Rajesh kumar

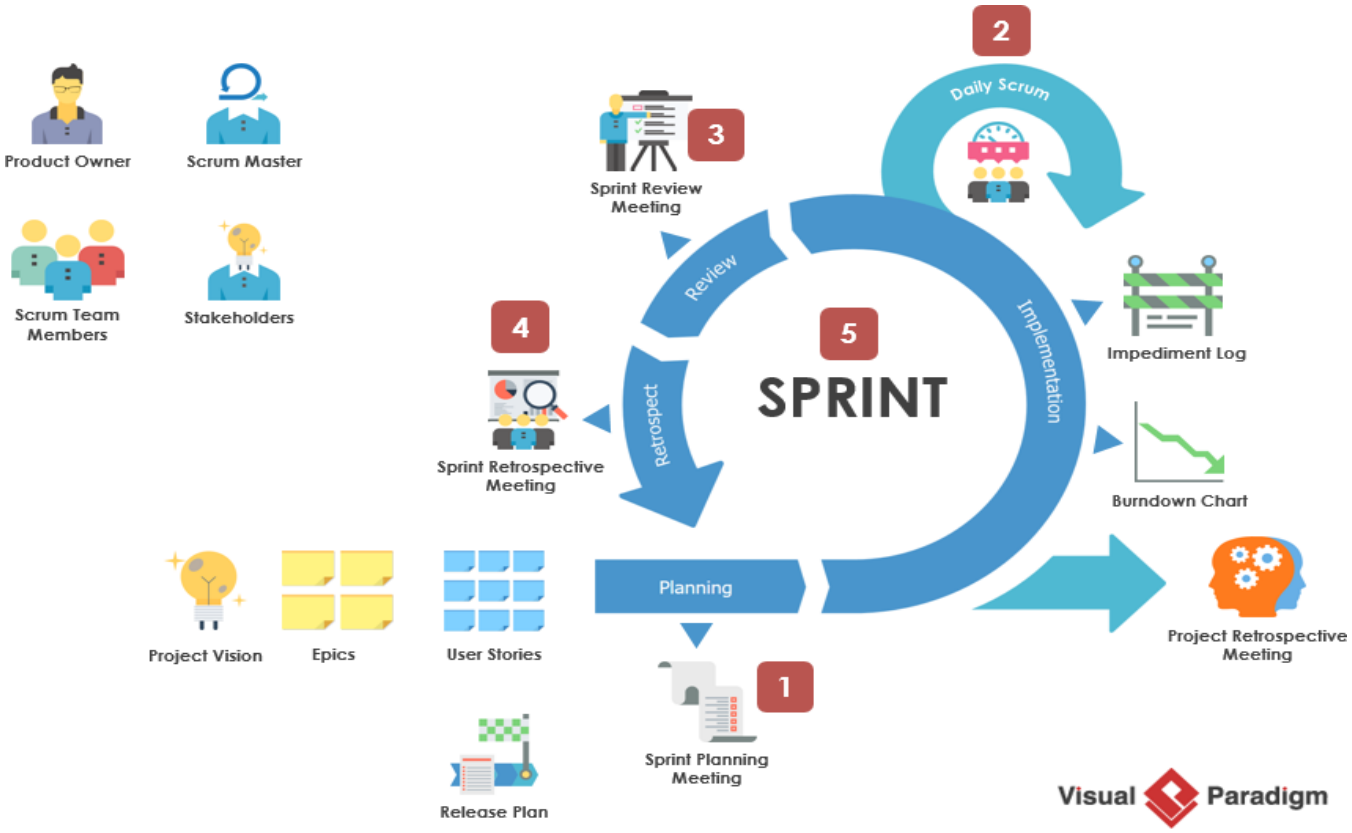
2017UCO2570

Q. Write short notes on following

- **Scrum**
- **Lean Development**
- **Extreme programming (XP)**
- **Adaptive Software Development (ASD)**
- **Feature Driven Development**

Scrum

A very popular method that borrows its title from the rugby scrum, and uses it as a metaphor for the daily progress update meeting. Scrum has short iterations (sprints) that each focus on delivering working software, a tightly prioritized ‘backlog’ for both the sprint and the product, and specifies a ‘Product Owner’ role who sets the priorities.



scrum master is the team role responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use.

product owner is a role on a product development team responsible for managing the product backlog in order to achieve the desired outcome that a product development team seeks to accomplish.

product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.

Sprint Planning Meeting is held at the start of every sprint to decide on the product backlogs etc.

Daily Scrum is a brief meeting to help the team stay on track.

Sprint Review Meeting is held at the end of every sprint to show the accomplishments of the team during the sprint.

Sprint Retrospective is held at the end of the sprint in which the team reflect on how well the scrum is working for them and what changes they might wish to see in the scrum to make it work better.

Extreme programming

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

Values

The five values of XP are communication, simplicity, feedback, courage, and respect and are described in more detail below.

Communication

Software development is inherently a team sport that relies on communication to transfer knowledge from one team member to everyone else on the team. XP stresses the importance of the appropriate kind of communication – face to face discussion with the aid of a white board or other drawing mechanism.

Simplicity

Simplicity means “what is the simplest thing that will work?” The purpose of this is to avoid waste and do only absolutely necessary things such as keep the design of the system as simple as possible so that it is easier to maintain, support, and revise. Simplicity also means address only the requirements that you know about; don’t try to predict the future.

Feedback

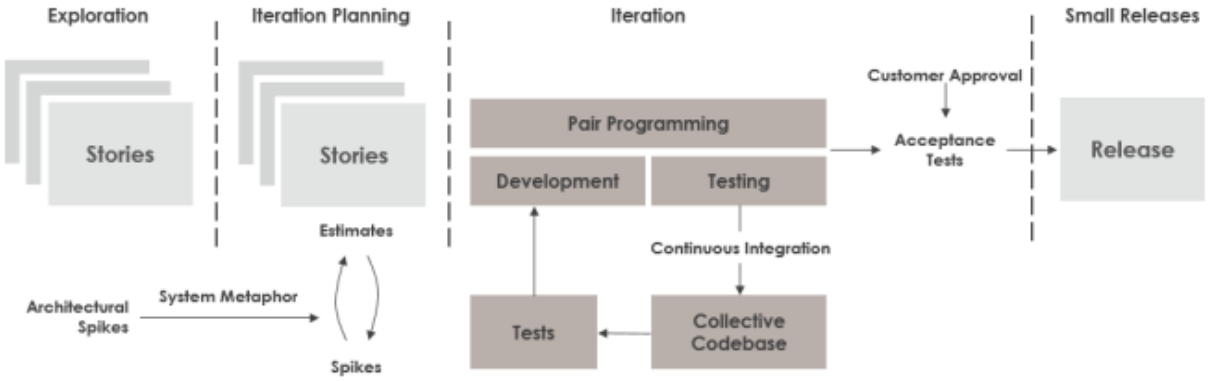
Through constant feedback about their previous efforts, teams can identify areas for improvement and revise their practices. Feedback also supports simple design. Your team builds something, gathers feedback on your design and implementation, and then adjust your product going forward.

Courage

Kent Beck defined courage as “effective action in the face of fear” (Extreme Programming Explained P. 20). This definition shows a preference for action based on other principles so that the results aren’t harmful to the team. You need courage to raise organizational issues that reduce your team’s effectiveness. You need courage to stop doing something that doesn’t work and try something else. You need courage to accept and act on feedback, even when it’s difficult to accept.

Respect

The members of your team need to respect each other in order to communicate with each other, provide and accept feedback that honors your relationship, and to work together to identify simple designs and solutions.



Lean software development is a set of principles that can be applied to software development to decrease programming effort, budgeting, and defect rates by one third. The principles were adapted from lean manufacturing by Mary and Tom Poppendieck. This approach is beneficial to an organization because agile iterations eliminate extensive pre-planned specifications. User stories rather than large upfront specs are easily understood by each team member and simpler to communicate.

Lean Development

Lean development makes it possible to gain information straight from the source, therefore eliminating the common problem of producing software that does not address the customers' needs. Short iterations provide an opportunity to communicate small sets of plans up front and allow the team to make decisions in order to adapt to unforeseen circumstances. Organizations that have the ability to complete fast, simple improvements in the shortest time frame gain powerful decision-making benefits.

Lean development methodology principles can be applied in any IT environment for improved programming practices. The practice is based on seven principles:

- Waste Elimination
- Amplifying Learning
- Late Decision Making
- Fast Delivery
- Team Empowerment
- Built-in Integrity
- View Applications as a Whole

Feature Driven Development

Feature-Driven Development (FDD) is one of the agile processes not talked or written about very much. Often mentioned in passing in agile software development books and forums, few actually know much about it. However, if you need to apply agile to larger projects and teams, it is worthwhile taking the time to understand FDD a little more

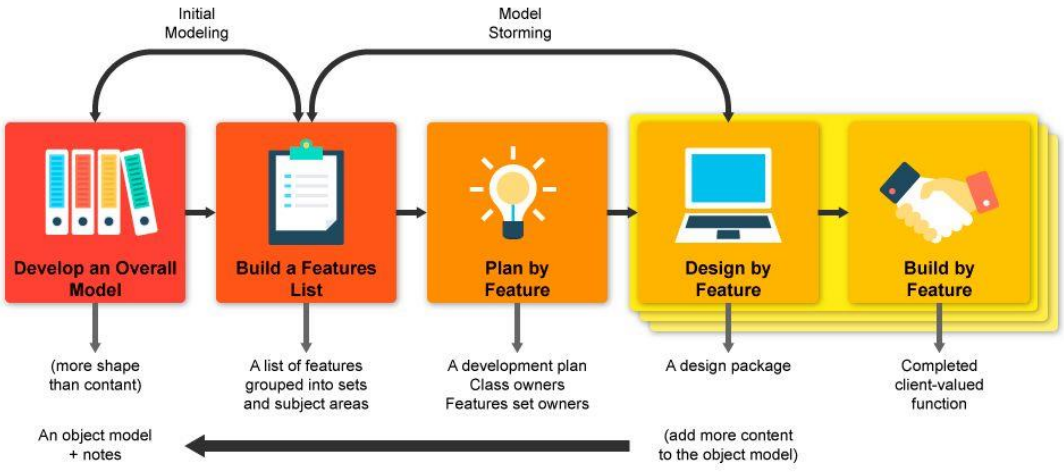
The natural habitat of Scrum and XP-inspired approaches is a small team of skilled and disciplined developers. It remains a significant challenge to scale these approaches to larger projects and larger teams. Some have been successful but many have struggled.

Feature-Driven Development (FDD) invented by Jeff De Luca is different. While just as applicable for small teams, Jeff designed FDD from the ground up to work for a larger team. Larger teams present different challenges. For example, a small team of disciplined and highly skilled developers by definition is likely to succeed regardless of which agile method they use. In contrast, it is unrealistic to expect that everyone in a larger team is equally skilled and disciplined. For this and

other reasons, FDD makes different choices to Scrum and XP in a umber of areas.

Adaptive Software Development (ASD)

Adaptive Software Development (ASD) is a direct outgrowth of an earlier agile framework, Rapid Application Development (RAD). It aims to enable teams to quickly and effectively adapt to changing requirements or market needs by evolving their products with lightweight planning and continuous learning. The ASD approach encourages teams to develop according to a three-phase process: speculate, collaborate,



learn.

Strengths and Weakness of ASD

ASD’s strengths include:

- Focused on the end users, which can lead to better and more intuitive products
- Allows for on-time and even early delivery
- Encourages more transparency between developers and clients

ASD’s weaknesses include:

- Demands extensive user involvement, which can be difficult to facilitate
- Integrates testing into every stage, which can add to a project’s costs
- Emphasis on rapid iterating and continuous feedback can lead to scope creep