International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 03 | Mar 2021 www.irjet.net p-ISSN: 2395-0072

Agile against conventional approaches of SDLC

F.Melbin Chandrika
Educator
RKV Senior Secondary School, India

Abstract: *The software life cycle is the most important element in the development of software. It indicates the production phases of software. The benefits and drawbacks of conventional SDLC approaches and agile methods are discussed in this article. It also recommends amendments to current and agile growth that make the SDLC lighter in project management more realistic.*

Keywords: SDLC, Agile, ROI

1. Introduction

The development and management of information systems [7] is SDLC. Application output the SDLC life cycle. Data model that allows for collecting cross-discipline data in an ALM system, and provide an example of how to use that collected data to build upon the data collection activities of an organization [6]. Usually it involves a range of phases from an early development review to product evaluation and testing after development. Included also is the model and approach that software building teams use to coordinate and manage the whole process of development.

A software program is designed to perform a certain set of tasks. These tasks require complicated calculation and care. The PC works as well. These tasks were well defined. In order to make end products very strong, durable and enduring, there must be a demanding and repeated task through the entire supply chain. In order to achieve these features of an effective system, a standard development process, capable of stressing the nature and complexity of the whole development process is important. Several device developers are currently using two SDLC methodologies, traditional and agile.

2. Conventional Methods

The software methodologies traditional such as waterfalls, V-models and RUP have been divided into heavyweight methodologies [1]. This approach is specifically aimed at defining requirements, designing solutions, evaluating and enforcing a variety of interventions. At the beginning of the project a stable set of specifications for traditional software development methods should be developed and documented.

The conventional way of designing software is characterized by four stages. The first step is to decide the project specifications for different building phases and how long it takes to prevent potential problems. The initial step is, if the requirements had been developed, would be to develop and design an architectural infrastructure as schedules or models. This raises future problems and provides developers with a workable strategy. The project will continue to be confronted as it progresses.

The project moves to the development phase, where code is developed before the fundamental goals at the end of the architectural and design plans are achieved. The manufacturing industry is also divided into a variety of teams of smaller, skill-based operations. Sometimes, the test process overlaps with the design phase to make sure the concerns are addressed early. Following the completion of the project and the developers' approach to project requirements, the customer is involved in the test and feedback process and the customer is satisfied with the project.

Predetermined and on-going software processes [1] are the cornerstone for software creation of conventional software systems. The progress of this project depends on knowing all the specifications before construction starts, which makes it incredibly difficult to adjust during the entire development cycle. However, it is much simpler to measure project costs, schedule and assign resources [2]. It is also possible to do this.

3. Agile methodology

The idea of gradual and iterative growth is based on agile development in which the developmental cycle stages are continually revisited. It uses input from customers to develop tools for designing solutions[5].

The growth cycle is divided into a smaller segment called "increments" or "iterations" that affect traditional stages of development rather than the use of the single large process model in the conventional SDLC. The following four are the key features of agile considerations according to the Agile Manifesto:

- Early participation of consumers
- The development of the Internet
- Teams autonomous
- Alteration of transition

Six methodologies, including crystal processes, software dynamics, functional creation, clean development software, scrubber processes and extreme programming, have been identified[8].

4. Discussion

In the previous approach, the capacity to efficiently and economically deliver outcomes in dynamic projects with unclear requirements represents a significant disparity between agile and traditional development approaches. Agile methods concentrate on personnel, software, customer cooperation and change response; traditional methods stress the value of the contracts, the plans.

The business Return of Investment [ROI] [9] focuses on agile growth strategies. The development teams typically contact the participants in the conventional development cycle and collect information in the early stages of the process. During the design process, the design teams begin the actual coding phase. Only after the whole coding process is completed begins the testing [4]. Only after the assessment process has not posed any problems will the final product be presented to stakeholders. The drawback to this traditional approach is the "one shot" nature of the structure of the production team. Assume that during the evaluation process, a failure happens, the main challenge being to fix the whole module. The typical SDLCs have another problem with the fact that stakeholders do not know what to do with the method in most situations, so that the application of a requirement model does not suffice in earlier stages. Customer or stakeholder requests may enter the market following submission and publication of the final items. Requests for modifications from different parties to bring the system in place will lead to numerous compatibility and software integrity issues. In a bigger framework these issues are all the more evident. The conventional SDLC therefore is not an enterprise-as-usual solution.

Agile SDLC eliminates the need to "keep in mind" in contrast with the conventional SDLCs. Stakeholders also lack sufficient experience at the outset of the project to identify all implementation requirements [11]. It's not true. The fact that users cannot select the features to be implemented into the system is a normal phenomenon. The system releases ample information to offer customers details to strengthen the specifications before the new update. This software is also proven in organized, agile practice and written. In agile practices the iterative approach allows clients to avoid making decisions, and so options could be delaying until any future iteration, when the alternative can be enhanced in terms of information and technology. Well before all demands are known, the benefits of agile SDLCs overriding conventional SDLCs are to build scalable SDLC.

For agile developments to provide the final product in order to meet the needs of consumers, iterative consumer preferences acquisition is a history. The completed modules are modified by the customers for any short iteration. These modules are not fully integrated, nor will any renovation or features lift production costs slightly. This allows developers to use all the features customers want and the integration of the system is only possible when customers no longer have any requirements. The entire system would apparently please customers greatly with the requested functions.

Stakeholders are allowed by agile framework approaches to optimize their return on investment. Development should also be tailored to the minimal cost of transition so that stakeholders are able to look at the economic factors at the beginning of each iteration in order to implement a higher functionality according to ROI [9]. It is also the duty of the development team to provide strategic input on the risk of evolving actors. In structures or processes that are focused on particular stakeholder needs, versatility of unique characteristics or artifacts is called versatile and lean. It is known as a modular function.

While agile methods transcend conventional methods in many respects, they are difficult to implement. That's hard. One way is for agile documentation to be substantially reduced and the code to be used itself as a document [12]. This allows developers to collaborate with agile methods to provide further feedback, such as clarification and code analysis. Newer developers or new team members cannot, however, carry out tasks because they cannot grasp the project correctly. Therefore, expert developers are asked multiple questions that could cause iterations to be delayed, thus increasing the cost of development. Conventional approaches, on the other hand, stress the importance of documentary information for providing the development team with feedback and clarification on projects, as developers are not interested in ignoring project information or the availability of a knowledgeable developer.

Agile methodology is well known for its emphasis on customer co-operation[8]. The development team and its customers share their job and provide feedback on the features they provide to upgrade current features and introduce new features in the framework for each deliverable iteration. The development team is meeting with customers. Daily meetings will often be weekly and repetitive, as their respective modules can be used on various occasions and modules, when variations in demand for each iteration change, for other members and clients. In addition, for each iteration, the time permitted is typically limited, usually within weeks. The schedule for each module is often tight for developers, particularly if complicated processing algorithms are needed. This results in each plan iteration such that efficient coordination cannot be formed between team members and customers. Traditional methodologies, in the coding phase leading up to implementation and coding procedures, include a simple model of specifications to provide the development team guidance in the development phase. Customers are not allowed to take part in this growth cycle because only the integrated system will be announced to customers as a final product when the entire system is finalized and introduced. In such situations, developers do not need to worry about day-to-day iteration meetings and can complete the system for a longer period to enhance their performance.

As mentioned above, agile development focuses on communication so that clients can produce the completed modules effectively with each iteration and explain the current progress and problems. For the entire development team, interpersonal and social competences are essential. The development team must also be fully aware of customers' expectations and changes and the need for good communication skills. But it has no good social skills for any developer. If the team developer has poor social skills, it is difficult for associated participants to know about developing the module, resulting in a lack of user specificity in future processes. Although developers can't really say what customers want, unnecessary functions are likely to occur in the completed module. This raises construction costs by renovating the module and increases uncertainty by depending on the social skills of developers during the development process.

The capacity to adapt demand to agile architecture results in two problems that are dependent on nature - stiffness and movement. Rigidity means a change in the device, while the motion refers to the failure of the system to accept reusable parts which implies too much effort and risk. In the event of such issues, a high degree of restructuring is needed in order to eliminate needless addicts[13]. The consequence of this issue is the infringement of the concept of interface separation[14],

which explains most of the problems during the deployment phase. Table 1 demonstrates the variations between AGILE and conventional methods.

Table. 1: Comparison of Agile and Traditional Approaches

	AGILE	TRADITIONAL
User requirement	Iterative acquisition	Detailed user requirements are well-defined before coding/implementation
Rework cost	low	high
Development direction	Readily changeable	Fixed
Testing	On every iteration	After coding phase completed
Customer involvement	high	low
Extra quality required for developers	Interpersonal skills & basic business knowledge	Nothing in particular
Suitable Project scale	low to medium-scaled	Large-scaled

5. Enhancement

Agile design approaches for customer satisfaction were developed with the goal of minimizing development cycles, reducing bug rates, and adjusting to evolving market demands during the process of development[11]. It is a very useful method for replacing the conventional heavy-weight life cycle in modern software development procedures. However, it has not yet been mastered and faces some difficulties in its implementation. In previous discussions, flexible methods simplifies documentation communication and reduce awareness of the framework in this aspect of agile development. There is no guarantee that if the program was initially created using several methods, the code would be a documentation. In order to increase its viability, the device must also be incorporated. This can be carried out by automated codification checks, in which the software tools monitor source codes to ensure that the laws or best practices are complied with. This helps identify code bugs, saving time and costs prior to the test. It offers complete code coverage as well as identification of defects and the origin of the flaw. In addition, reorganizing cross-sectional issues would enhance comprehensibility and help to define code for different markets and separate business and platform code. The developers should always evaluate dependencies of the module and conduct unusual reactivators to increase both interfaces in order to increase the flexibilité and performance.

The social skills of developers are another important factor in supporting agile growth in practice. Strong social and interpersonal skills can motivate developers in the agile development of projects. Either training the organization or developers should be mindful of the significance of the standard themselves and should always encourage regular engagement and socialization with the people around them to polish their interpersonal skills. You can understand the strategy and understand what others want to say by getting closer to other people.

In order to connect with customers equally, developers must have special entrepreneurial skills. A team of business experiences will prevent the products from providing the user with the consequence that consumers lose trust in manufacturing equipment. Developers must also not only excel in computer science, but must also have an understanding of fundamental commercial law. This can be accomplished by providing a quick organizational training or by inviting field experts to obtain a high degree of understanding to enhance team members' skills.

For developers to practice agile development, certain points should also be noted. Incremental changes are first required to make the project system plan and the artifacts more usable. They will also collect input to make sure those participants and stakeholders fulfill the project needs. Moreover, processes and objects should be discarded which do not contribute a sustainable value to an operational computing environment that gives added value to systems-supported processes [3], [10].

6. Conclusion

The software development life cycle describes the total phase of development used by a company to effectively manufacture software. Modern SDLCs are divided into conventional SDLCs and agile SDLCs into two main categories.

Agile SDLC exalts conventional SDLC, as previously stated. Agile SDLC has its own inconvenience, however. Even if agile SDLCs are better suited for large-scale small-scale projects, conventional SDLCs are more suitable. The development team therefore has an SDLC that suits the project best.

The development team will use certain parameters to determine the optimum SDLC to include team size, geographical location, software size and complexities, project type, business strategy, technology skills and where appropriate. Before making a decision, the team must also analyze each SDLC's distinction, benefits and disadvantages. In order to determine SDLC applicants against selection criteria, the team should also evaluate market environment, industry expectations and business objectives.In order to ensure that the company maximizes the ability to provide software efficiently and to ensure long-term management decision selection and implementation of the acceptable SDLC, the SDLC approach is essential for choosing and taking the software.

References

[1]Nikiforova, O., Nikulsins, V., Sukovskis, U.: Integration of MDA Framework into the Model of Traditional Software Development. In: Frontiers in Artificial Intelligence and Applications, Databases and Information Systems V, vol. 187, pp. 229–239. IOS Press, Amsterdam (2009)

[2]IBM: Rational Unified Process: Best practices for software development teams (2003), http://www.ibm.com/developerworks/rational/library/253.html

[3]Chiranjeevi Aradhya, "Safety Critical Systems and Agile Methodology For Avionics Software Development", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.8, Issue 12, pp.2461-2468, December 2020

[4]Chiranjeevi Aradhya. "A detailed survey and evaluation of various aspects of embedded software robustness testing." International Journal of Advance Research, Ideas and Innovations in Technology 7.1 (2021).

[5]Szalvay, Victor. An Introduction to Agile Software Development. Danube Technologies Inc. 2004.

[6]Chiranjeevi Aradhya. "Utilizing lifecycle management system approach, boost airworthiness certification of software-centric avionics systems" International Journal Of Advance Research And Innovative Ideas In Education Volume 7 Issue 1 2021 Page 1583-1589

[7] Systems Development Lifecycle: Objectives and Requirements. Bender RPT Inc. 2003.

[8]Peterson, Kai. A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case. Journal of System and Software. 2009.

[10] Abrahamsson, Pekka. Agile Software Development Methods: Review and Analysis. Julkaisua-Utgivare-Publisher.2002.

[9] Rico, David F. What is the ROI of agile vs. traditional Methods. 2008.

[10] Carayannis, E.G. Agile Project Management for IT Project. Greenwood Press / Quorum Books. 2002.

[11] Cho, Juyun. Issues and Challenges of agile software development with SCRUM. Issues in Information System.VOL IX, No. 2. 2008.

[12]Vijayasarathy, Leo R. Agile Software Development: A survey of early adopters. Journal of Information TechnologyManagement Volume XIX, Number 2. 2008.

[13] Henssen, Geir K. Maintenance and Agile Development: Challenges, Opportunities and Future Directions. 2009.

[14] Martin, R.C., Agile Software Development, Principles, Patterns and Practice. Prentice Hall. 2002.