Software Design

Software design is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. It deals with representing the client's requirement, as described in SRS (Software Requirement Specification) document, into a form, i.e., easily implementable using programming language.

The following items are designed and documented during the design phase: 

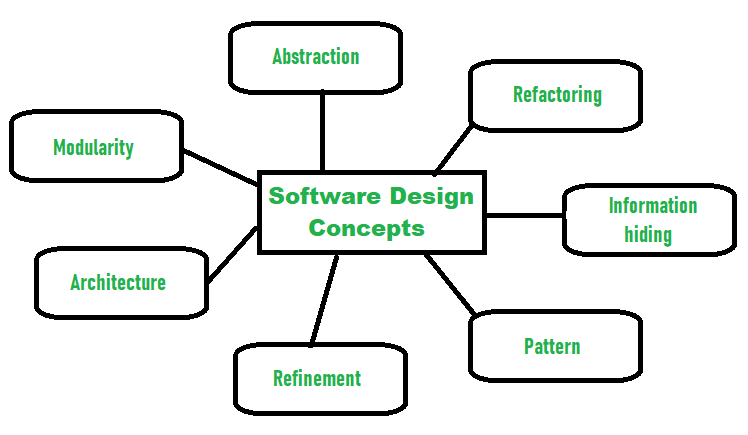
* Different modules required.
* Control relationships among modules.
* Interface among different modules.
* Data structure among the different modules.
* Algorithms required to implement among the individual modules.

## Objectives of Software Design

1. **Correctness:**   
   A good design should be correct i.e. it should correctly implement all the functionalities of the system.
2. **Efficiency:**   
   A good software design should address the resources, time, and cost optimization issues.
3. **Flexibility:**  
   A good software design should have the ability to adapt and accommodate changes easily. It includes designing the software in a way, that allows for modifications, enhancements, and scalability without requiring significant rework or causing major disruptions to the existing functionality.
4. **Understandability:**   
   A good design should be easily understandable, for which it should be modular and all the modules are arranged in layers.
5. **Completeness:**   
   The design should have all the components like data structures, modules, and external interfaces, etc.
6. **Maintainability:**   
   A good software design aims to create a system that is easy to understand, modify, and maintain over time.

Software Design concepts:

The **software design concept** simply means the idea or principle behind the design. It describes how you plan to solve the problem of designing software, the logic, or thinking behind how you will design software. It allows the software engineer to create the model of the system or software or product that is to be developed or built.



The following**points should be considered while designing Software:** 

1. **Abstraction- hide Irrelevant data**   
   Abstraction simply means to hide the details to reduce complexity and increases efficiency or quality. Different levels of Abstraction are necessary and must be applied at each stage of the design process so that any error that is present can be removed to increase the efficiency of the software solution and to refine the software solution.
2. **Modularity- sub divide the system**   
   Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project. In the same way, modularity in design means subdividing a system into smaller parts so that these parts can be created independently and then use these parts in different systems to perform different functions.
3. **Architecture - design a structure of something**  
   Architecture simply means a technique to design a structure of something. Architecture in designing software is a concept that focuses on various elements and the data of the structure. These components interact with each other and use the data of the structure in architecture.
4. **Refinement- removes impurities**  
   Refinement simply means to refine something to remove any impurities if present and increase the quality. The refinement concept of software design is actually a process of developing or presenting the software or system in a detailed manner that means to elaborate a system or software. Refinement is very necessary to find out any error if present and then to reduce it.
5. **Pattern- a repeated form**   
   The pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern. The pattern in the design process means the repetition of a solution to a common recurring problem within a certain context.
6. **Information Hiding - hide the information**   
   Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party. In software design, information hiding is achieved by designing the modules in a manner that the information gathered or contained in one module is hidden and can’t be accessed by any other modules.
7. **Refactoring- reconstruct something**  
   Refactoring simply means reconstructing something in such a way that it does not affect the behavior of any other features. Refactoring in software design means reconstructing the design to reduce complexity and simplify it without affecting the behavior or its functions.

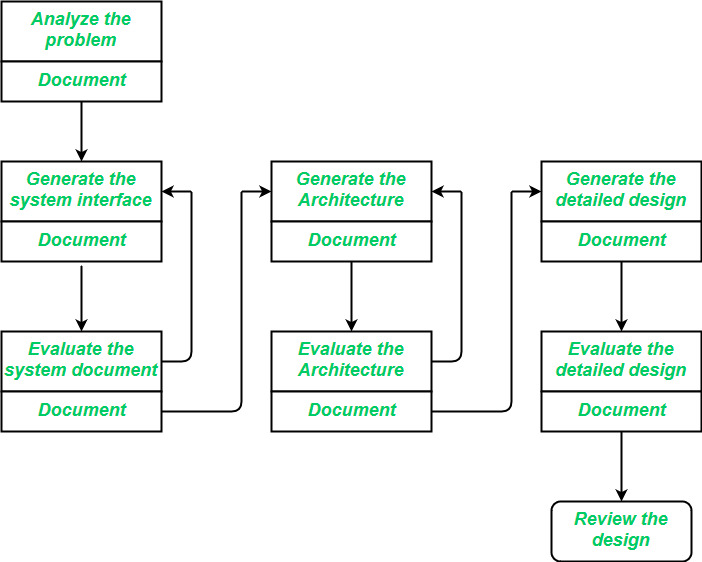
**Software Design Process**

The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language. The software design process can be divided into the following three levels of phases of design:

1. Interface Design
2. Architectural Design
3. Detailed Design

**Elements of a System:**

1. **Architecture –** This is the conceptual model that defines the structure, behavior, and views of a system. We can use flowcharts to represent and illustrate the architecture.
2. **Modules –** These are components that handle one specific task in a system. A combination of the modules makes up the system.
3. **Components –** This provides a particular function or group of related functions. They are made up of modules.
4. **Interfaces –**This is the shared boundary across which the components of a system exchange information and relate.
5. **Data –**This is the management of the information and data flow.

  
   
**Interface Design:** *Interface design* is the specification of the interaction between a system and its environment. this phase proceeds at a high level of abstraction with respect to the inner workings of the system i.e, during interface design, the internal of the systems are completely ignored and the system is treated as a black box. Attention is focused on the dialogue between the target system and the users, devices, and other systems with which it interacts. The design problem statement produced during the problem analysis step should identify the people, other systems, and devices which are collectively called *agents*. Interface design should include the following details:

* Precise description of events in the environment, or messages from agents to which the system must respond.
* Precise description of the events or messages that the system must produce.
* Specification of the data, and the formats of the data coming into and going out of the system.
* Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

**Architectural Design:** *Architectural design* is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored. Issues in architectural design include:

* Gross decomposition of the systems into major components.
* Allocation of functional responsibilities to components.
* Component Interfaces
* Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
* Communication and interaction between components.

The architectural design adds important details ignored during the interface design. Design of the internals of the major components is ignored until the last phase of the design.

**Detailed Design:** *Design* is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures. The detailed design may include:

* Decomposition of major system components into program units.
* Allocation of functional responsibilities to units.
* User interfaces
* Unit states and state changes
* Data and control interaction between units
* Data packaging and implementation, including issues of scope and visibility of program elements
* Algorithms and data structures

# Heuristic Evaluation

**The need for Heuristic Evaluation :**

Heuristic Evaluation is the process of thorough evaluation/assessment where the experts in a particular domain, used to measure the usability of the user interface. Usability can be defined as how easily a specific user can use a particular design or say interface without facing any problem. In general, we can say the Heuristic Evaluation is performed to detect the issues in the design of a product. It also identifies the ways to resolve those issues present in design and meet the user expectations.

**When to conduct Heuristic Evaluation :**

There is no such rule when to perform the Heuristics Evaluation, but it can be performed at any stage of the design process. Most of the time the heuristic evaluation is performed after the paper prototyping and usability test. As Heuristics Evaluation helps to optimize the design of the user-interface it becomes very important to be performed to evaluate the final design.

**How to conduct Heuristic Evaluation :**

**Define the Scope of Evaluation –**  
Mentioning the budget and deadline becomes very important at the time of evaluation. One should also define the different parameters where they want to conduct the usability test.

**Know the End-User –**   
As we know, different groups of people have different expectations from a product. So it becomes very important to know the end-user and their interest.

**Choose your Set of Heuristics –**  
Without a proper heuristic, the Heuristics Evaluation will produce unreliable and useless results if all the evaluators are not going to use the same guidelines.

**Setting-up an Evaluation System and Identifying Issues –**  
Decide the different categories in which a problem should be categories like a critical issue, minor issue, etc. Evaluators must follow the guidelines of system evaluation.

**Analyze and Summarize the Results –**  
It becomes very necessary to analyze the issue present in the design of user interface and solve those issues before the deadline.

**Advantages :**

* Reveals many hidden usability problems.
* It helps to determine the overall user experience.
* Heuristics evaluation can be combined with usability testing.
* Better Heuristics Evaluation helps to engage more users.
* It is cheaper and faster than conducting full-blown usability testing.

**Disadvantages :**

* Sometimes it is a bit hard for even experts to figure out some problems.
* It becomes hard to find experts to conduct the Heuristics Evaluation.
* We will need few expert evaluators, so that it will become easier for us to stick with usability testing.
* Flaws in design will affect the engagement of users in the product.
* Heuristics testing depends on the expertise level of only a few experts.

**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.

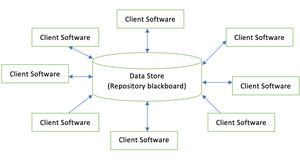
**Taxonomy of Architectural styles:** 

**1] Data 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***

**2] Data flow architectures:** 

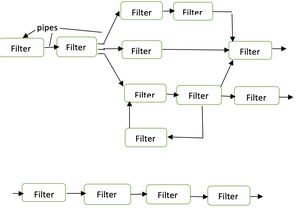
* This kind of architecture is used when input data is transformed into output data through a series of computational manipulative components.
* The figure represents pipe-and-filter architecture since it uses both pipe and filter and it has a set of components called filters connected by lines.
* Pipes are used to transmitting 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.

**Advantages of Data Flow architecture**

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

**The disadvantage of Data Flow architecture**

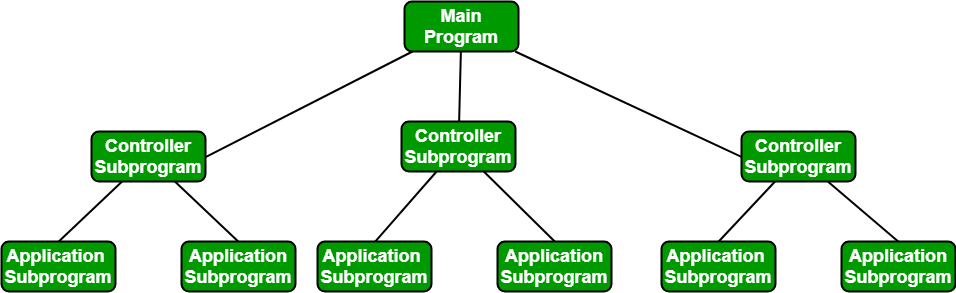
* It frequently degenerates to batch sequential system
* Data flow architecture does not allow applications that require greater user engagement.
* It is not easy to coordinate two different but related streams



***Data Flow architecture***

**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.



**4] 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**

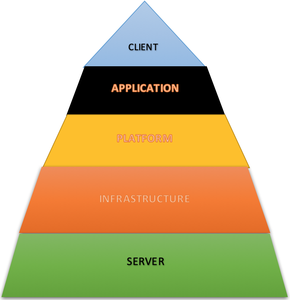
* It enables the designer to separate a challenge into a collection of autonomous objects.
* Other objects are aware of the implementation details of the object, allowing changes to be made without having an impact on other objects.

**5]** **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.

* system.



**Architectural Mapping using Data flow:**

A mapping technique, called structured design, is often characterized as a data flow-oriented design method because it provides a convenient transition from a data flow diagram to software architecture.

The transition from information flow to program structure is accomplished as part of a six step process:

(1) The type of information flow is established,

(2) Flow boundaries are indicated,

(3) The DFD is mapped into the program structure,

(4) Control hierarchy is defined,

(5) The resultant structure is refined using design measures.

(6) The architectural description is refined and elaborated.

**Transform Mapping**

• Transform mapping is a set of design steps that allows a DFD with transform flow characteristics to be mapped into a specific architectural style.

• To map these data flow diagrams into a software architecture, you would initiate the following design steps:

Step 1. Review the fundamental system model.

Step 2. Review and refine data flow diagrams for the software.

Step 3. Determine whether the DFD has transform or transaction flow characteristics.

Step 4. Isolate the transform center by specifying incoming and outgoing flow boundaries.

Step 5. Perform “first-level factoring.”

Step 6. Perform “second-level factoring.”

Step 7. Refine the first-iteration architecture using design heuristic for improved software quality.

# DFD for Library Management System

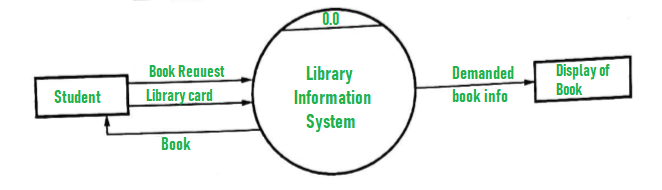
[Data Flow Diagram (DFD)](https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/) depicts the flow of information and the transformation applied when a data moves in and out from a system. The overall system is represented and described using input, processing and output in the DFD. The inputs can be:

* **Book request** when a student requests for a book.
* **Library card** when the student has to show or submit his/her identity as a proof.

The overall processing unit will contain the following output that a system will produce or generate:

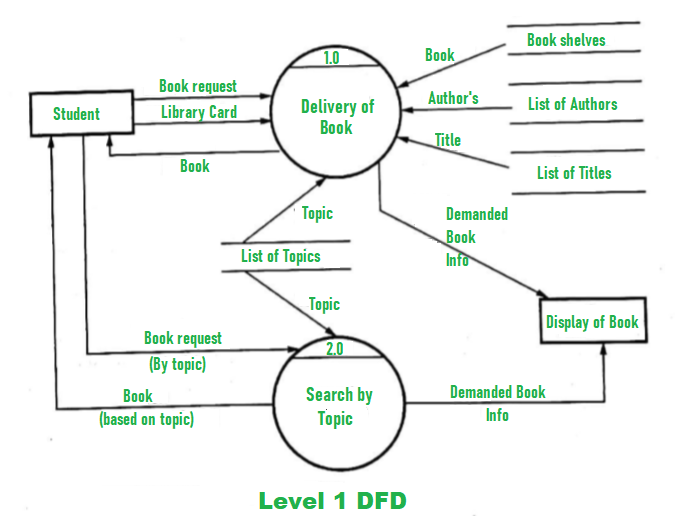
* Book will be the output as the book demanded by the student will be given to them.
* Information of demanded book should be displayed by the library information system that can be used by the student while selecting the book which makes it easier for the student.

1. **Level 0 DFD –**

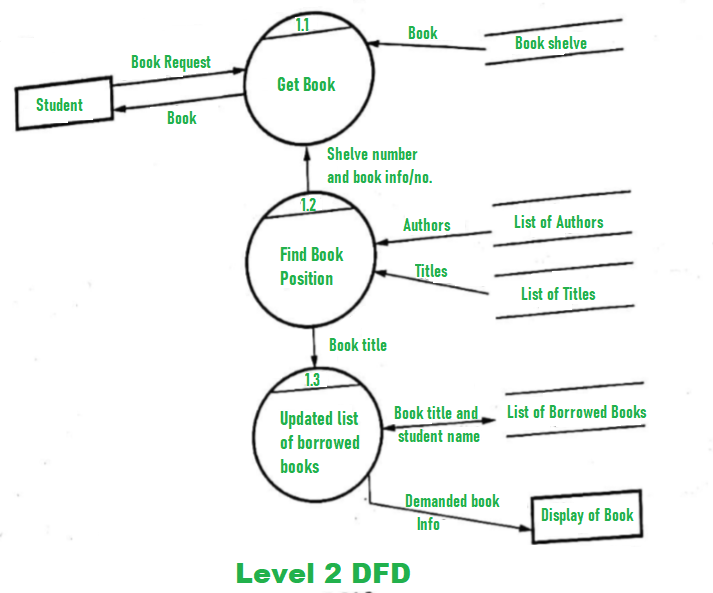


1. **Level 1 DFD –**  
   At this level, the system has to show or exposed with more details of processing.  
   The processes that are important to be carried out are:
   * Book delivery
   * Search by topic

List of authors, List of Titles, List of Topics, the bookshelves from which books can be located are some information that is required for these processes. **Data store** is used to represent this type of information.



1. **Level 2 DFD –**



# User Interface Design

The visual part of a computer application or operating system through which a client interacts with a computer or software. It determines how commands are given to the computer or the program and how data is displayed on the screen.

## Types of User Interface

There are two main types of User Interface:

* Text-Based User Interface or Command Line Interface
* Graphical User Interface (GUI)

**Text-Based User Interface:** This method relies primarily on the keyboard. A typical example of this is UNIX.

### Advantages

* Many and easier to customizations options.
* Typically capable of more important tasks.

### Disadvantages

* Relies heavily on recall rather than recognition.
* Navigation is often more difficult.

**Graphical User Interface (GUI):** GUI relies much more heavily on the mouse. A typical example of this type of interface is any versions of the Windows operating systems.

### Characteristics

|  |  |
| --- | --- |
| **Characteristics** | **Descriptions** |
| Windows | Multiple windows allow different information to be displayed simultaneously on the user's screen. |
| Icons | Icons different types of information. On some systems, icons represent files. On other icons describes processes. |
| Menus | Commands are selected from a menu rather than typed in a command language. |
| Pointing | A pointing device such as a mouse is used for selecting choices from a menu or indicating items of interests in a window. |
| Graphics | Graphics elements can be mixed with text or the same display. |

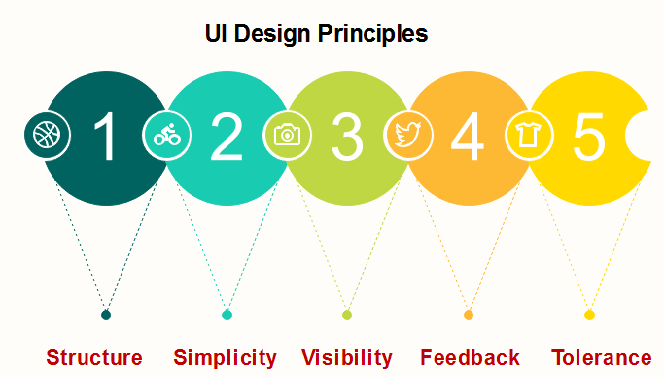
### Advantages

* Less expert knowledge is required to use it.
* Easier to Navigate and can look through folders quickly in a guess and check manner.
* The user may switch quickly from one task to another and can interact with several different applications.

### Disadvantages

* Typically decreased options.
* Usually less customizable. Not easy to use one button for tons of different variations.

## UI Design Principles



**Structure:** Design should organize the user interface purposefully, in the meaningful and usual based on precise, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall user interface architecture.

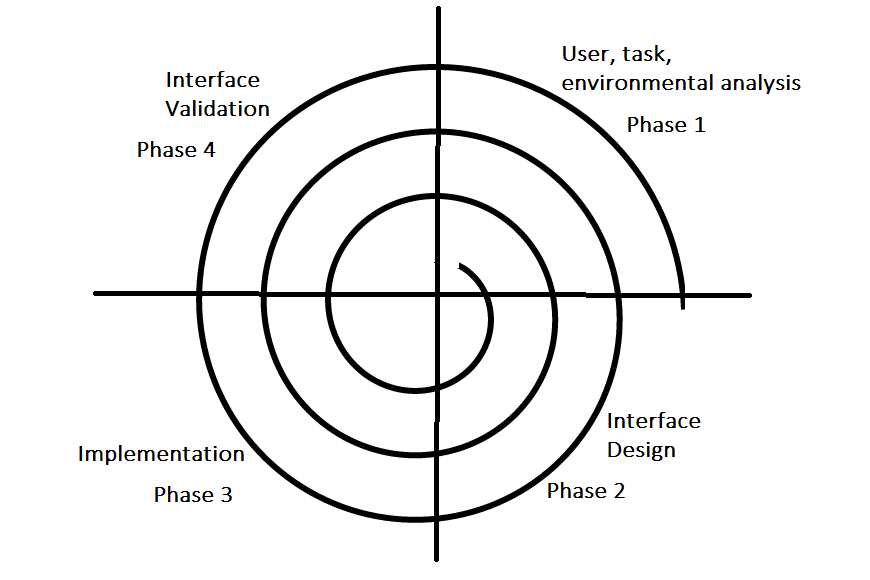
**Simplicity:** The design should make the simple, common task easy, communicating clearly and directly in the user's language, and providing good shortcuts that are meaningfully related to longer procedures.

**Visibility:** The design should make all required options and materials for a given function visible without distracting the user with extraneous or redundant data.

**Feedback:** The design should keep users informed of actions or interpretation, changes of state or condition, and bugs or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.

**Tolerance:** The design should be flexible and tolerant, decreasing the cost of errors and misuse by allowing undoing and redoing while also preventing bugs wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions.

**User Interface Design - Analysis:**



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.

## User Interface design

* User interface design helps in successing most of the software.
* It is part of the user and computer.
* Good interface design is user friendly.

**Types of user interface:**  
  
**1. Command Interpreter**  
Commands help the user to communicate with the computer system.  
  
**2. Graphical User Interfaces (GUI)**

* It is another approach to communicate with system.
* It allows a mouse-based, window-menu-based systems as an interface.

## The Golden Rules

The golden rules are known as interface design principles.  
  
**The golden rule are as follows:**  
  
**1. Place the user in control**

* The interaction should be defined in such a way that the user is not forced to implement unnecessary actions.
* The technical internal details must be hidden from the casual user.
* Design for the direct interaction with objects that appear on the screen.

**2. Reduce the user's memory load**

* The user interface must be designed in such a way that it reduces the demands on the user's short term memory.
* Create the meaningful defaults value as an advantage for the average users in the start of application.
* There must be a reset option for obtaining the default values.
* The shortcut should be easily remembered by the users.
* The interface screen should be friendly to users.

**3. Make the interface consistent**

* The system must allow the user to put task into meaningful context.
* Consistency should be maintained for all the interaction.
* Do not change the past system that is created by the user expectation unless there is a good reason to do that.

## User interface design issues

**The user interface design consist of following four issues:**  
  
**1. Response time of the system**  
Length and variability are the two important characteristic of the system response time.  
  
**2. User help facilities**  
The user of each software system needs the help facility or the user manual for the smooth use of the software.  
  
**3. Error information handling**  
Many error messages and warnings are created which irritate the users as they are not meaningful. Only the critical problems should be handled.  
  
**Error message guidelines are as follows:**

* The language of error message should be described in plain language i.e easily understandable for the users.
* For recovering the error,useful advice should be provided.
* The error message must have an audible or visual indications like beep, short flashing or the special error color.
* The error messages must indicate any negative result so that the user verifies it.
* The wordings of message should not be blamed on the user.

**4. Command labeling**  
The commands and menu labeling must be consistent, easy to understand and learn.