2017UCO1697

Saksham Bansal

COE - 3

***Write short notes on following***

***Scrum***

***· Lean Development***

***· Extreme programming (XP)***

***· Adaptive Software Development (ASD)***

***· Feature Driven Development***

**Scrum**

Scrum is a framework meant to promote team collaboration on building complex products. More specifically, it is a process framework that is used for managing product development as well as other knowledge work. Overall, scrum is meant to be a lightweight framework that is simple to understand yet it can be difficult to master.

Scrum framework is best suited when there is a significant amount of work that can be split into multiple 2-4 week iterations. There are five core values for scrum that are commitment, course, opens, focus and respect and all team members are expected to learn these values. The scrum principles, practices, artifacts, roles and life cycle are described in more detail below.

Scrum is based on three principles that are as follows:

1. Transparency - this is important so that everyone is aware of the latest issues that might be facing other team members.
2. Inspection - This is required so that the team can reflect and think on how the process has been working. This is done through the daily scrum meeting and the sprint review meeting.
3. Adaption - The team constantly adapts and revises the plan removing things that are not required or do not make sense.

The practices that are done as a part of the scrum framework are described below.

* Sprint - This is usually a timer-period of one month during which the team will potentially ship a product.
* Sprint planning- This is used to plan out the items from the product backlog that will be worked upon during the sprint. The result of this is a sprint backlog.
* Daily scrum - This is a short meeting (usually less than 15 minutes) where the team will coordinate their tasks and activities for the day.
* Sprint review - This is meant to review the final product of the sprint with the various stakeholders that might be involved. This is meant to demonstrate and give the various stakeholders a chance to see the product and get some feedback. The feedback is then placed in the product backlog for future consideration.
* Sprint retrospective - At the end of the sprint it is important that the team go over the effectiveness of the sprint and any adjustments that could be made moving forward.

The following artifacts are part of the scrum:

* Product backlog - The product backlog is a list of the changes that could be made to a given product. These are just meant to be options i.e. they are not guaranteed to be delivered.
* Sprint backlog - These are the product items from the product backlog that have been selected for delivery during the sprint.
* Increment - These are the items from the product backlog that are completed by the end of the sprint. The product owner can further decide whether to release these or further build upon in the future sprints.

The team members have the following roles in scrum:

* Product owner - it is a team role that is responsible for managing the product backlog, so that the team can achieve the desired outcomes as planned during the sprint. This role exists to address the challenges that the product development team has with what to build.
* Scrum master - is a role that is responsible to ensure that the team lives by its values and principles and is following the practices that were agreed upon during the sprint.
* Development team - as the name suggests this consists of people who will deliver the product increment inside a Sprint. Their main responsibility is to deliver the product that adds value in every Sprint.

Scrum is a framework that allows flexibility and adaptability in responding to changes. It has the following lifecycle that starts with prioritising the backlog, but does not provide any guidance on how to do so. Below are the key steps in the life cycle:

1. Establish the product backlog
2. Sprint planning is then conducted by the product owner and the development team
3. The development team then starts doing the necessary work for delivering the items on the product backlog.
4. The development team communicates their daily work on daily scrum.
5. At the end of the Sprint the team should be able to deliver the items selected for the sprint. These items are shown to the customer during the sprint review. The team also reflects on how the sprint has proceeded so far and adapting any processes required to be more effective in the future.
6. The team repeats the steps 2-5 until the final product is completed.

**Lean Development**

Lean software development is an agile software framework that is used for optimising the time and resources for the development of a product. It focuses on waste elimination and therefore, is often referred to as the minimum viable product (MVP) strategy i.e. a team would release a very basic product to the market and learn from the feedback of the user and then continue to add features based on the feedback.

Lean development borrows its core principles and philosophy from the manufacturing industry which used it to optimise their own production and assembly lines in order to minimise waste and maximise productivity. The seven lean development principles are described below:

1. Eliminate waste - This means to eliminate anything that does not add value for the customer. There can be several types of waste that can be eliminated such as over-production, unnecessary transportation, inventory etc.
2. Build quality in - Quality is hard to build in a product because in the process many teams end up creating waste for example through excessive testing. There are some tools that are used with lean development to ensure quality such as pair programming, test-driven development etc.
3. Create knowledge - This focuses on making sure that teams can properly document and retrain their learning over time, so that it can be used as a future reference. The tools that are used for this are pair programming, code reviews, documentation, wiki etc.
4. Defer commitment - This simply encourages teams to delay their decision if the necessary data for making the decisions is not available at the current time and further encourages them to keep their options open i.e. not plan for months in advance, not commit to ideas without much understanding etc.
5. Deliver fast - One of the main reasons teams are not able to deliver fast is because they plan too far in advance and create a lot of waste in the process of doing so. Lean development on the other hand is focused on building only a simple solution, putting it in front of customers and improving it incrementally based on customer feedback.
6. Respect people - This principle applies to all lean teams in the way they operate, communicate, deal with conflict, hire new members etc. Lean encourages teams to respect people through communicatively effectively, encouraging healthy conflict and trying to empower each other to do their best work.
7. Optimize the whole - This principle addresses a serious issue of trying to avoid the vicious cycle of suboptimization. This can happen when developers tend to sacrifice quality for speed which can increase the complexity of the code base resulting in more defects and more work to do for the developers. This leads to more pressure on the developers and the cycle continues.

**Extreme Programming**

Extreme programming is an agile software development methodology which is used to create higher quality software as well as provide better quality of life for the development team. Extreme programming is applicable in a variety of situations but is most commonly used when software requirements are dynamically changing, to mitigate risks caused by time constrained projects with new technology, and the technology that is being used allows for adding unit and functional tests.

There are five core values associated with extreme programming that are simplicity, feedback, respect, communication, and courage which are described below:

1. Communication - Communication is an important part of any software development project for the flow of knowledge from one team member to another. Extreme programming further stresses the importance of communication in software development. This can be done face to face with another person additionally, using the aid of a board or any other tool that might be available.
2. Simplicity - The second core value of XP is simplicity. In extreme programming we try to find out the simplest thing that would work. This is because we want to avoid wasting time and only work on things that are absolutely critical such as by keeping the architecture of the system as simple as possible in order to avoid unnecessary burdens and hassle during its maintenance, operation and support. Simplicity also implies that one should focus only on the requirements that are known and not on adding features that might not be required.
3. Feedback - Another important value of extreme programming is feedback. Feedback is important because it allows team members to identify their weaknesses and areas of improvement, so that they can constantly work to improve themselves. Simplicity and feedback play an important role together. Once your team has built something, they can gather feedback regarding their work and then adjust the product according to it.
4. Courage - The fourth value associated with extreme programming is courage which is the ability to take action even in the face of fear. Courage is important because otherwise oftentimes inaction can lead to decline in effectiveness of the team. In extreme programming there is preference for action because it allows to raise appropriate concerns and then act upon them. For example courage is required to accept the feedback that is received even though it might be difficult to accept. Courage is further required to accept your failure and move on to try something else, so that the team can move quickly in case of a failure.
5. Respect - Respect is important between the members of any team. It allows them to communicate with each and accept feedback from each other that allows them to improve themselves and work together effectively to design simple solutions to complex problems.

**Adaptive Software Development**

Adaptive software development (ASD) has directly evolved from another agile framework known as rapid application development (RAD). The Adaptive Software development approach allows teams to develop in three-phase processes i.e. speculate, collaborate and learn.

1. Speculate - During this phase the team is encouraged to explore and experiment with different ideas and iterates continuously with feedback. This is because with complex problems there is a lot of uncertainty that needs to be acknowledged.
2. Collaborate - Building complex applications require collecting and analysing diverse information specific to the problem at hand. This results in a very broad range of requirements that can only be handled through team collaboration. Collaboration is required to jointly produce results as well as share knowledge and make collective decisions. Collaboration also deals with maintaining a balance with managing and maintaining a neutral collaborative environment that is helpful to work productively.
3. Learn - This is a crucial stage of the adaptive development process. Every team member should focus on constantly enhancing their knowledge through practices/techniques such as technical reviews, focus groups and project retrospectives. Reviews should be done regularly throughout the project. The developers need to examine their assumption and learn about how to progress forward using the results of each development cycle. Overall they learn about product changes and changes in their assumption that affect the product being developed.

**Feature Driven Development**

Feature Driven development (FDD) is a type of agile framework that organises the process of software development through the progress on features. Feature driven development has five stages in the project life cycle that are discussed below.

1. Develop the overall model

This stage is used for deciding the domain or the scope of the problem that the software product tries to solve. During this stage the team works closely to define the scope and context of the system. If there are multiple models they should be merged together into one single model.

1. Build a features list

As the name suggests this stage is used for deciding the features that should be worked upon. Features in FDD are expressed as a combination of action, result and object. Each feature should be small enough so that it can be developed under two weeks. If a feature will take longer than that it should be broken into multiple smaller features.

1. Plan by feature

This stage is used to determine the order in which each feature should be developed and implemented. This is done considering various factors such as dependencies, workload, blockers or any other obstacle that might come in the way of feature development. Each feature is then assigned to a programmer whose skill set overlaps with that required for developing the feature within the specific timeframe.

1. Design by feature

This stage is used to decide which features will be built and implemented in a two-week iteration plan. This is based on feature priorities usually decided by the chief programmer. A final design review is done by the entire team before moving forward to the final stage.

1. Build by feature

In this final stage of the feature driven development process all the items required for the support of the feature are finally implemented. The UI is designed and the backend is implemented and a feature prototype is built for testing and review. If the feature is approved it is integration with the main build and is made available to all the customers.