Software Engineering | Design Engineering

Architectural Design

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.

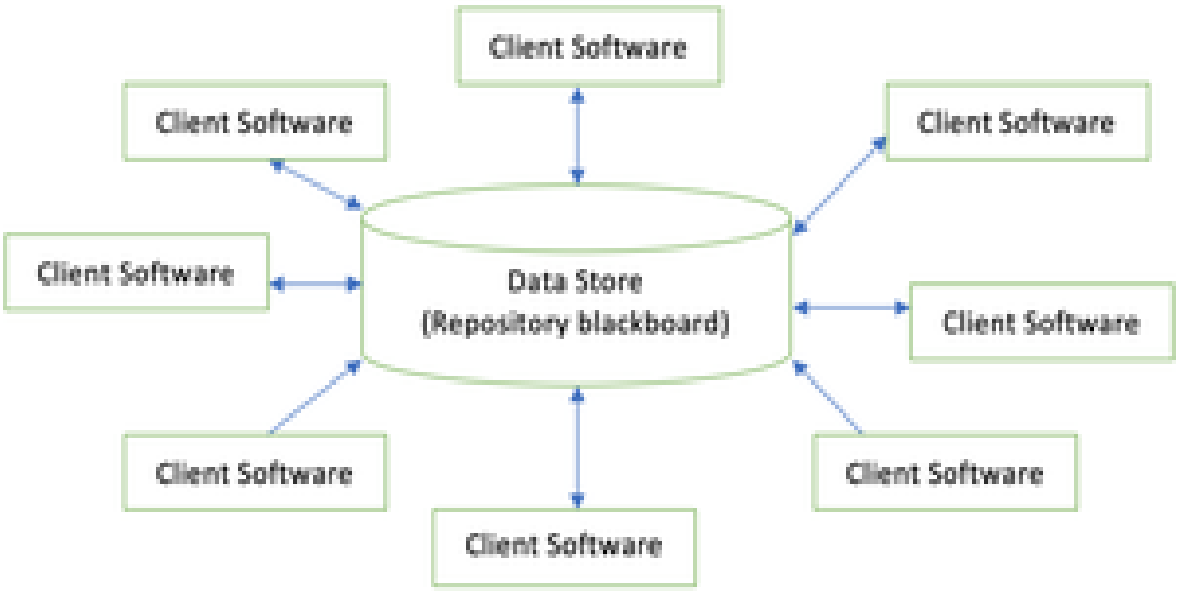
Taxonomy of Architectural styles:

1. **Data 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.
    - 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.

Advantage of Data centered architecture

- Repository of data is independent of clients

- Client work independent of each other
- It may be simple to add additional clients.
- Modification can be very easy



Data centered architecture

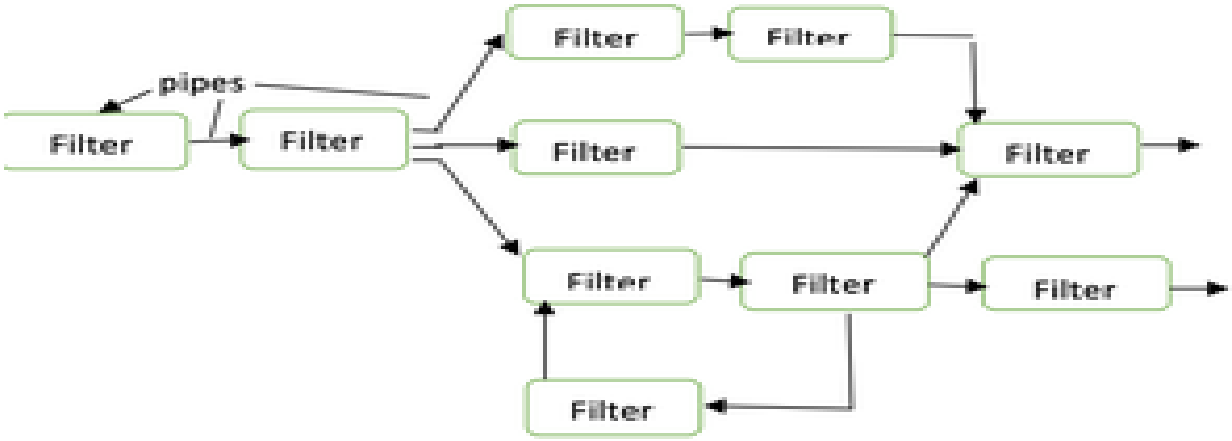
1. Data flow architectures:
    - This kind of architecture is used when input data to be transformed into output data through a series of computational manipulative components.
    - The figure represents pipe-and-filter architecture since it uses both pipe and filter and it has a set of components 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.

Advantage of Data Flow architecture

- It encourages upkeep, repurposing, and modification.
- With this design, concurrent execution is supported.

Disadvantage of Data Flow architecture

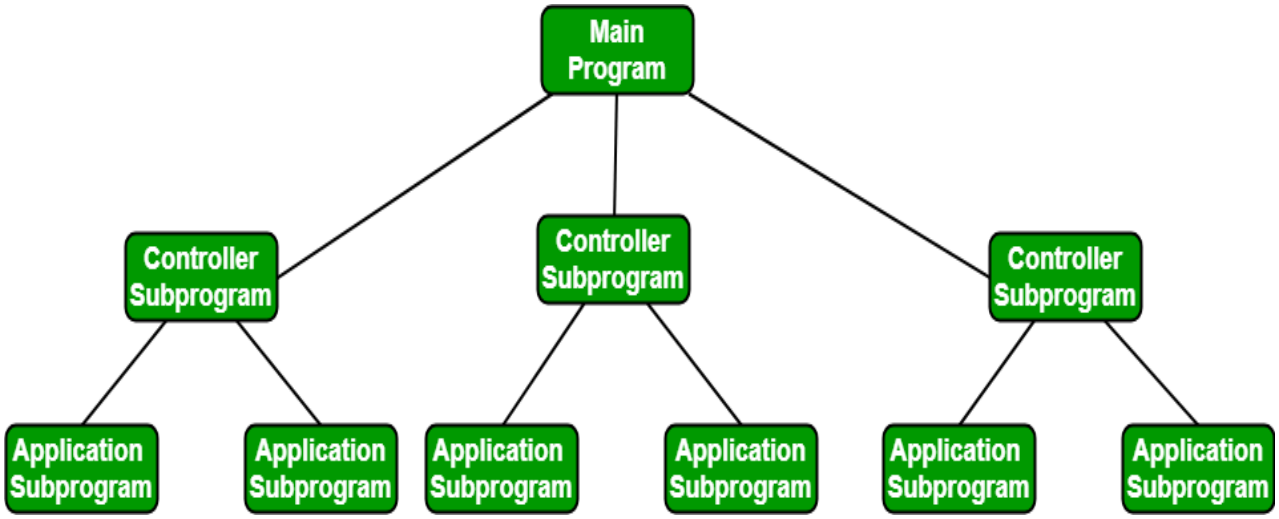
- It frequently degenerates to batch sequential system
- Data flow architecture does not allow applications that require greater user engagement.
- It is not easy to coordinate two different but related streams



Data Flow architecture

1. 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.

Characteristics of Object Oriented architecture

- Object protect the system's integrity.
- An object is unaware of the depiction of other items.

Advantage of Object Oriented architecture

- It enables the designer to separate a challenge into a collection of autonomous objects.
- Other objects are aware of the implementation details of the object, allowing changes to be made without having an impact on other objects.

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.
- One common example of this architectural style is OSI-ISO (Open Systems Interconnection-International Organisation for Standardisation) communication system.

Layered architecture

User Interface Design

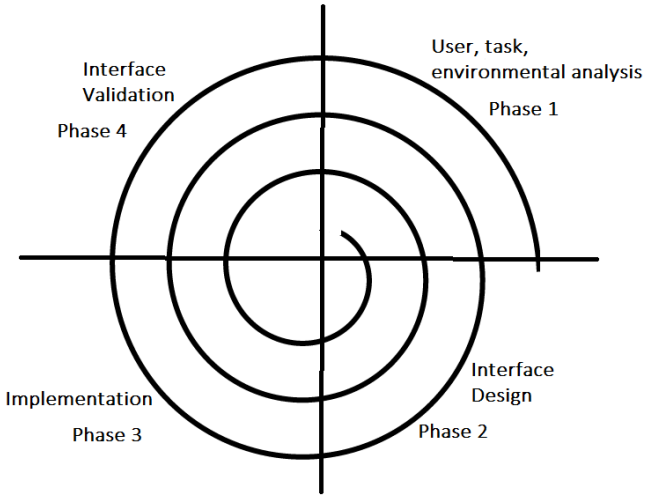
User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interface screens

There are two types of User Interface:

1. Command Line Interface: Command Line Interface provides a command prompt, where the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.
2. Graphical User Interface: Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.

User Interface Design Process:



The analysis and design process of a user interface is iterative and can be represented by a spiral model. The analysis and design process of user interface consists of four framework activities.

1. User, task, environmental analysis, and modeling: Initially, the focus is based on the profile of users who will interact with the system, i.e. understanding, skill and knowledge, type of user, etc, based on the user's profile users are made into categories. From each category requirements are gathered. Based on the requirements developer understand how to develop the interface. Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified, described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be asked are:
    - Where will the interface be located physically?
    - Will the user be sitting, standing, or performing other tasks unrelated to the interface?
    - Does the interface hardware accommodate space, light, or noise constraints?
    - Are there special human factors considerations driven by environmental factors?
2. Interface Design: The goal of this phase is to define the set of interface objects and actions i.e. Control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system. Specify the action sequence of tasks and subtasks, also called a user scenario. Indicate the state of the system when the user performs a particular task. Always follow the three golden rules stated by Theo Mandel. Design issues such as response time, command and action structure, error handling, and help facilities are considered as the design model is refined. This phase serves as the foundation for the implementation phase.

3. Interface construction and implementation: The implementation activity begins with the creation of prototype (model) that enables usage scenarios to be evaluated. As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.

4. Interface Validation: This phase focuses on testing the interface. The interface should be in such a way that it should be able to perform tasks correctly and it should be able to handle a variety of tasks. It should achieve all the user's requirements. It should be easy to use and easy to learn. Users should accept the interface as a useful one in their work.

Golden Rules:

The following are the golden rules stated by Theo Mandel that must be followed during the design of the interface. Place the user in control:

- Define the interaction modes in such a way that does not force the user into unnecessary or undesired actions: The user should be able to easily enter and exit the mode with little or no effort.

- Provide for flexible interaction: Different people will use different interaction mechanisms, some might use keyboard commands, some might use mouse, some might use touch screen, etc, Hence all interaction mechanisms should be provided.

- Allow user interaction to be interruptible and undoable: When a user is doing a sequence of actions the user must be able to interrupt the sequence to do some other work without losing the work that had been done. The user should also be able to do undo operation.

- Streamline interaction as skill level advances and allow the interaction to be customized: Advanced or highly skilled user should be provided a chance to customize the interface as user wants which allows different interaction mechanisms so that user doesn't feel bored while using the same interaction mechanism.

- Hide technical internals from casual users: The user should not be aware of the internal technical details of the system. He should interact with the interface just to do his work.

- Design for direct interaction with objects that appear on screen: The user should be able to use the objects and manipulate the objects that are present on the screen to perform a necessary task. By this, the user feels easy to control over the screen.

Reduce the user's memory load:

- Reduce demand on short-term memory: When users are involved in some complex tasks the demand on short-term memory is significant. So the interface should be

designed in such a way to reduce the remembering of previously done actions, given inputs and results.

- Establish meaningful defaults: Always initial set of defaults should be provided to the average user, if a user needs to add some new features then he should be able to add the required features.
- Define shortcuts that are intuitive: Mnemonics should be used by the user. Mnemonics means the keyboard shortcuts to do some action on the screen.
- The visual layout of the interface should be based on a real-world metaphor: Anything you represent on a screen if it is a metaphor for real-world entity then users would easily understand.
- Disclose information in a progressive fashion: The interface should be organized hierarchically i.e. on the main screen the information about the task, an object or some behavior should be presented first at a high level of abstraction. More detail should be presented after the user indicates interest with a mouse pick.

Make the interface consistent:

- Allow the user to put the current task into a meaningful context: Many interfaces have dozens of screens. So it is important to provide indicators consistently so that the user know about the doing work. The user should also know from which page has navigated to the current page and from the current page where can navigate.
- Maintain consistency across a family of applications: The development of some set of applications all should follow and implement the same design, rules so that consistency is maintained among applications.
- If past interactive models have created user expectations do not make changes unless there is a compelling reason.