**NAME: AMBAR JHA ROLL NO: 18 CLASS:TE-CMPN**

**Assignment No 1**

**Software Engineering**

**CO1: Understand and demonstrate basic knowledge in software engineering**

1. **Define software engineering. Explain umbrella activities in detail.**

**Ans**. Software Engineering is the method of applying scientific and technological knowledge, procedures and rules to design, develop and maintain software product.

## **Umbrella Activities in software engineering**

1. Software Project Tracking and Control
2. Formal Technical Reviews
3. Software Quality Assurance
4. Software Configuration Management
5. Document Preparation and Production
6. Re-usability Management
7. Measurement and Metrics
8. Risk Management

### Software Project Tracking and Control

Before the actual development begins, a schedule for developing the software is created. Based on that schedule the development will be done. However, after a certain period of time it is required to review the progress of the development to find out actions which are in need to be taken to complete the development, testing etc. in time. The outcome of the review might require the development to be rescheduled.

### Formal Technical Reviews

Software engineering is done in clusters or modules, after completing each module, it is good practice to review the completed module to find out and remove errors so their propagation to the next module can be prevented.

### Software Quality Assurance

The quality of the software such user experience, performance, load handling capacity etc. should be tested and confirmed after reaching predefined milestones. This reduces the task at the end of the development process. It should be conducted by dedicated teams so that the development can keep going on.

### Software Configuration Management

Software configuration management (SCM) is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products.

### Document preparation and production

All the project planning and other activities should be hardly copied and the production get started here.

### Re-usability Management

This includes the backing up of each part of the software project they can be corrected or any kind of support can be given to them later to update or upgrade the software at user/time demand.

### Measurement & Metrics

This will include all the measurement of every aspects of the software project.

### Risk Management

[Risk management is a series of steps](https://onlineclassnotes.com/2016/05/what-are-types-or-categories-of-risks-in-software-engineering-explain-the-process-of-risk-management.html) that help a software team to understand and manage uncertainty. It’s a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan that─ ‘should the problem actually occur’.

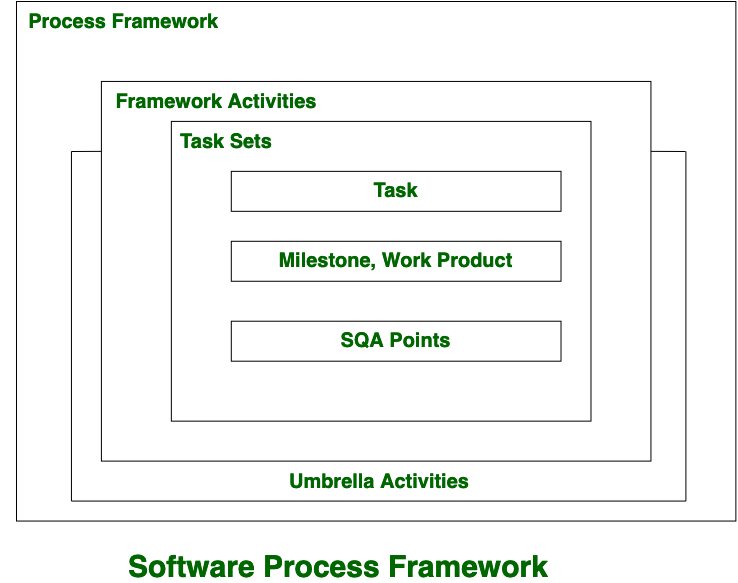
1. **Comparison between waterfall model, v-model, incremental model, iterative model, and spiral model**

**Ans.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Features** | **Waterfall** | **Iterative** | **Incremental** | **Spiral** | **V Model** |
| Well Defined Requirements | Yes | No | No | No | Yes |
| User Involvement in all phases | Only at beginning | High | Yes | High | No |
| Risk Analysis | Only at beginning | No Risk Analysis | No Risk Analysis | Yes | Only at beginning |
| Overlapping Phases | No | Yes | No | Yes | No |
| Cost | Low | High | Low | Expensive | Expensive |

**C) Software Framework Activities in detail**.

**Ans.** Software Process Framework is a foundation of complete software engineering process. Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.



A generic process framework encompasses five activities which are given below one by one:

**Communication:**   
In this activity, heavy communication with customers and other stakeholders, requirement gathering is done.

**Planning:**   
In this activity, we discuss the technical related tasks, work schedule, risks, required resources etc.

**Modeling:**   
Modeling is about building representations of things in the ‘real world’. In modeling activity, a product’s model is created in order to better understanding and requirements.

**Construction:**   
In software engineering, construction is the application of set of procedures that are needed to assemble the product. In this activity, we generate the code and test the product in order to make better product.

**Deployment:**   
In this activity, a complete or non-complete products or software are represented to the customers to evaluate and give feedback. on the basis of their feedback, we modify the products for supply better product.

**D)Explain Kanban model in detail.**

The basic elements of a Kanban software development process

### 1. Visualize Work

Creating a Kanban board is the first step towards visualizing your software development process. By creating a visual model of your work and workflow, you can observe the flow of work moving through your Kanban system.

The first step towards visualizing your work is to understand the distinct steps your work goes through as it moves from “To Do” to “Done.” These will become the lanes on your Kanban board.

Most software development teams follow some version of the following process: Gather Requirements > Design > Implementation > QA > Deploy > Maintain. We recommend undergoing a [value stream mapping exercise](https://blog.planview.com/value-stream-mapping-for-software-development/) to identify your team’s unique process steps.

Visualizing work in Kanban means not only visualizing the process, but visualizing each piece of work (represented by a Kanban card) as it moves through that process. Teams find that making the work visible, along with blockers, bottlenecks and queues, instantly leads to increased communication and collaboration.

Visualizing all work across the team also helps establish accountability and transparency across the team. This is invaluable, especially for software development teams, which require coordinated efforts to get their work done efficiently.

### 2. Limit Work in Process

Software development teams can easily get overwhelmed with their long list of to-dos; between new features, bug fixes, maintenance work, and other project work, teams often struggle to prioritize work in a disciplined, methodical way.

An important concept in the Kanban software development process, then, is limiting work in process, or WIP. By limiting how much-unfinished work is in process, you can reduce the time it takes an item to travel through the Kanban system. You can also avoid problems caused by task switching and reduce the need to constantly re-prioritize items.

[Limiting WIP](https://www.planview.com/resources/articles/benefits-wip-limits/) at both the personal and team levels can help software development teams move faster, reduce error, and collaborate more effectively.

### 3. Focus on Flow

A deliberate emphasis on optimizing for flow is a critical element of any Kanban software development process. Flow describes the way work moves through your Kanban system. Good flow means that work moves in a fairly linear path, from one step to another with little delay between them. Bad flow happens when work stops and starts abruptly, spends lots of time in wait states, or moves between different steps without making significant progress.

Using WIP limits is great first step towards optimizing for flow. By limiting WIP, you limit one of the biggest obstacles to good flow: an excess of active work. But this is not all; you must, as a team, work to keep each piece of work moving forward. Many teams discuss ways to improve flow at a daily standup, where they ask questions such as:

* What can we do to move existing cards off the board?
* Is there anything we can help to finish before we pull next work?
* Has anything been waiting in the same place for more than a day? What can we do to get it moving again?

You can also use process policies to maintain consistently good flow, by creating rules that all cards (and therefore, all team members) must follow. These might include:

* WIP limits on each lane, to prevent bottlenecks
* A policy on how to handle blockers
* A policy that dictates conditions that must be met before a new card can be pulled into an active lane
* A policy for “stale” work – work that has been sitting for a period that is longer than usual

Focusing on optimizing for flow can help software development teams stay productive, efficient, and ever-evolving on their path of continuous improvement.

### 4. Continuous Improvement

Kanban is a system of evolution, not revolution; it is not meant to force your team to completely change everything it does. A Kanban system is most effective when it reflects reality, which is why we always recommend that teams start by designing a board that accurately reflects their existing process.

From there, you can thoughtfully, carefully practice continuous improvement, a perpetual process of making small, meaningful changes that help your team perform more efficiently and effectively.

As you use your Kanban board, you’ll notice opportunities for practicing continuous improvement. Implement these changes one at a time, and measure their impact so you can maximize your learning.

Follow these four basic elements of the Kanban software development process consistently, and you’ll find the balance between discipline and flexibility that your team needs to succeed.