Agile Methodologies and Its Processes

[1,]Akanksha, [2,]Akansha Rakheja , [3,]Latika Kapur , [4,]Kanika Ahuja
[1,2,3,]Information Tech IT)Dronacharya College of Engineering, Gurgaon

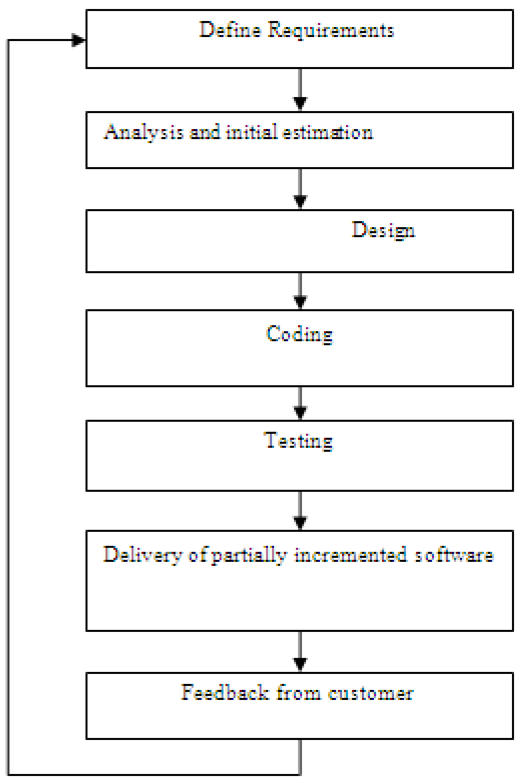
ABSTRACT:
"Agile"-denotes the quality of being agile. This paper deals with the concept of agile methodology. It will include some of the software development process models used. In particular the aim of the agile process is to satisfy the customer through early and continuous delivery of valuable software. This paper compares the agile processes with other software development life cycle models. Advantages and disadvantages of the agile processes are also included in this paper.

Keywords: Extreme Programming (XP), Scrum, Software Development Life Cycle (SDLC)

I. INTRODUCTION

Agile software development (also called "agile") isn't a set of tools or a single methodology, but a philosophy put to paper in 2001 with an initial 17 signatories. Agile was a significant departure from the heavyweight document –driven software development methodologies such as waterfall. Plan driven methods are those that begin with the solicitation and documentation of a set of requirements that is as complete as possible. Based on these requirements, one can then formulate a plan of development. Usually, the more complete the requirement the better the plan. Some examples of plan driven methods are various waterfall approaches. An underlying assumption in plan driven processes is that the requirements are relatively static. On the other hand, iterative methods such as spiral-model based approaches described in recently agile approaches count on change and recognize that only constant is changed. With the passage of time, the software industry, software technology and customer expectations were moving very quickly and the customers were becoming increasingly less able to fully state their need up front. As a result, agile methodologies and practises emerged as an explicit attempt to formally embrace higher rates of requirements change. Agile methods are a subset of iterative and evolutionary methods and are based on iterative enhancement and opportunistic development process. In all iterative products, each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test. Each iteration leads to an iteration release (which may be only an internal 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 . There is quantitative evidence that frequent deadlines reduce the variance of a software process and, thus, may increase its predictability and efficiency. The pre-determined iteration length serves as a timebox for the team. Scope is chosen for each iteration to fill the iteration length. Rather than increase the iteration length to fit the chosen scope, the scope is reduced to fit the iteration length. A key difference between agile methods and past iterative methods is the length of each iteration. In the past, iterations might have been three or six months long. With agile methods, iteration lengths vary between one to four weeks, and intentionally do not exceed 30 days. Research has shown that shorter iterations have lower complexity and risk, better feedback, and higher productivity and success rates.

Agile process is an iterative approach in which customer satisfaction is at highest priority as the customer has direct involvement in evaluating the software. The agile process follows the software development life cycle which includes requirements gathering, analysis, design, coding, testing and delivers partially implemented software and waits for the customer feedback. In the whole process, customer satisfaction is at highest priority with faster development time. The following figure depicts the software development life cycle of Agile Process.


Define Requirements
Analysis and initial estimation
Design
Coding
Testing
Delivery of partially incremented software
Feedback from customer

II. CHARACTERISTICS OF AGILE PROCESS

The core agile delivery principle is that although agile methods differ somewhat in their practices, all of them advocate these core principles--iterative and incremental delivery, collaboration and continuous improvement and here are top reasons to adopt agile methodology.

- Iterative and incremental delivery: Project delivery is divided into small functional releases or increments to manage risk and to get early feedback from customers and end users. These small releases are delivered on a schedule using iterations that typically last between one and four weeks each. Plans, requirements, design, code and tests are created initially and updated incrementally as needed to adapt to project changes.
- Collaboration: All core project team members including an on-site customer are co-located in a shared, open area to facilitate face-to-communication and conduct interactions. Team members self-organize by continuously completing tasks collaboratively without top-down management control.
- Continuous Improvement: Practices that enable delivery process inspection and adaptation are integrated into agile methods. Project Reflections are meetings conducted while the project is underway to facilitate regular reflection on its successes and failures, and any of the tools and techniques applied.
- Modularity: Agile process decomposes the complete system into manageable pieces called modules. Modularity plays a major role in software development processes.
- Convergent: All the risks associated with each increment are convergent in agile process by using iterative and incremental approach.
- Adaptive: Due to the iterative nature of agile process new risks may occurs. The adaptive characteristic of agile process allows adapting the processes to attack the new risks and allows changes in the real time requirements.
- Time Boxing: As agile process is iterative in nature, it requires the time limits on each module with respective cycle.
- Parsimony: In agile processes parsimony is required to mitigate risks and achieve the goals by minimal number of modules.

III. AGILE METHODOLOGIES

This section provides a brief introduction to three agile methodologies. The three were chosen to demonstrate the range of applicability and specification of the agile methodologies. For each methodology we provide an overview of its process.

3.1. Extreme Programming (XP)

Extreme Programming (XP) originators aimed at developing a methodology suitable for "object-oriented projects using teams of a dozen or fewer programmers in one location." The methodology is based upon five underlying values:

a. Communication: XP has a culture of oral communication and its practices are designed to encourage interaction.
b. Simplicity: Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.
c. Feedback: The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration.
d. Courage: The other three values allow the team to have courage in its actions and decision making.
e. Respect: Team members need to care about each other and about the project

XP is the most successful method of developing agile software because of its focus on customer satisfaction. XP requires maximum customer interaction to develop the software. It divides the entire software development life cycle into several number of short development cycles. It welcomes and incorporates changes or requirements from the customers at any phase of the development life cycle.

The Extreme Programming software development process starts with planning, and all iterations consist of four basic phases in its life cycle: designing, coding, testing, and listening. The overriding values that drives the XP life cycle are continual communication with the customer and amongst the team, simplicity by harping on the minimalist solution, frequent feedback through unit and acceptance testing, and the courage to take on problems proactively and integrate testing and changes in the development phase.

A. Planning:

The first phase of Extreme Programming life cycle is planning, where customers or users meet with the development team to create 'user stories' or requirements. The development team converts user stories into iterations that cover a small part of the functionality or features required. A combination of iterations provides the customer with the final fully functional product. The programming team prepares the plan, time, and costs of carrying out the iterations, and individual developers sign up for iterations. One planning approach is the critical path method, grouping iterations essential for project progress in a linear fashion, and arranging for completion of other iterations parallel to the critical path.

B. Designing

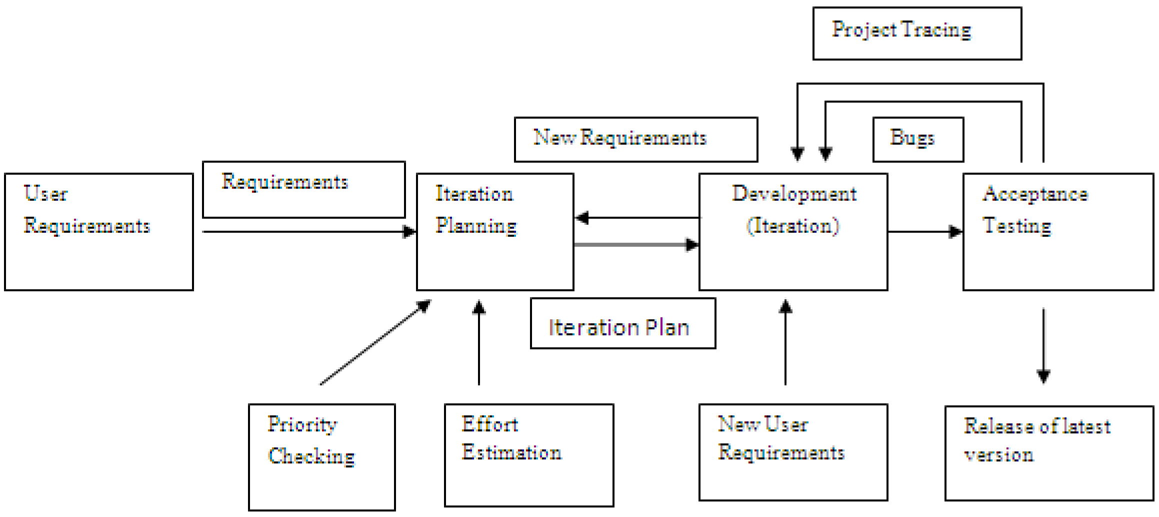
An iteration of XP programming starts with designing. Thrust on simplicity by expressing a thing only once and not adding functionality in anticipation. It uses systems metaphor or standards on names, class names and methods, and agreeing on uniform styles and formats to ensure compatibility among the work of different team members. It uses Software Class Responsibilities and Collaboration (CRC) Cards that allow for a departure from the traditional procedural mindset and make possible object oriented technology. Such cards allow all members of the project team to contribute ideas, and collate the best ideas into the design. It creates spike solutions or simple programs that explore potential solutions for a specific problem, ignoring all other concerns, to mitigate risk.

C. Coding

Coding constitutes the most important phase in the Extreme Programming life cycle. XP programming gives priority to the actual coding over all other tasks such as documentation to ensure that the customer receives something substantial in value at the end of the day.
Standards related to coding include:

- Developing the code based on the agreed metaphors and standards, and adopting a policy of collective code ownership.
- Pair programming or developing code by two programmers working together on a single machine, aimed at producing higher quality code at the same or less cost.

- Strict adherence to 40-hour workweeks with no overtime. This ensures the developers work in the peak of their mental and physical faculties.
- Frequent integration of the code to the dedicated repository, with only one pair integrating at a time to prevent conflicts, and optimization at the end.



D. Testing

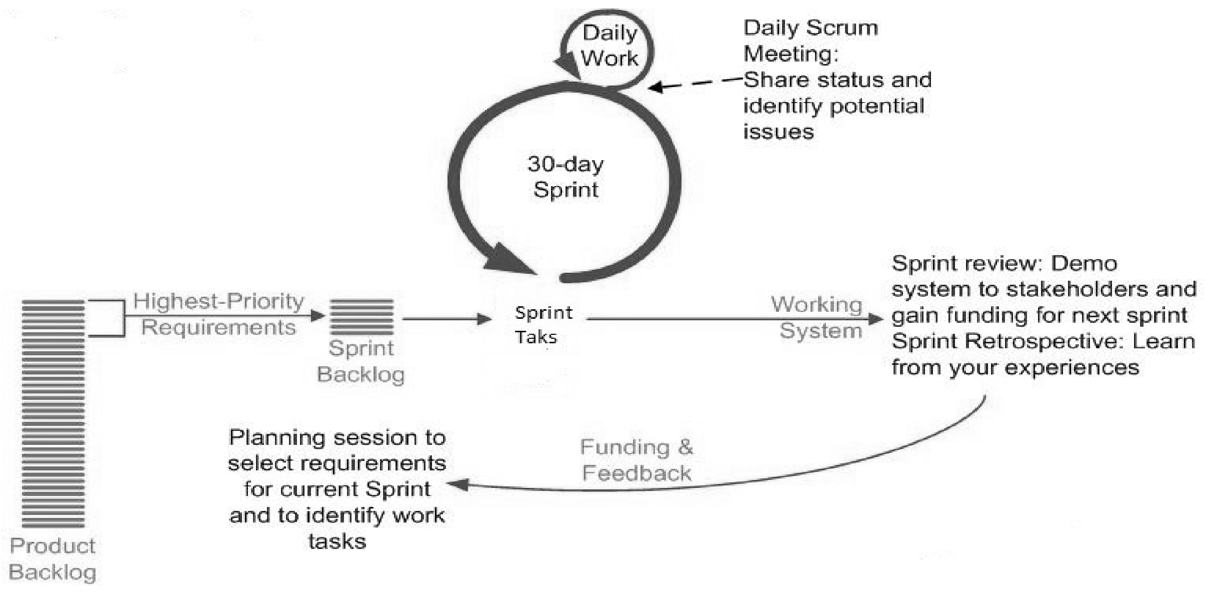
Extreme program integrates testing with the development phase rather than at the end of the development phase. All codes have unit tests to eliminate bugs, and the code passes all such unit tests before release. Another key test is customer acceptance tests, based on the customer specifications. Acceptance test run at the completion of the coding, and the developers provide the customer with the results of the acceptance tests along with demonstrations.

E. Listening

The basis of extreme programming is a continuous mechanism of customer involvement through feedback during the development phase. Apart from the customer, the developer also receives feedback from the project manager.The basis of feedback is the customer acceptance tests. Each feedback of the customer that specifies revised requirement becomes the basis of a new design, and the process of design-coding-tests-listening repeats itself. If the customer remains satisfied with the test results the iteration ends there, and the design for the new iteration starts, which again follows the design-coding-testing-listening cycle.

3.2. SCRUM

Scrum is another popular method of agile development through which productivity becomes very high. It is basically based on incremental software development process. In scrum method the entire development cycle is divided into a series of iteration where each iteration is called as a sprint. Maximum duration of a sprint is 30 days. The main idea of Scrum is that systems development involves several environmental and technical variables (eg. requirements, time frames, technology and resources) that are likely to change during the process. This makes the development process unpredictable and complex, requiring flexibility of the systems development process for it to be able to respond to the changes. As a result of the development process, a system is produced which is useful when delivered.


Daily Work
Daily Scrum Meeting: Share status and identify potential issues
30-day Sprint
Highest-Priority Requirements
Sprint Backlog
Sprint Taks
Working System
Sprint review: Demo system to stakeholders and gain funding for next sprint Sprint Retrospective: Learn from your experiences
Funding & Feedback
Planning session to select requirements for current Sprint and to identify work tasks
Product Backlog

The method starts with collecting user requirements but it is not expected that all the requirements should come out from the user at the beginning. User can change their mind at any time during development; they can add new features, remove or update some existing features. Next phase is to prioritize the requirements and the list is known as product backlog. A proper planning for sprint should be done i.e. how many sprints are needed to develop the software, duration of the sprint, and what are the requirements from the product backlog should be implemented in each sprint. This particular list is known as sprint backlog. During each sprint one sprint meeting is held every day to take the feedback how much work has been done. After each sprint review is taken to determine whether all the requirements for that particular sprint have already been implemented or not and to decide the requirements should be implemented at the next sprint. After each sprint we get a working increment of the software. A sprint is the basic unit of development in Scrum. The sprint is a "time boxed" effort, i.e. it is restricted to a specific duration. The duration is fixed in advance for each sprint and is normally between one week and one month.

IV. COMPARISON

4.1. Between various process models:

FEATURES	PROCESS MODELS	
	AGILE	WATERFALL
Architecture	Informal and incremental	Well documented and completed before coding
Developer	Shares ownership of code	Responsible for one area
Functionality	Focus on completing stories in short iterations	Focus on completing modules at large milestones
Process	Light process and documentation	Heavy process and documentation
Main roles	Developer	Developer,Arcitect
Communication	Open door policy	Happens only at beginning or at milestones
Orientation	People oriented	Process oriented
Deliverables	Delivers early partial working solutions	No working solutions is provided until late
Cost of change	Low cost of change	High cost of change
Testing	Fully automated, continuous testing at both functional and unit level	Testing as separate phase at the end of project
Knowledge Required	Product and Domain	Product and Domain
Time Boxing	Available	Not Available

4.2. Between agile methologies:

CHARACTERICTICS	XP	SCRUM
Development approach	Iterative increments	Iterative increments
Recommended iteration time period	One to six weeks	Two to four weeks
Project team	Smaller team	All sizes
Team communication	Informal daily meetings	Informal daily meetings
Project size	Smaller projects	All types of projects
Customer involvement	Customer involved	Customer involved through the role of Product owner
Project Documentation	Only basic documentation	Only basic documentation
Advantages	Open workspace, customer as a part of team, well defined best practices, feedback	High level of communication and collaboration
Disadvantages	Weak documentation, lack of discipline, customer presence is mandatory	Weak documentation, poor control over projects

V. ADVANTAGES

- Agile methodology has an adaptive team which is able to respond to the changing requirements.Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face to face communication and continuous inputs from customer representative leaves no space for guesswork.
- Continuous attention to technical excellence and good design
- A reduced budget
- Less defects in the final product
- Become responsive by supporting scope adjustments every iteration
- Decrease risk by always having working software

VI. DISADVANTAGES

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- When change comes so quickly, it is difficult to avoid resistance from stakeholders and complications to end user training.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.
- Because agile methods are not process-oriented and require quick response to change, a lack of documentation is often a primary characteristic
- Agile is not a silver bullet – Agile can be over-hyped, thus leading to unrealistic expectations

VII. CONCLUSION

In this paper we have discussed the agile development life cycle models, characteristics of agile process, methologies of agile process, advantages and disadvantages. In the comparative study of agile software development with other software development models we conclude that agile project is much better than other software development process in terms of productivity, performance, faster time cycles, risk analysis. Agile processes are implemented in important applications such as web based, testing tools, etc.

REFERENCES

[1] Sheetal Sharma et al. / International Journal on Computer Science and Engineering (IJCSE) ISSN , Agile Processes and Methodologies: A Conceptual Study.

[2] [Aoyama 1998] Mikio Aoyama, "Agile Software Process and Its Experience", Proceedings of the 1998 International Conference on Software Engineering,

[3] [Gilb 1988] Tom Gilb. Principles of Software Engineering Management, Addison Wesley.

[4] Williams, L., "Agile Software Development Methodologies and Practices ", in *Advances in Computers*, Volume 80,, 2010

[5] Layman, L., Williams, L., Cunningham, L., *Exploring Extreme Programming in Context: An Industrial Case Study*, Agile Development Conference 2004

[6] http://en.wikipedia.org/wiki/Agile_software_development