Write short notes on following

Lakshay Dabas

2017UCO1624

- ## Scrum

Scrum is a framework for project management that emphasises teamwork, accountability and iterative progress toward a well-defined goal. The framework begins with a simple premise: Start with what can be seen or known. After that, track the progress and tweak as necessary. The three pillars of Scrum are transparency, inspection and adaptation.

The framework, which is often part of Agile software development, is named for a rugby formation. Everyone plays a role. When it comes to product development, Scrum roles include product owner, Scrum master and Scrum development team.

Product owner: This team member serves as the liaison between the development team and its customers. The product owner is responsible for ensuring expectations for the completed product have been communicated and agreed upon.
Scrum master: This team member serves as a facilitator. The Scrum master is responsible for ensuring that Scrum best practices are carried out and the project is able to move forward.
Scrum development team: This is a group that works together for creating and testing incremental releases of the final product.

<u>The Scrum process</u>

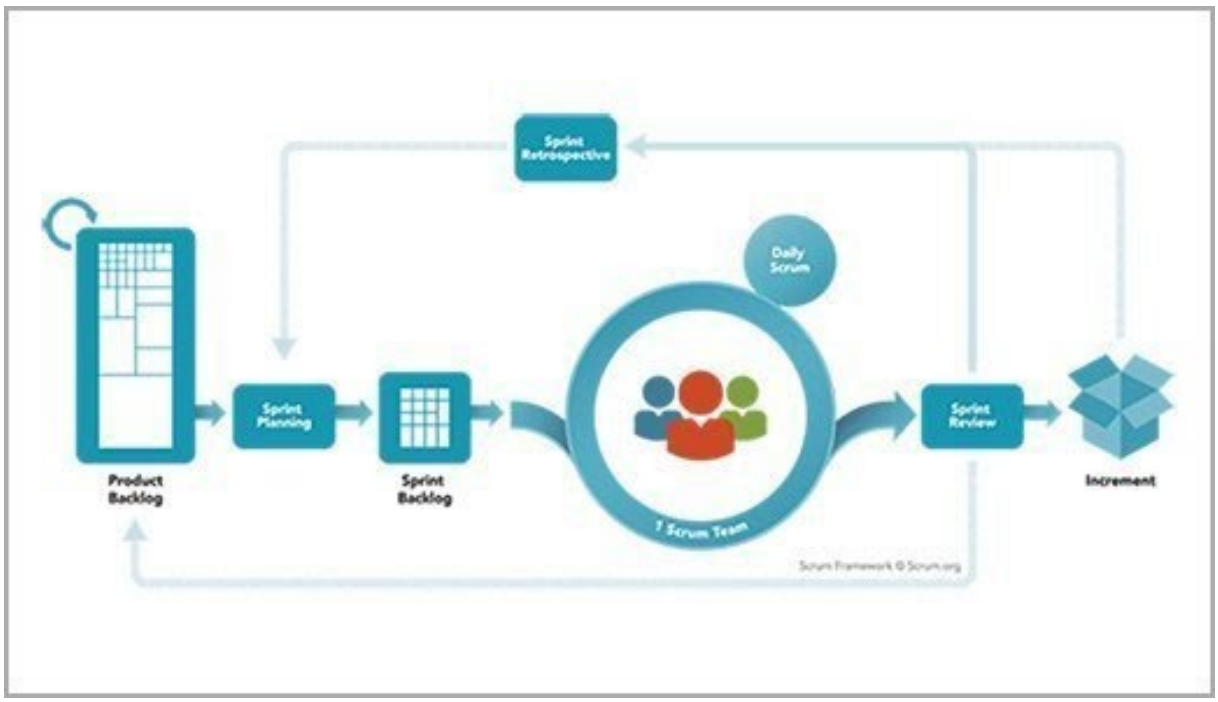
The Scrum process encourages practitioners to work with what they have and continually evaluate what is working and what is not working. Communication, which is an important part of the process, is carried out through meetings, called Events. Scrum Events include:

Daily Scrum . The Daily Scrum is a short stand-up meeting that happens at the same place and time each day. At each meeting, the team reviews work that was completed the previous day and plans what work will be done in the next 24 hours. This is the time for team members to speak up about any problems that might prevent project completion.

Sprint Planning Meeting. A Sprint refers to the time frame in which work must be completed, and it's often 30 days. Everyone participates in setting the goals, and at the end, at least one increment -- a usable piece of software -- should be produced.

Sprint Review. This is the time to show off the increment.

Sprint Retrospective. A Sprint Retrospective is a meeting that's held after a Sprint ends. During this meeting, everyone reflects on the Sprint process. A team-building exercise may also be offered. An important goal of a Sprint Retrospective is continuous improvement.



The Scrum framework shows how the elements of Scrum revolve around the Scrum team.

Scrum artifacts

An artifact is something of historical interest that deserves to be looked at again. In Scrum product development, artifacts are used to see what's been done and what is still in the queue. Scrum artifacts, which include product backlog, Sprint backlog, product increment and burn-down, are useful to look at in Sprint Planning Meetings.

Product backlog. This refers to what remains on the "to be done" list. During a product backlog grooming session, the development team works with the business owner to prioritize work that has been backlogged. The product backlog may be fine-tuned during a process called backlog refinement.

Sprint backlog. This is a list of tasks that must be completed before selected product backlog items can be delivered. These are divided into time-based user stories.

Product increment. This refers to what's been accomplished during a Sprint -- all the product backlog items -- as well as what's been created during all previous Sprints. The product increment reflects how much progress has been made.

Burn-down. The burn-down is a visual representation of the amount of work that still needs to be completed. A burn-down chart has a Y axis (work) and an X axis (time). Ideally, the chart illustrates a downward trend, as the amount of work still left to do over tim

- # Lean Development

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 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 Software Development Principles

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

Waste is defined as anything that is capable of reducing code quality, hindering time and effort, or reducing delivered business value. It can be things such as unnecessary code or functionality, programming delays, unclear requirements, and insufficient testing. Lean development focuses on eliminating these factors, learning required technologies, and gaining understanding about what the user really needs.
Additionally, by waiting until the last minute to make decisions, the cost of change remains much less. Iterative development is utilized to deliver new applications or enhancements as quick as possible. Integrity is built into the software to ensure architecture and that system components flow well together. Organizations incorporate lean development principles to achieve continuous improvement as changes are rapidly implemented.

• Extreme programming (XP)

Extreme programming (XP) is one of the most important software development framework of Agile models. It is used to improve software quality and responsive to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

Good practices needs to practiced extreme programming: Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team come up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop a good quality software. So, everybody should design daily.
- **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic principles of Extreme programming: XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User story is a conventional

description by the user about a feature of the required system. It does not mention finer details such as the different scenarios that can occur. On the basis of User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some feature. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using XP model are given below:

- **Coding:** The concept of coding which is used in XP model is slightly different from traditional coding. Here, coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system and choosing among several alternative solutions.

- **Testing:** XP model gives high importance on testing and considers it be the primary factor to develop a fault-free software.

- **Listening:** The developers needs to carefully listen to the customers if they have to develop a good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, it is desirable for the programmers to understand properly the functionality of the system and they have to listen to the customers.

- **Designing:** Without a proper design, a system implementation becomes too complex and very difficult to understand the solution, thus it makes maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasised.

- **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.

- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in present time, rather than trying to build something that would take time and it may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.

Applications of Extreme Programming (XP): Some of the projects that are suitable to develop using XP model are given below:

- **Small projects:** XP model is very useful in small projects consisting of small teams as face to face meeting is easier to achieve.

- **Projects involving new technology or Research projects:** This type of projects face changing of requirements rapidly and technical problems. So XP model is used to complete this type of projects.

• Adaptive Software Development (ASD)

Adaptive Software Development is a method to build complex software and system. ASD focuses on human collaboration and self-organisation. ASD **“life cycle”** incorporates three phases namely:

1. Speculation
2. Collaboration
3. Learning

These are explained as following below.

1. **Speculation:**
   During this phase project is initiated and planning is conducted. The project plan uses project initiation information like project requirements, user needs, customer mission statement etc, to define set of release cycles that the project wants.

2. Collaboration:
It is the difficult part of ASD as it needs the workers to be motivated. It collaborates communication and teamwork but emphasizes individualism as individual creativity plays a major role in creative thinking. People working together must trust each others to

- Criticize without animosity,
- Assist without resentment,
- Work as hard as possible,
- Possession of skill set,
- Communicate problems to find effective solution.

3. Learning:
The workers may have a overestimate of their own understanding of the technology which may not lead to the desired result. Learning helps the workers to increase their level of understanding over the project.
Learning process is of 3 ways:

1. Focus groups
2. Technical reviews
3. Project postmortem

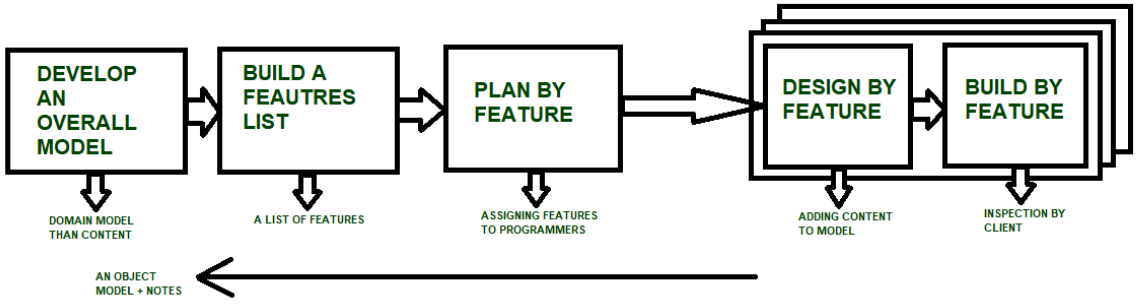
ASD's overall emphasis on the dynamics of self-organising teams, interpersonal collaboration, and individual and team learning yield software project teams that have a much higher likelihood of success.

• Feature Driven Development

It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive os feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels.

FDD Lifecycle

- Build overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature



Characteristics of FDD

- **Short iterative:** FDD lifecycle works in simple and short iterations to efficiently finish the work on time and gives good pace for large projects.
- **Customer focused:** This agile practice is totally based on inspection of each feature by client and then pushed to main build code.
- **Structured and feature focused:** Initial activities in lifecycle builds the domain model and features list in the beginning of timeline and more than 70% of efforts are given to last 2 activities.
- **Frequent releases:** Feature-driven development provides continuous releases of features in the software and retaining continuous success of the project.

Advantages of FDD

- Reporting at all levels leads to easier progress tracking.
- FDD provides continuous success for larger size of teams and projects.
- Reduction in risks is observed as whole model and design is build in smaller segments.
- FDD provides greater accuracy in cost estimation of the project due to feature segmentation.

Disadvantages of FDD

- This agile practice is not good for smaller projects.
- There is high dependency on lead programmers, designers and mentors.
- There is lack of documentation which can create an issue afterwards.