Write short notes on following

**· Scrum**

Scrum is a project management framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. It is lightweight and simple to understand. It emphasizes teamwork, accountability and iterative progress toward a well-defined goal.

It was developed by *Ken Schwaber* and *Jeff Sutherland* in the early 1990s to help organizations struggling with complex development projects.

Scrum makes clear the relative efficacy of your product management and work techniques so that you can continuously improve the product, the team, and the working environment. It works on the following values: Courage, Focus, Commitment, Respect, and Openness.

The framework begins with a simple premise: Start with what can be seen or known. After that, track the progress and tweak as necessary. The three pillars of Scrum are:

* Transparency
* Inspection
* Adaptation

The Scrum framework consists of

*Scrum Teams and their associated roles*: The Scrum Team consists of a *Product Owner*, the *Development Team*, and a *Scrum Master*. Scrum Teams are self-organizing — choose how best to accomplish their work — and cross-functional — teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity.

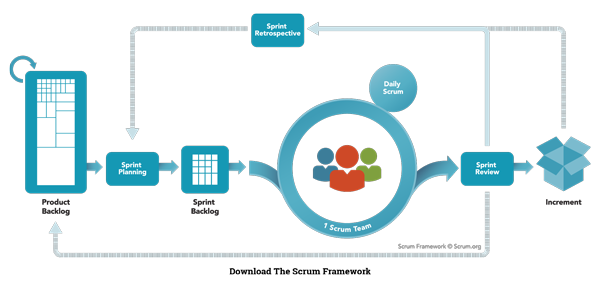
*Events*: Events in Scrum are time-boxed (of a fixed duration) and they create regularity and minimize the need for meetings not defined in Scrum. Once a Sprint begins, its duration is fixed and cannot be altered. The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process. The Scrum Events are:

* Sprint
* Sprint Planning
* Daily Scrum
* Sprint Review
* Sprint Retrospective

*Artifacts*: Scrum’s artifacts represent work or value to provide transparency and opportunities for inspection and adaptation. Artifacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact. The Scrum Artifacts are:

* Product Backlog
* Sprint Backlog
* Increment

*Rules*: The rules of Scrum bind together the roles, events, and artifacts, governing the relationships and interaction between them. Events in Scrum include:

**· Lean Development** 

It is a translation of lean manufacturing principles and practices in the form of a software development process. It is emerging within the Agile community. Lean offers a solid conceptual framework, values and principles, as well as good practices, that support agile organizations.

The term lean software development originated in the book by the same name, written by Mary Poppendieck and Tom Poppendieck, in 2003.

It can be summarized by 7 principles:

1. *Eliminate waste*: Anything and everything that doesn’t add value to the customer is considered as waste and should be eliminated
2. *Amplify learning*: The learning process is based on iterations when writing code and is sped up by usage of short iteration cycles. This leads to more communication with the customer hence amplifying learning
3. *Decide as late as possible*: decisions are delayed as much as possible until they can be made based on facts and not on uncertain assumptions and predictions
4. *Deliver as fast as possible*: The sooner the end product is delivered without major defects, the sooner feedback can be received, and incorporated into the next iteration
5. *Empower the team*: Lean approach follows the Agile Principle, i.e.build projects around motivated individuals [...] and trust them to get the job done. Encouraging progress, catching errors, and removing impediments, but not micro-managing the workers helps empower the team
6. *Build integrity in*: Integrity is how the software is being advertised, delivered, deployed, accessed, how intuitive its use is, its price and how well it solves problems. Lean incorporates this into the software building process
7. *Optimize the whole*: Software isn’t just about its individual parts, its about the product of their combined product. Lean emphasizes on having clear relationships between teams working on different parts of the product, this helps produce a system with smoothly interacting components

**· Extreme programming (XP)**

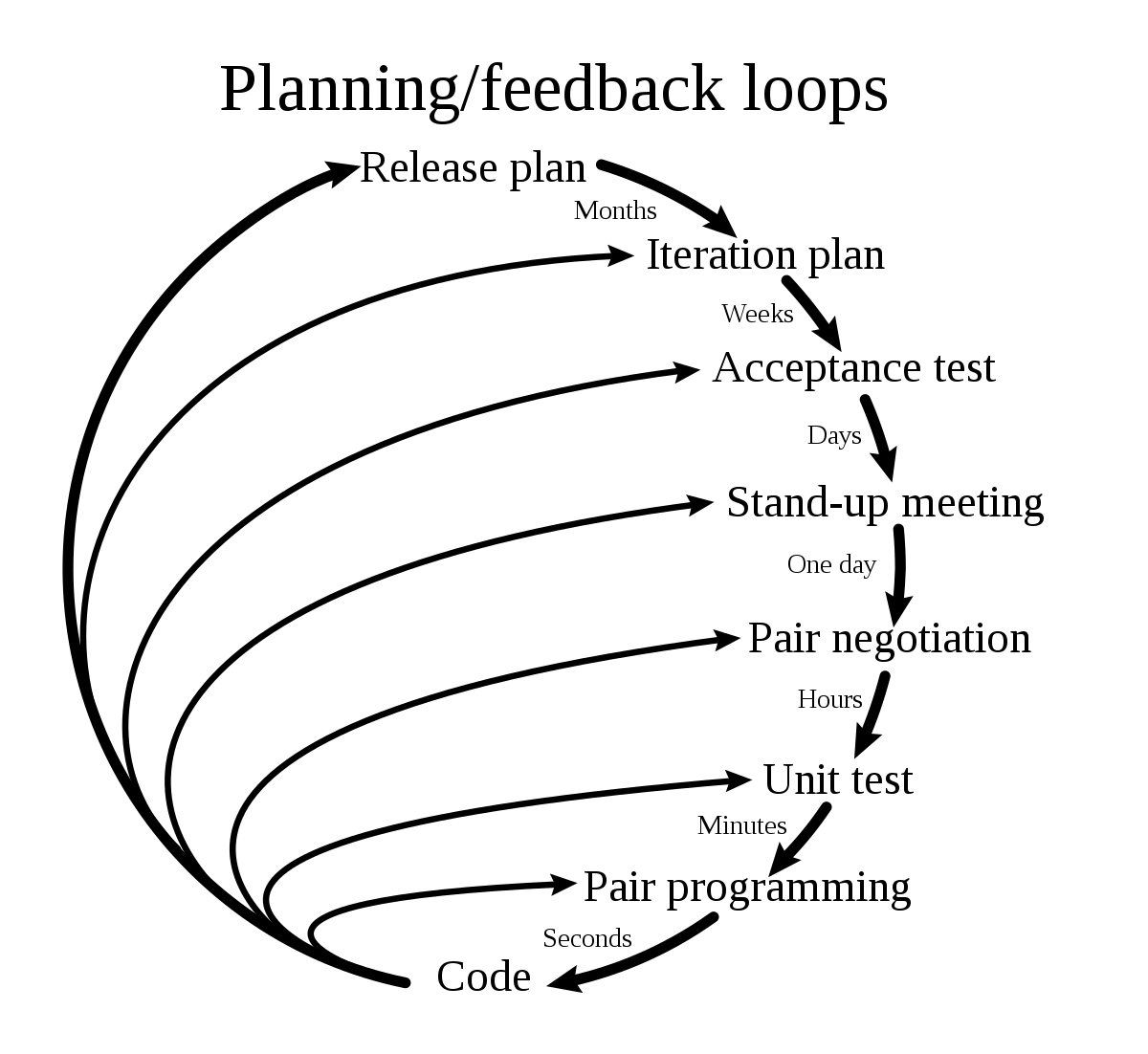
The first XP project was started on March 6, 1996. XP is one of several popular Agile Processes and has been very successful at many companies of all different sizes and industries world wide.

XP *stresses on customer satisfaction*: It delivers the software that the customer needs and as when the customer needs it. XP empowers your developers to confidently respond to changing customer requirements, even late in the life cycle.

XP *emphasizes teamwork*: Managers, customers, and developers are all equal partners in a collaborative team. XP implements a simple, yet effective environment in which the team self-organizes around the problem to solve it as efficiently as possible, thus being highly productive.

*The five essential ways XP improves a software project*:

* *Communication*: Extreme Programmers constantly communicate with their customers and fellow programmers
* *Simplicity*: design is kept clean & simple
* *Feedback*: testing the software starts from day 1, creating feedback loops. Delivery to the customers is as early as possible and changes are implemented as suggested
* *Respect*: each small success deepens the respect for the unique contributions of every team member
* *Courage*

Extreme Programming has *simple rules*: The rules that may seem awkward and perhaps even naive at first, are based on sound values and principles. Rules set expectations between team members but are not the end goal themselves. They define an environment that promotes team collaboration and empowerment.

**· Adaptive Software Development (ASD)**

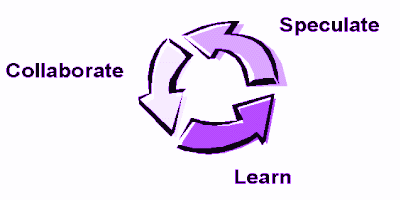
Jim Highsmith’s and Sam Bayer’s work on rapid application development (RAD) led to the creation of the *Adaptive Software Development* process. It revolves around the principle that continuous adaptation of the process to the work at hand is the normal state of affairs.

ASD replaces the traditional waterfall cycle model with a repeating series of *speculate*, *collaborate*, and *learn* cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, timeboxed, risk driven, and change tolerant. Just as RAD is, ASD is also a precedent to agile software development.

*Speculate*: The paradox of planning – it is more likely to assume that all stakeholders are comparably wrong for certain aspects of the project’s mission – is the reason why the term ‘*speculate*’ replaces planning. During speculation, the project is initiated and adaptive cycle planning is conducted. Adaptive cycle planning uses project initiation information (the customer’s mission statement, project constraints (e.g., delivery dates or user descriptions), and basic requirements) to define the set of release cycles, or software increments, that will be required for the project.

*Collaborate*: it refers to the efforts for balancing the work based on predictable parts of the environment (planning and guiding them) and adapting to the uncertain surrounding mix of changes caused by various factors, such as technology, requirements, stakeholders, software vendors.

*Learn*: The learning cycles, challenging all stakeholders, are based on the short iterations with design, build and testing. During the iterations, the knowledge is gathered by making small mistakes based on false assumptions and correcting those mistakes, thereby leading to greater experience and eventually mastery in the problem domain.



**· Feature Driven Development (FDD)**

FDD is an iterative and incremental agile (or lightweight) software development process. It is a blend of a number of industry-recognized best practices that are driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner in accordance with the Principles behind the Agile Manifesto.

History: FDD was initially devised by Jeff De Luca in 1997. It was initially a set of five processes with the first two processes being heavily influenced by Peter Coad's ideas & approaches and the other processes being a result of Jeff De Luca's experience.

Overview: FDD is a model-driven short-iteration process that consists of five basic activities. For accurate state reporting and keeping track of the software development project, milestones that mark the progress made on each feature are defined. During the first two sequential activities, i.e. developing the overall model & building the feature list, an overall model shape is established. The final three activities, i.e. *Plan*, *Design* & *Build by feature*, are iterated for each feature.

