Agile Vs Traditional Methodology: A Comparative View

Jyotsna[1] Mukul Varshney[2] Shivani Garg[3] Abha Kiran Rajpoot[4]
[1,2,3,4]Department of Computer Science & Engineering
[1,2,3,4]Sharda University

Abstract— One of the first decisions we face for each of our project implementations at Segue is “Which development methodology should we use?” Most software companies nowadays aim to produce valuable software in short time period with minimal costs, and within unstable, changing environments. Agile Methodologies were thus introduced to meet the new requirements of the software development companies. This paper gives the comparative view of agile development and traditional development.

Key words: Agile Methodology, Traditional Methodology

I. Introduction

A software application or an information system is designed to perform a particular set of tasks. Often, this set of tasks that the system will perform provides well-defined results, which involve complex computation and processing. It is therefore a harsh and tedious job to govern the entire development process to ensure that the end-product comprises of high degree of integrity and robustness, as well as user acceptance. Thus, a systematic development process which is able to emphasize on the understanding of the scope and complexity of the total development process is essential to achieve the said ¬characteristics of a successful system. There are two SDLC methodologies which are utilized by most system developers, namely the traditional development and agile development. These two approaches will be explained along with their advantages and disadvantages.

II. Waterfall Methodology

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of software development model is basically used for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model software testing starts only after the development is complete. In waterfall model phases do not overlap[4,5].

As this process is sequential, once a step has been completed, developers can’t go back to a previous step – not without scratching the whole project and starting from the beginning. There’s no room for change or error, so a project outcome and an extensive plan must be set in the beginning and then followed carefully.

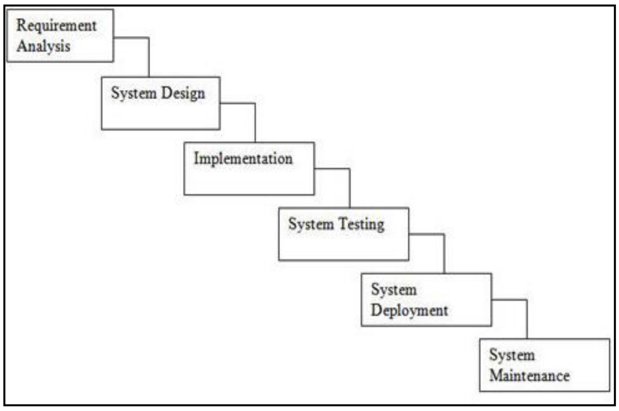

Fig. 1: Traditional Approach: Waterfall

A. Advantages of the Waterfall Methodology

1) The waterfall methodology stresses meticulous record keeping. Having such records allows for the ability to improve upon the existing program in the future[1,2].
2) With the waterfall methodology, the client knows what to expect. They’ll have an idea of the size, cost, and timeline for the project. They’ll have a definite idea of what their program will do in the end.
3) In the case of employee turnover, waterfall’s strong documentation allows for minimal project impact

B. Disadvantages of the Waterfall Methodology

1) Once a step has been completed, developers can’t go back to a previous stage and make changes.
2) Waterfall methodology relies heavily on initial requirements. However, if these requirements are faulty in any manner, the project is doomed.
3) If a requirement error is found, or a change needs to be made, the project has to start from the beginning with all new code.
4) The whole product is only tested at the end. If bugs are written early, but discovered late, their existence may have affected how other code was written. Additionally, the temptation to delay thorough testing is often very high, as these delays allow short-term wins of staying on-schedule.
5) The plan doesn’t take into account a client’s evolving needs. If the client realizes that they need more than they initially thought, and demand change, the project will come in late and impact budget.

C. When to use waterfall methodology?

1) When there is a clear picture of what the final product should be.
2) When clients won’t have the ability to change the scope of the project once it has begun.
3) When definition, not speed, is key to success

III. Agile Methodology

Agile came about as a solution to the disadvantages of the waterfall methodology. Instead of a sequential design process, the Agile methodology follows an incremental approach.

Agile methods are a subset of iterative and evolutionary methods and are based on iterative enhancement and opportunistic development processes. In all iterative products, each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test[1]. Each iteration leads to an iteration release that integrates all software across the team and is a growing and evolving subset of the final system. The purpose of having short iterations is so that feedback from iterations N and earlier, and any other new information, can lead to refinement and requirements adaptation for iteration N + 1. The customer adaptively specifies his or her requirements for the next release based on observation of the evolving product, rather than speculation at the start of the project. Agile is to have frequent inspection and adaptation in line with customer expectations and needs[2]. Agile methodologies include: Extreme Programming, Agile Modeling, SCRUM, Crystal methodologies, Feature-Driven Development, Adaptive Software Development. Agile is a set of software development methodologies, which are fundamentally based on some common principles

Agile is based on a few principles detailed below:

1) KIS [Keep it simple] approach
2) Customer satisfaction
3) Working software is the measure of progress
4) Working software is delivered in frequent intervals
5) Late changes in requirements can be accommodated
6) Face to face communication within teams
7) Self organizing teams
8) Attention to technical excellence and good design
9) Adoption to change

Developers start off with a simplistic project design, and then begin to work on small modules. The work on these modules is done in weekly or monthly sprints, and at the end of each sprint, project priorities are evaluated and tests are run. These sprints allow for bugs to be discovered, and customer feedback to be incorporated into the design before the next sprint is run[3].

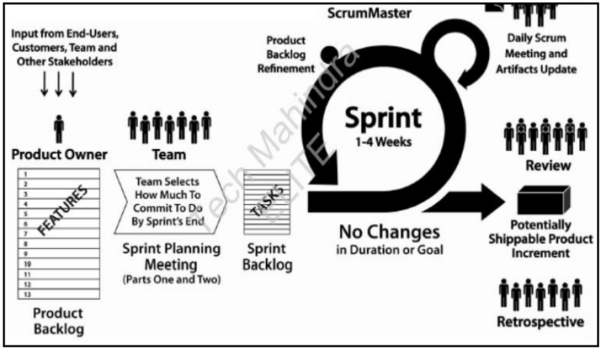
ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Review
FEATURES
1
2
3
4
5
6
7
8
9
10
11
12
13
Product Backlog
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting
(Parts One and Two)
TASKS
Sprint Backlog
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Retrospective

Fig. 2: Agile Development: Scrum

A. Advantages of the Agile Methodology

1) The Agile methodology allows for changes to be made after the initial planning. Re-writes to the the program, as the client decides to make changes, are expected.
2) Because the Agile methodology allows you to make changes, it's easier to add features that will keep you up to date with the latest developments in your industry.
3) At the end of each sprint, project priorities are evaluated. This allows clients to add their feedback so that they ultimately get the product they desire.
4) The testing at the end of each sprint ensures that the bugs are caught and taken care of in the development cycle. They won't be found at the end.
5) Because the products are tested so thoroughly with Agile, the product could be launched at the end of any cycle.

B. Disadvantages of Agile Methodology

1) With a less successful project manager, the project can become a series of code sprints. If this happens, the project is likely to come in late and over budget.
2) As the initial project doesn't have a definitive plan, the final product can be grossly different than what was initially intended[4].

C. When should you use Agile methodology?

1) When rapid production is more important than the quality of the product.
2) When clients will be able to change the scope of the project.
3) When there isn't a clear picture of what the final product should look like.
4) When you have skilled developers who are adaptable and able to think independently.
5) When the product is intended for an industry with rapidly changing standards.

IV. CONCLUSION

Both the Agile and waterfall methodologies have their strengths and weaknesses. The key to deciding which is right for you comes down to the context of the project. Is it going to be changing rapidly? If so, choose Agile. Do you know exactly what you need? Good. Then maybe waterfall is the better option.

REFERENCES

[1] Agile Alliance. Manifesto for Agile Software Development. [Online] Retrieved 16th March 2009. Available at: http://www.agilemanifesto.org .

[2] Agile Modeling Home Page. Effective Practices for Modeling and Documentation. [Online] Retrieved 17th March 2009. Available at: www.agilemodeling.com

[3] 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)

[4] Thesis, Jonna Kalermo and Jenni rissanen, "Agile software development in theory and practice"

[5] Chan, F., & Thong, J. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. Decision Support Systems (46), 803-814.

[6] Sharp, H., Robinson, H., & Petre, M. (2009). The role of physical artefacts in agile software development: Two complementary perspectives. Interacting with Computers , 108-106.