Agile vs waterfall: A Comparative Analysis

Vaishnavi Kannan
Student
Delhi Technological University

Smita Jhajharia
Guest Faculty
Delhi Technological University

DR.Seema Verma
Associate Professor
Banasthali University

***Abstract*--Every software company requires to follow a well defined methodology for the proper development of its product. Software Development Life Cycle Models help companies to produce good quality software within the time limit and cost budget. Two widely used SDLC's are discussed in this paper. This paper also discusses their use along with their pro's and con's. Software development in Microsoft is also discussed .**

***Index Terms*--Software Development Life Cycle, Waterfall model, Agile models.**

I INTRODUCTION

SDLC is the process consisting of a series of well planned activities to develop or modify the software products [1]. They usually contain a series of steps that provide a model for the development and management. SDLC's aim at improving the quality of the software product and also of the development method. It helps produce a software that is cost-effective, competent and of high quality. Once an application is created, the SDLC maps the proper delivery and retirement of the .The SDLC's usually contain the following stages: requirements analysis, design, implementation (construction), testing, release and maintenance. There are two types of SDLC that are widely in use today: waterfall and agile. The waterfall model is more customary and requires a well thought out plan and defined set of requirements in contrast to the agile SDLC which has less stringent guidelines and makes adjustments as needed.

WATERFALL MODEL

It is considered as the most basic software development life cycle model and is a linear and sequential approach to developing software. In this model, each process requires the need to be completed before moving on to the next phase. Progress is compared to a waterfall which gives the model its name. Also just like natural waterfalls where once the water has flowed through the edge of the cliff and began flowing downwards never turns back to reach the top of the hill the progress in the waterfall model can never be reverted i.e. we cannot turn back from one phase to the previous one. We have only one option and that is to move forward. This model is also called the **classic life cycle model** as it suggests a systematic sequential and classical approach to software development[2]. In a waterfall process, normally a documents is the output of each phase which serves as the input to the next phase. Team members cannot change the outputs that the project has certified. However, requirements are subject to change. There is thus a need for a mechanism which ensures that the modifications are done in a controlled manner without affecting the product and its progress. Companies like Toyota use this model for development.

The phases of a waterfall model can be generalized as follows-

1. Requirement Analysis and Specification Phase It is probably the most error prone, time consuming and the costliest phase. This phase aims to understand, gather and document the requirements of the user. Requirements of the user are the main reason for the development of software and hence are needed to be noted with precision. Requirement elicitation is a joint effort of the customers and the developers as the goal is to document all the functions, performance and interfacing requirements of the software product. This phase leads to the development of a large document written in natural language and which contains the "what"of system without describing the "how". This resultant document is called the *software requirement specification* (SRS) document. The SRS may act as a contract between the developer and the customer. In case of failure to fulfill the

ISSN: 2278 – 7798

requirements in the SRS, the product may be termed as a failure.

2. Design Phase – The SRS document produced in the previous stage serves as the input of this phase. The requirements are to be transformed into a structure suitable for design and implementation in some suitable programming language. The overall architecture is defined and the high level detailed and designed work is performed. The work / progress is documented and is known as the software design description document (SDD). The SDD may contain technical jargons and tells also about the "how "of the software. The information provided in the SDD should be sufficient to start the designing.

3. Implementation and Unit Testing - Based on the information provided by the SDD the designers have to start building/developing the software. If the SDD contains all the required information and is complete, consistent and accurate then this phase should go smoothly. Testing focuses on the examination, correction and modification of the code. In unit testing the software is tested one module at a time independent of the other. Usually, this process of integration and system testing requires reworking the modules and writing new code to correct problems in the operation or interactions of the pieces due to unforeseen problems as well as mistakes, miscommunications, or changes that have crept into the design of the parts during the project. If this integration work goes well, the team will ship the product when there are no serious bugs (errors or defects) remaining.

4. Integration and System Testing – Unit testing poses the following problems a) How do we test individual modules without a driver function? b) It doesn't tell us anything about how different modules interact with each other. System testing overcomes this issue by testing the software as a whole. Effective testing will ensure higher quality of our software, more satisfied users, lesser maintenance cost and more accurate and reliable results. This phase is a very expensive activity and consumes upto one-half the cost of the total software budget.

5. Operation and Maintenance Phase- It is the phase that comes after the software has been released, installed and is operational. The maintenance phase starts with the very release of the software and continues throughout the life cycle of the software. Software maintenance includes error correction, enhancement of functions and capabilities, introduction of new functions, removal of obsolete ones and optimization of the software [3].

DISADVANTAGES

1. It requires all the requirements to be specified at the very start which is not possible for real life projects mostly because customers are notorious for changing their stated requirements. Also unless the developers and the requirement specifiers are highly skilled, it is hard to know exactly what is needed a phase until some time is on that phase. Thus the software is required to be highly flexible and adaptable with taking into considerations that requirements are subject to change.

2. Since the model is linear it does not allow changes to be accommodated.

3. A working product is not available till late.

4. Real projects are rarely sequential and hence the waterfall model doesn't scale up well to large projects.

5. It doesn't accommodate risks and uncertainty. Unless there is iteration and feedback, there may be no way to improve initial imperfections in one of the later phases.

6. It is a poor model for large and ongoing projects.

7. It is difficult to measure the progress at every stage of the model .also the time and cost taken by each phase is not determinable.

8. Integration is done at the very end which doesn't help in the identification of challenges and business bottlenecks early in the life cycle.

9. A lot of time is wasted while waiting for one phase to be completed. For e.g., When the designers are busy designing the software the builder's time is wasted. Constant testing of the design, its implementation and verification of every phase is required to validate the phases preceding them. Some may argue that that if we follow a designed process and don't allow any room for mistakes then there should be no need to constantly validate the preceding phases.

10. Development of the SRS may take a lot of time for largeprojects.

11. Coordination of the work of a large group of people building different components of the software requires a high level of management that the simple

ISSN: 2278 – 7798

sequence of activities in the waterfall model cannot guarantee. To implement this synchronization and simultaneously provide programmers with the required amount of freedom is perhaps an problem that managers face.

ADVANTAGES

1. The major advantage of the waterfall or lifecyclemodel is that it provides a structure for organizing and controlling a software development project.

2. Design details and errors are captured by the model before any software is written and hence saving time during the development.

3. A proper technical document is made that makes it easy or the customers to know what they should expect from the software.The documentation also help in the process of maintenance.

4. If the procedure are followed properly then the cost and time estimation can be accurately done.

5. Because of its sequential and linear nature faults in one phase can be detected before moving on to the other.

6. A lot of emphasis is done on paperwork. Any new worker joining the development team may find It easy to catch up with the help of these documents.

7. For smaller projects it is the best model to use. Also it requires less resources compared to the others.

8.It allows departmentalization and managerial control. This allows the product to be completed on time by setting a schedule for every phase.

9. A project plan can be utilized for indistinguishable or comparable projects in the future.

Best for projects that deal with services oriented and non physical deliverables like code, copywriting and design projects.

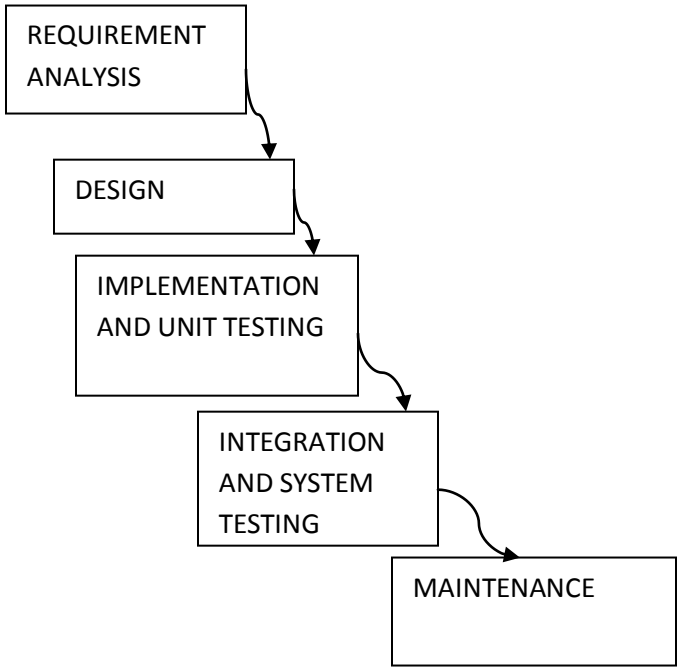
WHEN TO USE WATERFALL

1. When a clear picture of the final product is available.

2. When the requirements are well defined and understood and are not subject to change.

3. When the interest is on the final product to be developed rather that time being the concern.

The waterfall model works pretty well where the product development mainly consists of adding limited functionalities to an existing set of functionality. It also scales well for projects where changes in design can be introduced in a controlled way and the development can be continued without the requirement of customers or competitors. However waterfall model fails when there is so much new content or so many uncertainties to resolve that they become inevitable. For e.g. In the PC software market, for example, both hardware technologies and customer requirements change very rapidly. In such situations, capturing the specifications of a project at the very beginning is difficult. Consequently, the development cannot occur in a linear fashion. Any change in any part of the product for e.g., due to feedback from customers or evolution in particular hardware or software technologies, or even just to add a feature that a competitor has just introduced, may end up with pieces that no longer are compatible.



The testing fails. The project team now has to rework the pieces .Much of the developed product may now seem a waste .With these types of problems the software projects may exceed the time limit and end up being late, over budget, and with bugs. Alternatives to the waterfall have been proposed that use"iterative" approaches to development. These approaches see developers moving around in phases, vacillating between the designing, coding, and testing phase as the development of the software proceeds. This type of development in iteration is what many developers undergo in projects the though in an unplanned manner. Some companies refer to these iterations as "incremental builds". Many companies have started to move away from the classic waterfall model in practice although Companies still using elements of a waterfall process to plan and control software development can be found. For e.g., the U.S. Department of Defense, information system providers required the need to follow, as well as

ISSN: 2278 – 7798

document their procedure and used the waterfall model.

AGILE MODEL

Agile development is in itself a huge umbrella term that includes other agile methodologies also. These initially include the Scrum, XP, Crystal, FDD, and DSDM [4]. However lean agile methodologies also have been included as they have emerged as valuable agile methodologies in the past. Agile models were specifically designed keeping the adaptability of changing requirements in mind. An agile method is a combination of iterative and incremental process models keeping in mind the flexibility and the timely delivery of the software. Agile models consider every development process to be different and believe that the existing methods need to be personalized in order to best suit the project requirements. Agile provides methods to assess the development and risks and also the direction throughout the development lifecycle.

The product is developed in a series of rounds known as iterations. Each iteration typically lasts for 2-3 weeks and involves various team's working simultaneously on various areas like requirement gathering, requirement elicitation, planning, design, coding and testing. Each iteration guarantees an enhancement in features of the product of the previous iteration and the final iteration product involves all the features demanded by the customer [5].Agile works on the following approaches-

1. Individuals and Interactions –In agile, self organization and encouragement are as important as are interactions like pair programming and co-location.

2. ***Working Software-***Agile requires the need to develop a working prototype early in the life of a software and this prototype is often the best way to communicate with the stakeholders who can't appreciate the "how" of the software and technical jargons.

3. ***Customer Involvement-***Since agile doesn't require all the requirements at the beginning of the software, it relies on customer interaction throughout the development.

4. Quick Adaptability- Agile should be able to adapt to changing requirements quickly and as the need may be for the proper and timely development of the software. This "inspect and adapt" technique significantly decreases the development costs and time for shipment [6]. Teams can start developing their software in chorus with the requirement gathering process, which reduces the impact of the 'analysis paralysis 'on the progress. A team's work cycle is limited to 2 weeks because of which stakeholders have frequent opportunities to regulate releases for success. Agile helps develop the right software. It empowers companies to constantly enhance their products and plan their releases in order for better acceptance and greater success. Hence agile makes sure that the effort put in doesn't go waste and that the products market relevance doesn't drop and that the teams work doesn't wind up on a shelf, unreleased. An adaptive agile team will have difficulties describing how much progress will be observed in the next iteration or probably what risks might be encountered but it might be able to tell what are the functionalities that will be added to the product. The further away the deadline the vaguer is the idea. Each agile team contains a customer representative who is chosen by stakeholders to operate on their behalf and be available to answer questions that the developers have. At the end of each iteration, stakeholders and the customer representative evaluate the progress and reassess the priorities in order to optimize the return on investment (ROI) and for better alignment with customer needs and company goals. Companies like Pivotal Labs, Microsoft, RailsCarma etc use Agile mode of development.

ADVANTAGES

1. It takes the requirements and changing nature of software development in consideration and hence scales well for real life projects irrespective of their sizes.

2. It promotes team work. Everyone involved in the development process work together simultaneously on different areas of the software.

3. A working software product is available In the early phases of the software.

4. Agile doesn't dictate the need to document the work and minimal rules are employed. It also requires the minimum amount of resources compared to other models.

5. It makes planning just a voluntary phase and makes the process easy to manage and flexible.

ISSN: 2278 – 7798

6. Developers and customers constantly communicate with each other and interaction with the customers is given more importance than the process and tools. This also helps in the understanding of what the users expect from the final product.

7. Best for projects that deal with services oriented and non physical deliverables like code, copywriting and design projects.

DISADVANTAGES

1. It is not suitable for handling complex dependencies and has more risk of extensibility, sustainability and maintainability. In case of some software deliverables it is difficult to assess the effort at the beginning of the process.

2. Though it doesn't require an initial planning phase it still requires an overall development plan. A good management system, an agile leader and a PM practice is required. Without them the project might not work.

3. Completion of work within a strict period of time with good quality and specified requirements makes it difficult to manage.

4. Customer involvement is a must. Without the cooperation of users the development process would be hampered and may lead to failure.

5. No importance on documentation is there and hence any new member joining the team may find it difficult to adjust. Also existing team members might need to reference the SRS sometimes and its absence can stir the development in a wrong direction.

6. Some decisions require the expertise of the senior software developers; hence novice developers are not really welcomed unless combined with experienced resources.

7. Due to development in iterations errors and risks are identified early.

8. Not suited for projects with strictly defined requirements and scope.

9. Requires vigilant backlog and documentation maintenance , and tech debt management.

WHEN TO USE AGILE

1. When the requirements of the software are not well specified or when the requirements are expected to change during later phases of the development process .In agile new changes can be implemented at a very low cost because of the frequency of new increments that are produced.

2. If freedom of options and time is required by both the developers and the stakeholders. Having options provides them with the facility to leave important decisions until more or better data are available and also the project can be continued without the fear of it being a failure.

II WHY NOT TRADITIONAL

Software is not hardware. Software cannot be developed like an automobile where each part is added in sequential phases , each phase depending on the previous one. *DR. Winston Royce* condemned the phase based approach in which there was a need to go sequentially in the order of requirement elicitation, design, implementation and then testing. He specifically dejected the idea of lack of communication between the customers and the developers.

It is easy to see why waterfall model is not feasible. Knowing all the requirements in the very start and not expecting them to change made no sense. A team might build the software on time with all the specifications taken care of but in the interim the realities might have changed dramatically that the product may seem irrelevant and may not be able to score well.

III AGILE VS TRADITIONAL

Agile Methods are considered to be at the opposite end of the "plan driven " or "disciplined" methods rainbow. Projects usually are considered as lying between the "adaptive" to "disciplined" range. Agile methods occupy the "adaptive" bandwidth. An adaptive agile team will have difficulties describing how much progress will be observed in the next iteration or probably what risks might be encountered but it might be able to tell what are the functionalities, that will be added to the product. The further away the deadline the more vague is the idea. Whereas the waterfall model is predictive which in contrast to the agile model focuses on planning for the future well ahead of time. A predictive can tell exactly how and by how much the software is going to develop and what changes might be observed. However the predictive teams are not ready for unidentified or unaccepted changes and have difficulty adjusting to them. The plan is set and accommodating new changes may mean that the

ISSN: 2278 – 7798

process might have to be started all over again. Because of such an issue only the most important changes will be given a chance of being implemented[7].

IV WHAT IS YOUR METHODOLOGY?

Use Agile if...

1. If the requirements and functions are subject to change.

2. The product needs to be developed In a limited time.

3. A quick prototype with only certain functionalities is needed before the final product is available.

4. Agile requires the active participation of the stakeholders.

5. If the team is working on a highly collaborative, self organizing manner. Self organizing ensures that the team members are actively planning and estimating their own work.

6. If the team is willing to develop the software in installments.

Use Waterfall if...

1. The requirements are well defined and fixed.

2. There is no time limit or a reasonable amount of time is available to develop the software.

3. There is a need to document everything.

4. You need a safe software with a high reliability level.

5. You want to use the finished software.

6. You do not have resources for face-to-face everyday work.

V SIMILARITIES

1. Both provide methods, tools, techniques and well defined processes.

2. Both require a disciplined approach and have a pre decided finish time.

3. Both want customer satisfaction and want to adhere to the decided time and cost schedule.

VI MICROSOFT CASE STUDY

Many companies have begun to deviate from the waterfall style and make changes in order to be able to improve their capability to control changes in the software . One such company is Microsoft. After it had a number failed projects it could no longer have products involving small groups of programmers using *ad hoc* practices. As its customer support expanded there was a need to develop quality products and delivery problems became unacceptable. The dare Microsoft was facing was to introduce more structure and predictability into their loosely organized development practice and still have flexibility and creativity. Microsoft wanted a different process that would allow incremental approach to designing, respond to customer inputs, Build prototypes, and allows changes[8]. Microsoft includes the following phases in its product development.

1. Review and Planning- The number of releases required are planned with an explanation on why the particular release is required, and Their interdependencies with other products. The marketing team create sales forecasts on the basis of the product plans. The budget is then planned and the expected profit is calculated. Based on this analysis, a head-count is determined.

2. Development- Microsoft divides the development into three or four milestones, where each milestone has its own coding and testing phase. Groupings of features determine the milestones. The most difficult and important features are built first in contrast to the traditional SDLC's where the focus is on complete implementation of the specifications. As a result,
the pieces are put three or four times during the development with testing, debugging, and Integration activities all done in parallel in every milestone. Also it is assumed that specifications may change, so no specification documents are made at the start of a project. Microsoft creates a "build" of the product every day.

3.Testing Phase-Microsoft relies heavily on automated and manual testing by testers on a daily builds. Testing includes documentation, tutorials, set-up, hardware configurations, primary functions, and supporting utilities. Also, three types of beta tests are used to ensure better acceptance. Microsoft has introduced a parallel, incremental, and iterative but planned approach to product development that offers several benefits to the development organization:

1. It ***divides the large products into smaller controllable pieces*** (that can be created in a few months).

2. It ***enables projects to proceed methodically even when a complete and stable product cannot be determined*** at the project's beginning.

3. It ***allows large teams to work like small teams*** by dividing work into chunks, arranged in parallel but synchronized, and incremental.

VII CONCLUSION

Every project is different and requires to be handled differently. Hence, it is better not to grip on one particular methodology. the desires of the company

ISSN: 2278 – 7798

and project are subject to change regularly, and one needs to be elastic in how to approach these projects for them to be successful. No single methodology is the panacea, so the trick is to determine which method works and tailor it to suit the individual needs.

VIII. ACKNOWLEDGMENT

I would like to take this opportunity to thank and express my deep respect and regard for ***MS.Smita Jhajharia(****Guest Faculty, DTU, Delhi****)*** *and* ***DR. Seema Verma (****Associate professor, Banasthali University)* for their exemplary guidance, constant encouragement and valuable feedback throughout the preperation of this paper.Their valuable suggestions were immensly helpful and working under them was a knowledgeable experience.

REFERENCES

[1] http://www.tutorialspoint.com/sdlc/

[2]http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t

[3] Youssef Bassil " A Simulation Model for the Waterfall Software Development Life Cycle" International Journal of Engineering & Technology (IJET), ISSN: 2049-3444 May 2012

[4]http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Agile_Model

[5] Malik Hneif and Siew Hock Ow "Review Of Agile Methodologies In Software Development", International Journal of Research and Reviews in Applied Sciences ISSN: 2076-734X, October 2009

[6]http://www.ambysoft.com/essays/agileLifecycle.html

[7] http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/

[8] Michael A. Cusumano and Stanley Smith "Beyond the Waterfall:Software Development at Microsoft", August 16, 1995

ISSN: 2278 – 7798