Architectural Design Architectural design is a creative process here you design a system organization that will satisfy the functional and non-functional requirements of a system. Because it is a creative process, the activities within the process depend on the type of system being developed, the background and experience of the system architect, and the specific requirements for the system

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.

Because of the close relationship between non-functional requirements and software architecture, the particular architectural style and structure that you choose for a system should depend on the non-functional system requirements:

1. **Performance** If performance is a critical requirement, the architecture should be designed to localize critical operations within a small number of components, with these components all deployed on the same computer rather than distributed across the network. This may mean using a few relatively large components rather than small, fine-grain components, which reduces the number of component communications. You may also consider run-time system organizations that allow the system to be replicated and executed on different processors.

2. **Security** If security is a critical requirement, a layered structure for the architecture should be used, with the most critical assets protected in the innermost layers, with a high level of security validation applied to these layers.

3. Safety If safety is a critical requirement, the architecture should be designed so that safety-related operations are all located in either a single component or in a small number of components. This reduces the costs and problems of safety validation and makes it possible to provide related protection systems that can safely shut down the system in the event of failure.

4. Availability If availability is a critical requirement, the architecture should be designed to include redundant components so that it is possible to replace and update components without stopping the system.

5.Maintainability If maintainability is a critical requirement, the system architecture should be designed using fine-grain, self-contained components that may readily be changed. Producers of data should be separated from consumers and shared data structures should be avoided.

Architectural views Architectural views,discuss two issues that are relevant to both of these:

1. What views or perspectives are useful when designing and documenting a system's architecture?

2. What notations should be used for describing architectural models?

It is impossible to represent all relevant information about a system's architecture in a single architectural model, as each model only shows one view or perspective of the system. It might show how a system is decomposed into modules, how the run-time processes interact, or the different ways in which system components are distributed across a network. All of these are useful at different times so, for both design and documentation, you usually need to present multiple views of the software architecture. There are different opinions as to what views are required.

Krutchen (1995), in his well-known 4+1 view model of software architecture, suggests that there should be four fundamental architectural views, which are related using use cases or scenarios. The views that he suggests are:

1. A logical view which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.

2. A process view which shows how, at run-time, the system is composed of interacting processes. This view is useful for making judgments about nonfunctional system characteristics such as performance and availability.

3. A development view which shows how the software is decomposed for development, that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.

4. A physical view which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

Hofmeister et al. (2000) suggest the use of similar views but add to this the notion of a conceptual view. This view is an abstract view of the system

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

1. **Data centered 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.
    - 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.

- 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.

1. 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.
2. 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.

Reference:

https://www.geeksforgeeks.org/software-engineering-architectural-design/

https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf