Agile Software Development

Simran Bhatti, Vandana Tayal, Pooja Gulia

Student, Computer Science, Maharishi Dayanand University
Gurgaon, Haryana, India
sim.bhatti4@gmail.com

Student, Computer Science, Maharishi Dayanand University
Gurgaon, Haryana, India
vandana.tayal94@gmail.com

Student, Computer Science, Maharishi Dayanand University
Gurgaon, Haryana, India
pg.gulia2@gmail.com

Abstract

In software application development, agile software development (ASD) is a methodology for the creative process that predicts the need for flexibility and applies a level of soberness into the delivery of the finished product. It is about how to work together to achieve a common goal. This paper focuses on how technology team, work together to plan, build and deliver software. It is a group of software development methods in which requirements and solutions grow through fraternization between self-organizing, cross-functional teams.

Keywords: Application development, software product, agile manifesto, methodology.

I. BRIEF HISTORY

Incremental software development methods trace back to 1957.[3] In 1974, E. A. Edmonds wrote a paper that introduced an adaptive software development process.[4][5]Concurrently and independently, the same methods were developed and deployed by the New York Telephone Company's Systems Development Center under the direction of Dan Gielan.

Later, Ken Schwaber with others founded the Scrum Alliance and created the Certified Scrum Master programs and its derivatives. Schwaber left the Scrum Alliance in the fall of 2009, and founded Scrum.org.

In 2005, a group headed by Alistair Cockburn and Jim Highsmith wrote an addendum of project management principles, the Declaration of Interdependence to guide software project management according to agile software development methods.

In 2009, a movement spearheaded by Robert C Martin wrote an extension of software development principles, the Software Craftsmanship Manifesto, to guide agile software development according to professional conduct and mastery.

In 2011 the original Agile Alliance created the Guide to Agile Practices, an evolving open-source compendium of the working definitions of agile practices, terms, and elements, along with interpretations and experience guidelines from the world-wide community of agile practitioners.

The PRINCE2 project management methodology, used on many British Government projects, is being enhanced to manage projects that use Agile techniques.

II. THE AGILE MANIFESTO

In February 2001, 17 software developers met at the Snowbird resort in Utah to discuss some lightweight software development methods. They published the ***Manifesto for Agile Software Development.***

The manifesto comprises of some concepts which are described as below:

- ***Individuals and interactions***: self-organization and motivation are important, as are interactions like co-location and pair programming.
- ***Working software***: working software is more useful and welcome than just presenting documents to clients in meetings.
- ***Customer collaboration***: requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- ***Responding to change***: agile methods are focused on quick responses to change and continuous development.

III. THE AGILE PRINCIPLES

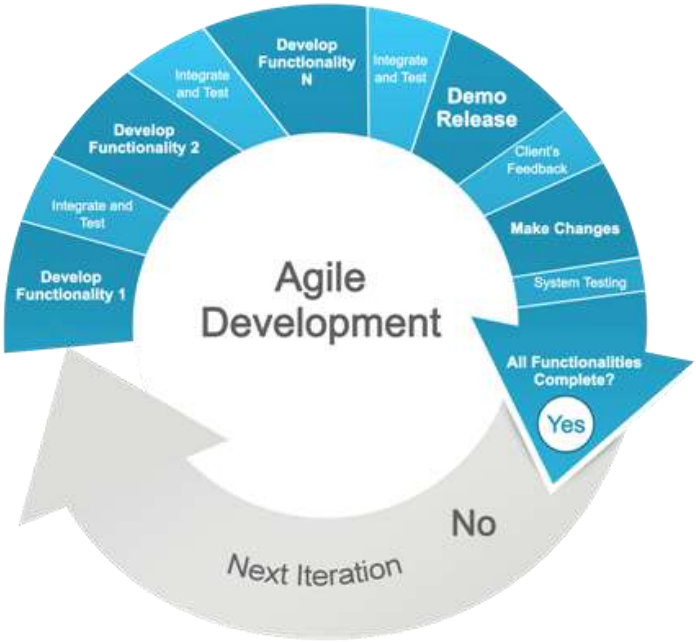
The Agile Manifesto is based on 12 principles:

1. Customer satisfaction by rapid delivery of useful software.
2. Welcome changing requirements, even late in development.
3. Working software is delivered frequently (weeks rather than months).
4. Close, daily cooperation between business people and developers.
5. Projects are built around motivated individuals, who should be trusted.
6. Face-to-face conversation is the best form of communication (co-location).
7. Working software is the principal measure of progress.
8. Sustainable development, able to maintain a constant pace.
9. Continuous attention to technical excellence and good design.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. Self-organizing teams.
12. Regular adaptation to changing circumstance.

- **Why Agile?**

Agile development provides opportunities to assess the direction throughout the development lifecycle. This is achieved through regular cadences of work, known as Sprints or iterations, at the end of which teams must present a potentially shippable product increment. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology is described as "iterative" and "incremental." In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development — requirements, design, etc. — is continually revisited. When a team stops and re-evaluates the direction of a project every two weeks, there's time to steer it in another direction. This "inspect-and-adapt" approach to development reduces development costs and time to market. Because teams can develop software at the same time they're gathering requirements, "analysis paralysis" is less likely to impede a team from making progress. And because a team's work cycle is limited to two weeks, stakeholders have recurring opportunities to calibrate releases for success in the real world. Agile development helps companies build the right product. Agile development preserves a product's critical market relevance and ensures a team's work doesn't wind up on a shelf, never released.

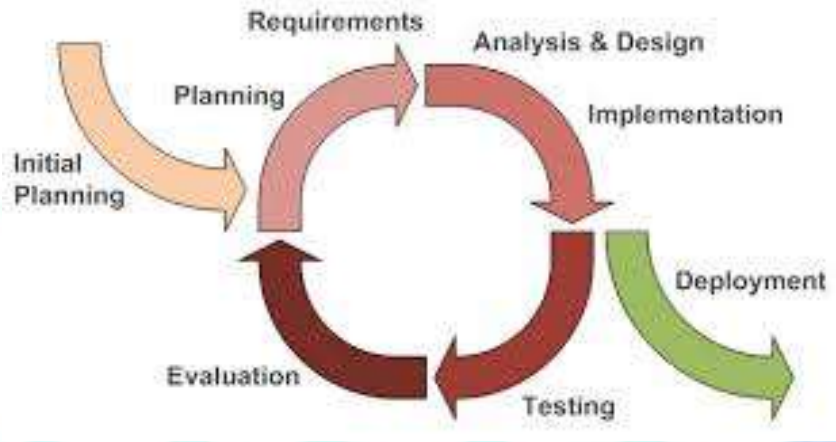
IV. AGILE DEVELOPMENT - DIAGRAMATIC REPRESENTATION



- **Agile Phases :**

The 5 phases we will discuss here are feasibility, planning, development, adapting, and deployment:

1. The ***feasibility*** phase is used to determine if an idea has enough merit to carry forward planning. An individual or small group will inspect the idea for customer value, company value, and risk.
2. If an idea is viable it will proceed to ***planning***. The project team will be assembled at this time and the team will work together to identify features. Features will be examined for value and risk and eventually estimated so they can be assigned to an iteration plan.
3. ***Development*** iterations convert the iteration plan into working code. Features are built, tested, and demonstrated to the customer and stakeholders at the end of each and every iteration.
4. The team ***adapts*** between development iterations. Customer feedback is used to adjust the plan for the forthcoming iteration.
5. When all iterations are complete the team ***deploys*** the working code into a production environment.



V. SOME AGILE PITFALLS

Organizations and teams implementing agile development often face difficulties transitioning from more traditional methods such as waterfall development, such as teams having an agile process forced on them.These are often termed ***agile anti-patterns*** or more commonly ***agile smells***. Below are some common examples:

- Lack of overall project design
- Adding stories to a sprint in progress
- Lack of sponsor support
- Insufficient training
- Product owner role is not properly filled
- Teams are not focused
- Excessive preparation/planning

- Problem-solving in the daily scrum
- Assigning Tasks
- Scrum master as a contributor
- Lacking test automation
- Allowing technical debt to build up
- Attempting to take on too much in a sprint
- Fixed time, resources, scope and quality

VI. REFERENCES

- http://agilemethodology.org/
- http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t
- http://en.wikipedia.org/wiki/Agile_software_development
- www.agile-developmenttools.com