CED12   
Emerging Paradigms in Programming

Assignment



Name- Mohit Yadav

Roll No- 2017UCO1656

# 

# Q. Write short notes on following :

## # Scrum

**Scrum** is a type of **Agile Framework.** It encourages teams to learn through experiences, self-organize while working on a problem, and introspect about the achievements and shortcomings to continuously improve. Scrum like agile is an **iterative process** which prioritizes high **productivity** and **swift development** of application.

**Scrum keywords**

1. **Increment** : Increment or Sprint-Goal is the working end product of each sprint. It is presented to the user for thorough testing and reviews.
2. **Product Backlog** : It is a dynamic list that enlists all the functional and nonfunctional requirements of the product. It is generally maintained by the product owner or manager.
3. **Sprint Backlog** : It is the list of items that are selected for implementation in the current sprint. It is compiled before each sprint in the sprint planning meeting with the help of the whole team.

**Lifecycle of scrum**

1. **Product Backlog Meeting** : In this meeting the product backlog list is updated and reorganised according to priority.
2. **Sprint planning meeting** : In this meeting the team decides the sprint backlog meeting that enlists the things that need to be achieved in the following sprint.
3. **Implementation** : This is the time period within which the scrum team works together to implement the increment. It usually spans from 2 weeks to a month.
4. **Sprint Review** : At the end of the sprint, the team gets together for an informal session to view a demo of, or inspect, the increment. The development team showcases the backlog items that are now ‘Done’ to stakeholders and teammates for feedback. The product owner can decide whether or not to release the increment
5. **Release** : If the product is complete it is released to production.
6. **Retrospection** : In this stage quality or status of the product is checked and all the achievements and shortcomings of the increment are discussed.

## # Lean Development

**Lean Software Development** is an **Agile** framework based on optimizing development time and resources, eliminating waste, and ultimately delivering only what the product needs. The Lean approach is also often referred to as the Minimum Viable Product (MVP) strategy, in which a team releases a bare-minimum version of its product to the market, learns from users what they like, don’t like and want to be added, and then iterates based on this feedback.

**Lean Principles**

1. **Eliminate waste** : Lean philosophy regards everything not adding value to the customer as waste and aims to remove any of that waste. Such waste may include:
   1. Partially done work
   2. Extra features
   3. Relearning
   4. Task switching
   5. Waiting
   6. Handoffs
   7. Defects
   8. Management activities
2. **Build quality** : Lean principles aim to make sure that the quality of product in terms of both functional and nonfunctional requirements be excellent.
3. **Create Knowledge** : This principle encourages Lean teams to provide the infrastructure to properly document and retain valuable learning.
4. **Defer commitment** : This Lean principle encourages teams to demonstrate responsibility by keeping their options open and continuously collecting information, rather than making decisions without the necessary data.
5. **Deliver Fast** : This principle is in the core of agile development. We aim to deliver the application as fast as possible.
6. **Optimizing the whole** : Modern software systems are not simply the sum of their parts, but also the product of their interactions. So instead of suboptimizing the parts, the lean principle aims to optimize the product as whole.

## 

## # Extreme programming (XP)

**Extreme programming** is another agile development framework which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

**Activities in XP**

1. **Coding** : This is the most and sometimes the only important part of product implementation. It involves writing down program that a computer can interpret and produce the desired results
2. **Testing** : It is the process of mocking the use cases of product and analyzing how it performs. We use unit and acceptance tests here.
3. **Listening** : Process of getting information regarding the requirements from users by listening to them.
4. **Designing** : If the system is too complex we can use this step to organize and plan the coding step in XP.

**Values in XP**

1. **Communication** : Building software systems requires communicating system requirements to the developers of the system. Also it is imperative that the developers are also communicating between themselves.
2. **Simplicity** : XP encourages starting with simplest solution because a simple design with very simple code could be easily understood by most programmers in the team.
3. **Feedback** : Feedback is essential and can be taken from System ( performance feedback though unit tests ), Users and also the team.
4. **Courage** : Several practices embody courage. One is the commandment to always design and code for today and not for tomorrow. This is an effort to avoid getting bogged down in design and requiring a lot of effort to implement anything else.
5. **Respect** : The respect value includes respect for others as well as self-respect. Programmers should never commit changes that break compilation, that make existing unit-tests fail, or that otherwise delay the work of their peers. Members respect their own work by always striving for high quality and seeking for the best design for the solution at hand through refactoring.

## # Adaptive Software Development (ASD)

**Adaptive Software Development** is a method to build complex software and systems. ASD focuses on human collaboration and self-organisation.

**Phases of ASD**

1. **Speculation** : During this phase the project is initiated and planning is conducted. The project plan uses project initiation information like project requirements, user needs etc to define the increments.
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.
3. **Learning** : The workers may have an 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.

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

## 

## # Feature Driven Development

**Feature-driven development (FDD)** is an iterative and incremental software development process. It is a lightweight or Agile method for developing software. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are driven by user or client based functional and nonfunctional requirements.

**Activities in FDD**

1. **Develop overall model** : The FDD project starts with a high-level walkthrough of the scope of the system and its context. Next, detailed domain models are created for each modelling area by small groups and presented for peer review. One or more of the proposed models are selected to become the model for each domain area. Domain area models are progressively merged into an overall model.
2. **Build feature list** : Using user stories and experiences we build a list of use cases and features that application needs to offer. We usually break these features in action, result and object to better describe them.
3. **Plan by feature** : After the feature list is completed, the next step is to produce the development plan and assign ownership of features as classes to programmers.
4. **Design by feature** : Design documentation is laid down for each feature and some of these features are selected for implementation and their detailed sequence diagram is modeled.
5. **Build by feature** : After a successful design inspection for each activity to produce a feature is planned, the class owners develop code for their classes. After unit testing and successful code inspection, the completed feature is promoted to the main build.