**Name:** Imran Adeel

**ID #** 14323

Q.1:

**Methods for Designing of Software:**

 Generally, there are 6 methods are used for designing of software:

1. Waterfall Model.
2. Feature - Driven Development Model.
3. Agile Model.
4. Scrum Model.
5. Extreme Programming Model.
6. Lean Model.
7. **Waterfall Model:**

When it comes to software development, Waterfall is the most traditional and sequential choice. Although it’s usually viewed as an ”old school” or outdated method, it’s helpful to understand the history and structure of Waterfall to better appreciate the flexibility of more modern methodologies. First created in 1970, Waterfall was one of the most prominent methodologies for several decades because of its plan-driven approach.

Waterfall requires plenty of structure and documentation up front. It is divided into self-contained stages or steps. The first stage is vital, requiring a full understanding by both developers and customers of the project’s demands and scope before anything begins. The stages are relatively rigid and often follow this sequence: determine the project’s requirements and scope, analyze those requirements, design, implement, test, deploy and finally, maintain.

There’s a lack of flexibility with this approach, meaning what is decided by the customer and developer at the beginning must be seen through. Should any changes need to be made or mistakes addressed toward the end stages, the Waterfall method generally requires a full restart.

Typically, one stage must be finished before the next can begin, which can help with organization and assignments. And because the full scope of the project is understood in advance, software progress can easily be measured. Waterfall is often utilized by large, plan-driven teams who have a very clear understanding of the project’s scope;—however, development teams who don’t operate in a vacuum will likely find better results with the flexibility and agility of more modern methodologies.

1. **Feature - Driven Development Model:**

An iterative and incremental approach to software development, Feature-Driven Development

(FDD) is derived from the Agile methodology and is considered one way to implement it. Similar to Waterfall, FDD is typically viewed as an older methodology, a sort of precursor to modern Lean/Agile implementations. FDD still focuses on the goal of delivering working software frequently and is an especially client-centric approach, making it a good fit for smaller development teams.

Features are a foundational piece of FDD. Features are client-valued pieces of work that, according to the FDD approach, should be delivered every two weeks.

To produce tangible software often and efficiently, FDD has five steps, the first of which is to develop an overall model. Next, build a feature list and then plan by each feature. The final two steps—design by feature and build by feature—will take up the majority of the effort. At each step, status reporting is encouraged and helps to track progress, results, and possible errors. Although efficient response to change is one of FDD’s better attributes, an understanding of the client’s requirements and the overall model at the beginning of the project can reduce any surprises during development.

Additionally, any feature that takes longer than two weeks to design and build must be further broken down into separate features until it meets the two-week rule. The rigid structure of FDD make it less desirable to teams who balance project-driven and break-fix types of work.

1. **Agile Model:**

The Agile methodology was developed as a response to growing frustrations with Waterfall and other highly structured, inflexible methodologies. This approach is designed to accommodate change and the need to produce software faster.

Agile values individuals and their relationships and interactions over tools; it features customer collaboration throughout the development process; it responds to change instead of following a set-in-stone plan; and it focuses on presenting working software, rather than documentation.

Unlike Waterfall, Agile is well equipped to handle the complexity and variability involved in development projects.Using the Agile approach, teams develop in short sprints or iterations, each of which includes a defined duration and list of deliverables, but in no particular order. During sprints, teams work towards the goal of delivering working software (or some other tangible, testable output).

Agile is collaboration-heavy, focusing on team strengths and efficiency, along with internal feedback from various departments and clients. Client satisfaction is the highest priority with the Agile approach, which teams achieve by continuously delivering working, tested, prioritized features.

1. **Scrum Model:**

Another way to implement the Agile approach, Scrum borrows from Agile’s foundational beliefs and philosophy that teams and developers should collaborate heavily and daily.

With Scrum, software is developed using an iterative approach in which the team is front and center—experienced and disciplined workers on smaller teams might find the most success with this method, as it requires self-organization and self-management.

Team members break down end goals into smaller goals at the beginning and work through them using fixed-length iterations—or sprints—to build software and showcase it often (which usually last two weeks). Meetings play an important role in the Scrum approach, and during each sprint, daily planning meetings and demos take place to follow progress and gather feedback. This incremental method promotes quick changes and development and adds value to complex projects. Scrum incorporates the structure and discipline of more traditional software development methodologies with the flexibility and iterative practices of modern Agile.

1. **Extreme Programming Model:**

Another Agile framework, Extreme Programming (or XP) focuses on producing higher quality software using the best practices in software development. As with most Agile approaches, XP allows for frequent releases in short development sprints that encourage change when needed.

In general, XP follows a set of values, rather than steps, including simplicity (develop what is required, nothing more); communication (teams must collaborate and work together on every piece of the software); consistent feedback; and respect.

Extreme Programming requires developers to first plan and understand the customer’s user stories—their informal descriptions of certain features. Other practices include: scheduling and dividing work into iterations. Design with simplicity in mind, code and test often, which helps to create fault-free software. Listen to feedback to best understand the functionality, and then test more.

1. **Lean:**

Lean is at once a workflow methodology and a mindset, incorporating principles and practices from the manufacturing space and applying them broadly to a variety of industries, including software development. While Agile is an excellent methodology for the practical application of development best practices, it does not include instructions for scaling these practices across the organization or applying them outside of development-type work.

This is why many organizations who practice Agile at the team level begin to incorporate Lean philosophies, practices, and tools to help to innovate at scale. Lean’s basic principles—optimize the whole, eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, and respect people—can help to guide decision-making across the organization in a way that can help to unearth potential issues and maintain a healthy organizational culture.

Combining the best of Lean thinking and Agile software development practices can create a healthy, sustainable culture of innovation that benefits not only the development organization, but the system as a whole.