way that it will not be misconstrued by the advanced and the less experienced users.

Flexibility, data availability and generalizability concerns seem to represent the main areas that need to be enhanced for the system to function effectively. By addressing these challenges, the system can provide information for different software development projects, which should improve results in this field.

5.9 Assumptions Made During the Development Process

During the development of the proposed recommendation system for selecting software development methodologies, several assumptions were made that could impact the system's performance. Recognizing these assumptions is essential for understanding the potential limitations and areas for improvement in the system. Below are key assumptions that were considered:

1. Availability of Historical Data

Assumption: It is assumed that organizations have access to sufficient historical project data, including metrics related to project complexity, team composition, and past methodology effectiveness.

Impact: If organizations lack comprehensive historical data or if the data is of poor quality, the system's ability to generate accurate and relevant recommendations may be compromised.

2. Consistency in Project Parameters

Assumption: The system assumes that project parameters (e.g., scope, complexity, resource availability) remain relatively stable throughout the project lifecycle.

Impact: In reality, project parameters can change frequently due to evolving requirements or external factors. This variability may lead to recommendations that are no longer applicable, affecting the system's reliability.

3. User Familiarity with Methodologies

Assumption: It is assumed that users have a basic understanding of various software development methodologies and their characteristics.

Impact: If users lack familiarity with the methodologies, they may misinterpret the recommendations or fail to provide accurate input data, leading to suboptimal decision-making.

4. Uniformity of Team Dynamics

Assumption: The system assumes that team dynamics and collaboration styles are relatively uniform across different projects and organizations.

Impact: Variability in team dynamics, such as communication styles and collaboration practices, can influence the effectiveness of different methodologies. The system may not account for these differences, potentially leading to inappropriate recommendations.

5. Static Nature of Methodology Effectiveness

Assumption: It is assumed that the effectiveness of specific methodologies remains constant over time and across different projects.
Impact: The effectiveness of methodologies can vary based on context, industry, and project type. If the system does not adapt to these changes, it may provide outdated or irrelevant recommendations.
6. Simplified User Input
Assumption: The system assumes that users can provide simplified input regarding project parameters without needing extensive training or guidance.
Impact: If users find the input process complex or confusing, they may provide incomplete or inaccurate information, which can negatively affect the quality of the recommendations.
7. Homogeneity of Stakeholder Preferences
Assumption: The system assumes that stakeholder preferences and requirements are relatively homogeneous within a project.
Impact: In reality, stakeholders may have diverse and sometimes conflicting preferences, which can complicate the recommendation process. The system may struggle to reconcile these differences, leading to less effective recommendations.
The assumptions made during the development of the recommendation system can significantly impact its performance and effectiveness. By acknowledging these assumptions, developers and users can better understand the system's limitations and work towards mitigating potential issues. Continuous evaluation and refinement of the system, along with user feedback, will be essential for addressing these assumptions and enhancing the overall performance of the recommendation system.

6. Conclusion& Future Work

This Research article proposes recommendation system for choosing the best software development methodology based on the particular data parameters and type of project. It tackles the decisive problem of method selection by putting forth a two-phase recommender model, which provides direction from initial approach selection to particular model selection. By utilizing machine learning methods, the suggested approach improves decision-making by taking a variety of project aspects and preferences into account. Existing systems concentrate on particular areas such as tool selection or requirements elicitation; in contrast, the proposed model is unique in that it addresses methodology as well as recommendations for process models. By streamlining the development process, this integrated solution seeks to ensure efficacy, efficiency, and alignment with project goals. Such all-encompassing recommendation systems are becoming more and more necessary as software development progresses to effectively face the challenges of project management and produce positive results.
Each approach to software development has advantages and disadvantages. For projects with dynamic needs and early user involvement, rapid application development, or RAD, is a good fit. With prototyping, improvements may be made quickly and projects with changing demands can benefit greatly. Tasks with ambiguous objectives can be effectively divided into manageable segments using the Spiral approach. Relentless client collaboration and iterative software development are the hallmarks of Extreme Programming. To meet client requests, Scrum places a strong emphasis on collaboration and iterative sprints. Workplace collaboration, comprehension of project requirements, and the delivery of superior features are prioritized in feature-driven development. Selecting the best approach for a project depends on its requirements and associated risks.
Future work could involve expanding the analysis to include additional software development methodologies within both traditional and agile approaches, integrating them into a recommendation system for IT developers. The recommendation system could be designed to suggest multiple models, potentially including hybrid approaches, aimed at developing higher-quality software. This enhancement aims to provide developers with a more comprehensive and effective tool for selecting suitable methodologies for their projects.

6. References

[1] Saleh SM. Comparative Study on the Software Methodologies for Effective Software Development. International Journal of Scientific and Engineering Research.2017;8(4):1018-1025, May 2017. Available online:
https://www.researchgate.net/publication/316753858_Comparative_Study_on_the_Software_Methodologies_for_Effective_Software_Development.
[2] Gurung G.,Shah R.,Jaiswal D.Software Development Life Cycle Models-A Comparative Study. International Journal of Scientific Research in Computer Science, Engineering and information Technology.202;DOI: 10.32628/CSEIT206410.
[3] Samuel Gbli T. Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. Asian Journal of Research in Computer Science. 2024; 17 (5): 30-42, DOI: 10.9734/AJRCOS/2024/v17i5436.
[4] Hossain MI. Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management. International Journal for Multidisciplinary Research.2023;5(5). http://dx.doi.org/10.36948/ijfmr.2023.v05i05.6223.
[5] Alam I, Sarwar N, Noreen I. Statistical analysis of software development models by six-pointed star framework. PLoS ONE. 2022; 17(4): e0264420. https://doi.org/10.1371/journal. pone.0264420.
[6] Mishra A.,Alzoubi Y. Structured software development versus agile software development: a comparative analysisInt J Syst Assur Eng Manag (August 2023) 14(4):1504–1522. https://doi.org/10.1007/s13198-023-01958-5.
[7] Sankhe P, Dixit M. A Review and Survey of Software Development Methodologies. International Journal of Creative Research Thoughts.2023;11(5): 2320-2882. Available online: ijcrt.org/papers/IJCRT2305598.pdf.
[8] Hossain M.I. Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management. International Journal for Multidisciplinary Research (IJFMR).2023;5(5). DOI: 10.36948/ijfmr.2023.v05i05.6223
[9] Risener, K. (2022). A Study of Software Development Methodologies. Computer Science and Computer Engineering Undergraduate Honors Theses Retrieved from https://scholarworks.uark.edu/csceuht/103.
[10] Otieno M. Odera D. Ounza J. Theory and practice in secure software development lifecycle: A comprehensive Survey. World Journal of Advanced Research and Reviews, 2023, 18(03), 053–078. https://doi.org/10.30574/wjarr.2023.18.3.0944.
[11] Fagarasan C,Popa O, Pisla A, Cristea C. Agile, waterfall and iterative approach in information technology projects. IOP Conf. Ser.: Mater. Sci. Eng.2021; 1169 012025. doi:10.1088/1757-899X/1169/1/012025.
[12] Alam I, Sarwar N, Noreen I Statistical analysis of software development models by six-pointed star framework. PLoS ONE:2022; 17(4): e0264420. https://doi.org/10.1371/journal. pone.0264420.
[13] Dilmini I.G.U.,Rathnayaka, Kumara BTGS. A Review of Software Development Methodologies in Software Engineering. International Journal of Advance Research and Innovative Ideas in Education.2020;6(4). Available online:
https://ijariie.com/AdminUploadPdf/A_Review_of_Software_Development_Methodologies_in_Software_Engineering_ijariie12553.pdf.
[14] Narayan R. STUDY OF VARIOUS SOFTWARE DEVELOPMENT METHODOLOGIES. EPRA International Journal of Multidisciplinary Research (IJMR).2021;7(4). DOI: 10.36713/epra2013.
[15] Havstorm, T., Karlsson, F. Software developers reasoning behind adoption and use of software development methods – a systematic literature review International journal of information systems and project management,2023; 11(2): 47-78 https://doi.org/10.12821/ijispm110203.
[16] M. Weber, Economy and society. Berkeley, CA: University of California Press, 1978.

[17] Hafeez A.,Hassan S., Javeed S.,Quershi B.,Aziz A.,Furqan M.,and Hussain I. Software Engineering Process Models Strengths and Limitations with SimSE. Indian Journal of Science and Technology, 2019;12(31). DOI: 10.17485/ijst/2019/v12i31/146617.
[18] Music, Semir, "Selecting a Software Development Methodology Based on Project Characteristics" (2018). Graduate Research Papers. 3884. https://scholarworks.uni.edu/grp/3884.
[19] Gurianov D., Myshenkov K., Terekhov V. Software Development Methodologies: Analysis and Classification.5th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE).IEEE xplore. March 2023; https://doi.org/10.1109/REEPE57272.2023.10086852.
[20] Jiujiu Yu.Research Process on Software Development Model.ACMME IOP Publishing IOP Conf. Series: Materials Science and Engineering 394 ;2018: 032045 doi:10.1088/1757-899X/394/3/032045., 2018.
[21] Saranya P, Monica V, Priyadharshini J. Comparative Study of Software Development Methodologies. International Research Journal of Engineering and Technology (IRJET).2017;4(5). Available online: https://www.irjet.net/archives/V4/i5/IRJET-V4I529.pdf.
[22] Saravanos, A.; Curinga, M.X. Simulating the Software Development Lifecycle: The Waterfall Model. Appl. Syst. Innov. 2023; 6, 108. https://doi.org/10.3390/asi6060108.
[23] Adenowo A, Adenowo B. Software Engineering Methodologies: A Review of the Waterfall Model and ObjectOriented Approach. International Journal of Scientific & Engineering Research. 2020 July; 4(7).
[24] ide E. T. Akinsola, Afolakemi S. Ogunbanwo, Olatunji J. Okesola, Isaac J. Odun-Ayo, Florence D. Ayegbusi & Ayodele A. Adebiyi," Comparative Analysis of Software Development Life Cycle Models (SDLC)", Computer Science On-line Conference CSOC 2020: Intelligent Algorithms in Software Engineering August 2020 pp 310–322. DOI: 10.1007/978-3-030-51965-0_27.
[25] Herawati S., Negara Y., Febriansyah H., and Fatah D. Application of the Waterfall Method on a Web-Based Job Training Management Information System at Trunojoyo University Madura. E3S Web of Conferences 328, 04026 (2021). https://doi.org/10.1051/e3sconf/202132804026.
[26] Daraghmi Y.-A., Daraghmi E.-Y. RAPD: Rapid and Participatory Application Development of Usable Systems. IEEE Access. Sep.2022; 10. doi: 10.1109/ACCESS.2022.3203582.
[27] Sasmito GW, Wibowo D, Dairoh, Implementation of Rapid Application Development Method in the Development of Geographic Information Systems of Industrial Centers. Journal of Information & communication convergence engineering. Sep.2020; 18(3):194-200.
[28] Bjarnason E, Lang F, Mjöberg A. An empirically based model of software prototyping: a mapping study and a multi-case study. Empirical Software Engineering. 2023; 28:115. Doi:https://doi.org/10.1007/s.10664-023-10331-w.
[29] Susanto A., Meiryani.System Development Method with The Prototype Method", International Journal of Scientific & Technology Research.2019;8(7). Available online: http://www.ijstr.org/final-print/july2019/System-Development-Method-With-The-Prototype-Method.pdf
[30] Doshi D., Jain L.,Gala K.Review of The Spiral Model and Its Applications. International Journal of Engineering Applied Sciences and Technology.2021;5(12): 311-316. Available online: https://www.ijeast.com/papers/311-316,Tesma512,IJEAST.pdf.
[31] Doshi D, Jain L, Gala K. REVIEW OF THE SPIRAL MODEL AND ITS APPLICATIONS. International Journal of Engineering Applied Sciences and Technology. 2021; 5(12). 311-316.
[32] Komatsu T, Nagatani K, Hirata Y. Spiral model development of retroftted robot for tele-operation of conventional hydraulic excavator. ROBOMECH Journal springerseries.2023; 10:28. https://doi.org/10.1186/s40648-023-00267-7.

[33] Alsharari AS, Nazmee W, Zainon W, Letchmunan S. A Review of Agile Methods for Requirement Change Management in Web Engineering. International Conference on Smart Computing and Application (ICSCA). 2023 IEEE; DOI: 10.1109/ICSCA57840.2023.10087734.
[34] Moloto M., Harmse A, Zuva T, "Impact of Agile Methodology Use on Project Success in Organizations - A Systematic Literature Review," in Software Engineering Perspectives in Intelligent Systems. CoMeSySo 2020. Advances in Intelligent Systems and Computing, R. Silhavy, P. Silhavy, and Z. Prokopova Eds. Cham: Springer, 2020, pp. 267- 280.
[35] Edison H, Wang X,Conboy K. Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.2022 August; 48(8). http://dx.doi.org/10.1109/TSE.2021.3069039.
[36] Sofia F,Barata P, Determinants of E-Commerce, Artificial Intelligence, and Agile Methods in Small- and Medium-Sized Enterprises. IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT.2023; 0018-9391.https:// doi 10.1109/TEM.2023.3269601.
[37] Akhtar A, Bakhtawar B, Akhtar S, EXTREME PROGRAMMING VS SCRUM: A COMPARISON OF AGILE MODELS. International Journal of Technology, Innovation and Management (IJTIM). 2022; 2(2).
Online at: https://doi.org/10.54489/ijtim.v2i1.77
[38] Manisha, Khurana M, Kaur K.Impact of Agile Scrum Methodology on Team's Productivity & Client Satisfaction- A Case Study. IEEE, 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N).2021; DOI: 10.1109/ICAC3N53548.2021.9725505, INSPEC Accession Number: 21666322.
[39] Saeedi, K.; Visvizi, A. Software Development Methodologies, HEIs, and the Digital Economy. Educ. Sci. 2021, 11, 73. https://doi.org/10.3390/educsci 11020073.

[40] Lee, W.-T.; Chen, C.-H. Agile Software Development and Reuse Approach with Scrum and Software Product Line Engineering. Electronics. 2023;12, 3291. https:// doi.org/10.3390/electronics12153291.
[41] Riady S, Sofi K, Shadiq J, Arifin R. selection of Feature Driven Development (FDD) Model in Agile Method for Developing Information System of Mosque Management. Journal of Computer Networks, Architecture and High Performance Computing. July 2022; 4(2). https://doi.org/10.47709/cnahpc.v4i2.1469.
[42] Nawaz Z, Proposal of Enhanced FDD Process Model, I. J. Education and Management Engineering. 2021; 4, 43-50. DOI: 10.5815/ijeme.2021.04.05
[43] Nalendra A K. Rapid Application Development (RAD) model method for creating an agricultural irrigation system based on internet of things. The 5th Annual Applied Science and Engineering Conference (AASEC 2020) IOP Conf. Series: Materials Science and Engineering 1098. 2021; 022103.doi:10.1088/1757-899X/1098/2/022103.
[44] Saari M, Soini J, Grönman J, Rantanen P., Mäkinen T, Sillberg P. Modeling the Software Prototyping Process in a Research Context. Information Modelling and Knowledge Bases XXXII, doi:10.3233/FAIA200823.
[45] Shrivastava A, Jaggi I, Katoch N, Gupta D, Gupta S. A Systematic Review on Extreme Programming. Journal of Physics: Conference Series 1969. 2021; 012046. IOP Publishing doi:10.1088/1742-6596/1969/1/012046.
[46] Sankhe P, Dixit M, Bano T, Mathur S. Review of an Agile Software Development Methodology with SCRUM & Extreme Programming. IEEE International Conference on Current Development in Engineering and Technology.2022;978(1):5415-5. https://doi.org/10.1109/CCET56606.2022.10080640.
[47] Alexander Dada O, Sanusi I. The adoption of Software Engineering practices in a Scrum environment. African Journal of Science, Technology, Innovation and Development.2022;14(6):1429-1446, DOI:10.1080/20421338.2021.1955431.

[48] Hanslo R, Tanner M, Machine Learning models to predict Agile Methodology adoption. Proceedings of the Federated Conference on Computer Science and Information Systems.2020;697–704. https://doi: 10.15439/2020F214.
[49] Jain H, Khunteta A, Srivastava S.Churn prediction in telecommunication using logistic regression and logit boost. Procedia Computer Science.2020;167: 101-112. https://doi.org/10.1016/j.procs.2020.03.187.
[50] Gadekallu T, Khare N, Bhattacharya S, Reddy P, Maddikunta, In-Ho R, Alazab M. Early detection of diabetic retinopathy using PCA-firefly based deep learning model.Electronics.2020;9(2):274. https://doi.org/10.3390/electronics9020274.
[51] Suyal M, Goyal P. A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms based on Supervised Learning. International Journal of Engineering Trends and Technology.2022 July; 70(7):43-48.https://doi.org/10.14445/22315381/IJETT-V70I7P205.
[52] Sarker I, Kayes A,Watters P.Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage. Springer Journal of *Big Data*. 2019 July;57. https://doi.org/10.1186/s40537-019-0219-y.
[53] Akram F, Ahmad T, Sadiq M. Recommendation systems-based software requirements elicitation process—a systematic literature review. Journal of Engineering and AppliedScience.2024;71(29) https://doi.org/10.1186/s44147-024-00363-4.
[54] Wei L, Capretz L. Recommender Systems for Software Project Managers. Proceedings of the Evaluation and Assessment in Software Engineering. 2021; 412-417, https://doi.org/10.1145/3463274.3463951.
[55] Rocco JD, Ruscio DD, Sipio CD, Nguyen P, Rubei R, Development of recommendation systems for software engineering: the CROSSMINER experience. Empirical SoftwareEngineering.2021;69(26). https://doi.org/10.1007/s10664-021-09963-7.
[56] Michael B. Recommendation of Project Management Practices: A Contribution to Hybrid Models. IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT.2022December;69(6). https://doi.org/10.1109/TEM.2021.3101179.