Write short notes on the following -

- Rahul Meena

- 2017UCO1598

**A. Scrum**

Scrum is the type of Agile framework. It is a framework within which people can address complex adaptive problems while productivity and creativity of delivering product is at highest possible values. Scrum uses Iterative process.

Life cycle of scrum proceeds as follows -

1. A Scrum team is formed. It usually consists of 5-9 people.Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint.

2. Product owner: The product owner is often someone from product management or marketing, a key stakeholder or a key user.

3. Scrum master: The Scrum Master is responsible for making sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside, and so on.

4. Product backlog: According to the prioritize features the product is organized.

5. Sprint planning meeting: At the start of each sprint, a sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team. The Scrum team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog,

6. Daily Scrum: Each day during the sprint, a brief meeting called the daily scrum is conducted. This meeting helps set the context for each day’s work and helps the team stay on track.

7. Sprint review meeting: If the product still has some non-achievable features then it will be checked in this stage and then the product is passed to the Sprint Retrospective stage.

8. Sprint retrospective: In this stage quality or status of the product is checked.

9. Sprint: A Sprint is a time-box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.

**B. Lean Development**

Lean Software Development (LSD) is an agile framework based on optimizing development time and resources, eliminating waste, and ultimately delivering only what the product needs.

The seven Lean principles are:

1. Eliminate waste - One of the key elements of practicing Lean is to eliminate anything that does not add value to the customer.
2. Build quality in

1. Create knowledge - This principle encourages Lean teams to provide the infrastructure to properly document and retain valuable learning. This can be done by using Pair programming, Code reviews, Training, Documentation etc.
2. Defer commitment - This Lean principle encourages team to demonstrate responsibility by keeping their options open and continuously collecting information, rather than making decisions without the necessary data.
3. Deliver fast
4. Respect people - The Lean principle of Respect for People is often one of the most neglected, especially in the fast-paced, burnout-ridden world of software development. It applies to every aspect of the way Lean teams operate, from how they communicate, handle conflict, hire and onboard new team members, deal with process improvement, and more.
5. Optimize the whole - optimizing the whole is a Lean development principle that encourages Lean organizations to eliminate these sorts of vicious cycles by operating with a better understanding of capacity and the downstream impact of work.

**C. Extreme programming (XP)**

Extreme programming (XP) is one of the most important software development frameworks 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.A few of the basic steps followed in Software Development using XP are mentioned 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 the primary factor to develop a fault-free software.

**Listening:** The developers need 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 the functionality of the system properly 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 in the elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.

**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 the 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.

**D. Adaptive Software Development (ASD)**

ASD was created to replace traditional Waterfall variants. In a general sense, ASD processes represent a repeating series of **speculate, collaborate and learn cycles.**

**Speculate** refers to the planning paradox—outcomes are unpredictable, therefore, endless suppositions on a product’s look and feel are not likely to lead to any business value. Some important dimensions are more than likely wrong. Hypothesize a rough idea of what you need and plan to adapt. The big idea behind ‘speculate’ is when we plan a product to its smallest detail as in a requirements up front Waterfall variant, we produce the product we intend and not the product the customer needs.

**Collaboration** portrays a balance between managing the doing and creating and maintaining the collaborative environment. Speculation says we can’t predict outcomes. If we can’t predict outcomes, we can’t plan. If we can’t plan, traditional project management theory suffers. Collaboration counterpoises speculation in that a project manager plans the work between the predictable parts of the environment and adapts to the uncertainties of various factors—stakeholders, requirements, software vendors, technology, etc.

**Learning** cycles challenge all stakeholders and project team members. Based on short iterations of design, build and testing, knowledge accumulates from the small mistakes we make due to false assumptions, poorly stated or ambiguous requirements or misunderstanding the stakeholders’ needs. Correcting those mistakes through shared learning cycles leads to greater positive experience and eventual mastery of the problem domain.

**E. Feature Driven Development**

An Agile methodology for developing software, Feature-Driven Development (FDD) is customer-centric, iterative, and incremental, with the goal of delivering tangible software results often and efficiently. FDD in Agile encourages status reporting at all levels, which helps to track progress and results.

As with all Agile methodologies, the first step in FDD is to gain an accurate understanding of content and context of the project, and to develop a clear, shared understanding of the target audience and their needs.

1. Develop an overall model

This stage is when the outline is drafted. Using the primary goal as a guide, the team will develop detailed domain models, which will then be merged into one overall model that acts as a rough outline of the system. As it develops and as the team learns, details will be added.

1. Build a features list

Use the information assembled in the first step to create a list of the required features.Make a list of features (that can be completed in two weeks’ time), and keep in mind that these features should be purposes or smaller goals, rather than tasks.

3. Plan by Feature

Analyze the complexity of each feature and plan tasks that are related for team members to accomplish. During the planning stage, all members of the team should take part in the evaluation of features with the perspective of each development stage in mind. Then, use the assessment of complexity to determine the order in which each feature will be implemented, as well as the team members that will be assigned to each feature set.This stage should also identify class owners, individual developers who are assigned to classes.

4. Design by Feature

A chief programmer will determine the feature that will be designed and built. He or she will also determine the class owners and feature teams involved, while defining the feature priorities. Part of the group might be working on technical design, while others work on framework. By the end of the design stage, a design review is completed by the whole team before moving forward.

5. Build by Feature

This step implements all the necessary items that will support the design. Here, user interfaces are built, as are components detailed in the technical design, and a feature prototype is created. The unit is tested, inspected and approved, then the completed feature can be promoted to the main build.

6. Conclusion

Feature-Driven Development is a practical Agile approach suited for long-term, complex projects. It is a suitable choice for development teams seeking a simple but structured Agile method that is scalable and delivers predictable results.