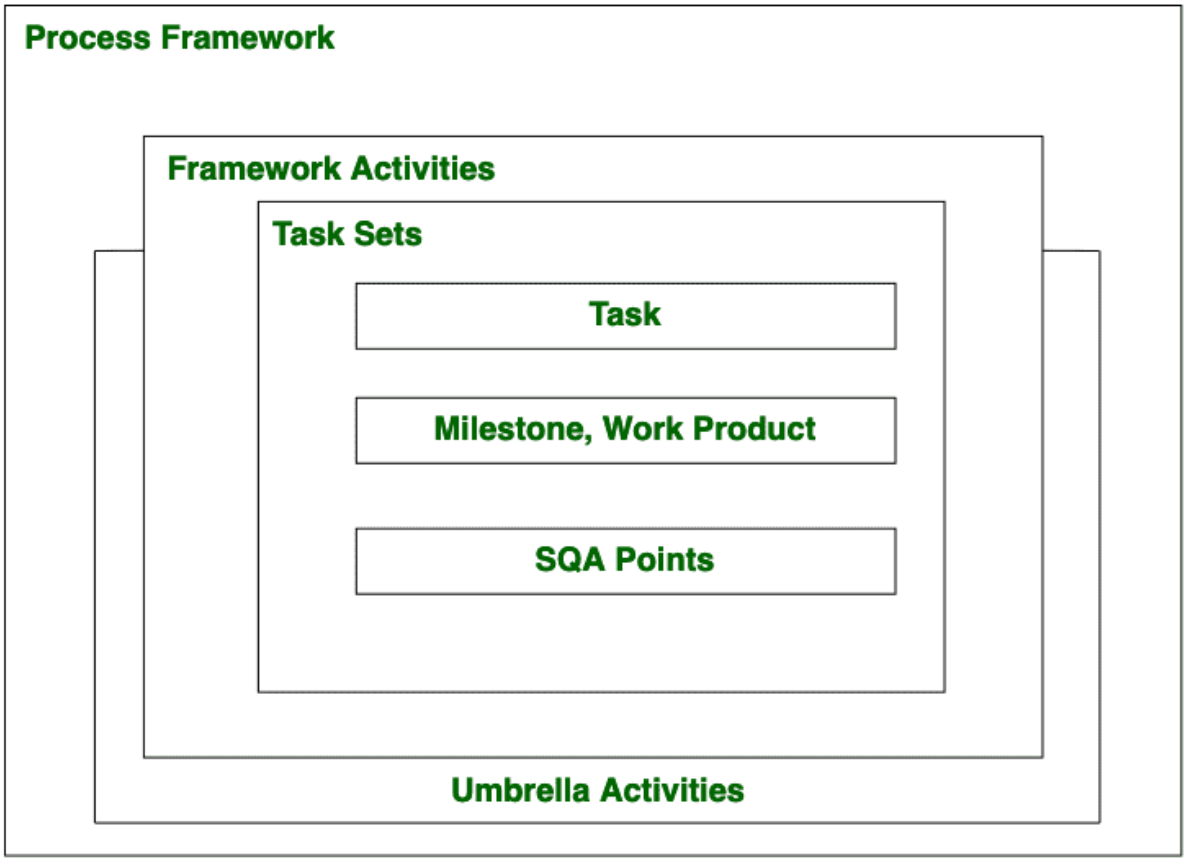
Software Process Framework

Framework is a Standard way to build and deploy applications. **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.



Software Process Framework

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

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

2. **Planning:**
   In this activity, we discuss the technical related tasks, work schedule, risks, required resources etc.
3. **Modeling:**
   Modelling is about building representations of things in the 'real world'.In modelling activity, a product's model is created in order to better understanding and requirements.
4. **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.
5. **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.

Umbrella activities include:

- Risk management
- Software quality assurance(SQA)
- Software configuration management(SCM)
- Measurement
- Formal technical reviews(FTR)

Umbrella Activities:

Software engineering is a collection of co-related steps. These steps are presented or accessed in different approaches in *different software process models*. Umbrella activities are a set of steps or procedure that the software engineering team follows to maintain the progress, quality, change and risks of the overall development tasks. These steps of umbrella activities will evolve through the *phases of generic view of software development*.

Umbrella Activities are as follows:

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 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'.

Architectural Design

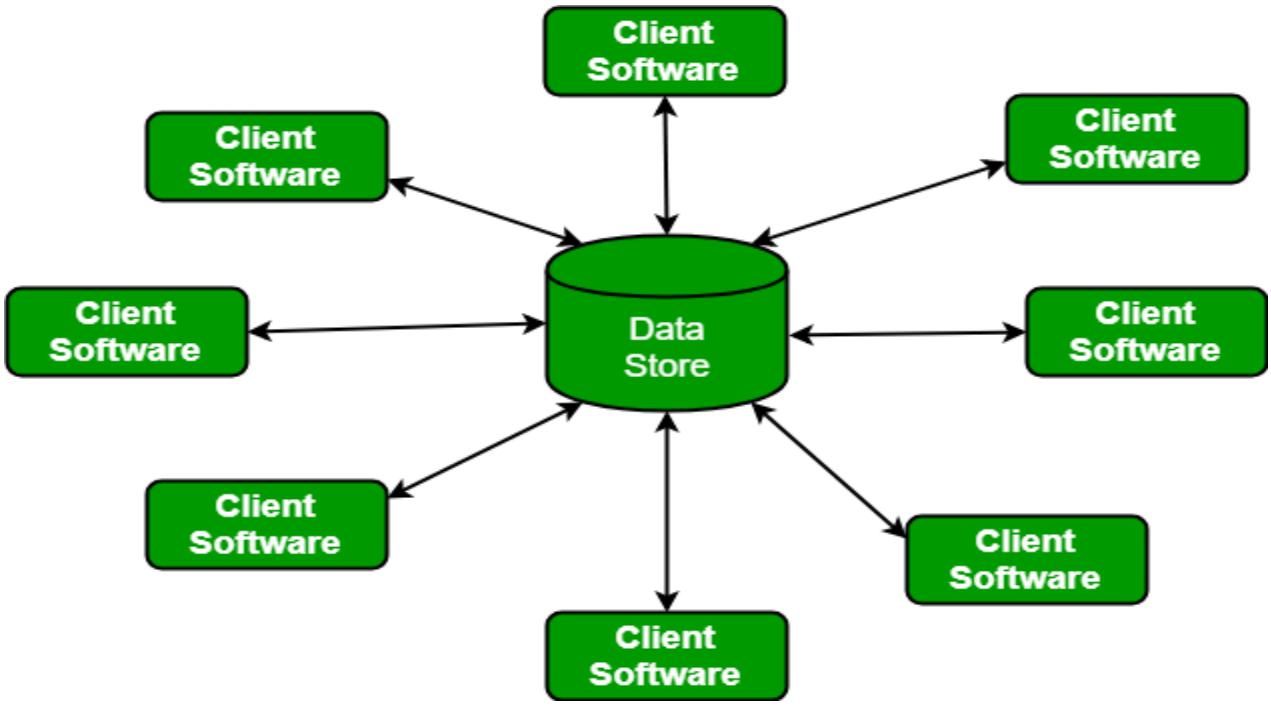
Introduction: The software needs the architectural design to represents the design of software. IEEE defines architectural design as "the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system." The software that is built for computer-based systems can exhibit one of these many architectural styles.
Each style will describe a system category that consists of :

- A set of components(eg: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

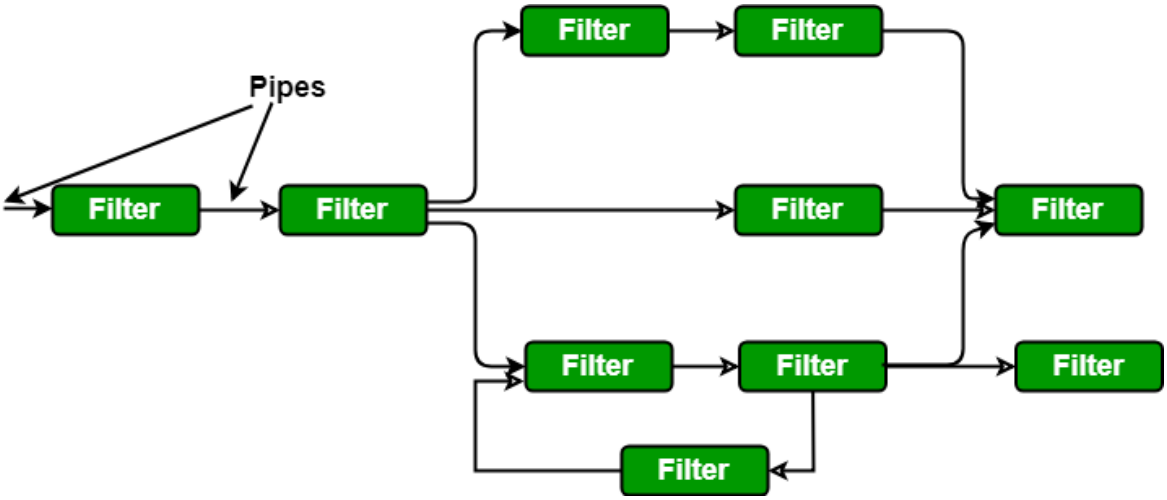
The use of architectural styles is to establish a structure for all the components of the system.

Taxonomy of Architectural styles:

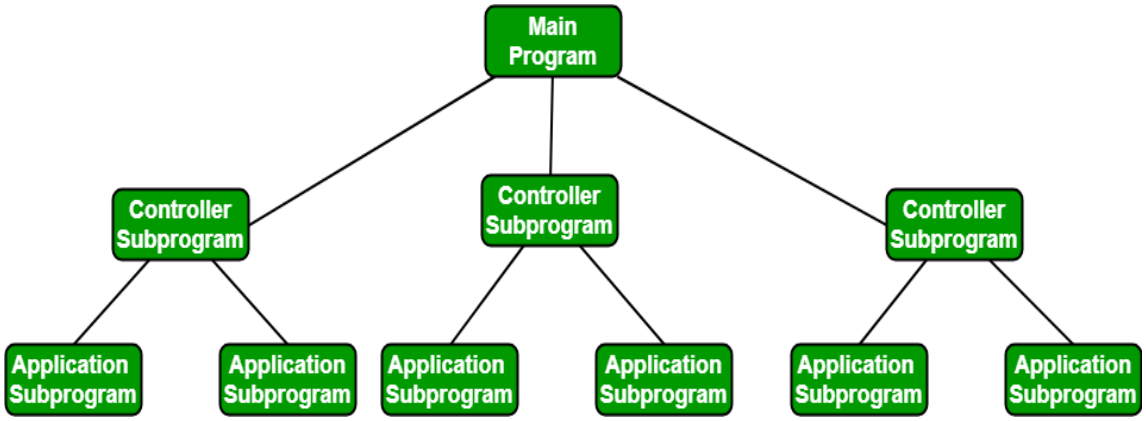
1. **Data centred architectures:**
   - A data store will reside at the center of this architecture and is accessed frequently by the other components that update, add, delete or modify the data present within the store.
   - The figure illustrates a typical data centered style. The client software access a central repository. Variation of this approach are used to transform the repository into a blackboard when data related to client or data of interest for the client change the notifications to client software.
   - This data-centered architecture will promote integrability. This means that the existing components can be changed and new client components can be added to the architecture without the permission or concern of other clients.
   - Data can be passed among clients using blackboard mechanism.



2. **Data flow architectures:**
   - This kind of architecture is used when input data to be transformed into output data through a series of computational manipulative components.
   - The figure represents pipe-and-filter architecture since it uses both pipe and filter and it has a set of components called filters connected by pipes.
   - Pipes are used to transmit data from one component to the next.
   - Each filter will work independently and is designed to take data input of a certain form and produces data output to the next filter of a specified form. The filters don't require any knowledge of the working of neighboring filters.
   - If the data flow degenerates into a single line of transforms, then it is termed as batch sequential. This structure accepts the batch of data and then applies a series of sequential components to transform it.



3. **Call and Return architectures:** It is used to create a program that is easy to scale and modify. Many sub-styles exist within this category. Two of them are explained below.
    - **Remote procedure call architecture:** This components is used to present in a main program or sub program architecture distributed among multiple computers on a network.
4. **Main program or Subprogram architectures:** The main program structure decomposes into number of subprograms or function into a control hierarchy. Main program contains number of subprograms that can invoke other components.



5. **Object Oriented architecture:** The components of a system encapsulate data and the operations that must be applied to manipulate the data. The

coordination and communication between the components are established via the message passing.

6. **Layered architecture:**
   - A number of different layers are defined with each layer performing a well-defined set of operations. Each layer will do some operations that becomes closer to machine instruction set progressively.
   - At the outer layer, components will receive the user interface operations and at the inner layers, components will perform the operating system interfacing(communication and coordination with OS)
   - Intermediate layers to utility services and application software functions.