User Interface Design

User Interface Design

- User interface design creates an effective communication medium between a human and a computer.
- Following a set of interface design principles, design identifies interface objects and actions and then creates a screen layout that forms the basis for a user interface prototype.

User Interface Design

- If software is difficult to use,
- if it forces you to mistakes, or
- if it frustrates your efforts to accomplish your goals,
  you won’t like it,
- Regardless of the computational power it exhibits, the content it delivers, or the functionality it offers.
- Thus interface has to be right because it moulds a user’s perception of the software and facilitates the use of software

User Interface Design

- User interface design begins with the identification of users, tasks, and environmental requirements.
- Once user tasks have been identified, user scenarios are created and analysed to define a set of interface objects and actions.

User Interface Design

- These form the basis for the creation of screen layout that depicts graphical design and placement of icons, definition of descriptive screen text, specification and titling of windows, and specification of major and minor menu items.
- Tools are used to prototype and ultimately implement the design model, and the result is evaluated for quality.

User Interface Design

- **Easy to learn?**
- **Easy to use?**
- **Easy to understand?**

User Interface Design

<u>Typical Design Errors</u>

- **lack of consistency**
- **too much memorization**
- **no guidance / help**
- **no context sensitivity**
- **poor response**
- **unfriendly**

Golden Rules

- Three Golden Rules for designing user interface are :

1: Place the user in control

2: Reduce the user’s memory load

3: Make the interface consistent

These golden rules actually form the basis for a set of user interface design principles

1: Place the user in control

- **Define interaction modes in a way that does not force a user into unnecessary or undesired actions.**

An interaction mode is the current state of the interface.

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. The user should be able to enter and exit the mode with little or no effort.

1:Place the user in control

- **Provide for flexible interaction.**

  As different users have different interaction preferences, therefore, choices should be provided. For example, software might allow a user to interact via keyboard commands, mouse movement, a digitizer pen, a multi touch screen, or voice recognition commands.

- **Allow user interaction to be interruptible**

  Even when involved in a sequence of actions, the user should be able to interrupt the sequence to do something else (without losing the work that had been done). The user should also be able to “undo” any action.

1: Place the user in control

- **Streamline interaction as skill levels advance and allow the interaction to be customized.** Users often find that they perform the same sequence of interactions repeatedly. It is worthwhile to design a “macro” mechanism that enables an advanced user to customize the interface to facilitate interaction.
- **Hide technical internals from the casual user.** The user interface should move the user into the virtual world of the application. The user should not be aware of the operating system, file management functions etc

1: Place the user in control

- **Design for direct interaction with objects that appear on the screen.**

The user feels a sense of control when able to manipulate the objects that are necessary to perform a task in a manner similar to what would occur if the object were a physical thing. 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

- A well-designed user interface does not rely on a user’s memory because the more
  a user has to remember, the more error-prone the interaction will be.
- Whenever possible, the system should “remember” pertinent information and assist the user with an interaction scenario that assists recall. The design principles that enable an interface to reduce the user’s memory load are:

2: Reduce the User’s Memory Load

A set of design principles that help to reduce the load on user’s memory:

- **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, inputs, 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: Reduce the User’s Memory Load

- **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.
- **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).

2: Reduce the User’s Memory Load

- **The visual layout of the interface should be based on a real-world metaphor.**

For example, a bill payment system should use a checkbook 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 interaction sequence.

2: Reduce the User's Memory Load

- **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.

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 design rules that are maintained throughout all screen displays

(2) Input mechanisms are constrained to a limited set that is used consistently throughout the application, and

(3) Mechanisms for navigating from task to task are consistently defined and implemented.

3:Make the Interface Consistent

A set of design principles that help make the interface consistent:

- **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.

3:Make the Interface Consistent

- **Maintain consistency across a family of applications.** A family of applications (i.e., a product line) should implement the same design rules so that consistency is maintained for all interaction.
- **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 will cause confusion.

USER INTERFACE ANALYSIS & DESIGN

- The overall process for analysing and designing a user interface begins with the creation of different models of system function (as perceived from the outside).
- You begin by outlining the human and computer oriented tasks that are required to achieve system function and then considering the design issues that apply to all interface designs.
- Tools are used to prototype and ultimately implement the design model, and the result is evaluated by end users for quality.

Interface Design Models

- Four different models come into play when a user interface is analyzed and designed

  1: Design model – Created by a software engineer

  2: User model – Established by a human engineer or software engineer

  3: Implementation model – Created by the software implementers

  4: User's mental model / *system perception*
  – Developed by the end-user when interacting with the application

- The role of the interface designer is to reconcile these differences and derive a consistent representation of the interface

1: Design Model

- Derived from the analysis model of the requirements
- Incorporates data, architectural, interface, and procedural representations of the software
- Constrained by information in the requirements specification that helps to define the users of the system

2: User Model

- Establishes the profile of the end-users of the system
  - Based on age, gender, physical abilities, education, cultural or ethnic background, motivation, goals, and personality
- Considers <u>syntactic</u> knowledge of the user
  - The mechanics of interaction that are required to use the interface effectively
- Considers <u>semantic</u> knowledge of the user
  - The underlying sense of the application; an understanding of the functions that are performed, the meaning of input and output, and the objectives of the system
- Categorizes users as
  - Novices
    - No syntactic knowledge of the system, little semantic knowledge of the application, only general computer usage
  - Knowledgeable, intermittent users
    - Reasonable semantic knowledge of the system, low recall of syntactic information to use the interface
  - Knowledgeable, frequent users
    - Good semantic and syntactic knowledge (i.e., power user), look for shortcuts and abbreviated modes of operation

3:Implementation Model/System Image

- Consists of the look and feel of the interface combined with all supporting information (from books, videos, help files) that describe system syntax and semantics
- When the implementation model and the user's mental model are coincident; then users feel comfortable with the software and use it effectively
- The design model must have been developed to accommodate the information contained in the user model, and the system image must accurately reflect syntactic and semantic information about the interface .

4:User's Mental Model

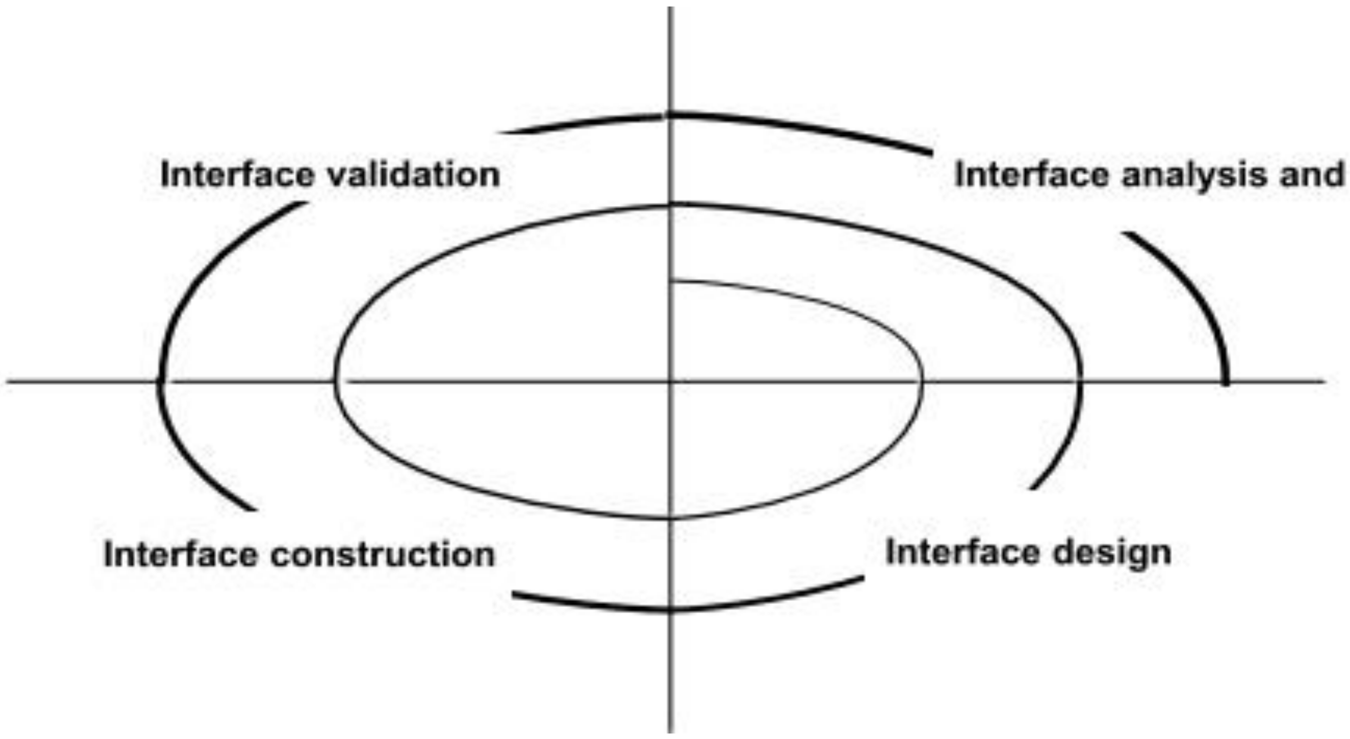
- Often called the user's system perception
- Consists of the perception of the system that users carry in their minds.
- For example, if the user of a particular word processor were asked to describe its operation, the system perception would guide the response. Accuracy of the description depends upon the user’s profile and overall familiarity with the software in the application domain.

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.

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

The User Interface Design Process



1. User, task, and environment analysis and modeling (Interface analysis and modelling)

- The initial analysis activity focuses on the profile of the users who will interact with the system.
- Skill level, business understanding, and general receptiveness to the new system are defined; and different user categories are defined. For each user category, requirements are elicited.

1. User, task, and environment analysis and modeling (Interface analysis and modelling)

- Once general requirements have been defined, a more detailed task analysis is conducted.
- Those tasks that the user performs to accomplish the goals of the system are identified, described, and elaborated (over a number of iterative passes through the spiral).

1. User, task, and environment analysis and modeling (Interface analysis and modelling)

- Finally, analysis of the user environment focuses on the characteristics of the physical work environment (e.g., location, lighting, position constraints).
- The information gathered as part of the analysis action is used to create an analysis model for the interface. Using this model as a basis, the design activity commences.

2. Interface design

- The goal of interface design is to define a set of interface objects and actions (and their screen representations) that enable a user to perform all defined tasks in a manner that meets every usability goal defined for the system.

3. Interface construction,

- Interface construction
  - Begins with a prototype that enables usage scenarios to be evaluated
  - Continues with development tools to complete the construction

4. Interface validation

- Interface validation, focuses on

  i) The ability of the interface to implement every user task correctly, to accommodate all task variations, and to achieve all general user requirements

  ii) The degree to which the interface is easy to use and easy to learn

  iii) The users' acceptance of the interface as a useful tool in their work

1: Interface Analysis

1.1:Elements of the User Interface

- To perform user interface analysis, the practitioner needs to study and understand four elements
  - The <u>people(end users)</u> who will interact with the system through the interface
  - The <u>tasks</u> that end users must perform to do their work
  - The <u>content</u> that is presented as part of the interface
  - The <u>work environment</u> in which these tasks will be conducted

1.2: User Analysis

- The analyst strives to converge the end user's mental model and the design model by understanding
  - The users themselves
  - How these people use the system
- Information can be obtained from
  - <u>User interviews</u> with the end users
  - <u>Sales input</u> from the sales people who interact with customers and users on a regular basis
  - <u>Marketing input</u> based on a market analysis to understand how different population segments might use the software
  - <u>Support input</u> from the support staff who are aware of what works and what doesn't, what users like and dislike, what features generate questions, and what features are easy to use
- A set of questions should be answered during user analysis (see next slide)

1.2: User Analysis Questions

1) Are the users trained professionals, technicians, clerical or manufacturing workers?
2) What level of formal education does the average user have?
3) Are the users capable of learning on their own from written materials or have they expressed a desire for classroom training?
4) Are the users expert typists or are they keyboard phobic?
5) What is the age range of the user community?
6) Will the users be represented predominately by one gender?
7) How are users compensated for the work they perform or are they volunteers?
8) Do users work normal office hours, or do they work whenever the job is required?
9) Is the software to be an integral part of the work users do, or will it be used only occasionally?
10) What is the primary spoken language among users?
11) What are the consequences if a user makes a mistake using the system?
12) Are users experts in the subject matter that is addressed by the system?
13) Do users want to know about the technology that sits behind the interface?

1.3:Task Analysis and Modeling

- The goal of task analysis is to know and understand:
  - What work will the user perform in specific circumstances?
  - What tasks and subtasks will be performed as the user does the work?
  - What specific problem domain objects will the user manipulate as work is performed?
  - What is the sequence of work tasks—the workflow?
  - What is the hierarchy of tasks?

- To answer these questions, these techniques are applied to the user interface.

I. Use Cases.
II. Task Elaboration.
III. Object Elaboration.
IV. Workflow Analysis.
V. Hierarchical Representation.

1.3:Task Analysis and Modeling

i. Use Cases

- Use case describes the manner in which an actor (in the context of user interface design, an actor is always a person) interacts with a system.
- When used as part of task analysis, the use case is developed to show how an end user performs some specific work-related task.
- In most instances, the use case is written in an informal style (a simple paragraph).

1.3:Task Analysis and Modeling

ii. Task Elaboration.

- Stepwise elaboration (also called functional decomposition or stepwise refinement) is used as a mechanism for refining the processing tasks that are required for software to accomplish some desired function.
- Task analysis for interface design uses an elaborative approach that assists in understanding the human activities the user interface must accommodate.
- You should define and classify the human tasks that are required to accomplish the goal of the system or app

1.3:Task Analysis and Modeling

iii. Object Elaboration.

- Rather than focusing on the tasks that a user must perform, you can examine the use case and other information obtained from the user and extract the physical objects that are used by the interior designer.
- These objects can be categorized into classes. Attributes of each class are defined, and an evaluation of the actions applied to each object provides a list of operations.

1.3:Task Analysis and Modeling

iv. Workflow Analysis.

- When a number of different users, each playing different roles, makes use of a user interface, it is sometimes necessary to go beyond task analysis and object elaboration and apply *workflow analysis.*
- This technique allows you to understand how a work process is completed when several people (and roles) are involved.

1.3:Task Analysis and Modeling

v. Hierarchical Representation

- A process of elaboration occurs as you begin to analyse the interface.
- Once workflow has been established, a task hierarchy can be defined for each user type. The hierarchy is derived by a stepwise elaboration of each task identified for the user.

1.4: Content Analysis / Analysis of Display Content

- The user tasks lead to the presentation of a variety of different types of content
- The display content may range from character-based reports, to graphical displays, to multimedia information
- Display content may be
  1. generated by components (unrelated to the interface) in other parts of an application,
  2. acquired from data stored in a database that is accessible from the application, or
  3. transmitted from systems external to the application in question
- The format and aesthetics of the content (as it is displayed by the interface) needs to be considered. A set of questions that should be answered during content analysis:

1.4: Content Analysis / Analysis of Display Content

1) Are various types of data assigned to consistent locations on the screen (e.g., photos always in upper right corner)?
2) Are users able to customize the screen location for content?
3) Is proper on-screen identification assigned to all content?
4) Can large reports be partitioned for ease of understanding?
5) Are mechanisms available for moving directly to summary information for large collections of data?
6) Is graphical output scaled to fit within the bounds of the display device that is used?
7) How is color used to enhance understanding?
8) How are error messages and warnings presented in order to make them quick and easy to see and understand?

1.5:Analysis of the Work Environment

- People do not perform their work in isolation. They are influenced by:

  the activity around them, the physical characteristics of

  the workplace, the type of equipment they are using, and

  the work relationships they have with other people.

- In some applications the user interface for a computer-based system is placed in a “user-friendly location” (e.g., proper lighting, good display height, easy keyboard access),
- For other applications (e.g., a factory floor or an airplane cockpit), lighting may be suboptimal, noise may be a factor, a keyboard or mouse or touch screen may not be an option, display placement may be less than ideal.

1.5: Analysis of the Work Environment

- Thus software products need to be designed to fit into the work environment, otherwise they may be difficult or frustrating to use
- Factors to consider include:
  - Type of lighting
  - Display size and height
  - Keyboard size, height and ease of use
  - Mouse type and ease of use
  - Surrounding noise
  - Space limitations for computer and/or user
  - Weather or other atmospheric conditions
  - Temperature or pressure restrictions
  - Time restrictions (when, how fast, and for how long)

1.5:Analysis of the Work Environment

- In addition to physical environmental factors, the workplace culture also comes into play.
- Will system interaction be measured in some manner (e.g., time per transaction or accuracy of a transaction)?

  Will two or more people have to share information before

  an input can be provided?

  How will support be provided to users of the system?
- These and many related questions should be answered before the interface design commences.

2:Interface Design Steps

- Once interface analysis has been completed, all tasks (or objects and actions) required by the end user have been identified in detail, the interface design activity commences.
- Interface design, like all software engineering design, is an iterative process. Each user interface design step occurs a number of times, elaborating and refining information developed in the preceding step.
- Although many different user interface design models have been proposed, all suggest some combination of the following steps:

1) Using information developed during user interface analysis, Define user interface <u>objects</u> and <u>actions</u> (operations) applied to them
2) Identify events (user actions) that will cause the state of the user interface to change
3) Depict the representation of each state
4) Indicate how the user interprets the state of the system from information provided through the interface

Interface Design Steps

- During all of these steps, the designer must
  - Always follow the three golden rules of user interfaces
  - Model how the interface will be implemented
  - Consider the computing environment (e.g., display technology, operating system, development tools) that will be used

Design Issues to Consider

- Four common design issues usually surface in any user interface :

  1: System response time

  2: User help facilities

  3: Error information handling

  4: Menu and command labeling
- Many designers do not address these issues until late in the design or construction process
  - This results in unnecessary iterations, project delays, and customer frustration
  - Therefore, it is better to consider each issue at the beginning of software design, when changes are easy and costs are low.

1: System response time

- Response time has two important characteristics: Length and variability.
- If system response is too long, user frustration and stress are inevitable.
- *Variability* refers to the deviation from average response time. Low variability enables the user to establish an interaction rhythm, even if response time is relatively long. For example, a 1-second response to a command will often be preferable to a response that varies from 0.1 to 2.5 seconds. When variability is significant, the user wonders whether something “different” has occurred behind the scenes.

2: User help facilities

- When is it available, how is it accessed, how is it represented to the user, how is it structured,
- Almost every user of an interactive, computer-based system requires help now and then. Modern software should provide online help facilities that enable a user to get a question answered or resolve a problem without leaving the interface.

3: Error information handling

- How meaningful to the user, how much descriptive of the problem
- An error message or warning should have the following characteristics (guidelines):

1. The message should describe the problem in <u>plain language</u> that a typical user can understand
2. The message should provide <u>constructive advice</u> for recovering from the error
3. The message should indicate any <u>negative consequences</u> of the error (e.g., potentially corrupted data files) so that the user can check to ensure that they have not occurred (or correct them if they have)
4. The message should be accompanied by an <u>audible or visual cue</u> such as a beep, momentary flashing, or a special error color
5. The message should be <u>non-judgmental</u> that is message should never blame the user

4: Menu and command labeling

- Earlier typed command was the most common mode of interaction between user and system software. Today, the use of window-oriented, point-and pick interfaces has reduced reliance on typed commands.
- A number of design issues arise when typed commands or menu labels are provided as a mode of interaction:

4: Menu and command labeling

Questions/design issues for Menu Labeling and Typed Commands

- Will every menu option have a corresponding command?
- What form will a command take? Options include a control sequence (e.g., alt-P), function keys, or a typed word.
- How difficult will it be to learn and remember the commands?
- What can be done if a command is forgotten?
- Can commands be customized or abbreviated by the user?
- Are menu labels self-explanatory within the context of the interface?
- Are submenus consistent with the function implied by a master menu item?
- Have appropriate conventions for command usage been established across a family of applications?

4: Menu and command labeling

Application Accessibility.

- As computing applications become ubiquitous, software engineers must ensure that interface design encompasses mechanisms that enable easy access for those with special needs.
- *Accessibility* for users who may be physically challenged (with visual, hearing, mobility, speech, and learning impairments) is imperative for ethical, legal, and business reasons.

4: Menu and command labeling

Internationalization.

- The challenge for interface designers is to create “globalized” software. That is, user interfaces should be designed to accommodate a generic core of functionality that can be delivered to all who use the software.
- A variety of internationalization guidelines are available for this purpose. These guidelines address broad design issues (e.g., screen layouts may differ in various markets) and discrete implementation issues (e.g., different alphabets may create specialized labelling and spacing requirements).

The *Unicode* standard has been developed to address the challenge of managing dozens of natural languages with hundreds of characters and symbols

Design and Prototype Evaluation

- The prototyping approach is effective, but is it possible to evaluate the quality of a user interface before a prototype is built?
- If you identify and correct potential problems early, the number of loops through the evaluation cycle will be reduced and development time will shorten.
- If a design model of the interface has been created, a number of evaluation criteria can be applied during early design reviews:

Design and Prototype Evaluation
Evaluation criteria:

1: The amount of learning required by the users

- The length and complexity of the requirements model or written specification of the system and its interface provide an indication of the amount of learning required by users of the system

2: The interaction time and overall efficiency

- Derived from the number of user tasks specified and the average number of actions per task

3: The memory load on users

- Derived from the number of actions, tasks, and system states

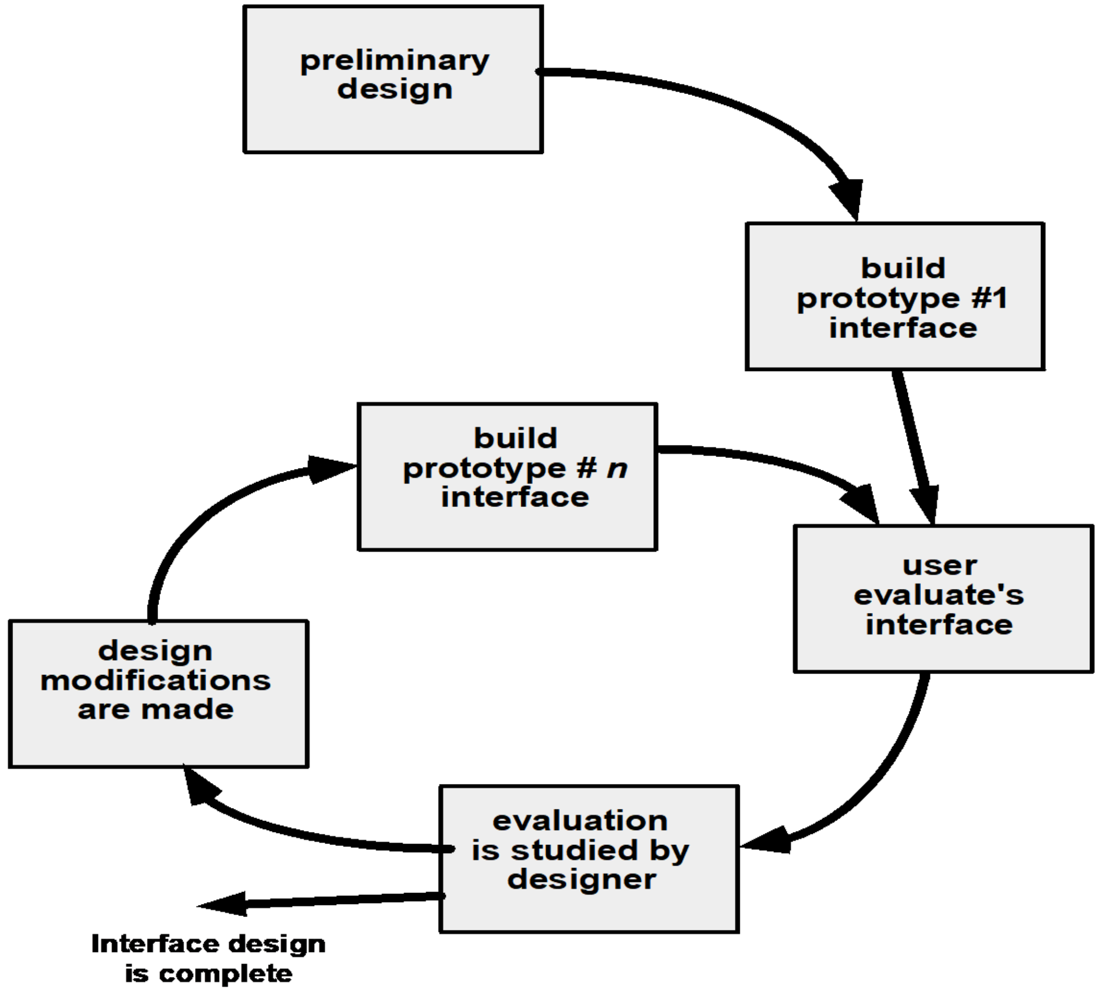
4: The complexity of the interface and the degree to which it will be accepted by the user

- Derived from the interface style, help facilities, and error handling procedures

Design and Prototype Evaluation

- Prototype evaluation can range from an informal test drive to a formally designed study using statistical methods and questionnaires
- The prototype evaluation cycle consists of prototype creation followed by user evaluation and back to prototype modification until all user issues are resolved
- The prototype is evaluated for
  - Satisfaction of user requirements
  - Conformance to the three golden rules of user interface design
  - Reconciliation of the four models of a user interface

Design Evaluation Cycle


preliminary design
build prototype #1 interface
build prototype # *n* interface
user evaluate's interface
design modifications are made
evaluation is studied by designer
Interface design is complete