

# Software Engineering

## User Interface Design



**Puneet Kumar Jain**

CSE Department

**National Institute of Technology Rourkela**

# Reference



- Most of the content of the presentation belongs to the following books:
  - R.S. Pressman & Associates, Inc. *Software Engineering: A Practitioner's Approach, 6/e*
  - Rajib Mall, *Introduction to Software Engineering*

# Content

- Introduction
- Characteristics of a good user interface design
- Golden rules of user interface design
- Reconciling four different models
- User interface analysis
- User interface design
- User interface evaluation
- Types of user interfaces

# Introduction



- In the early days of computer, no software product had any user interface:
  - all computers were batch systems
  - no interactions with users were supported.
- We know that things are very different now:
  - almost every software product is highly interactive.

# Introduction



- Users interact with a software product through its user interface:
  - user-interface portion of any software is directly relevant to the users.
  - many users judge a software from its user interface.
- User interface design:
  - a practical and important problem.

# Introduction



- Aesthetics apart, a difficult to use interface:
  - leads to higher levels of user errors
  - leads to user dissatisfaction.
- Users become particularly irritated when a system behaves in unexpected ways,
  - issued commands do not carry out actions according to intuitive expectations of users.

# Introduction



- A significant portion of the total development effort:
  - spent in developing the user interface.
- For many interactive applications:
  - as much as 50% of the total development effort is spent on the user interface part.

# Background



- Interface design focuses on the following
  - The design of interfaces between software components
  - The design of interfaces between the software and other nonhuman producers and consumers of information
  - The design of the interface between a human and the computer
- Graphical user interfaces (GUIs) have helped to eliminate many of the most horrific interface problems
- However, some are still difficult to learn, hard to use, confusing, counterintuitive, unforgiving, and frustrating
- User interface analysis and design has to do with the study of people and how they relate to technology

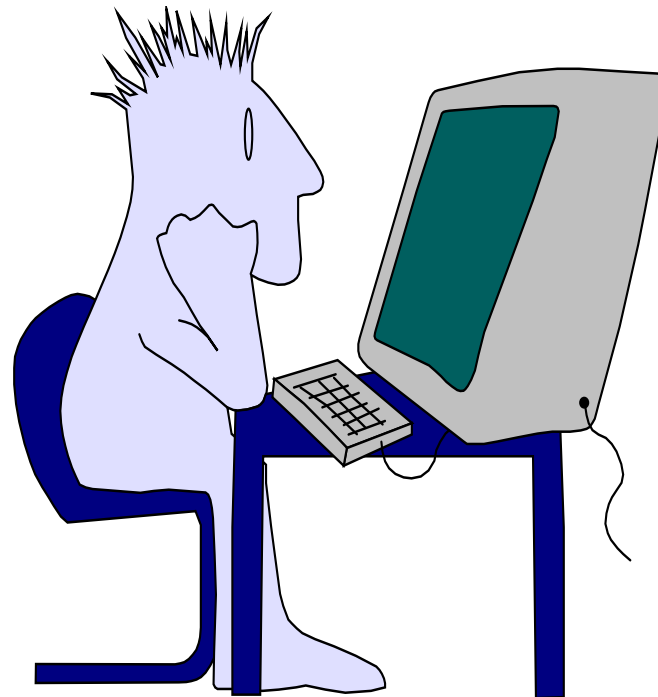


# Interface Design

**Easy to learn**

**Easy to use**

**Easy to understand**



# Characteristics of Good User Interfaces

- **Speed of learning**
- **Speed of use**
- **Speed of recall**
- **Minimum scope of error**
- **Attractiveness**
- **Consistency**
- **Feedback facility**
- **Support for multiple skill level**
- **User guidance and on-line help**

# Speed of learning

- A good user interface should be simple to learn.
- A good user interface should not require
  - users to memorize commands.
- An important factor affecting the speed of learning is consistency:
  - Once, a user learns about a command, should be able to use similar commands
- Users can learn about an interface faster, if it is based on:
  - day-to-day real life examples (aka metaphors)
  - use concepts with which users are already familiar.

# Speed of learning

- For example, interface of a text editor:
  - can use concepts similar to writing on paper:
    - such as cutting lines and paragraphs and pasting it at other places,
    - users can immediately relate to it.
- Also, learning is facilitated by:
  - intuitive command names
  - symbolic command issue procedures.

# Speed of use



- Speed of use is determined by:
  - the time and effort necessary to initiate and execute different commands.
  - The time and user effort necessary to execute different commands should be minimal.
  
- Examples of bad interfaces:
  - users required to type in lengthy commands
  - command issue involves moving the mouse to widely different areas of the screen
  - can slow down the operating speed of users.

# Speed of recall

- Once users learn how to use an interface:
  - their speed of recall about using the interface should be maximized.
- The speed of recall is improved if the interface is based on:
  - metaphors
  - symbolic command issue procedures
  - intuitive command names.

# Error rate

- A good user interface should
  - minimize the scope of committing errors.
- Error rate can be easily measured:
  - count the errors committed by different users.
- Error monitoring can be automated:
  - instrument user interface with monitoring code
  - record the frequency and types of errors committed by different users
  - later display statistics of various kinds of user errors.

# How to reduce error possibilities?

- Consistency of command names,
- Consistency of command issue procedures,
- Consistency in behaviour of similar commands
- Simplicity of command issue procedure, etc.



# Error Recovery (Undo facility)

- All categories of users commit errors.
  - A good interface should allow users to undo mistakes..
- Users are inconvenienced:
  - if they can not recover from even simple errors.

# Attractiveness

- A good user interface should be attractive:
  - An attractive user interface catches user attention and fancy.
  - In this respect,
    - graphics-based user interfaces have a definite advantage over text-based interfaces.

# Consistency

- Consistency of commands is very desirable.
  - allow users to generalize the knowledge about one aspect of the interface to another.
- Consistency helps in:
  - speed of learning,
  - speed of recall,
  - also helps in reduction of error rate.

# Feedback



- A good user interface must provide constant feedback to user actions:
  - For example, if any user request takes more than few seconds to process,
    - the user must be informed that his/her request is still being processed.
- In the absence of any response from the computer for a long time:
  - a novice user might even start recovery/shutdown procedures in panic.

# Support for multiple skill levels

- A good user interface:
  - should support different levels of sophistication in command issue procedures:
  - users with different experience levels prefer different types of interfaces.
- Experienced users are more concerned about speed of command issue:
  - whereas novice users pay prime importance to usability aspects.
- As users become familiar with an interface:
  - look for faster command issue procedures such as "hot-keys", "macros", etc.

# User Guidance and On-line Help

- Users might need guidance
  - or seek help from the system.
  
- User Guidance is provided through two broad category of methods:
  - On-line help system
  - Guidance and error messages produced
    - in response to user actions.

# On-line Help System

- Generic help messages are not very useful:
  - on-line help messages should be tailored to the context in which help is invoked.
  
- A good on-line help should:
  - provide messages in context-dependent way.
    - keep track of what a user is doing
  - help messages should be tailored to user's experience level.
  - should take advantage of graphics capabilities of the screen
    - not just be a copy of the user manual.

# Guidance Messages

- The guidance messages should be carefully designed:
  - prompt the user:
    - next actions he/she might take,
    - current status of the system,
    - progress made so far in processing the command



# Error Messages

- Error messages should be polite.
- Error messages should not be associated with noise:
  - might embarrass the user.
- The messages should suggest how a given error can be rectified.
- If appropriate,
  - the user should be given the option of invoking on-line help
  - to find out more about the error situation.



Improper Error Message

**File not found!**

# Guidelines for Error Messages

- The message should describe the problem in plain language that a typical user can understand
- The message should provide constructive advice for recovering from the error
- The message should indicate any negative consequences of the error (e.g., potentially corrupted data files)
- The message should be accompanied by an audible or visual cue such as a beep, momentary flashing, or a special error color
- The message should be non-judgmental
  - The message should never place blame on the user

An effective error message philosophy can do much to improve the quality of an interactive system and will significantly reduce user frustration when problems do occur

# Golden Rules

- Place the user in control
- Reduce the user's memory load
- Make the interface consistent

# Place the User in Control

- Define interaction modes in a way that does not force a user into unnecessary or undesired actions
  - The user shall be able to enter and exit a mode with little or no effort (e.g., spell check → edit text → spell check)
- Provide for flexible interaction
  - The user shall be able to perform the same action via keyboard commands, mouse movement, or voice recognition
- Allow user interaction to be interruptible and "undo"able
  - The user shall be able to easily interrupt a sequence of actions to do something else (without losing the work that has been done so far)
  - The user shall be able to "undo" any action

# Place the User in Control

- Streamline interaction as skill levels advance and allow the interaction to be customized
  - The user shall be able to use a macro mechanism to perform a sequence of repeated interactions and to customize the interface
- Hide technical internals from the casual user
  - The user shall not be required to directly use operating system, file management, networking. etc., commands to perform any actions. Instead, these operations shall be hidden from the user and performed "behind the scenes" in the form of a real-world abstraction
- Design for direct interaction with objects that appear on the screen
  - The user shall be able to manipulate objects on the screen in a manner similar to what would occur if the object were a physical thing (e.g., stretch a rectangle, press a button, move a slider)

# Reduce the User's Memory Load

- Reduce demand on short-term memory
  - The interface shall reduce the user's requirement to remember past actions and results by providing visual cues of such actions
- Establish meaningful defaults
  - The system shall provide the user with default values that make sense to the average user but allow the user to change these defaults
  - The user shall be able to easily reset any value to its original default value
- Define shortcuts that are intuitive
  - The user shall be provided mnemonics (i.e., control or alt combinations) that tie easily to the action in a way that is easy to remember such as the first letter

# Reduce the User's Memory Load

- The visual layout of the interface should be based on a real world metaphor
  - The screen layout of the user interface shall contain well-understood visual cues that the user can relate to real-world actions
- Disclose information in a progressive fashion
  - When interacting with a task, an object or some behavior, the interface shall be organized hierarchically by moving the user progressively in a step-wise fashion from an abstract concept to a concrete action (e.g., text format options → format dialog box)

The more a user has to remember, the more error-prone interaction with the system will be

# Make the Interface Consistent

- The interface should present and acquire information in a consistent fashion
  - All visual information shall be organized according to a design standard that is maintained throughout all screen displays
  - Input mechanisms shall be constrained to a limited set that is used consistently throughout the application
  - Mechanisms for navigating from task to task shall be consistently defined and implemented
- Allow the user to put the current task into a meaningful context
  - The interface shall provide indicators (e.g., window titles, consistent color coding) that enable the user to know the context of the work at hand
  - The user shall be able to determine where he has come from and what alternatives exist for a transition to a new task



# Make the Interface Consistent

- Maintain consistency across a family of applications
  - A set of applications performing complimentary functionality shall all 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., alt-S to save a file), the application shall continue this expectation in every part of its functionality

# Reconciling Four Different Models

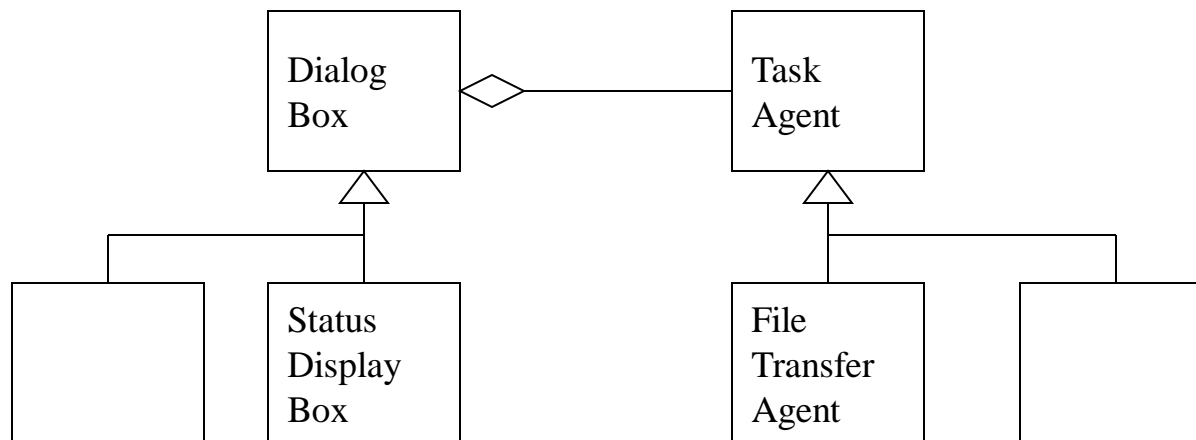
- Four different models come into play when a user interface is analyzed and designed
  - **User profile model** – Established by a human engineer or software engineer
  - **Design model** – Created by a software engineer
  - **Implementation model** – Created by the software implementers
  - **User's mental model** – Developed by the 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

# User Profile 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 syntactic knowledge of the user
  - The mechanics of interaction that are required to use the interface effectively
- Considers semantic 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

# 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 define the user of the system



# Implementation Model

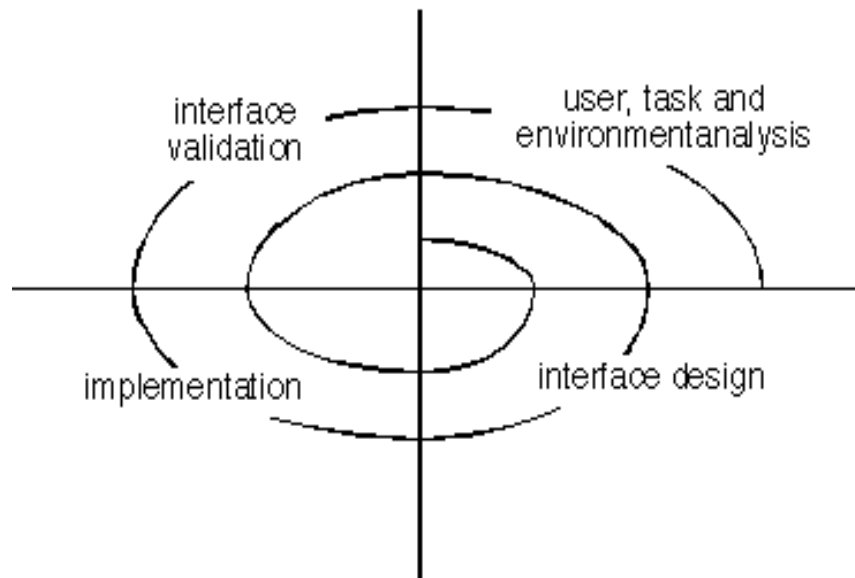
- Consists of the look and feel of the interface combined with all supporting information (books, videos, help files) that describe system syntax and semantics
- Strives to agree with the user's mental model; users then feel comfortable with the software and use it effectively
- Serves as a translation of the design model by providing a realization of the information contained in the user profile model and the user's mental model

# User's Mental Model

- Often called the user's system perception
- Consists of the image of the system that users carry in their heads
- Accuracy of the description depends upon the user's profile and overall familiarity with the software in the application domain

# User Interface Design Process: A Spiral Process

- User interface development follows a spiral process
  - Interface analysis (user, task, and environment analysis)
  - Interface design
  - Interface construction
  - Interface validation, focuses on



# User Interface Analysis



# Elements of the Interface analysis

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

# User Analysis

- The analyst strives to get the end user's mental model and the design model to converge by understanding
  - The users themselves
  - How these people use the system
- Information can be obtained from
  - User interviews with the end users
  - Sales input from the sales people who interact with customers and users on a regular basis
  - Marketing input based on a market analysis to understand how different population segments might use the software
  - Support input 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)

# 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?

# Task Analysis and Modeling

- Task analysis strives to know and understand
  - The work the user performs in specific circumstances
  - The tasks and subtasks that will be performed as the user does the work
  - The specific problem domain objects that the user manipulates as work is performed
  - The sequence of work tasks (i.e., the workflow)
  - The hierarchy of tasks
- Use cases
  - Show how an end user performs some specific work-related task
  - Enable the software engineer to extract tasks, objects, and overall workflow of the interaction
    - Task elaboration refines interactive tasks
    - Object elaboration identifies interface objects (classes)
    - Workflow analysis defines how a work process is completed when several people (and roles) are involved

# Content Analysis

- The display content may range from character-based reports, to graphical displays, to multimedia information
- Display content may be
  - Generated by components in other parts of the application
  - Acquired from data stored in a database that is accessible from the application
  - 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 should be answered during content analysis (see next slide)

# Content Analysis Guidelines

- 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?

# Work Environment Analysis

- 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)

# User Interface Design



# UI designing



- Two points must be addressed in interactive systems design
  - How should information from the user be provided to the computer system?
  - How should information from the computer system be presented to the user?

# User interface design

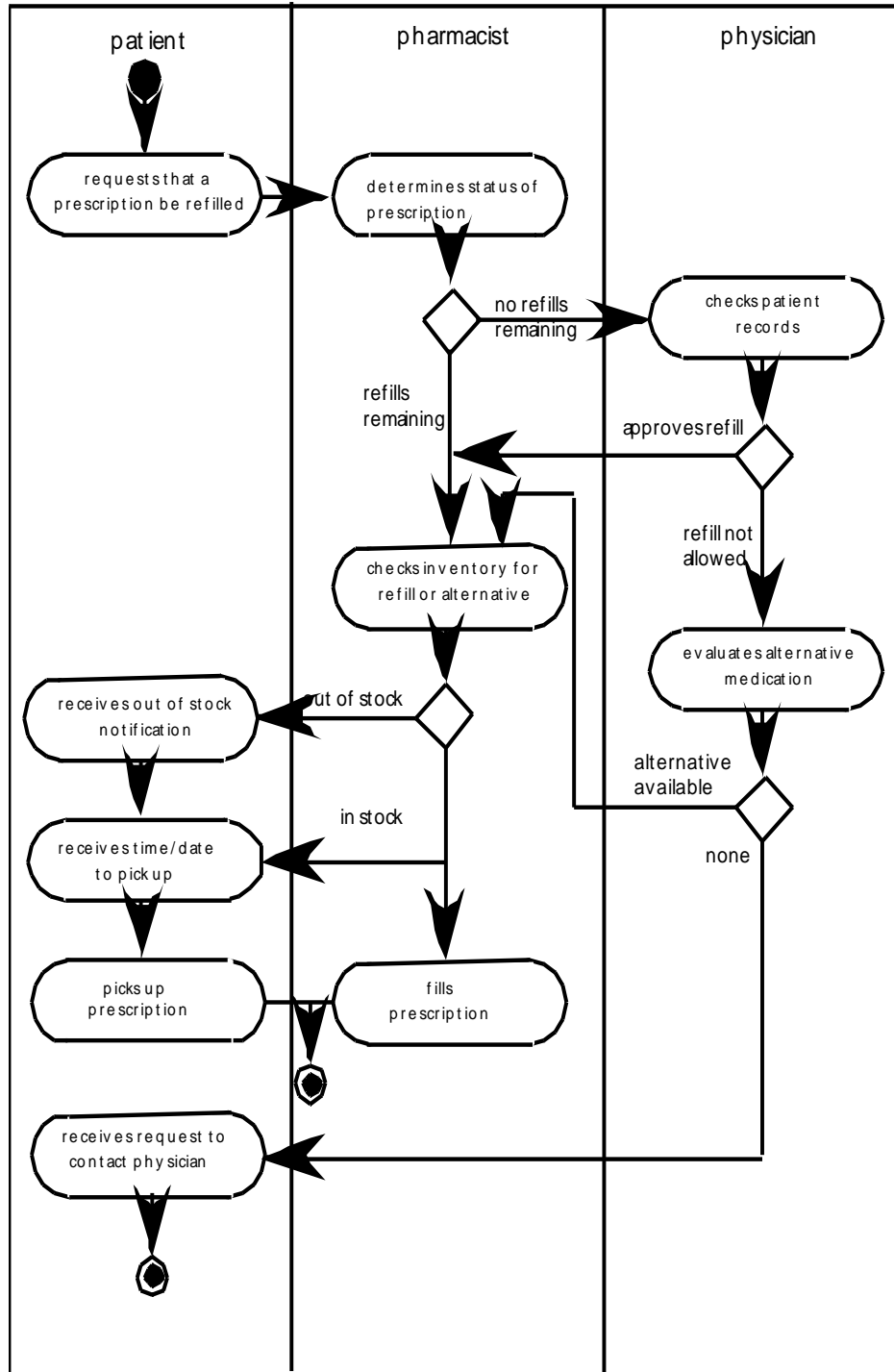
- General steps for user interface design
  - 1) Using information developed during user interface analysis, define user interface objects and actions (operations)
  - 2) Define events (user actions) that will cause the state of the user interface to change; model this behavior
  - 3) Depict each interface state as it will actually look to the end user
  - 4) Indicate how the user interprets the state of the system from information provided through the interface
- 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

# Interface Objects and Actions

- Interface objects and actions are obtained from the use cases and the software problem statement
- Interface objects are categorized into types: source, target, and application
  - A **source object** is dragged and dropped into a **target object** such as to create a hardcopy of a report
  - An **application object** represents application-specific data that are not directly manipulated as part of screen interaction such as a list
- After identifying objects and their actions, an interface designer performs screen layout which involves
  - Graphical design and placement of icons
  - Definition of descriptive screen text
  - Specification and titling for windows
  - Definition of major and minor menu items
  - Specification of a real-world metaphor to follow

# Design Issues to Consider

- Four common design issues usually surface in any user interface
  - System **response time** (both length and variability)
  - **User help facilities**
    - When is it available, how is it accessed, how is it represented to the user, how is it structured, what happens when help is exited
  - **Error information handling** (more on next slide)
    - How meaningful to the user, how descriptive of the problem
  - Menu and command **labeling** (more on upcoming slide)
    - Consistent, easy to learn, accessibility, internationalization
- Many software engineers do not address these issues until late in the design or construction process
  - This results in unnecessary iteration, project delays, and customer frustration



# Swimlane Diagram

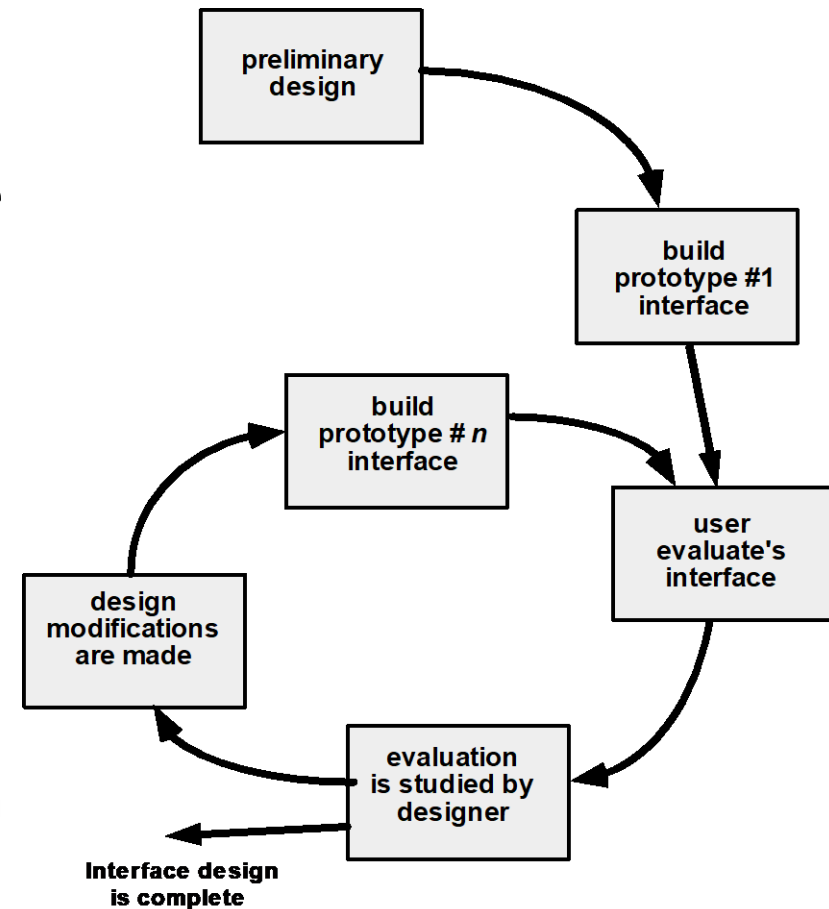
# User Interface Evaluation

# Design and Prototype Evaluation

- Before prototyping occurs, a number of evaluation criteria can be applied during design reviews to the design model itself
  - The **amount of learning** required by the users
    - Derived from the length and complexity of the written specification and its interfaces
  - The **interaction time and overall efficiency**
    - Derived from the number of user tasks specified and the average number of actions per task
  - The **memory load** on users
    - Derived from the number of actions, tasks, and system states
  - 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
- 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





# Analysis of Display Content

- Are different types of data assigned to consistent geographic locations on the screen (e.g., photos always appear in the upper right hand corner)?
- Can the user customize the screen location for content?
- Is proper on-screen identification assigned to all content?
- If a large report is to be presented, how should it be partitioned for ease of understanding?
- Will mechanisms be available for moving directly to summary information for large collections of data.
- Will graphical output be scaled to fit within the bounds of the display device that is used?
- How will color to be used to enhance understanding?
- How will error messages and warning be presented to the user?

# Types of User Interface

# GUI versus Text-Based User Interface



- In a GUI:
  - several windows with different information can be simultaneously displayed on user's screen.
  - This is perhaps the biggest advantage of GUI
  - user can simultaneously interact with several related items at any time
  - can even run many unrelated applications

# GUI versus Text-Based User Interface



- Iconic information representation and symbolic information manipulation is possible in a GUI.
- Symbolic information manipulation:
  - such as pulling an icon representing a file into a trash can for deleting
  - intuitively very appealing
    - user can instantly remember it.

# GUI versus Text-Based User Interface



- A GUI can support command selection:
  - using an attractive and user-friendly menu selection system.
  
- In a GUI, a pointing device can be used:
  - a mouse or a light pen to issue commands.
  - The use of a pointing device
    - makes command issue procedure simpler.

# GUI versus Text-Based User Interface



- On the flip side, a GUI requires:
  - special terminals with graphics capabilities
  - requires special input devices such a mouse.
  
- In contrast, a text-based interface:
  - can run even on cheap alphanumeric terminals:
  - Graphics terminals are usually much more expensive than alphanumeric terminals.

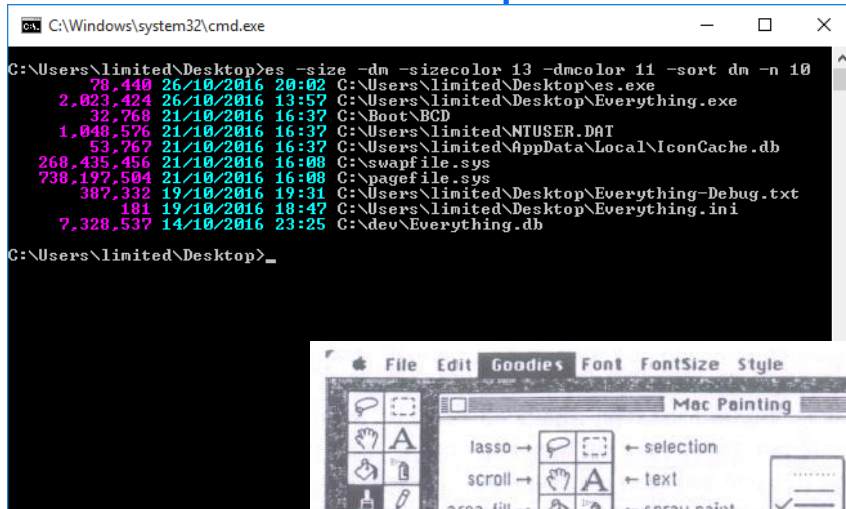
# GUI versus Text-Based User Interface



- Increasing availability of:
  - terminals with graphics capability
  - bit-mapped high-resolution displays
  - significant amount of local processing power.
- We will concentrate our attention to GUIs

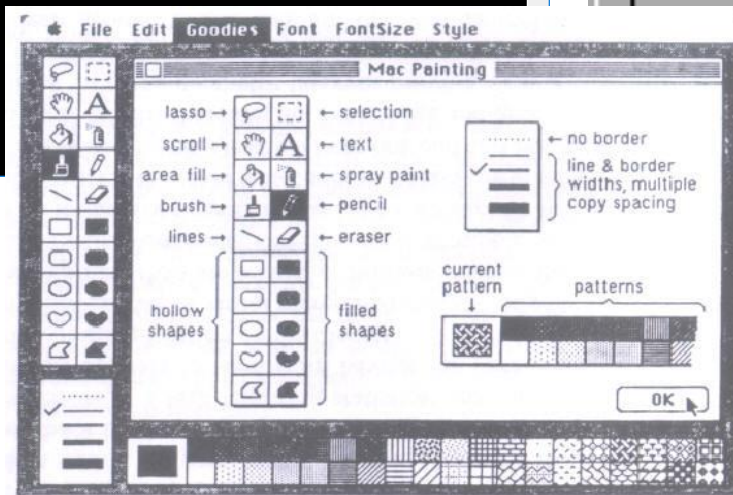
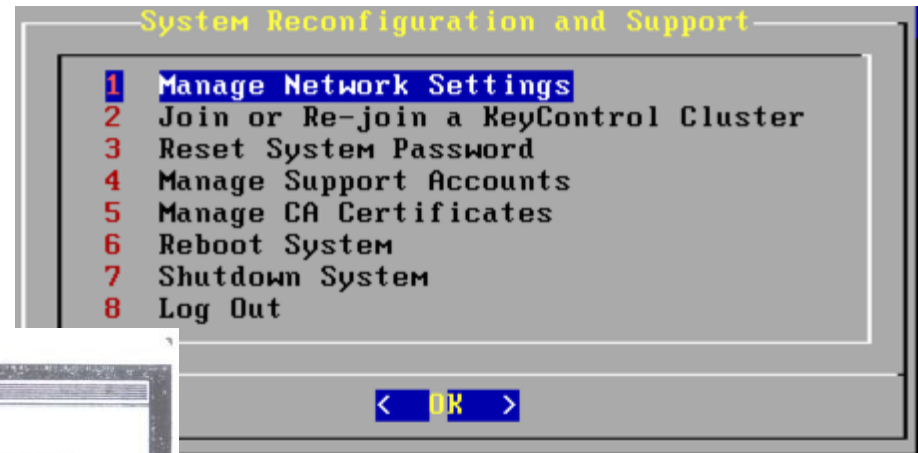
# Types of User Interfaces

- User interfaces can be classified into three categories:
  - Command language-based interface
  - Menu-based interface
  - Direct manipulation interface



```

C:\Windows\system32\cmd.exe
C:\Users\limited\Desktop>es -size -dm -sizecolor 13 -dmcolor 11 -sort dm -n 10
78,440 26/10/2016 20:02 C:\Users\limited\Desktop\es.exe
2,023,424 26/10/2016 13:57 C:\Users\limited\Desktop\Everything.exe
32,768 21/10/2016 16:37 C:\Boot\BCD
1,048,576 21/10/2016 16:37 C:\Users\limited\NTUSER.DAT
268,435,456 21/10/2016 16:08 C:\Users\limited\AppData\Local\IconCache.db
738,197,504 21/10/2016 16:08 C:\swapfile.sys
387,332 19/10/2016 19:31 C:\Users\limited\Desktop\Everything-Debug.txt
181 19/10/2016 18:47 C:\Users\limited\Desktop\Everything.ini
7,328,537 14/10/2016 23:25 C:\dev\Everything.db
C:\Users\limited\Desktop>_
    
```





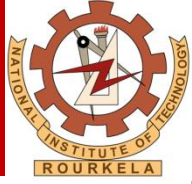
# Types of User Interfaces

- Each category of interface has its advantages and disadvantages:
  - Modern applications sport a combination of all the three types of interfaces.

# Choice of Interface

- Which parts of the interface should be implemented using what type of interface?
  - No simple guidelines available
  - to a large extent depends on the experience and discretion of the designer.
  - a study of characteristics of the different interfaces would give us some idea.

# Command Language-Based Interface



- As the name itself suggests:
  - incorporates some language to form commands.
- Users frame the required commands in the language:
  - type them in whenever required.

# Design of command language interface

- Simple command language interface:
  - determine all the commands needed to be supported
  - assign unique names to the different commands.

# Design of command language interface

- A more sophisticated command language interface:
  - allow users to compose primitive commands to form more complex commands.
    - Consider `cat x.dat|grep 123`
  - Like a programming language.

# Advantages of Command Language Interfaces



- Easy to develop:
  - compiler writing techniques are well developed.
- Can be implemented even on cheap alphanumeric terminals.
- Much more efficient:
  - compared to other types of interfaces.

# Disadvantages of Command Language Interfaces

- Difficult to learn:
  - Require the user to memorize primitive commands.
- Require the user to type in commands.
- Users make errors while:
  - formulating commands in the command language
  - typing them in.
- All interactions are through key-board:
  - cannot take advantage of effective interaction devices such as a mouse.
  - For casual and inexperienced users,
    - command language interfaces are not suitable.

# Issues in Designing a Command Language Interface

- The designer has to decide:
  - what **mnemonics** are to be used for the commands.
  - mnemonics should be meaningful
    - yet be concise to minimize the amount of typing required.
  - whether users will be allowed to redefine command names to suit their own preferences.
    - but increases complexity of user interface development.
  - whether it should be possible to compose primitive commands to create more complex commands.
  - syntax and semantics of command composition options has to be clearly and unambiguously decided.



# Menu-Based Interface

- Advantages of a menu-based interface over a command language interface:
  - users are not required to remember exact command names.
  - typing effort is minimal:
    - menu selections using a pointing device.
    - This factor becomes very important for the occasional users who can not type fast.

# Menu-Based Interface

- For experienced users:
  - menu-based interfaces is slower than command language interfaces
  - experienced users can type fast
  - also get speed advantage by composing simple commands into complex commands.

# Menu-Based Interface

- Composition of commands in a menu-based interface is not possible.
  - actions involving logical connectives (and, or, all, etc.)
    - awkward to specify in a menu-based system.

# Menu-Based Interface

- If the number of choices is large,
  - it is difficult to design a menu-based interface.
  - Even moderate sized software needs hundreds or thousands of menu choices.
- A major problem with the menu-based interface:
  - structuring large number of menu choices into manageable forms.

# Structuring Menu Interface

- Any one of the following options is adopted to structure menu items.
  - Walking menu
  - Scrolling menu
  - Hierarchical menu

# Scrolling Menu

- Used when the menu options are highly related.
  - For example text height selection in a word processing software.
- Scrolling of menu items
  - lets the user to view and select the menu items that can not be accommodated on one screen.



# Walking Menu



- Walking menu is commonly used to structure large menu lists:
  - when a menu item is selected,
  - it causes further menu items to be displayed adjacent to it in a submenu.



Welcome to NIT Rourkela



राष्ट्रीय प्रौद्योगिकी संस्थान राउरकेला  
National Institute of Technology Rourkela



Call Us  
+91661 246 2020



Mail Us  
registrar[at]nitrkl.ac.in

THE INSTITUTE

ACADEMICS

STUDENTS

FACULTY & STAFF

ALUMNI & INDUSTRY

Faculty Directory

Career

Centralised Instrumentation Facility

Officer Directory

Circulars & Notices

SRICCE

Staff Directory

Rules & Regulations

Purchase & Store Section

Campus Life

Official Forms

Reach your Student

# Walking Menu

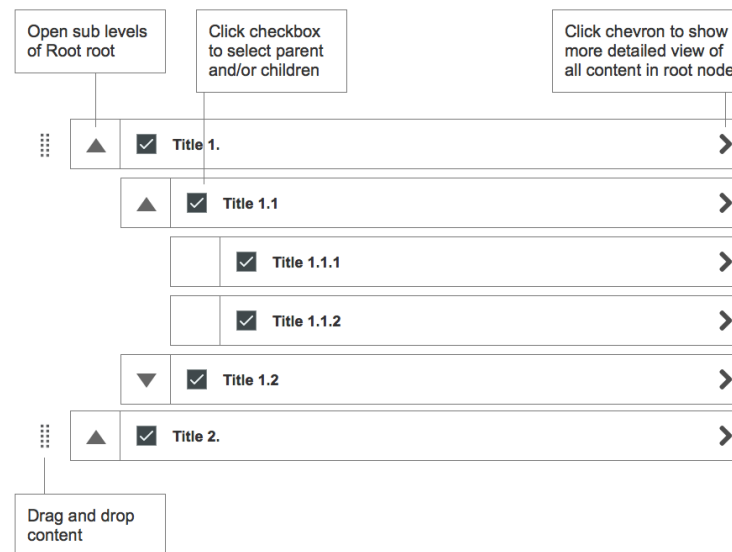
- A walking menu can successfully structure commands only if:
  - there are tens rather than hundreds of choices
  - each adjacently displayed menu does take up some screen space
    - the total screen area is after all limited.



# Hierarchical menu

- Menu items are organized in a hierarchy or tree structure.
  - Selecting a menu item causes the current menu display to be replaced by an appropriate submenu.
  - One can consider the menu and its various submenu to form a hierarchical tree-like structure.

Example layout



Img Ref: <https://ux.stackexchange.com/questions/89835/what-is-the-best-layout-for-a-hierarchical-menu-with-multiple-actions>

# Hierarchical menu

- Walking menu are a form of hierarchical menu:
  - practicable when the tree is shallow.
- Hierarchical menu can be used to manage large number of choices,
  - but, users face navigational problems
    - lose track of where they are in the menu tree.

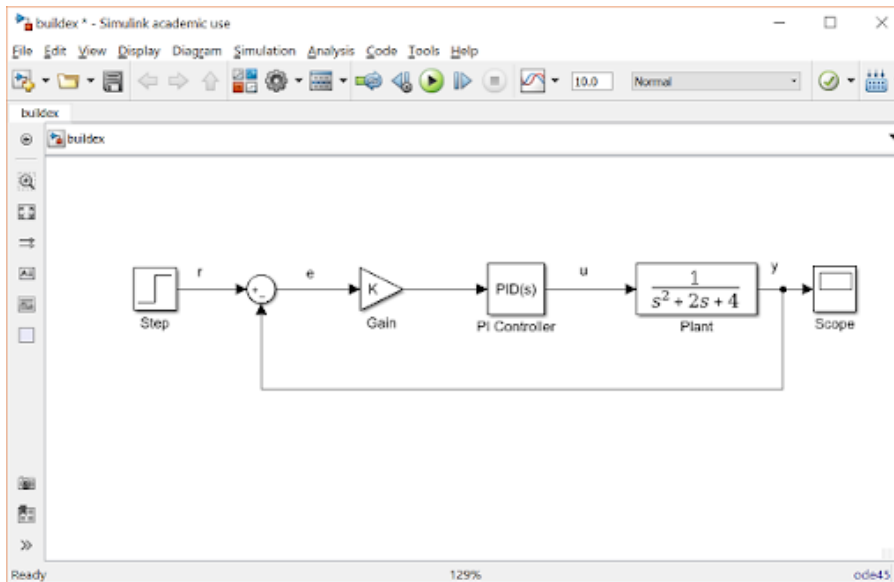
# Direct Manipulation Interface

- Present information to the user
  - as **visual models or objects**.
- Actions are performed on the visual representations of the objects, e.g.
  - pull an icon representing a file into an icon representing a trash box, for deleting the file.
- Direct manipulation interfaces are sometimes called as **iconic interfaces**.

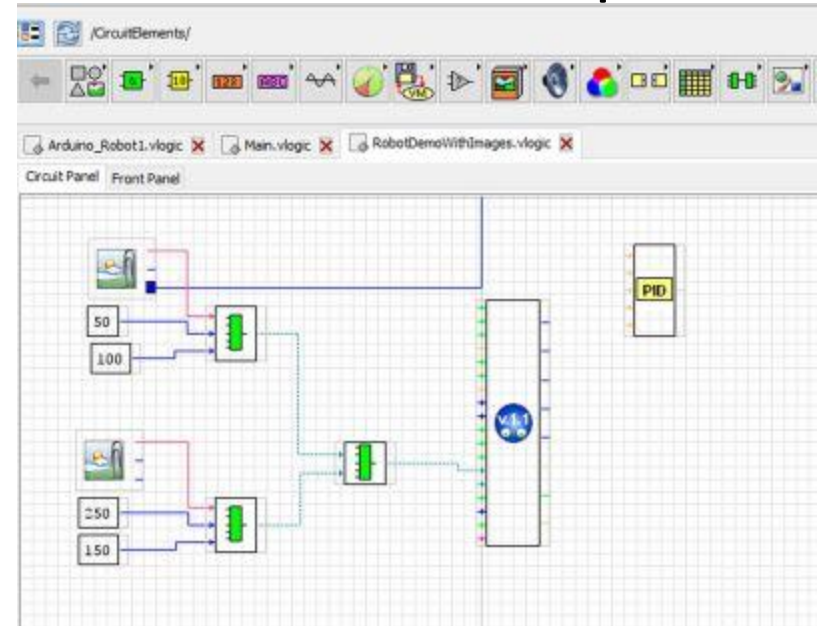
# Direct Manipulation (Iconic) Interface



- Important advantages of iconic interfaces:
  - icons can be recognized by users very easily,
  - icons are language-independent.
- However, experienced users consider direct manipulation interfaces too slow.



Simulink®



Labview®

# Direct Manipulation (Iconic) Interface



- It is difficult to form complex commands using a direct manipulation interface.
- For example, if one has to drag a file icon into a trash box icon for deleting a file:
  - to delete all files in a directory one has to perform this operation again and again
  - very easily done in a command language-interface by issuing a command `delete *.*`

- End of Chapter

*Thanks*