Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT III

Chapter 2 – User Interface Design

Concepts:

Introduction

The user interface is a device or a program (application program) that facilitates the user to communicate with the computer to convey what they want to do or what they want from the computer. The user interface is what the users see when they use the computer.

The user interface can be categorized into three types those are:

1. Command Line Interface

The command-line interface was the only mode of interaction with the computer in the starting days of computing. Here, the user communicates with the computer using textual instructions or commands. For example, if one needs to delete a file from the system, the command would be: del c:\file.txt.

2. Menu Based Interface

It is somewhat easier than the command line interface as users interacting with the menu-based interface 'do not have to learn the command name', nor do they have to put effort into writing the commands. Here, the user does not have to type the commands; thus, the syntax error is automatically avoided.

A very popular example of a menu-based interface is ATM. A user uses menu buttons to select the options from the menu.

3**. Graphical User Interface (GUI)**

Over a period of time, there has been a tremendous evolution in the computer's user interface. Today's computer has a high-resolution screen with pointing devices (mouse). The modern-day computer's interface has windows, scroll bars, text boxes, buttons, icons, and menus. To pick and point out the elements of GUI, we have a mouse.

Golden Rules to Design UI

Theo Mandel, an interface designer, has put up three golden rules in his book on 'interface design', which must be considered by software engineers while designing the user interface as it helps software engineers to design a user-friendly interface., an interface designer, has put up three golden rules in his book on 'interface design', which must be considered by software engineers while designing the user interface as it helps software engineers to design a user-friendly interface.

The golden rules are as follows:

1. Place the User in Control

An interface designer may enforce some constraints to simplify the creation of an interface, but it may result in an annoying interface. So, while introducing the restrictions and constraints, the designer should be intended to simplify the interface. There is much more to placing a user in control while interacting with the system.

Mandel [MAN97] defines a number of design principles that allow the user to maintain control:

1. **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. For example, if spell check is selected in a word-processor menu, the software moves to a spell-checking mode. There is no reason to force the user to remain in spell checking mode if the user desires to make a small text edit along the way

2. **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.

3. **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.

4. **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.

5.**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.

6.**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. For example, an application interface that allows a user to "stretch" an object (scale it in size) is an implementation of direct manipulation.

2. Reduce the User's Memory Load

If the user has to remember more while interacting with the system, it will be more prone to the error. Whenever possible, the system should "remember" pertinent information and assist the user with an interaction scenario that assists recall.

Mandel [MAN97] defines design principles that enable an interface to reduce the user's memory load:

1. Reduce demand on short-term memory: When users are involved in complex tasks, the demand on short-term memory can be significant. The interface should be designed to reduce the requirement to remember past actions and results. This can be accomplished by providing visual cues that enable a user to recognize past actions, rather than having to recall them.

2.Establish meaningful defaults: The initial set of defaults should make sense for the average user, but a user should be able to specify individual preferences. However, a "reset" option should be available, enabling the redefinition of original default values.

3.Define shortcuts that are intuitive: When mnemonics are used to accomplish a system function (e.g., alt-P to invoke the print function), the mnemonic should be tied to the action in a way that is easy to remember (e.g., first letter of the task to be invoked).

4.**The visual layout of the interface should be based on a real-world metaphor**: For example, a bill payment system should use a check book and check register metaphor to guide the user through the bill paying process. This enables the user to rely on well-understood visual cues, rather than memorizing an arcane interaction sequence.

5. Disclose information in a progressive fashion: The interface should be organized hierarchically. That is, information about a 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. An example, common to many word-processing applications, is the underlining function. The function itself is one of a number of functions under a text style menu. However, every underlining capability is not listed. The user must pick underlining, then all underlining options (e.g., single underline, double underline, dashed underline) are presented.

3. Make the Interface Consistent

The interface should present and acquire information in a consistent fashion. This implies that

(1) all visual information is organized according to a design standard that is maintained throughout all screen displays,
(2) input mechanisms are constrained to a limited set that are used consistently throughout the application, and
(3) mechanisms for navigating from task to task are consistently defined and implemented.

Mandel [MAN97] defines a set of design principles that help make the interface consistent:

1.Allow the user to put the current task into a meaningful context: Many interfaces implement complex layers of interactions with dozens of screen images. It is important to provide indicators (e.g., window titles, graphical icons, consistent color coding) that enable the user to know the context of the work at hand. In addition, the user should be able to determine where he has come from and what alternatives exist for a transition to a new task.

2.Maintain consistency across a family of applications: A set of applications (or products) should all implement the same design rules so that consistency is maintained for all interaction.

3.If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so: Once a particular interactive sequence has become a de facto standard (e.g., the use of alt-S to save a file), the user expects this in every application he encounters. A change (e.g., using alt-S to invoke scaling) will cause confusion.
The interface design principles discussed in this and the preceding sections provide basic guidance for a software engineer. In the sections that follow, we examine the interface design process itself.

User Interface Design

The overall process for designing a user interface begins with the creation of different models of system function (as perceived from the outside). The human- and computer-oriented tasks that are required to achieve system function are then delineated; design issues that apply to all interface designs are considered; tools are used to prototype and ultimately implement the design model; and the result is evaluated for quality.

Four different models come into play when a user interface is to be designed. The software engineer creates a **design model**, a human engineer (or the software engineer) establishes a **user model**, the end-user develops a mental image that is often called the user's model or the system perception, and the implementers of the system create a **system image** [RUB88]. Unfortunately, each of these models may differ significantly. The role of interface designer is to reconcile these differences and derive a consistent representation of the interface.

The user model establishes the profile of end-users of the system. To build an effective user interface, "all design should begin with an understanding of the intended users, including profiles of their age, sex, physical abilities, education, cultural or ethnic background, motivation, goals and personality" [SHN90]. In addition, users can be categorized as

• **Novices**: No syntactic knowledge of the system and little semantic knowledge of the application or computer usage in general.

• **Knowledgeable, intermittent users**: Reasonable semantic knowledge of the application but relatively low recall of syntactic information necessary to use the interface.

• **Knowledgeable, frequent users**: Good semantic and syntactic knowledge that often leads to the "power-user syndrome"; that is, individuals who look for shortcuts and abbreviated modes of interaction.

The system perception (user's model) is the image of the system that end-users carry in their heads. For **example,** if the user of a particular word processor were asked to describe its operation, the system perception would guide the response. The accuracy of the description will depend upon the user's profile (e.g., novices would provide a sketchy response at best) and overall familiarity with software in the application domain. A user who understands word processors fully but has worked with the specific word processor only once might actually be able to provide a more complete description of its function than the novice who has spent weeks trying to learn the system.

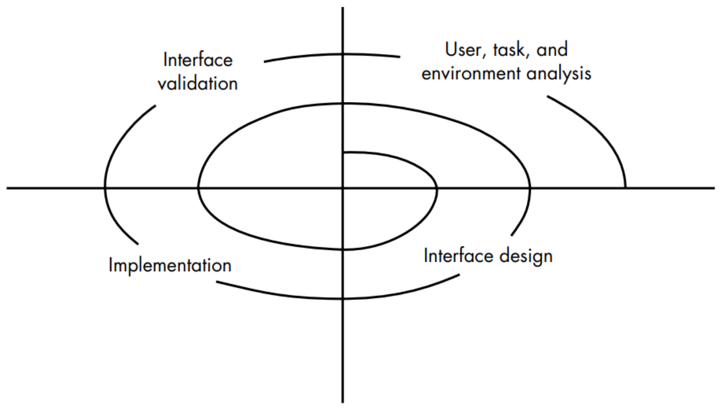
The system image combines the outward manifestation of the computer-based system (the look and feel of the interface), coupled with all supporting information (books, manuals, videotapes, help files) that describe system syntax and semantics. When the system image and the system perception are coincident, users generally feel comfortable with the software and use it effectively.

The User Interface Design Process

The design process for user interfaces is iterative and can be represented using a spiral model. The user interface design process encompasses four distinct framework activities [MAN97]:

1. User, task, and environment analysis and modeling
2. Interface design
3. Interface construction
4. Interface validation

The user
interface
design process



The spiral shown in Figure implies that each of these tasks will occur more than once, with each pass around the spiral representing additional elaboration of requirements and the resultant design. In most cases, the implementation activity involves prototyping—the only practical way to validate what has been designed.

1. User, task, and environment analysis and modeling

Before designing any solution, you need to know what actually you have to design. In this framework, you have to study the user's profile who will use this interface.

In this phase, the designer must understand the people who will interact with the system using the interface. What type of task the user has to perform to achieve the goal? Further, the designer must investigate the content that has to present as a part of the interface**.** The designer must also analyze the environment where the user will interact with the system via an 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

After completing the interface analysis, the designer identifies the task that the end-user requires. Interface designing is also an iterative process. The designer first identifies the objects and the operations performed on those objects.

The designer also has to define the events that would change the state of the user interface. Further, the designer has to outline each state of the user interface as it would appear to the end-user The designer also has to define the events that would change the state of the user interface. Further, the designer has to outline each state of the user interface as it would appear to the end-user.

3. Interface construction

After identifying the objects, operations and events the designer creates a prototype. This prototype helps in the evaluation of the real-world scenario. This process is also iterative and the construction continues till the prototype is approved for conducting real-world scenarios.

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

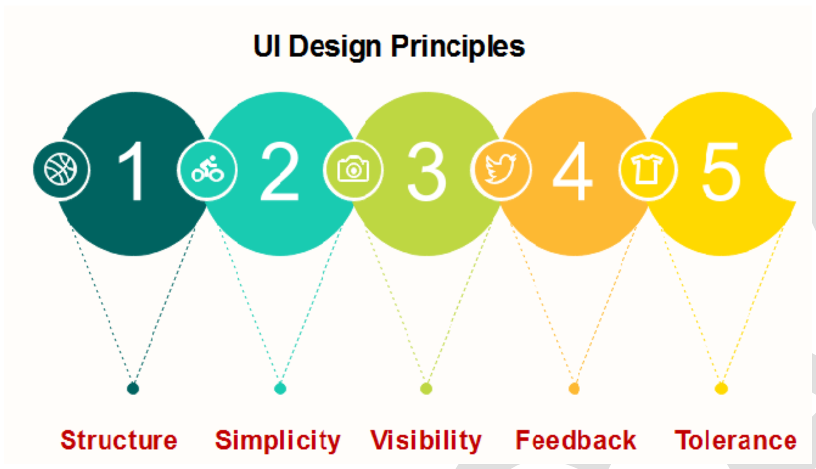
The validation phase is performed to see whether the interface can perform user tasks along with all the variations in the real world. Like, as to whether the interface can perform all general user tasks? Is it easy to use, and understand and we can accept it as a useful tool?

UI Design Principles

User Interface Design is the design of the interface or system which is directly accessible by the user and they interact with in order to do a task. It establishes the way with which the user will interact with the product.

Its main aim is to enhance the appearance of the product, the quality of technology used and the usability of the product. It refers to the software or the hardware of the system which the user can see and also the various ways or commands to control or use the product.

It focuses on the looks or how the app or software is looking. Attributes like theme, animations, colors, etc. constitute the user interface.



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.