"Comparative Analysis of Agile Methodologies in Software Development"

Sahana A R[1], Siri Nadig K Y[2], Disha S Kanavi[3], Abhishek M Raikar[4] , Poojitha C H[5]

[1]Bachelor of Engineering, Information Science and Engineering, Bapuji Institute of Engineering and technology, Karnataka, India
[2]Bachelor of Engineering, Information Science and Engineering, Bapuji Institute of Engineering and technology, Karnataka, India
[3]Bachelor of Engineering, Information Science and Engineering, Bapuji Institute of Engineering and technology, Karnataka, India
[4]Bachelor of Engineering, Information Science and Engineering, Bapuji Institute of Engineering and technology, Karnataka, India
[5]Bachelor of Engineering, Information Science and Engineering, Bapuji Institute of Engineering and technology, Karnataka, India

---

Abstract*:* Agile process comparison is the focus of this work. Other models of the software development process will be guided by this work. Software project management, software schedule management, and other related fields can benefit greatly from the use of agile procedures. Agile processes specifically aim to reduce the rate of faults, speed up development, and satisfy the customer. The agile procedures are contrasted with various software development life cycle models in this article. Although this paper discusses the benefits and cons of agile processes, it should be noted that these procedures are not always beneficial.

Keywords: Software Development Life Cycle (SDLC), Agile Development.

1.INTRODUCTION

There are two primary factors to take into account in the software development life cycle: the process itself and the software's quality. Agile software development methods are incremental and iterative, with specifications that can be modified to meet the demands of the client. It supports time boxing, iterative development, and adaptive planning. It is a theoretical framework that encourages anticipated interactions at every stage of development. Each of the many SDLC models—spiral, waterfall, and RAD—has advantages of its own. The actions carried out at every stage of a software development life cycle are described by the SDLC framework [1].

Planning, analysis, design, coding, testing, and maintenance are all aspects of software development that must be carried out in accordance with client requests. The selection of a particular model is contingent upon the different uses. However, we will examine agile processes and their approaches in this paper. The agile process is a software development process in and of itself [2]. Because the customer is directly involved in assessing the program, customer satisfaction is given top importance in the iterative agile process [3].

The software development life cycle, which comprises requirements collecting, analysis, design, coding, and testing, is followed by the agile method, which produces partially implemented software while awaiting client feedback. Customer satisfaction is the top emphasis throughout the entire process, and development time is accelerated. The Agile Process software development life cycle is shown in Figure 1 below.

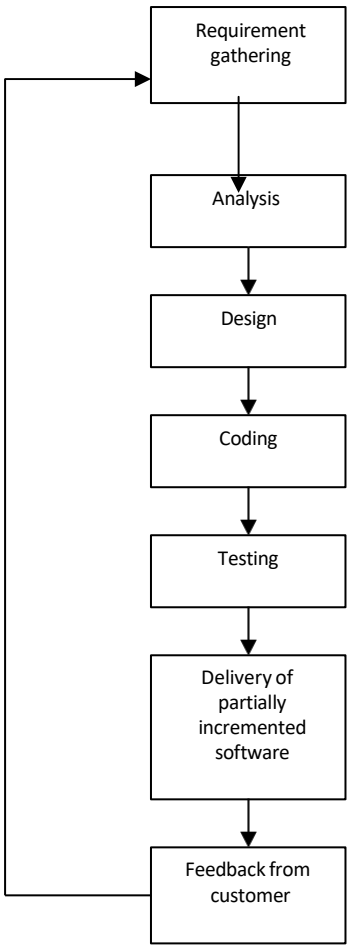

Figure 1. Phases of Agile Process.

The structure of this document is as follows: Section II lists the attributes of Agile processes. The agile approaches are covered in Section III. The benefits of the Agile Process are covered in Section IV, its drawbacks in Section V, and a comparison of Agile with other SDLC models in Section VI. Section VII concludes with a discussion of the findings.

II. CHARACTERISTICS OF AGILE PROJECTS

Agile processes break up jobs into manageable chunks and require less planning. Short-term projects including teamwork that adheres to the software development life cycle are intended for the agile process. The following stages are included in the software development life cycle: 1. Collecting requirements; 2. Analysis; 3. Design; 4. Coding; 5. Testing; and 6. Maintenance. Software risks are decreased when customers and software team management are involved. Iterative in nature, this agile method allows for adjustments based on consumer satisfaction. Multiple iterations make it simple to introduce new features in an agile approach.

1. Iterative
Agile software techniques concentrate on a single need with numerous iterations because their primary goal is customer satisfaction.

2. Modularity
The entire system is broken down into manageable components known as modules using an agile process. A key component of software development methods is modularity.

3. Time-boxing
Because the agile approach is iterative, each module must have a time restriction with a corresponding cycle.

4. Parsimony
Parsimony is necessary in agile processes to reduce risks and accomplish objectives with the fewest possible modules.

5. Incremental
Because the agile process is iterative, the system must be developed in steps, each of which must be independent of the others, and finally, all of the steps must be combined to form the entire system.

6. Flexible
Because the agile method is iterative, new hazards could arise. Agile procedures' adaptable nature enables them to be modified to address emerging risks and accommodate real-time demand changes.

7. Convergent
In an agile process that uses an incremental and iterative approach, all of the risks related to each increment converge.

8. Cooperation
Because the agile method is modular, the software development team must communicate well with one another.

At the conclusion of the software development process, various modules must be integrated. 9.People-focused

Customer satisfaction takes precedence over procedure and technology in agile processes. Software performance and productivity are enhanced by a competent development team.

III. METHODOLOGIES

Agile projects can be implemented using a variety of approaches. The three approaches that are most frequently employed in industry have been covered here. Various facets of the software development life cycle are the focus of agile approaches. While some concentrate on software project management (the scrum approach), others emphasize the behaviors (extreme programming, pair programming).

A. XP, or Extreme Programming

XP's emphasis on customer satisfaction makes it the most effective approach to creating agile software. To develop the software, XP needs to engage with customers as often as possible. The full software development life cycle is broken down into a number of shorter development cycles. At any stage of the development life cycle, it accepts and integrates customer modifications or requirements.

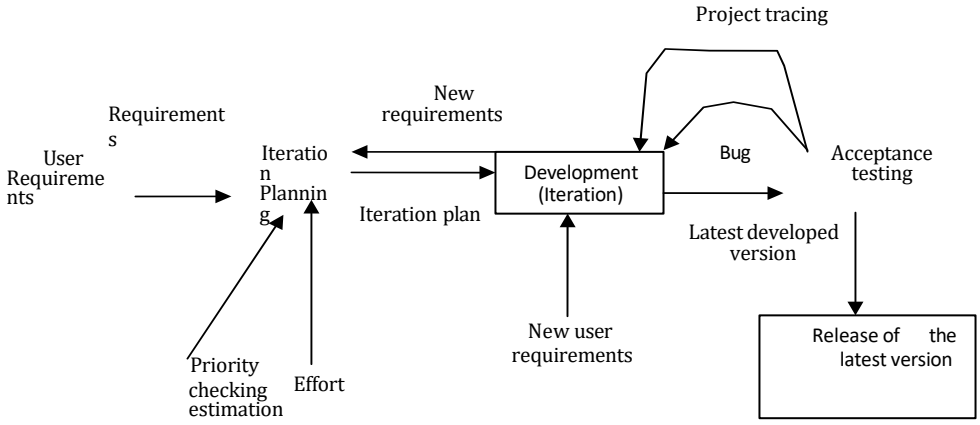

Figure 2: Method of Developing Agile Processes using Extreme Programming

The entire process of creating an agile process using the XP approach is depicted in the diagram above. Gathering user needs is the first step in extreme programming. The entire development process is broken down into a number of smaller cycles based on these requirements. Iteration planning, which includes determining the number of cycles, ranking the requirements, and calculating the amount of work needed to execute each cycle, is the next stage. Pair programming is

now used to develop each literation. The iteration plan should be modified in accordance with any new user requirements that may arise during the development phase. The most recent version will then be tested for bugs; if any are found, they will be fixed in the following iteration.

Following each acceptance test, a project tracing should be conducted to get feedback on the amount of work completed thus far.

Pair programming, thorough code review, code rewriting, and an open workspace are just a few of the additional features that Windows XP has brought to developers [4].

B. Scrum

Another well-liked agile development technique that boosts efficiency is scrum. The incremental software development process is essentially its foundation. The scrum technique breaks down the whole development cycle into a sequence of iterations, each of which is referred to as a sprint. A sprint can last no more than 30 days.

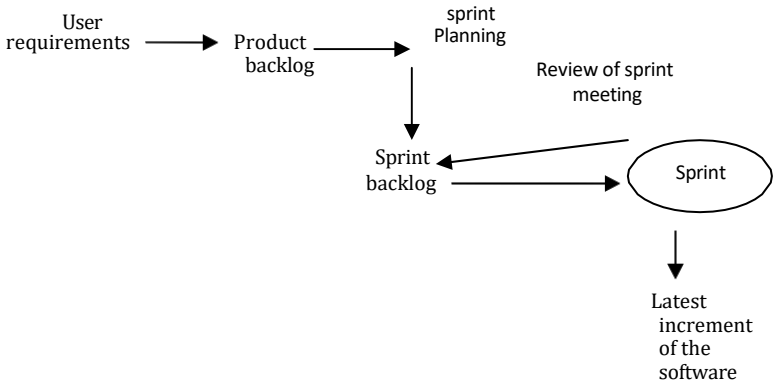

Figure 3: Method of Developing Agile Processes using Scrum

The conclusion of each sprint to ascertain whether or not all of the requirements for that specific sprint have been implemented, and to determine which requirements should be implemented in the following sprint. We receive a functional software increment following each sprint.

C. FDD, or feature-driven development

Among the agile development techniques is FDD. Designing the software's domain before development is the main benefit of this approach.

TABLE I. PRCOESS MODELS

Features	Different Process Models		
	Agile Process	***Spiral Model***	***RAD Model***
Definition	Agile process is the ability to both create and respond to changing requirements of software.	Spiral model is the software development model which focuses on managing risks.	RAD model is "high speed adaptation of linear sequential model, in which component based construction is used.
Adaptability	?	?	X
Testing Phase	Unit, Integration , System testing	Unit, Integration and System testing	Unit
Quality Factors	?	?	X
Risk Analysis	X	?	X
Off-the- Tools	X	X	?
Failure normally due to	Code	Code	Architecture and design
Knowledge Required	Product and domain	Product and domain	Domain
Entry & exit Criteria	X	X	?
Mock up	?	?	X
Extendability	?	?	X
Project management involvement	?	X	?
Higher Reliability	?	?	X
Time Boxing	?	X	?
Features	**Different Process Models**		
	Agile Process	***Spiral Model***	***RAD Model***
Status of Development Team	Less experience required	Less experience required	More experience required
Use of reusable components	X	X	?
Flexibility	?	?	X
Customer Involvement	?	?	X

CONCLUSION

The software development life cycle models, the spiral model, the agile process's features, and its benefits and drawbacks have all been covered in this essay. Agile projects are significantly superior to other software development processes in terms of productivity, performance, quicker time cycles, and risk analysis, according to a comparison of agile software development with other software development methods. Important applications like web-based testing tools, etc., use agile procedures.

REFERENCES

[1] Chan, Frank KY, and James YL Thong. "Acceptance of agile methodologies: A critical review and conceptual framework." Decision support systems 46.4 (2009): 803-814.

[2] Srivastava, Apoorva, Sukriti Bhardwaj, and Shipra Saraswat. "SCRUM model for agile methodology." 2017 International Conference on Computing, Communication and Automation (ICCCA). IEEE, 2017

[3] Salza, Pasquale, Paolo Musmarra, and Filomena Ferrucci. "Agile methodologies in education: A review." Agile and lean concepts for teaching and learning: Bringing methodologies from industry to the classroom (2019): 25-45

[4] Kumar, Gaurav, and Pradeep Kumar Bhatia. "Impact of agile methodology on software development process." International Journal of Computer Technology and Electronics Engineering (IJCTEE) 2.4 (2012): 46-50.

[5] Sunner, Daminderjit. "Agile: Adapting to need of the hour: Understanding Agile methodology and Agile techniques." 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT). IEEE, 2016.

[6] Stavru, Stavros. "A critical examination of recent industrial surveys on agile method usage." Journal of Systems and Software 94 (2014): 87-97

[7] Livermore, Jeffrey A. "Factors that impact implementing an agile software development methodology." Proceedings 2007 IEEE SoutheastCon. IEEE, 2007.

[8] Trivedi, Devharsh. "Agile methodologies." International Journal of Computer Science & Communication 12.2 (2021): 91-100.

[9] Shankarmani, Radha, et al. "Agile methodology adoption: Benefits and constraints." International Journal of Computer Applications 58.15 (2012).

[10] Conboy, Kieran, and Brian Fitzgerald. "Method and developer characteristics for effective agile method tailoring: A study of XP expert opinion." ACM Transactions on Software Engineering and Methodology (TOSEM) 20.1 (2010): 1-30.

[11] Khalil, Md Asif, and Bonthu Kotaiah. "Implementation of agile methodology based on SCRUM tool." 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). IEEE, 2017.