Write short notes on following

Scrum

· Lean Development

· Extreme programming (XP)

· Adaptive Software Development (ASD)

· Feature Driven Development

**Answer 1: Scrum**

Scrum is a programming framework for agile software development used for development of complex programs where the program is broken down into small tasks, which can be completed in 1 month or 2 weeks, called sprints and the development in done in teams which are self-organized, meaning that there is no overall team leader, and they communicate with each other, conduct daily meetings called scrum, hence the name. At the end of every sprint the team conducts a review of the sprint, draw burnout charts, discuss progress and draw sprint retrospective.

The term “Scrum” was first coined in 1986 which was adopted from rugby where scrum is a formation. It is an evolutionary model which is a light-weight, iterative and incremental method to develop complex systems. The roles in scrum include

* product owner: who are the stakeholders of the product
* development team: it is the team of 3-9 members who are responsible for developing the product
* scrum master: who is responsible for removing obstacles of the development team.

The process of Scrum development can be broken down into the following parts:

1. Sprint Planning: The team meets prior to each sprint to discuss the scope of the sprint, the requirements, select product backlog that can be achieved in the sprint, set a sprint goal.
2. Daily Scrum: It is a daily meeting which happens at the same place, everyday without delay for 15 minutes where each member states the task completed yesterday, the task planned for today and any hindrances obstacles that may hinder development.
3. Sprint review: It is a meeting held at the end of every sprint to review the product developed, present it to stakeholder and plan on what to do next
4. Sprint Retrospective: It is a meeting held at the end of sprint to reflect the product in the past sprint and discuss what can be optimised further.
5. Backlog Refinement:

**Answer 2: Lean Development**

Lean Development or Lean Software development is a programming methodology that takes into account the Lean Manufacturing Principles and applies them to software development. Lawn Manufacturing is a production methodology introduced by Toyota. In a nutshell Lean manufacturing aims to “**do more and more with less and less**”. In other words, it tries to use less resources, less manpower to deliver less wastage and deliver products that suit the users’ needs more effectively at each iteration. It takes 5 key principles which are value, value stream, pull, flow, perfection.

Lean Development uses users’ experience to develop a solid conceptual framework for the workflow, key principles and values along with good practices which provide agility to the software that is developed. As the product is based on user’s experience with the product, it creates a kind of feedback loop where the users’ requirements may change over time and since Lean Development takes into account users’ experience, it makes room for the agility in the software.

The term lean software development was first introduced in 2003 by Mary Poppendieck and Tom Poppendieck. Lean Development is characterized by 7 principles which are in close association with Lean Manufacturing principles:

* **Eliminate Waste** - waste, here, is defined as anything that does not add value to the product from the user’s perspective. It may include extra features, defects, partial code, waiting, building wrong features, etc. Managerial overhead which won’t provide any real value to the product is also a waste from the user's perspective. Lean methodology eliminates these wastes by first identifying them by value streaming and then eliminating one by one in each iteration.
* **Amplify Learning** - Lean development is based on iteration and the value of the software is based on how efficiently it fits to the user’s needs and not restricts itself to the requirements. This means the user's need may change over time and so should the product and this is possible only when we know about the user’s need which is achieved through learning. We achieve this by presenting them with the screens and getting their input.
* **Decide as late as possible** - It aims to keep as many options open for the user as possible until the user has realized the need better. It also helps in agility of the software and makes it more generic.
* **Deliver as soon as possible**- It aims to develop and roll out the software to the market as soon as possible so that the feedback is collected faster and wastes can be eliminated in the next iteration. The shorter the iterations, better is the learning. It also helps with keeping up with the user’s current needs so the developers don’t have to worry about what the user may need in the production phase and they would actually know what they need via the software that was released in the previous iteration.
* **Empower the team** - It is based on agile principles that “work around motivated individuals and trust them to get the job done”. Here the developers come together and discuss the solution and strategies and share it with the managers who would suggest improvements or the course of action to the team.
* **Build integrity** - It include refactoring of code and building integration tests so that we keep only the essential parts of the code at each iteration and modularise it
* **Optimize** - keep up with modern trends to solve a particular problem

**Answer 3: Extreme Programming**

Extreme Programing(XP) is a software programming methodology high quality software that is susceptible and responsive to the change in user’s needs over time. By this very definition XP falls under the paradigm of Agile Software Development. It focuses on frequent releases and changing according to the user’s need in every iteration. It gets its name from the fact that the beneficial elements of the traditional software development are taken to ‘extreme’ level.

XP was developed by Kent Beck in 1996 while working in Crystal on the C3 project. It gained rapid attention from the development community in the 1990's and 2000’s because of external influences like shifting from sequential programming to object oriented programming and the rapid rise of the internet which caused a rise in speed-to-market and increased competition in the market.

There are 5 values followed in XP vis-a-vis:

1. Communication - Software development is a team sport and effective communication is necessary for rapid development.
2. Simplicity - Keep the code as simple as possible, don’t over engineer solution
3. Feedback - feedback is important to pinpoint the areas where there is room for improvement
4. Courage - raise organizational issues when they cause a hindrance to the team’s efficiency
5. Respect - respect your team members suggestions

There are 12 programming practices adopted in XP which are categorized into 4 fields.

1. Pair Programming- Coding is done by two people in a single workstation performing a single task.one person actually writes and thinks the code for the task while the other person thinks about testing, viability, changes and reviews.
2. Planning game- It consists of a planning meeting for every iteration and is done one a week. It is of 2 types vis-a-vis release planning and iteration planning.
3. Test driven Development- unit test are thought of before even beginning the coding phase for the individual tasks.
4. Whole team- customer who actually uses the system should be part of the team.
5. Continuous integration- Code is released in small iterations and the new piece of code is continuously integrated with the old code.
6. Refactoring- the code is refactored and optimized at every iteration and the non essential components or the obsolete components are removed at every iterations
7. Small releases- The release of the product is done in multiple stages and continuously so the releases are small and any risks or bugs can be identified and mitigated easily
8. Coding standards- XP dwells on very high coding standards including commenting and proper modularization
9. Collective code ownership- It means that everyone is responsible for the code that is written so everyone has editing rights.
10. Simple design- the Simplest design that solves the problem is considered best.
11. System metaphor- It incorporates naming methodology that would make it easier for the people and/or programmers to understand the functionality of the code
12. Sustainable pace- It encourages developers to not overwork for developing a solution but to keep a steady pace.

**Answer 4: Adaptive Software Development**

Adaptive Software Development(ASD) is software programming methodology that was developed by Sam Bayes and Jim highsmith while working on Rapid Application Development. It is an Agile development methodology which is iterative instead of a single release model wherein each iteration consists of Speculation, Collaboration and Learning phases which lets the software adapt to the user’s needs and requirements. It is a Cyclic Evolutionary model built to develop complex systems based on human interaction and self-organization.

The Characteristics of ASD include:

* Timeboxed
* Feature driven - development is done on the basis of the features required to accomplish the task
* Iterative - small continuous releases are adopted than a single bulk release. At each release new features are added to the software and the obsolete functionalities are removed, risks are mitigated.
* Risk driven - risks are mitigated at every iteration of the release. Interaction with the users helps identify the risks and bugs and are resolved in successive iterations.
* Change tolerant

There are three main phases in each iteration of the software development:

1. Speculation: In this phase the project or the release is initiated, planning is done as to what all needs to be done and what all could be done to get the desired results. Requirements are discussed and drafted, user statements are reports and a mission statement is compiled. This phase acknowledges the uncertainties in the project and planning and uses exploration and experimentation.
2. Collaboration: Complex projects are hard to build not only because the coding part is complex but also because the information and the knowledge required to build a stable application is also diverse and is only possible with team collaboration. So the team must communicate with one another to come up with detailed risks, methods,challenges and ideas to achieve the tasks to build the application. The team should be able to voice out their suggestions and criticisms without anonymity, etc to find efficient and effective solution.
3. Learning: This phase is aimed to increase the knowledge of the developers to the underlying technologies, which when not thoroughly studied, may sometimes yield undesired results. Learning is done with the help of Focus Groups, Technical Reviews and Project retrospective.

**Answer 5: Feature Driven Development**

Feature Driven Development(FDD) is a client-centric, architecture-centric evolutionary programming methodology which is both iterative and incremental. It is a fast and lightweight Agile development methodology which was introduced in 1999 by Jeff DeLuca. Features are an important part of FDD. Features are defined as small user-valued functionality provided to the system. In FDD, it is imperative to deliver tangible, working software continuously in small releases.

FDD is a model-driven short-iteration process that consists of five basic activities. The first two activities determine the overall model shape for the system and the next three activities define each feature. The activities are:

1. Develop overall model: This includes a high level design of the system followed by creation of domain models
2. Build feature list: A list of all the required features for the system are compiled that are necessary for the proper functionality of the system
3. Plan by feature: Feature or a set of features from the list are assigned to programmers
4. Design by feature: Programmers select a subset of features assigned to them and develop a sequence diagram and flow models to refine the working of the system that would be developed within a short time frame (usually 2 weeks).
5. Build by feature: After the designing of the feature is complete code is written for the same, unit tests are written and tested.

FDD is based on a core set of Software engineering best practices vis-a-vis:

* Domain object modelling:
* Developing by feature:
* Individual class ownership
* Feature teams
* Inspection