Chapter 3

INTRODUCTION TO AGILE

3.1 Introduction

This chapter highlights the complete introduction about the agile. The characteristics of agile are described. The software which posses such characteristics are said to be agile. Here some principles are defined on the basis of which all agile methodology works. The strengths and the limitations of agile approaches are discussed here. The reasons for adopting agile approaches are also mentioned in this chapter. So this chapter shows the complete picture of agile.

3.2 Characteristics of Agile Processes

Any process that is said to be agile, is having the following characteristics:

1. **Modularity**
   It is a key element of any major process. It allows a process to be broken into components i.e called activities.

2. **Iterative**
   Agile Processes focus on short cycles. A set of activities are completed with in each cycle. These cycles will be started and completed in a matter of weeks. However, single cycle i.e iteration will probably not be enough to get the element 100% correct. Therefore, the short cycle is repeated many times to improve the deliverables [27].

3. **Incremental**
   The entire system is not build at once by an agile process; instead, it partitions the nontrivial system into increments which may be developed in parallel, at different times, and at different rates. At each increment, unit test is performed independently. When an increment is completed and tested, it is integrated into the system.
4. **Adaptive**
   During iteration, new risks may be added which require some activities that were not planned. The agile process found new risks. By using the activities planned during the iteration if the goal cannot be achieved, new activities can be added to allow the goal to be reached. Similarly, activities may be discarded if the risks turn out to be ungrounded.

5. **People-Oriented**
   Here people are favored over process and technology. They develop through adaptation in a natural manner. Developers, who are authorized, raise their productivity, quality, and performance. After all, they are the best individuals in the organization to know how to make these changes.

6. **Time Bound**
   Iterations become the perfect unit for planning the software development project. For each iteration time limits can be set and scheduled accordingly. All of the activities of the process cannot be scheduled in a single iteration, instead attempt only those activities necessary to achieve the goals set out at the beginning of the iteration. Activities may be rescheduled if they cannot be completed within the allotted time period.

7. **Collaborative**
   Agile processes promote communication among team members. Communication is a very important part of any software development project. Because in an agile process, a project is developed in pieces, understanding how the pieces fit together is important to create the finished product. There is more to integration than simple communication. Quickly integrating a large project while increments are being developed in parallel requires collaboration.

### 3.3 Principles of Agile Methodology according to Agile Manifesto

According to Agile Manifesto [38], each agile methodology has some principles. These are as follows:
1. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face to-face conversation.

7. Working software is the primary measure of progress. Agile processes promote sustainable development.

8. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. The best architectures, requirements, and designs emerge from self-organizing teams.

11. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

12. Simplicity—the art of maximizing the amount of work not done—is essential.

3.4 Strengths of Agile

Agile is a very powerful technique. Some of its strengths are as follows [9]:

1. **Welcome Changing Requirements, Even Late in Development**

   The underlying philosophy of process-oriented traditional methods is that the requirements are completely locked in and frozen before the succeeding phases. There is a need for flexible and adaptable methods that allow developers to make late changes in specifications. Changes should be allowed throughout the whole
development process. The use of iterations and prototype is proposed, means in order to allow the changes.

2. **Simple**

   In agile methods user is encouraged to invent simple solutions. The reason is that early solutions should be easy to change and that there should be less to change. Program code should be kept as simple as possible in order to support early delivery. Rapid evolution and speed goes hand in hand with simplicity.

3. **Reduced overheads**

   The approach of doing changes at any stage not only reduces overheads, it also helps in the upgrading of programmes.

4. **Cost and Time Reduction**

   The time to develop software in agile methods is shorter than in traditional methods. Consequently, the cost for development process also reduces. With agile methods, only 20 to 40 people in a team are needed. Therefore, the salary which is paid for all of them decreases. The cost in this case reduces a lot comparing with traditional methods where hundreds of members involve in a project. Work is done with face-to-face communication so there will not be money loss for communication fees.

5. **Launchable Product**

   At the end of each tested stage, one has a launchable product. So if any errors are found, they caught and eliminate at the development cycle and product is again tested at the next stage.

6. **Satisfying Stakeholders and Users**

   Agile methods encourage changing requirements from customers, even in the end of development process. Customers work daily with other members in the team. They can make any changes in requirements during the software process. The reason of creating a new iteration is adding new functionalities for software which satisfy new requirements. They themselves write functional tests at the end of each iteration. Because of this, the latest version meets what customers need at the highest level. In the development process, they still continue thinking about new changes. The development process only stops when customers do not require any
change in requirements. Errors are found at the earliest stages, so customers are more satisfied. It means that the final product has all functionalities which the customer wants.

7. **Iterative-Incremental Development**
   The development process in agile methods is an incremental process with a lot of short iterations. Hence, software is developed more and more complicated after each iteration. In addition, it is done throughout all necessary phases in each iteration. For this reason, risks are reduced a lot.

8. **Frequent Integration**
   In most of the agile methodologies studied, integration occurs continuously during the development and production processes. This is because the development process in agile methodologies is iterative-incremental, and executable increments are released in the very first iteration. To achieve this, and also to be able to test the product in the user environment and receive feedback, integration becomes an essential activity.

9. **Effective Planning**
   Since most Agile methodologies are iterative-incremental, planning is an issue taken very seriously. Agile methodologies are also extremely wary of the “Death by Planning” risk encountered in non-Agile processes, and therefore stress the need for regular plan reviews.

10. **Prioritizing Requirements**
    Prioritizing the requirements is a common practice in agile methodology. It facilitates frequent release of executable software, better support for iterative incremental production of software, and helps focus on satisfying the stakeholders.

11. **Teamwork and Collaborative Decision Making**
    Team work is defined as “when a group of people work well together”. A strong encouragement is that business people and developers work closely to each other. Face-to-face conversations are encouraged and the project is carried out with a constant pace where there are no delays or no interruptions.
12. **Rapid Development**

Software systems are developed in an incremental process with several iterations. Necessary phases such as analysis of requirements, design, coding and testing are included in each iteration to make a complete product which means that a new version is released after each iteration. In the case of the XP method, a new version is released in no more than two months and with Scrum method, the duration of each iteration is from one week to one month. It is shorter than spiral or incremental processes in traditional methods. The very short release is the main characteristic to distinguish a software process in agile methods with an incremental process in traditional methods.

Short releases with agile methods can be done because of the following reasons.

a. The customer is required to work alongside members of team. Therefore, developers have information about changes in requirements in a short time and throughout face to face communication.

b. The principle “simplicity” means that the design plan is simple. Hence, programmers can implement it rapidly.

c. All members are in small environments with face-to-face communication, so tasks are rapidly moved to next phase. For example, testers test what programmers have done after a few minutes and give feedbacks.

d. Team includes senior members with much experience in the area of project so they can finish tasks in a short time.

The fast development of software is very important in software industry nowadays. It saves a lot of money, minimizes risks, and so on. The long development process can not bring the benefit for products. Software technologies are improved extremely fast. For this reason of long development process, when the product is finished, it is already obsolete and can not compete with other products.

13. **Less Documentation**

The system developers are encouraged to keep the code simple and straightforward. One of the greatest misunderstandings about agile methods is that
this means that you don’t have to create any documentation at all. What it means is that the documentation burden is should be kept as minimal as possible.

14. Higher Utilization of Work Loads
   Means large scale software are developed with a fixed number of developers.

15. Higher Quality
   It is the strength of agile that higher quality can be achieved by earlier feedback from the customers [2].

16. Higher Flexibility to change of management and development plans [2].

17. Reduces the ‘idle’ time to minimum and thus, increases resource utilization.

18. Helps visualize the final deliverable in early stages of development.

19. Gives a quick estimation of ‘How far are we from final product’.

3.5 Limitations of Agile Methodologies

1. Handling of Larger Teams
   Agile Methodologies only works for small to medium –sized teams. These methods are not suitable to handle large teams. In case of large teams agile mechanisms fail to act effectively. For this conventional methods are more suitable.

2. Requirement of Highly Skilled and Highly Motivated individuals
   Agile development requires highly skilled and highly motivated individuals. The existence of such skilled and motivated individuals is very difficult to find.

3. Difficult to understand exactly where project stands
   In a business or in an environment upper management always want to know that where the project stands but because of having iterative approach and when the project is on track it is difficult to tell exactly about the project.

4. Insufficient necessary documentation
   In agile methods emphasis is given on face to face communication and rapid development. It does not support documentation. It lacks an overall requirements specification document. Customers do not make their requirements at the beginning of the process. They do this thing during the process. Therefore, requirements are not systematic. At the end of project, there will not a virtual
requirement document and a design plan with agile methods. Software is developed in iterations. After each of the iterations, the design plan document is replaced with new one which is different from the previous one. Hence, an overall design plan document is missing at the end of the development process.

5. **Insufficient support for big teams**

Face to face communication is not fit for large teams. It will hard to communicate directly between teams member due to a large number of members. In this case, other communication approaches such as meetings, conferences are used in large environments. In fact, as less documentation is done in agile methods, so large teams can not understand the whole project. Therefore, in these large teams, programmers cannot get feedback in a short time.

6. **Insufficient support for teams with junior developers**

Senior developers play an important role in agile methods. There are huge gap between senior developers and junior developers. Experience is the most important thing in developing software. Senior developers often have much more experience and they can start their tasks immediately while junior developers need more time to study about what the tasks are and how to do them. Programmers which implement the tasks rapidly are required in agile methods which are impossible for junior programmers. The delay occurs because of this reason which affects all phases in the whole project.

7. **Insufficient support for distributed development environments like MNCs**

Working in a same location and face to face communication is required in agile methods. Therefore, these methods can not be used in spread out development environment like multinational companies. In these companies face-to-face communication is impossible because a project is developed by members in different locations. Customers cannot work together with other members. Real time communication technologies like Video conferences are expensive and can not use all the time. Members stay in different time zones so they can not work together in the same time. Documentation is also not supported by Agile Methods. A few documents are released in the development process. This thing is very difficult in distributed development environments. With a few documents,
members cannot understand what work has been done and where the development process is at that time. Rapid changes in design plan documents and the development process are also hard for team members to catch up.

8. **Insufficient support for large software systems**
   Overall architecture is constructed at the beginning of the project and update during the development process which is necessary while developing large software systems. These software systems are normally complicated and work cannot be done on them without a designed plan. Agile methods do not support the overall architecture. Maintenance work is required in large software systems. Maintenance process is unable due to lack of necessary documents like requirement specifications, designed plan, implementation plan.

9. **Need for customer presence during development**
   Developing software according to the requirements of customer is the responsibility of the development team. So they force the customers to participate which may not be acceptable or possible in many situations.

10. **Heavy dependence on Tools**
    Heavy dependence on tools is one of the problem while using agile methodologies. For example XP is completely based on Collective code ownership and therefore not applicable and successful without the appropriate support tools. No agile methodology is practical without such type of tools.

11. **Heavily dependence on User**
    The success of agile methodology heavily dependent on user involvement. and the success will depend on the co-operation and communication of the user.

12. **Limitations in developing Safety Critical Software**
    Quality Control Mechanisms incorporated in agile methodologies do not provide the features required for the purpose of developing critical software, so agile methodologies alone are not sufficient for developing it.

3.6 Conclusion
This chapter gives the complete picture of agile approach. Various reasons to adopt agile approaches are described in this chapter. Next chapter explains various existing agile methodologies and comparisons among different methodologies are also given.