Write short notes on following

Scrum

Scrum is a framework that provides a template which teams can follow for any project. Some important parts in the scrum framework are as follows :

* Scrum Master : Its his/her responsibility to make sure that the scrum framework is being diligently and effectively being followed by all the members of the team and also protects the team from external situations.
* Sprint planning meeting : At the beginning of each Sprint to plan out the tasks that will be completed by each team member in the upcoming Sprint. The scrum master leads this meeting.
* Sprint : A Sprint is a time period of around a month or less during which a certain number of tasks will have to be completed by each team member.
* Sprint review meeting : In this meet the performance of the team is evaluated, and all the team members put forth the tasks they were able to complete and the ones which were left incomplete.
* Daily Scrum : This is a 10-15 minute meet held everyday in which everyone states the work they did in the previous day and the work they’ll do today. This helps keep everyone up to date on each others tasks.

Having done a summer internship in Amazon India, I’ve personally witnessed how Scrum works and can testify that this framework indeed helps a lot to keep team members as efficient as possible.

Lean Development

Lean Development is the application of the Lean Principles to software development. Lean principles were initiated in manufacturing to optimize production lines and maximise the value to the customer. The seven Lean Development principles are -

1. Eliminate waste  
   The main idea is here is to eliminate all that does not add any value to the customer. Some things that can be removed are :
   1. Unnecessary code or functionality
   2. Unclear or constantly changing requirements
   3. Bureaucracy
   4. Task switching
   5. Partial work done
2. Build quality in  
   Building quality is something that appears obvious but is seldom implemented effectively if not taken as a disciplined practice. Some things that can be done in this regard are -
   1. Pair programming - using the skills of two developers to ensure quality
   2. Test-driven development - can be done by writing the criteria the code should meet to meet the business requirements
   3. Incremental development and constant feedback
   4. Minimize wait states
   5. Automation
3. Create knowledge  
   Creating knowledge should go hand in hand with development. This can be done by pair programming, code reviews, documentation, well commented code, training, etc
4. Defer commitment  
   This does not mean to keep postponing things and be lazy. Rather it means not to plan months in advance as requirements may change, not to commit to ideas or projects without understanding the full breadth, and to constantly collect data and information regarding any important decision.
5. Deliver fast
6. Respect for people  
   This can be done by -
   1. Communication proactively and effectively
   2. Encouraging healthy conflicts
   3. Empowering each other to do effective work
7. Optimize the whole

Extreme programming (XP)

Extreme programming is a framework that is best used when there are dynamic software requirements, small teams co existing closeby and automated unit and functional tests are possible. There are 5 main tenets of Extreme Programming :

1. Communication  
   Extreme Programming very specifically highlights the importance of Communication, especially in a situation like software development where team members sharing information with each other is crucial for the success of the team. It also stresses the importance the appropriate communication to take place : through a board sometimes, face to face, etc
2. Simplicity  
   There is no need to re-invent the wheel again and again. This value highlights the importance of keeping things simple so that the structures are easy to understand and revise and support. It also requires that you only address the problems that you are currently facing.
3. Feedback  
   Constant feedback on the previous performance of the team can help identify patterns and weak points much earlier and then improve then right now rather than discovering the deficiencies in the product later on which could be much more difficult to correct.
4. Courage  
   Courage is one of the most important values that can massively improve the outcomes. It takes courage to try something new, to listen to feedback and work on the shortcomings and it takes courage to raise concerns against the team’s shortcomings.
5. Respect  
   It is important that all the team members respect each other and their decisions in order to provide constructive criticism and to communicate effectively with each other.

Adaptive Software Development (ASD)

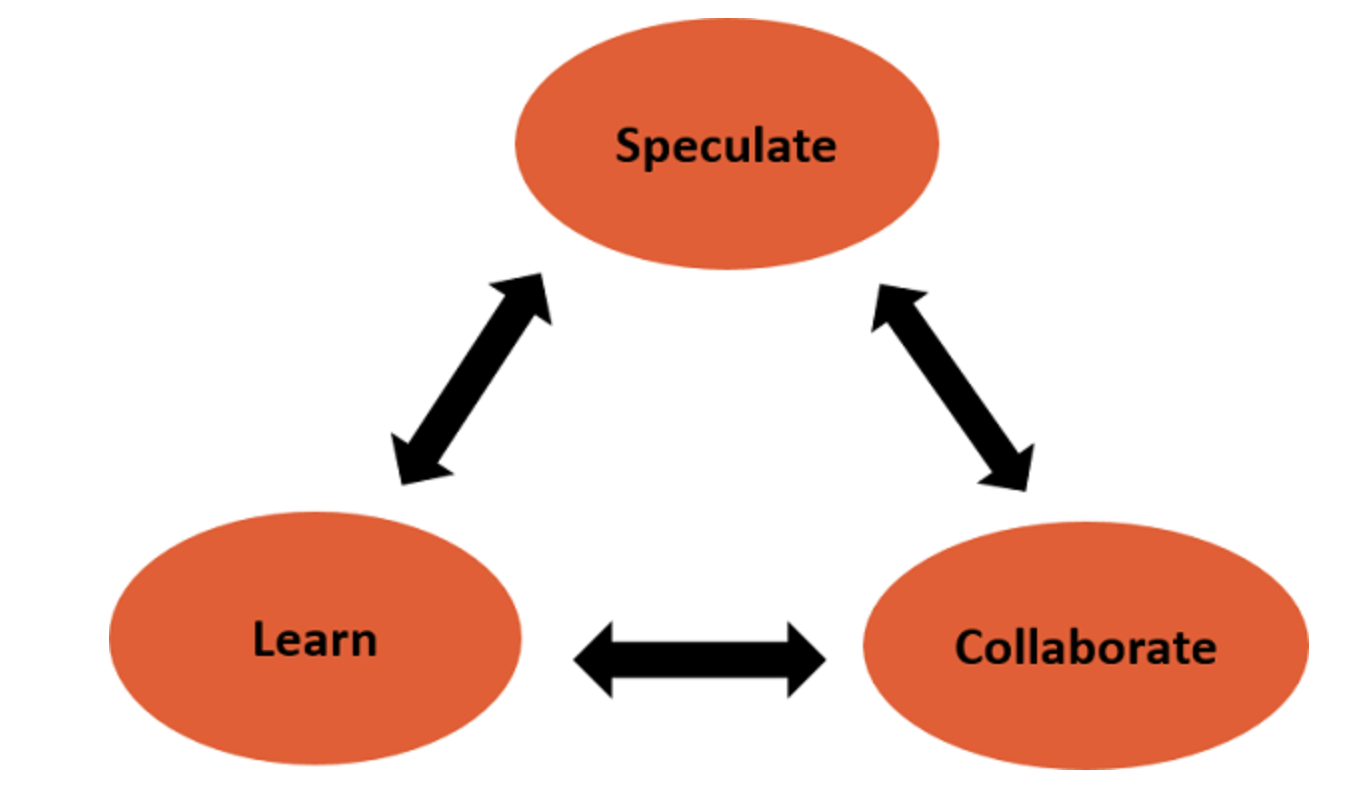
Adaptive Software Development was published in 2000 by Jim Highsmith. It is used for a wide range of applications, often large projects, where the specific outcome cannot be determined in the beginning. There are 3 phases of development in Adaptive Software Development. This framework is cyclical and all the three phases are interrelated, reflecting the complexity of a large project.The three phases are -

1. Speculate - The term speculate is deliberately used instead of planning to reflect that often the planning cannot be done right at the beginning. Planning is definitely a part of this phase, but the word speculate is used to describe the reality and uncertainty in a complex problem. In the ASD cycle, small iterations are preferred.
2. Collaborate - In a complex problem, it is imperative that all the information available to all the members is shared openly and work is divided according to the expertise of the various members and that they help each other.This requires the ability to share information, working together effectively to produce results and and to make decisions. The diverse knowledge requirements in a complex problem can only be solved by the collaboration of all the knowledge holders.
3. Learn - Small iterations in ASD give ample scope for members to learn. The team can constantly enhance its knowledge by -
   1. Technical reviews
   2. Project Retrospectives
   3. Customer Focus Groups

Regular reviews allow the customer and the developers to learn about -

1. Product changes
2. Fundamental changes in the understanding of how the product is to be developed

From looking at this Speculate-Learn-Collaborate cycle, it is clear that all of them are interrelated.



Feature Driven Development

In a Feature Driven Development model, as the name suggests, progress is marked by the number of features developed. Features do not imply what they are commonly understood to be, rather they are more like ‘stories’ in scrum. FDD is designed to follow a 5 step development process -

1. Develop an overall model
2. Build a features list
3. Plan by feature
4. Design by feature
5. Build by feature

FDD’s strengths are -

1. Simple model allows for rapid development
2. Allows larger teams to make progress with regular success
3. Leverages upon the predefined development standards, so developers can move quickly

Some weaknesses are -

1. Not suitable for smaller projects
2. Documentation is not practiced as heavily, which can lead to confusion
3. Very dependent on the head developer

This model can be used in a large company on a large project. However, everything happens in a top down fashion and individual ownership isnt promoted. The ownership lies upon the head developer alone.