MASENO UNIVERSITY
BACHELOR OF SCIENCE IN COMPUTER SCIENCE/
BACHELOR OF SCIENCE IN COMPUTER TECHNOLOGY
YEAR 3 SEMESTER 2
CCS 302/CCT 304: HUMAN COMPUTER INTERACTION
NOTES

Basic concept of human computer interaction (HCI)

Human-computer interaction

- **Human-computing interaction** (HCI) concerns the study of the design, evaluation, and implementation of the interfaces between computing devices and people
- HCI has three components: the human, the interaction, and the computer

The human user

- The human user may be an individual or a group of users who employ the computer to accomplish a task.
- The human user may be a novice, intermediate, or expert who uses the technological system.
- Further, the human user may be a child using the system to complete a homework assignment or an adult performing a task at work.
- Additionally, the human user may be a person who has a physical or cognitive limitation which impacts his/her use with the computer-based system.
- No matter who the human user is, the goal when interacting with a computer system is to have a seamless interaction which accomplishes the task.

The computer system

- A computer is defined as an electronic device designed to accept data, perform prescribed mathematical and logical operations at high speed, and display the results of these operations.
- However, as computers become more complex, users expect more than just a display of the results of their operations. The term computer system is used to represent technology and technological systems.
- As the human user becomes to depend on these technological systems more, the interaction between the user and the system becomes more complex.

The interaction

- Interaction is the communication between the user and the computer system.
- For computer systems to continue their wide spread popularity and to be used effectively, the computer system must be well designed. According to Sharp, Rogers, and Preece, a central concern of interaction design is to develop an interactive system that is usable (2007).

- More specifically, the computer system must be easy to use, easy to learn, thereby creating a user experience that is pleasing to the user.

Characteristics of a good user interface

Some 8 characteristics of a good user interface are:

1. Clear
2. Concise
3. Familiar
4. Responsive
5. Consistent
6. Attractive
7. Efficient
8. Forgiving

Clear

Clarity is the most important element of user interface design. Indeed, the whole purpose of user interface design is to enable people to interact with your system by communicating meaning and function. If people can't figure out how your application works or where to go on your website they'll get confused and frustrated.

Concise

Keep things clear but also keep things concise. When you can explain a feature in one sentence instead of three, do it. When you can label an item with one word instead of two, do it. Save the valuable time of your users by keeping things concise. Keeping things clear and concise at the same time isn't easy and takes time and effort to achieve, but the rewards are great.

Familiar

- Many designers strive to make their interfaces 'intuitive'. But what does intuitive really mean? It means something that can be naturally and instinctively understood and comprehended. But how can you make something intuitive? You do it by making it 'familiar'.
- Familiar is just that: something which appears like something else you've encountered before. When you're familiar with something, you know how it behaves — you know what to expect. Identify things that are familiar to your users and integrate them into your user interface.

Responsive

Responsive means a couple of things.

- First of all, responsive means fast. The interface, if not the software behind it, should work fast. Waiting for things to load and using laggy and slow interfaces is frustrating. Seeing things load quickly, or at the very least, an interface that loads quickly (even if the content is yet to catch up) improves the user experience.
- Responsive also means the interface provides some form of feedback. The interface should talk back to the user to inform them about what's happening. Have you pressed that button successfully? How would you know? The button should display a 'pressed' state to give that feedback. Perhaps the button text could change to "Loading…" and it's state disabled. Is the software stuck or is the content loading? Play a spinning wheel or show a progress bar to keep the user in the loop.

Consistent

- Consistent interfaces allow users to develop usage patterns — they'll learn what the different buttons, tabs, icons and other interface elements look like and will recognize them and realize what they do in different contexts.
- They'll also learn how certain things work, and will be able to work out how to operate new features quicker, extrapolating from those previous experiences.

Attractive

- A good interface should be attractive in a sense that it makes the use of that interface enjoyable.
- Yes, you can make your UI simple, easy to use, efficient and responsive, and it will do its job well — but if you can go that extra step further and make it attractive, then you will make the experience of using that interface truly satisfying. When your software is pleasant to use, your customers or staff will not simply be using it — they'll look forward to using it.

Efficient

- A good interface should allow you to perform those functions faster and with less effort.
- What you really need to do to make an interface efficient is to figure out what exactly the user is trying to achieve, and then let them do exactly that without any fuss.
- You have to identify how your application should 'work' — what functions does it need to have, what are the goals you're trying to achieve?

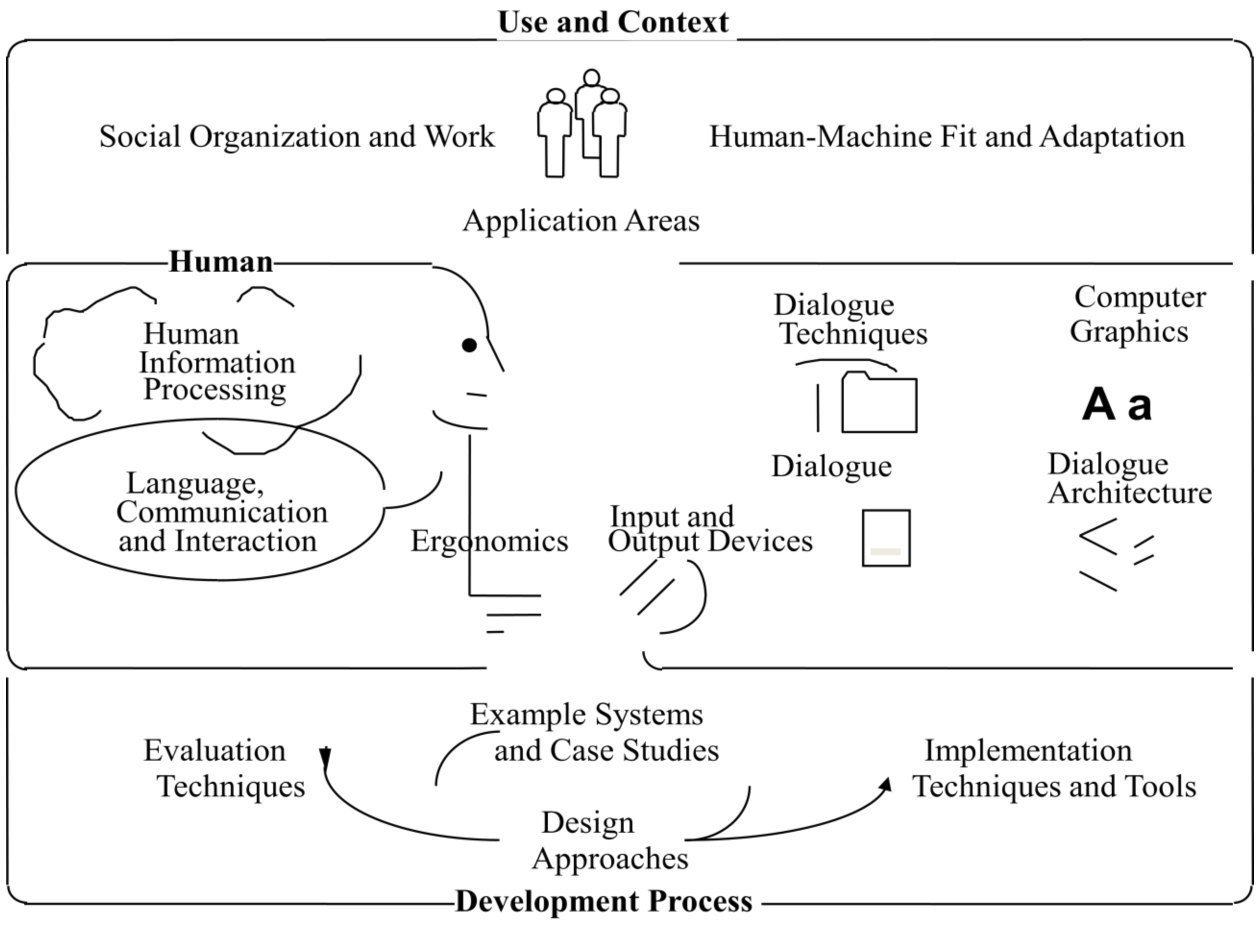
- Implement an interface that lets people easily accomplish what they want instead of simply implementing access to a list of features.

Forgiving

- Nobody is perfect, and people are bound to make mistakes when using your software or website. How well you can handle those mistakes will be an important indicator of your software's quality.
- Don't punish the user — build a forgiving interface to remedy issues that come up.
- A forgiving interface is one that can save your users from costly mistakes. For example, if someone deletes an important piece of information, can they easily retrieve it or undo this action?
- When someone navigates to a broken or nonexistent page on your website, what do they see? Are they greeted with a cryptic error or do they get a helpful list of alternative destinations?

The multidisciplinary nature of HCI

Overview: Map of Human Computer Interaction



Use and context of computers

Problems of fitting computers, *their uses, and the context of use together*

Social organization and work

- humans are interacting social beings
- considers models of human activity:
  - small groups, organizations, socio-technical systems
- quality of work life…

Application areas

- characteristics of application domains, e.g. individual vs group work
- popular styles
  - document production, communications, design, tutorials and help, multi-media information kiosks, continuous control (cockpits, process control), embedded systems (copiers, home appliances)

Human-machine fit and adaptation

- improve the fit between the designed object and its use
  - how systems are selected and adopted; how users improvise routine systems; how systems adapt to the user (customization); how users adapt to the system (training, ease of learning); user guidance (help, documentation, error-handling)

Human characteristics

To understand the human as an information-processing system, how humans communicate, and people's physical and psychological requirements

Human information processing

- characteristics of the human as a processor of information
  - memory, perception, motor skills, attention, problem-solving, learning and skill acquisition, motivation, conceptual models, diversity...

Language, communication and interaction

- aspects of language
  - syntax, semantics, pragmatics; conversational interaction, specialized languages

Ergonomics

- anthropometric and physiological characteristics of people and their relationship to workspace and the environment

    - arrangement of displays and controls; cognitive and sensory limits; effects of display technology; fatigue and health; furniture and lighting; design for stressful and hazardous environments; design for the disabled...

Computer system and interface architecture

The specialized components computers have for interacting with people

Input and output devices

- mechanics and characteristics of particular hardware devices, performance characteristics (human and system), esoteric devices, virtual devices

Dialogue techniques

- the basic software architecture and techniques for interacting with humans
    - e.g. dialog inputs and outputs; interaction styles; issues

Dialog genre

- The conceptual uses to which the technical means are put
    - e.g. interaction and content metaphors, transition management, style and aesthetics

Computer graphics

- basic concepts from computer graphics that are especially useful to HCI

Dialogue architecture

- software architecture and standards for interfaces

e.g., screen imaging; window managers; interface toolkits; multi-user architectures, look and feel, standardization and interoperability

The Development Process

The construction and evaluation of human interfaces

Design approaches

- the process of design
    - e.g. graphical design basics (typography, color, etc); software engineering; task analysis; industrial design...

Implementation techniques and tools

- tactics and tools for implementation, and the relationship between design, evaluation and implementation
    - e.g. prototyping techniques, dialog toolkits, object-oriented methods, data representation and algorithms

Evaluation techniques

- philosophy and specific methods for evaluation
    - e.g. productivity, usability testing, formative and summative evaluation

Example systems and case studies

- classic designs to serve as example of interface design genres

Why study human use of computer systems?

Business view:

- to use humans more productively/effectively
- the human costs now far outweigh hardware and software costs

Personal view:

- people view computers as appliances, and want it to perform as one

Marketplace view:

- everyday people using computers
  - now expect "easy to use system"
  - not tolerant of poorly designed systems
  - little vendor control of training
  - heterogeneous group
- if product is hard to use, people will seek other products
- eg Mac vs IBM (Microsoft Windows)

The *system* view:

- complex human
- complex computer
- complex interface between the two

The human factors view:

- humans have necessary limitations
- errors are costly in terms of
  - loss of time
  - loss of money
  - loss of lives in critical systems
  - loss of morale
- design can cope with such limitations!

The social view:

Computers contribute to critical parts of our society, and cannot be ignored

- educate our children
- take medical histories and provide expert advice
- keep track of our credit worthiness
- play(?) war games (and help form policies)
- control air and ground traffic flow
- book travel
- control chemical/oil/nuclear plants
- control space missions
- assist humans with their everyday tasks (office automation)

- control complex machines (aircraft, space shuttles, super tankers)
- help control consumer equipment (cars, washing machines)
- entertainment (games, intellectual stimulation)….

In all these views, economics and human best interests are aligned

HCI as a Multidisciplinary Field

The discipline of HCI

HCI is a field that brings many disciplines together and is regarded as a highly multidisciplinary field. There are several main disciplines that are encompassed within HCI.

1. **Art and Graphic Design**
    - While psychologists bring to the field of HCI the understanding of how humans act and react to technology, and computer scientists and engineers design and develop the computer systems, and the area of human factors enhances knowledge about the physical environment in which the system will be used and the social sciences help obtain accurate descriptions about the user, without the areas of art and graphic design, most systems would not be used.
    - Artists and graphic designers put a "face" on the system and thereby with a good design and use of color, artists and graphic designers help to make the interaction between the user and the system enjoyable and seamless.
    - Graphic designers often use typography, visual arts and rules for page layouts to assist with the design of an interface for a system.
    - Meanwhile, the discipline of art brings to HCI a creative process by which interaction takes place between the user and the computer system. Artists help to bridge the gap between designing the system for the user and making the system usable by the user.

2. **Business**

- Various areas of business include business administration, accounting, economics, finance, management, and sales and marketing. At the core of each of these areas lies a knowledge base that HCI uses to its benefit.
- Whether it includes how to sale and market a computer system that capitalizes on user sensory interaction with a system, or if it includes e-commerce management, all the areas of business contribute to the HCI discipline.

3. **Engineering**
    - Engineering is defined as "The creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of

their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property."

- Engineering plays a very specific role in HCI ensuring that systems are designed and developed according to specifications.

4. Ergonomics and Human Factors

- The term ergonomics (human factors) is traditionally the study of the physical characteristics of interaction.
- More specially the discipline is concerned with how the controls are designed, the physical qualities of the screen, and the physical environment in which the interaction between the user and the system takes place.
- The goal of human factors is to optimize human well-being and overall system performance.
- The discipline of human factors is important to the field of HCI as it focuses on the user's capabilities and limitations.

5. Technical Writing

- Technical writing is concerned with the presentation of information that helps a reader solve a specific problem. Technical writing has been called a form of technical communication that is frequently used to demystify technical terms and language.
- Technical communicators write and design many kinds of professional documents which include but are not limited to manuals, lab reports, web pages and proposals.
- Technical writing contributes to the field of HCI as it provides a form of communication that helps to enhance the interaction between the user and the computer system.

6. Computer Science

- Computer science is a discipline that is concerned with the study and the science of theories and methods that underlie technological systems.
- Computer science can also be thought of as the study of computer hardware, and the study, design and implementation of computer software. HCI for many years has been thought to be a sub-discipline of computer science. HCI encompasses the field of computer science as it does many other disciplines.
- Computer system design includes a variety of activities that range from hardware design to interface design.
- Careful interface design plays an essential role in the overall design of interaction between the user and the computer system.
- The themes from computer science's software design are therefore, very prominent in the user interface design of HCI.

7. **Social Sciences**
    - Although, HCI has often been linked with the "hard sciences" of computer science and engineering, it is the "soft sciences" of sociology and anthropology that bring to the forefront of the discipline the impact and influence that technology has on its users.
    - A major concern of the social sciences is to understand the interaction between the computer system and the user both during and after the event. Therefore, the reasons for including the social sciences in HCI are to obtain a more accurate description of the interaction between users, their tasks, the technological systems that they use and the environment in which they use the systems.

8. **Information Systems**
    - Information systems, is an applied discipline that studies the processes of the creation, operation, and social contexts and consequences of systems that manipulate information. It also includes the analysis, development, and management of systems.
    - The area of information systems has two distinguishing features that place information systems within the context of HCI: (1) business application and (2) management orientations (Zhang, 2004).
    - Consequently, information systems works well as one of the many disciplines of HCI because it is concerned with the study in which humans interact with information, technologies, and tasks in business, organizational, and cultural environments.
    - Simply put the discipline of information systems helps HCI to go beyond the theoretical concepts of computer science to a more applied approach while taking into account issues related to social and organizational constructs.

9. **Math and Statistics**
    - Evaluation is concerned with gathering information about the usability of a system in order to improve system performance (Benyon et at., 1993). Without evaluation, user requirements may not be met or system performance may be low, all leading to an unpleasant user experience.
    - However, in order to evaluate a system, data concerning the user's interaction with the system and the user's attitudes towards the system must be collected and analyzed.
    - Consequently math, primarily statistics, plays an important role in the evaluation of a system and the user.
    - Statistical testing helps to present the results of evaluation in a useful and meaningful manner.
    - Consequently, if researchers are observing the behavior between the user and the computer repeatedly, comparing one group of users to another group of users, studying one group of users comprised of individuals differing from one another, or simply presenting background information on a group of users, statistics is needed.

10. Cognitive Psychology

- In order to design a product for the user, it is important to know the user's capabilities and limitations.
- The discipline of cognitive psychology provides knowledge of the user's perceptual, cognitive and problem-solving skills.
- Cognitive psychology is needed in order to understand the manner in which humans act and react. More importantly, cognitive psychology is used to understand how users will interact with technological systems and devices.
- Of particular interest to HCI is the human information processing system which is akin to the computer information processing system.
- The human information processing system, according to various researchers consists of three subsystems which include: the perceptual system, which handles sensory information; the motor system, which controls actions; and, the cognitive system which provides the processing needed to connect the sensory information with the motor system (Card, et al., 1983).
- The computer information system includes: input devices which accepts information by apparatuses such as a keyboard or mouse; output devices which include peripheral devices; and, the central processing unit which combines the arithmetic and logic unit with the control unit to transform user input to output.

History (and future) of HCI

- Large displays
- Small displays
- Peripheral displays
- Alternative I/O
- Ubiquitous computing
- Virtual environments
- Implants
- Speech recognition
- Multimedia
- Video conferencing
- Artificial intelligence
- Software agents
- Recommender systems

...

Paradigms of interaction
What are Paradigms

- Predominant theoretical frameworks or scientific world views of style of interaction.

- Understanding HCI history is largely about understanding a series of paradigm shifts

Why study paradigms

Concerns

- how can an interactive system be developed to ensure its usability?
- how can the usability of an interactive system be demonstrated or measured?

History of interactive system design provides paradigms for usable designs

Paradigms of interaction

- New computing technologies arrive, creating a new perception of the human—computer relationship.
- We can trace some of these shifts in the history of interactive technologies.

Example Paradigm Shifts

- Batch processing
- Timesharing
- Networking
- Graphical display
- Microprocessor
- WWW
- Ubiquitous Computing

Batch processing

- A collection of program statements are put in a single file that is executed as a single unit.

Time-sharing

- single computer supporting multiple users

Video Display Units

- more suitable medium than paper
- computers for visualizing and manipulating data
- one person's contribution could drastically change the history of computing

Programming toolkits

- the right programming toolkit provides building blocks to producing complex interactive systems

Personal computing

- A system is more powerful as it becomes easier to user
- Future of computing in small, powerful machines dedicated to the individual

Window systems and the WIMP interface

- humans can pursue more than one task at a time
- windows used for dialogue partitioning, to “change the topic”
- 1981 – Xerox Star first commercial windowing system
- windows, icons, menus and pointers now familiar interaction mechanisms

Metaphor

- relating computing to other real-world activity is effective teaching technique
  - file management on an office desktop
  - word processing as typing
  - financial analysis on spreadsheets
  - virtual reality – user inside the metaphor
- Problems
  - some tasks do not fit into a given metaphor
  - cultural bias

Direct manipulation

- 1982 – Shneiderman describes appeal of graphically-based interaction
  - visibility of objects
  - incremental action and rapid feedback
  - reversibility encourages exploration
  - syntactic correctness of all actions
  - replace language with action
  - 1984 – Apple Macintosh
- the model-world metaphor
- What You See Is What You Get (WYSIWYG)

Language versus Action

- actions do not always speak louder than words!
- DM – interface replaces underlying system
- language paradigm
- interface as mediator
- interface acts as intelligent agent
- programming by example is both action and language

Hypertext

- 1945 – Vannevar Bush and the memex

- key to success in managing explosion of information
- mid 1960s – Nelson describes hypertext as non-linear browsing structure
- hypermedia and multimedia

Multimodality
- a mode is a human communication channel
- emphasis on simultaneous use of multiple channels for input and output

Computer Supported Cooperative Work (CSCW)
- CSCW removes bias of single user / single computer system
- Can no longer neglect the social aspects
- Electronic mail is most prominent success

The World Wide Web
- Hypertext, as originally realized, was a closed system
- Simple, universal protocols (e.g. HTTP) and mark-up languages (e.g. HTML) made publishing and accessing easy
- Critical mass of users lead to a complete transformation of our information economy.

Agent-based Interfaces
- Original interfaces
  - Commands given to computer
  - Language-based
- Direct Manipulation/WIMP
  - Commands performed on “world” representation
  - Action based
- Agents - return to language by instilling proactivity and “intelligence” in command processor
  - Avatars, natural language processing

Ubiquitous Computing
  - Design interactions that don’t demand our intention

Sensor-based and Context-aware Interaction
- Humans are good at recognizing the “context” of a situation and reacting appropriately
- Automatically sensing physical phenomena (e.g., light, temp, location, identity) becoming easier
- How can we go from sensed physical measures to interactions that behave as if made “aware” of the surroundings?

Human factors

The human

- Information i/o
    - visual, auditory, haptic, movement
- Information stored in memory
    - sensory, short-term, long-term
- Information processed and applied
    - reasoning, problem solving, skill, error
- Emotion influences human capabilities
- Each person is different

Vision

Two stages in vision

• Physical reception of stimulus

• Processing and interpretation of stimulus

The Eye - physical reception

- mechanism for receiving light and transforming it into electrical energy
- light reflects from objects
- images are focused upside-down on retina
- retina contains rods for low light vision and cones for colour vision
- ganglion cells (brain!) detect pattern and movement

Interpreting the signal

- Size and depth
    - visual angle indicates how much of view object occupies
      (relates to size and distance from eye)
    - visual acuity is ability to perceive detail (limited)
    - familiar objects perceived as constant size
      (in spite of changes in visual angle when far away)
    - cues like overlapping help perception of size and depth

- **Brightness**
    - subjective reaction to levels of light
    - affected by luminance of object
    - measured by just noticeable difference
    - visual acuity increases with luminance as does flicker

Reading

- Several stages:
    - visual pattern perceived
    - decoded using internal representation of language
    - interpreted using knowledge of syntax, semantics, pragmatics
- Word shape is important to recognition

Hearing

- Provides information about environment: distances, directions, objects.
- Physical apparatus:
    - outer ear – protects inner and amplifies sound
    - middle ear – transmits sound waves as vibrations to inner ear
    - inner ear – chemical transmitters are released and cause impulses in auditory nerve
- Sound
    - Pitch – sound frequency
    - loudness – amplitude
    - timbre – type or quality

- Humans can hear frequencies from 20Hz to 15kHz
    - less accurate distinguishing high frequencies than low.
- Auditory system filters sounds
    - can attend to sounds over background noise.
    - for example, the cocktail party phenomenon.

Touch

- Provides important feedback about environment.
- May be key sense for someone who is visually impaired.
- Stimulus received via receptors in the skin:
    - Thermoreceptors – heat and cold
    - Nociceptors – pain
    - Mechanoreceptors – pressure (some instant, some continuous)
- Some areas more sensitive than others e.g. fingers.
- Kinethesis - awareness of body position
    - affects comfort and performance.

Movement

- Time taken to respond to stimulus: reaction time + movement time
- Movement time dependent on age, fitness etc.
- Reaction time - dependent on stimulus type:

- visual ~ 200ms
- auditory ~ 150 ms
- pain ~ 700ms

Memory
There are three types of memory function:

$$\begin{tikzcd}
\text{Sensory memories} \arrow[rd] & & \\
& \textbf{Attention} & \\
& \text{Short-term memory or working memory} \arrow[rd] & \\
& & \textbf{Rehearsal} \\
& & \text{Long-term memory}
\end{tikzcd}$$

Sensory memory
- Buffers for stimuli received through senses
  - iconic memory: visual stimuli
  - echoic memory: aural stimuli
  - haptic memory: tactile stimuli
- Examples
  - “sparkler” trail
  - stereo sound
- Continuously overwritten

Short-term memory (STM)
- Scratch-pad for temporary recall
  - rapid access ~ 70ms
  - rapid decay ~ 200ms
  - limited capacity: 7± 2 chunks

Examples
212348278493202
0121 414 2626
HEC ATR ANU PTH ETR EET

Long-term memory (LTM)
- Repository for all our knowledge
  - slow access ~ 1/10 second

    - slow decay, if any
    - huge or unlimited capacity
- Two types
    - Episodic – serial memory of events
    - Semantic – structured memory of facts, concepts, skills

Semantic LTM derived from episodic LTM

- Semantic memory structure
    - provides access to information
    - represents relationships between bits of information
    - supports inference
- Model: semantic network
    - inheritance – child nodes inherit properties of parent nodes
    - relationships between bits of information explicit
    - supports inference through inheritance

LTM - Storage of information

- rehearsal
    - information moves from STM to LTM
- total time hypothesis
    - amount retained proportional to rehearsal time
- distribution of practice effect
    - optimized by spreading learning over time
- structure, meaning and familiarity
    - information easier to remember

LTM – Forgetting

Decay

- information is lost gradually but very slowly

interference

- new information replaces old: retroactive interference
- old may interfere with new: proactive inhibition

so may not forget at all memory is selective …

… affected by emotion – can subconsciously `choose' to forget

LTM – retrieval

Recall

- information reproduced from memory can be assisted by cues, e.g. categories, imagery

Recognition

- information gives knowledge that it has been seen before
- less complex than recall - information is cue

Emotion

- Various theories of how emotion works
  - James-Lange: emotion is our interpretation of a physiological response to a stimuli
  - Cannon: emotion is a psychological response to a stimuli
  - Schacter-Singer: emotion is the result of our evaluation of our physiological responses, in the light of the whole situation we are in
- Emotion clearly involves both cognitive and physical responses to stimuli
- The biological response to physical stimuli is called *affect*

- Affect influences how we respond to situations
  - positive → creative problem solving
  - negative → narrow thinking

“Negative affect can make it harder to do even easy tasks; positive affect can make it easier to do difficult tasks”

- Implications for interface design
  - stress will increase the difficulty of problem solving
  - relaxed users will be more forgiving of shortcomings in design
  - aesthetically pleasing and rewarding interfaces will increase positive affect

Machine factors

The Computer

A computer system is made up of various elements. Each of these elements affects the interaction

- input devices – text entry and pointing
- output devices – screen (small & large), digital paper
- virtual reality – special interaction and display devices
- physical interaction – e.g. sound, haptic, bio-sensing

    - paper – as output (print) and input (scan)
    - memory – RAM & permanent media, capacity & access
    - processing – speed of processing, networks

Text entry devices

Keyboards

- Most common text input device
- Allows rapid entry of text by experienced users
- Keypress closes connection, causing a character code to be sent
- Usually connected by cable, but can be wireless

Handwriting recognition

- Text can be input into the computer, using a pen and a digesting tablet
    - natural interaction
- Technical problems:
    - capturing all useful information - stroke path, pressure, etc. in a natural manner
    - segmenting joined up writing into individual letters
    - interpreting individual letters
    - coping with different styles of handwriting
    - Used in PDAs, and tablet computers …
      … leave the keyboard on the desk!

Speech recognition

- Improving rapidly
- Most successful when:
    - single user – initial training and learns peculiarities
    - limited vocabulary systems
- Problems with
    - external noise interfering
    - imprecision of pronunciation
    - large vocabularies
    - different speakers

Positioning, pointing and drawing devices

The Mouse

- Handheld pointing device
    - very common
    - easy to use

Touchpad

- small touch sensitive tablets
- ‘stroke’ to move mouse pointer
- used mainly in laptop computers

Trackball and thumbwheels

Trackball

- ball is rotated inside static housing
    - like an upsdie down mouse!
- relative motion moves cursor
- indirect device, fairly accurate
- separate buttons for picking
- very fast for gaming
- used in some portable and notebook computers.

Thumbwheels

- for accurate CAD – two dials for X-Y cursor position
- for fast scrolling – single dial on mouse

Joystick and keyboard nipple

Joystick

- Indirect pressure of stick = velocity of movement
- buttons for selection on top or on front like a trigger
- often used for computer games, aircraft controls and 3D navigation

Keyboard nipple

- for laptop computers
- miniature joystick in the middle of the keyboard

Touch-sensitive screen

- Detect the presence of finger or stylus on the screen.
    - works by interrupting matrix of light beams, capacitance changes or ultrasonic reflections
    - *direct* pointing device
- Advantages:
    - fast, and requires no specialised pointer
    - good for menu selection
    - suitable for use in hostile environment: clean and safe from damage.

- Disadvantages:
  - finger can mark screen
  - imprecise (finger is a fairly blunt instrument!)
    - difficult to select small regions or perform accurate drawing
  - lifting arm can be tiring

Stylus and light pen

Stylus

- small pen-like pointer to draw directly on screen
- may use touch sensitive surface or magnetic detection
- used in PDA, tablets PCs and drawing tables

Light Pen

- now rarely used
- uses light from screen to detect location
- BOTH …
- very direct and obvious to use
- but can obscure screen

Digitizing tablet

- Mouse like-device with cross hairs
- used on special surface - rather like stylus
- very accurate - used for digitizing maps

Eyegaze

- control interface by eye gaze direction - e.g. look at a menu item to select it
- uses laser beam reflected off retina - a very low power laser!
- mainly used for evaluation
- potential for hands-free control
- high accuracy requires headset
- cheaper and lower accuracy devices available

  - sit under the screen like a small webcam

Discrete positioning controls

- in phones, TV controls etc.
  - cursor pads or mini-joysticks
  - discrete left-right, up-down
  - mainly for menu selection

Display devices

Cathode ray tube

- Stream of electrons emitted from electron gun, focused and directed by magnetic fields, hit phosphor-coated screen which glows
- used in TVs and computer monitors

Health hazards of CRT

- X-rays: largely absorbed by screen (but not at rear!)
- UV- and IR-radiation from phosphors: insignificant levels
- Radio frequency emissions, plus ultrasound (~16kHz)
- Electrostatic field - leaks out through tube to user. Intensity dependant on distance and humidity. Can cause rashes.
- Electromagnetic fields (50Hz-0.5MHz). Create induction currents in conductive materials, including the human body. Two types of effects attributed to this: visual system - high incidence of cataracts in VDU operators, and concern over reproductive disorders (miscarriages and birth defects).

Health hints

- do not sit too close to the screen
- do not use very small fonts
- do not look at the screen for long periods without a break
- do not place the screen directly in front of a bright window
- work in well-lit surroundings

Liquid crystal displays

- Smaller, lighter, and no radiation problems.
- Found on PDAs, portables and notebooks,
  … and increasingly on desktop and even for home TV
- also used in dedicated displays:
  digital watches, mobile phones, HiFi controls

Special displays

Random Scan (Directed-beam refresh, vector display)

  - draw the lines to be displayed directly
  - no jaggies
  - lines need to be constantly redrawn
  - rarely used except in special instruments

Direct view storage tube (DVST)

- Similar to random scan but persistent => no flicker
- Can be incrementally updated but not selectively erased
- Used in analogue storage oscilloscopes

Large displays

- used for meetings, lectures, etc.
- technology
    - plasma – usually wide screen
    - video walls – lots of small screens together
    - projected – RGB lights or LCD projector
        - hand/body obscures screen
        - may be solved by 2 projectors + clever software

Situated displays

- displays in 'public' places
    - large or small
    - very public or for small group
- display only
    - for information relevant to location
- or interactive
    - use stylus, touch sensitive screen
- in all cases … the location matters
    - meaning of information or interaction is related to the location

3D displays

- desktop VR
    - ordinary screen, mouse or keyboard control
    - perspective and motion give 3D effect
- seeing in 3D
    - use stereoscopic vision
    - VR helmets
    - screen plus shuttered specs, etc.

Physical controls and sensors

Dedicated displays

- analogue representations:
    - dials, gauges, lights, etc.

- digital displays:
    - small LCD screens, LED lights, etc.
- head-up displays
    - found in aircraft cockpits
    - show most important controls

Sounds

- beeps, bongs, clonks, whistles and whirrs
- used for error indications
- confirmation of actions e.g. keyclick

Touch, feel, smell

- touch and feeling important
    - in games … vibration, force feedback
    - in simulation … feel of surgical instruments
    - called *haptic* devices
- texture, smell, taste
    - current technology very limited

Environment and bio-sensing

- sensors all around us
    - car courtesy light – small switch on door
    - ultrasound detectors – security, washbasins
    - RFID security tags in shops
    - temperature, weight, location
- … and even our own bodies …
        - iris scanners, body temperature, heart rate, galvanic skin response, blink rate

Paper: printing and scanning

Printing

- image made from small dots
    - allows any character set or graphic to be printed,
- critical features:
    - resolution
        - size and spacing of the dots
        - measured in dots per inch (dpi)
    - speed
        - usually measured in pages per minute

Scanners

- Take paper and convert it into a bitmap
- Two sorts of scanner
    - flat-bed: paper placed on a glass plate, whole page converted into bitmap
    - hand-held: scanner passed over paper, digitising strip typically 3-4” wide
- Shines light at paper and note intensity of reflection
    - colour or greyscale
- Typical resolutions from 600–2400 dpi

Used in

- desktop publishing for incorporating photographs and other images
- document storage and retrieval systems, doing away with paper storage
- special scanners for slides and photographic negatives

Optical character recognition(OCR)

- OCR converts bitmap back into text
- different fonts
    - create problems for simple “template matching” algorithms
    - more complex systems segment text, decompose it into lines and arcs, and decipher characters that way
- page format
    - columns, pictures, headers and footers

Memory

Short-term Memory – RAM

- Random access memory (RAM)
    - on silicon chips
    - 100 nano-second access time
    - usually volatile (lose information if power turned off)
    - data transferred at around 100 Mbytes/sec
- Some *non-volatile RAM* used to store basic set-up information

Long-term Memory – disks

- magnetic disks
    - floppy disks store around 1.4 Mbytes
    - hard disks typically 40 Gbytes to 100s of Gbytes
      access time ~10ms, transfer rate 100kbytes/s

- optical disks
    - use lasers to read and sometimes write
    - more robust that magnetic media
    - CD-ROM - same technology as home audio, ~ 600 Gbytes
    - DVD - for AV applications, or very large files

Blurring boundaries

- PDAs
    - often use RAM for their main memory
- Flash-Memory
    - used in PDAs, cameras etc.
    - silicon based but persistent
    - plug-in USB devices for data transfer

Speed and capacity

- some sizes (all uncompressed)
    - the Bible ~ 4.5 Mbytes
    - scanned page ~ 128 Mbytes
        - (11x8 inches, 1200 dpi, 8bit greyscale)
    - digital photo ~ 10 Mbytes
        - (2–4 mega pixels, 24 bit colour)
    - video ~ 10 Mbytes *per second*
        - (512x512, 12 bit colour, 25 frames per sec)

Virtual memory

- Problem:
    - running lots of programs + each program large
    - not enough RAM
- Solution - Virtual memory :
    - store some programs temporarily on disk
    - makes RAM appear bigger
- But swapping
    - program on disk needs to run again
    - copied from disk to RAM
    - slows things down

Compression

- reduce amount of storage required
- lossless

    - recover exact text or image – e.g. GIF, ZIP
    - look for commonalities:
        - text: AAAAAAAAAABBBBBCCCCCCCC 10A5B8C
        - video: compare successive frames and store change
- lossy
    - recover something like original – e.g. JPEG, MP3
    - exploit perception
        - JPEG: lose rapid changes and some colour
        - MP3: reduce accuracy of drowned out notes

Storage formats – text

- ASCII - 7-bit binary code for each letter and character
- UTF-8 - 8-bit encoding of 16 bit character set
- RTF (rich text format)

  - text plus formatting and layout information
- SGML (standardized generalised markup language)

  - documents regarded as structured objects
- XML (extended markup language)

  - simpler version of SGML for web applications
- Images:
    - many storage formats :

      (PostScript, GIFF, JPEG, TIFF, PICT, etc.)
    - plus different compression techniques

      (to reduce their storage requirements)
- Audio/Video
    - again lots of formats :

      (QuickTime, MPEG, WAV, etc.)
    - compression even more important
    - also 'streaming' formats for network delivery

Methods of access

- large information store
    - long time to search => use index
    - what you index -> what you can access
- simple index needs exact match
- forgiving systems:
    - Xerox "do what I mean" (DWIM)
    - SOUNDEX – McCloud ~ MacCleod
- access without structure …
    - free text indexing (all the words in a document)
    - needs lots of space!!

Processing and networks

Finite processing speed

- Designers tend to assume fast processors, and make interfaces more and more complicated
- But problems occur, because processing cannot keep up with all the tasks it needs to do
  - cursor overshooting because system has buffered keypresses
  - icon wars - user clicks on icon, nothing happens, clicks on another, then system responds and windows fly everywhere
- Also problems if system is too fast - e.g. help screens may scroll through text much too rapidly to be read

Moore's law

- computers get faster and faster!
- 1965 …
  - Gordon Moore, co-founder of Intel, noticed a pattern
  - processor speed doubles every 18 months
  - PC … 1987: 1.5 Mhz, 2002: 1.5 GHz
- similar pattern for memory
  - but doubles every 12 months!!
  - hard disk … 1991: 20Mbyte : 2002: 30 Gbyte
- baby born today
  - record all sound and vision
  - by 70 all life's memories stored in a grain of dust!

The myth of the infinitely fast machine

- implicit assumption … no delays - an infinitely fast machine
- what is good design for real machines?
- good example … the telephone :
  - type keys too fast
  - hear tones as numbers sent down the line
  - actually an accident of implementation
  - emulate in deisgn

Limitations on interactive performance

Computation bound

- Computation takes ages, causing frustration for the user

Storage channel bound

- Bottleneck in transference of data from disk to memory

Graphics bound

- Common bottleneck: updating displays requires a lot of effort - sometimes helped by adding a graphics co-processor optimised to take on the burden

Network capacity

- Many computers networked - shared resources and files, access to printers etc. - but interactive performance can be reduced by slow network speed

Networked computing

Networks allow access to

- large memory and processing
- other people (groupware, email)
- shared resources – esp. the web

Issues

- network delays – slow feedback
- conflicts - many people update data
- unpredictability

The Interaction

- interaction models
  - translations between user and system
- ergonomics
  - physical characteristics of interaction
- interaction styles
  - the nature of user/system dialog
- context
  - social, organizational, motivational

Some terms of interaction

Domain

- the area of work under study e.g. graphic design

Goal

- what you want to achieve e.g. create a solid red triangle

Task

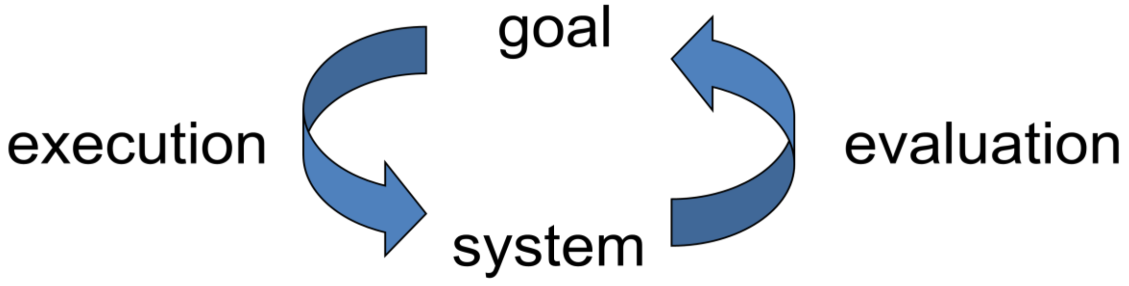
- how you go about doing it
- ultimately in terms of operations or actions

e.g. … select fill tool, click over triangle

Donald Norman's model

- Seven stages
    - user establishes the goal
    - formulates intention
    - specifies actions at interface
    - executes action
    - perceives system state
    - interprets system state
    - evaluates system state with respect to goal
- Norman's model concentrates on user's view of the interface

Execution/evaluation loop



Goal

- user establishes the goal

Execution

- formulates intention
- specifies actions at interface
- executes action

System

- perceives system state
- interprets system state

Evaluation

- evaluates system state with respect to goal

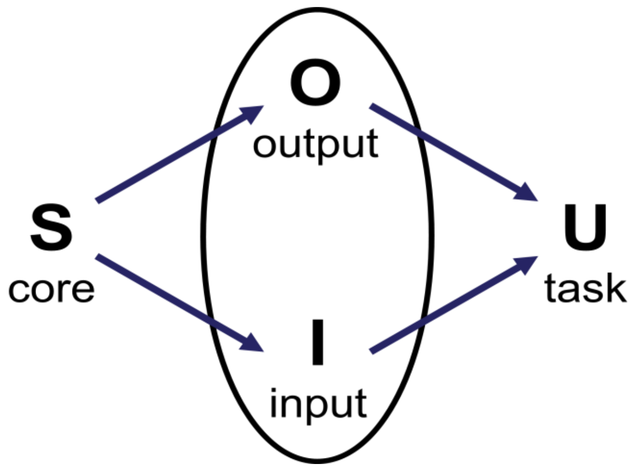
Using Norman’s model

Some systems are harder to use than others
Gulf of Execution
user’s formulation of actions
≠ actions allowed by the system

Gulf of Evaluation
user’s expectation of changed system state
≠ actual presentation of this state

Abowd and Beale framework

- It is an extension of Norman’s model
- Their interaction framework has 4 parts
    - user
    - input
    - system
    - output



- each has its own unique language
  interaction ⇒ translation between languages
- problems in interaction = problems in translation

Using Abowd & Beale's model

user intentions

→ translated into actions at the interface

→ translated into alterations of system state

→ reflected in the output display

→ interpreted by the user

General framework for understanding interaction

- not restricted to electronic computer systems
- identifies all major components involved in interaction
- allows comparative assessment of systems
- an abstraction

Ergonomics

- Study of the physical characteristics of interaction
- Also known as human factors – but this can also be used to mean much of HCI!
- Ergonomics good at defining standards and guidelines for constraining the way we design certain aspects of systems

Ergonomics – examples

- arrangement of controls and displays
  - e.g. controls grouped according to function or frequency of use, or sequentially
- surrounding environment
  - e.g. seating arrangements adaptable to cope with all sizes of user
- health issues
  - e.g. physical position, environmental conditions (temperature, humidity), lighting, noise,
- use of colour
  - e.g. use of red for warning, green for okay, awareness of colour-blindness etc.

Interaction styles

Common interaction styles

- command line interface
- menus
- natural language
- question/answer and query dialogue
- form-fills and spreadsheets
- WIMP
- point and click
- three–dimensional interfaces

Command line interface

- Way of expressing instructions to the computer directly
  - function keys, single characters, short abbreviations, whole words, or a combination
- suitable for repetitive tasks
- better for expert users than novices
- offers direct access to system functionality
- command names/abbreviations should be meaningful!
- Typical example: the Unix system

Menus

- Set of options displayed on the screen
- Options visible
  - less recall - easier to use
  - rely on recognition so names should be meaningful
- Selection by:
  - numbers, letters, arrow keys, mouse
  - combination (e.g. mouse plus accelerators)
- Often options hierarchically grouped
  - sensible grouping is needed
- Restricted form of full WIMP system

Natural language

- Familiar to user
- speech recognition or typed natural language
- Problems
  - vague

  - ambiguous
  - hard to do well!
- Solutions
  - try to understand a subset
  - pick on key words

Query interfaces

- Question/answer interfaces
  - user led through interaction via series of questions
  - suitable for novice users but restricted functionality
  - often used in information systems
- Query languages (e.g. SQL)
  - used to retrieve information from database
  - requires understanding of database structure and language syntax, hence requires some expertise

Form-fills

- Primarily for data entry or data retrieval
- Screen like paper form.
- Data put in relevant place
- Requires
  - good design
  - obvious correction facilities

Spreadsheets

- first spreadsheet VISICALC, followed by Lotus 1-2-3
  MS Excel most common today
- Sophisticated variation of form-filling.
  - grid of cells contain a value or a formula
  - formula can involve values of other cells e.g. sum of all cells in this column
  - user can enter and alter data spreadsheet maintains consistency

WIMP Interface

Windows

Icons

Menus

Pointers

… or windows, icons, mice, and pull-down menus!

- default style for majority of interactive computer systems, especially PCs and desktop machines

Point and click interfaces

- used in
  - multimedia
  - web browsers
  - hypertext
- just click something!
  - icons, text links or location on map
- minimal typing

Three dimensional interfaces

- virtual reality
- 3D workspaces
  - use for extra virtual space
  - light and occlusion give depth
  - distance effects

Elements of the wimp interface

Windows

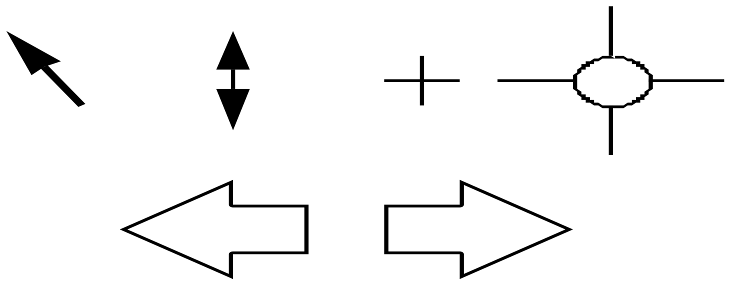
- Areas of the screen that behave as if they were independent
  - can contain text or graphics
  - can be moved or resized
  - can overlap and obscure each other, or can be laid out next to one another (tiled)
  - scrollbars
  - allow the user to move the contents of the window up and down or from side to side
- **title bars**
  - describe the name of the window

Icons

- small picture or image
- represents some object in the interface - often a window or action
- windows can be closed down (iconised) - small representation fi many accessible windows
- icons can be many and various - realistic representations.

Pointers

- important component
  - WIMP style relies on pointing and selecting things
- uses mouse, trackpad, joystick, trackball, cursor keys or keyboard shortcuts
- wide variety of graphical images



Menus

- Choice of operations or services offered on the screen
- Required option selected with pointer

Problem – take a lot of screen space

Solution – pop-up: menu appears when needed

Kinds of Menus

- Menu Bar at top of screen (normally), menu drags down
  - pull-down menu - mouse hold and drag down menu
  - drop-down menu - mouse click reveals menu
  - fall-down menus - mouse just moves over bar!
- Contextual menu appears where you are
  - pop-up menus - actions for selected object
  - pie menus - arranged in a circle
    - easier to select item (larger target area)
    - quicker (same distance to any option)

      … but not widely used!

Menus extras

- Cascading menus
  - hierarchical menu structure
  - menu selection opens new menu
  - and so on
- Keyboard accelerators
  - key combinations - same effect as menu item

        - two kinds
            - active when menu open – usually first letter
            - active when menu closed – usually Ctrl + letter

Menus design issues

- which kind to use
- what to include in menus at all
- words to use (action or description)
- how to group items
- choice of keyboard accelerators

Buttons

- individual and isolated regions within a display that can be selected to invoke an action
- Special kinds
    - radio buttons
      – set of mutually exclusive choices
    - check boxes
      – set of non-exclusive choices

Toolbars

- long lines of icons
- fast access to common actions
- often customizable:
    – choose *which* toolbars to see
    – choose *what* options are on it

Palettes and tear-off menus

- Problem
  menu not there when you want it
- Solution
  palettes – little windows of actions
    – shown/hidden via menu option
      e.g. available shapes in drawing package
  tear-off and pin-up menus
    – menu 'tears off' to become palette

Dialogue boxes

- Information windows that pop up to inform an important event or request information.
  e.g: when saving a file, a dialogue box is displayed to allow the user to specify the filename and location. Once the file is saved, the box disappears.
  Interaction design

What is interaction design?

- Designing interactive products to support people in their everyday and working lives
  - Sharp, Rogers and Preece (2002)
- The design of spaces for human communication and interaction
  - Winograd (1997)

Goals of interaction design

- Develop usable products
  - Usability means easy to learn, effective to use and provide an enjoyable experience
- Involve users in the design process

What to design

- Need to take into account:
  - Who the users are
  - What activities are being carried out
  - Where the interaction is taking place

- Need to optimise the interactions users have with a product
  - Such that they match the users activities and needs

Understanding users' needs

- Need to take into account what people are good and bad at

- Consider what might help people in the way they currently do things

- Listen to what people want and get them involved

- Use tried and tested user-based methods

Interaction design in business

- Increasing number of ID consultancies, examples of well known ones include:
  - Nielsen Norman Group: "help companies enter the age of the consumer, designing human-centered products and services"
  - Swim: "provides a wide range of design services, in each case targeted to address the product development needs at hand"
  - IDEO: "creates products, services and environments for companies pioneering new ways to provide value to their customers"

What do professionals do in the ID business?

- interaction designers - people involved in the design of all the interactive aspects of a product
- usability engineers - people who focus on evaluating products, using usability methods and principles
- web designers - people who develop and create the visual design of websites, such as layouts
- information architects - people who come up with ideas of how to plan and structure interactive products
- user experience designers - people who do all the above but who may also carry out field studies to inform the design of products

What is involved in the process of interaction design

- Identify needs and establish requirements
- Develop alternative designs
- Build interactive prototypes that can be communicated and assessed
- Evaluate what is being built throughout the process

Core characteristics of interaction design

- users should be involved through the development of the project
- specific usability and user experience goals need to be identified, clearly documented and agreed at the beginning of the project
- iteration is needed through the core activities

Usability goals

- Effective to use
- Efficient to use
- Safe to use
- Have good utility
- Easy to learn
- Easy to remember how to use

User experience goals

- Satisfying
- Fun
- Enjoyable
- Entertaining
- Helpful
- Motivating
- Aesthetically pleasing

- rewarding
- support creativity
- emotionally fulfilling

Design principles

- Generalizable abstractions for thinking about different aspects of design
- The do's and don'ts of interaction design
- What to provide and what not to provide at the interface
- Derived from a mix of theory-based knowledge, experience and common-sense

The five Design principles are: Visibility, Feedback, Constraints, Consistency and affordances

Visibility

- ▪ make relevant parts visible
- ▪ make what has to be done obvious

Feedback

- Sending information back to the user about what has been done
- Includes sound, highlighting, animation and combinations of these
    - e.g. when screen button clicked on provides sound or red highlight feedback

Constraints

- Restricting the possible actions that can be performed
- Helps prevent user from selecting incorrect options
- Three main types (Norman, 1999)
    - physical
    - cultural
    - logical

Physical constraints

- Refer to the way physical objects restrict the movement of things
    - E.g. only one way you can insert a key into a lock
- How many ways can you insert a CD or DVD disk into a computer?
- How physically constraining is this action?
- How does it differ from the insertion of a floppy disk into a computer?

Logical constraints

- Exploits people's everyday common sense reasoning about the way the world works
- An example is the logical relationship between physical layout of a device and the way it works as the next slide illustrates

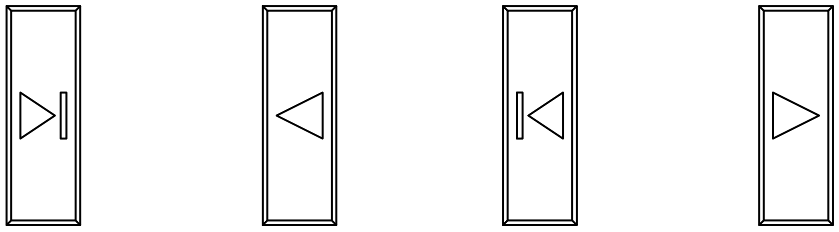
Cultural constraints

- Learned arbitrary conventions like red triangles for warning

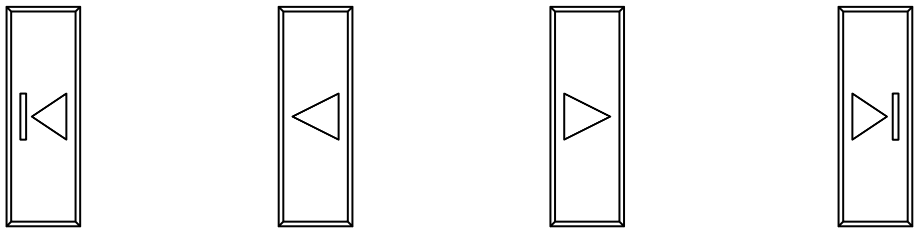
- Can be universal or culturally specific

Mapping

- Relationship between controls and their movements and the results in the world
- Why is this a poor mapping of control buttons?



- Why is this a better mapping?



- The control buttons are mapped better onto the sequence of actions of fast rewind, rewind, play and fast forward

Consistency

- Design interfaces to have similar operations and use similar elements for similar tasks
- For example:
  - always use ctrl key plus first initial of the command for an operation – ctrl+C, ctrl+S, ctrl+O
- Main benefit is consistent interfaces are easier to learn and use

When consistency breaks down

- What happens if there is more than one command starting with the same letter?
  - e.g. save, spelling, select, style
- Have to find other initials or combinations of keys, thereby breaking the consistency rule
  - E.g. ctrl+S, ctrl+Sp, ctrl+shift+L
- Increases learning burden on user, making them more prone to errors

Internal and external consistency

- Internal consistency refers to designing operations to behave the same within an application
  - Difficult to achieve with complex interfaces
- External consistency refers to designing operations, interfaces, etc., to be the same across applications and devices
  - Very rarely the case, based on different designer's preference

Affordances: to give a clue

- Refers to an attribute of an object that allows people to know how to use it
  - e.g. a mouse button invites pushing, a door handle affords pulling
- Norman (1988) used the term to discuss the design of everyday objects
- Since it has been much popularised in interaction design to discuss how to design interface objects
  - e.g. scrollbars to afford moving up and down, icons to afford clicking on

What does 'affordance' have to offer interaction design?

- Interfaces are virtual and do not have affordances like physical objects
- Norman argues it does not make sense to talk about interfaces in terms of 'real' affordances
- Instead interfaces are better conceptualised as 'perceived' affordances
  - Learned conventions of arbitrary mappings between action and effect at the interface
  - Some mappings are better than others

Usability principles

- Similar to design principles, except more prescriptive
- Used mainly as the basis for evaluating systems
- Provide a framework for heuristic evaluation

Usability principles (Nielsen 2001)

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Help users recognize, diagnose and recover from errors
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help and documentation

User Centered Design

The design is based upon a user's:

- abilities and needs
- context
- work
- tasks

Usability Engineering

Defining Usability

The ISO defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." [ISO, 1998].

Six Usability Attributes

1. Effectiveness: completeness with which users achieve their goal.
2. Learnability: ease of learning for novice users.
3. Efficiency: steady-state performance of expert users.
4. Memorability: ease of using system intermittently for casual users.
5. Errors: error rate for minor and catastrophic errors.
6. Subjective Satisfaction: how pleasant system is to use.

Measuring Usability Attributes

- Effectiveness: decide on definition of success. For example, number of substitution words spotted in a text, or binary measure of success (order completed or not).
- Learnability: pick novice users of system, measure time to perform certain tasks. Distinguish between no/some general computer experience.
- Efficiency: decide definition of expertise, get sample expert users (difficult), measure time to perform typical tasks.
- Memorability: get sample casual users (away from system for certain time), measure time to perform typical tasks.
- Errors: count minor and catastrophic errors made by users while performing some specified task. For example, number of deviations from optimal click path.
- Satisfaction: ask users' subjective opinion (questionnaire), after trying system for real task.

Usability Evaluation

There are three types of evaluation, according to the purpose of the evaluation:

- Exploratory - how is it (or will it be) used?
- Formative - how can it be made better?
- Summative - how good is it?

Exploratory Evaluation

This explores current usage and the potential design space for new designs.

- Done before interface development.
- Learn which software is used, how often, and what for.
- Collect usage data – statistical summaries and observations of usage.

Formative Evaluation

This informs the design process and helps improve an interface during design.

- Done during interface development.
- Learn why something went wrong, not just that it went wrong.
- Collect process data – qualitative observations of what happened and why.

Summative Evaluation

This assesses the overall quality of an interface.

- Done once an interface is (more or less) finished.
- Either compare alternative designs, or test definite performance requirements.
- Collect bottom-line data – quantitative measurements of performance: how long did users take, were they successful, how many errors did they make.

Usability Evaluation Methods

The methods of usability evaluation can also be classified according to who performs them:

- Usability Inspection Methods: Inspection of interface design by usability specialists using heuristics and judgement (no test users).
- Usability Testing Methods: Empirical testing of interface design with real users.

The Usability Engineering Lifecycle

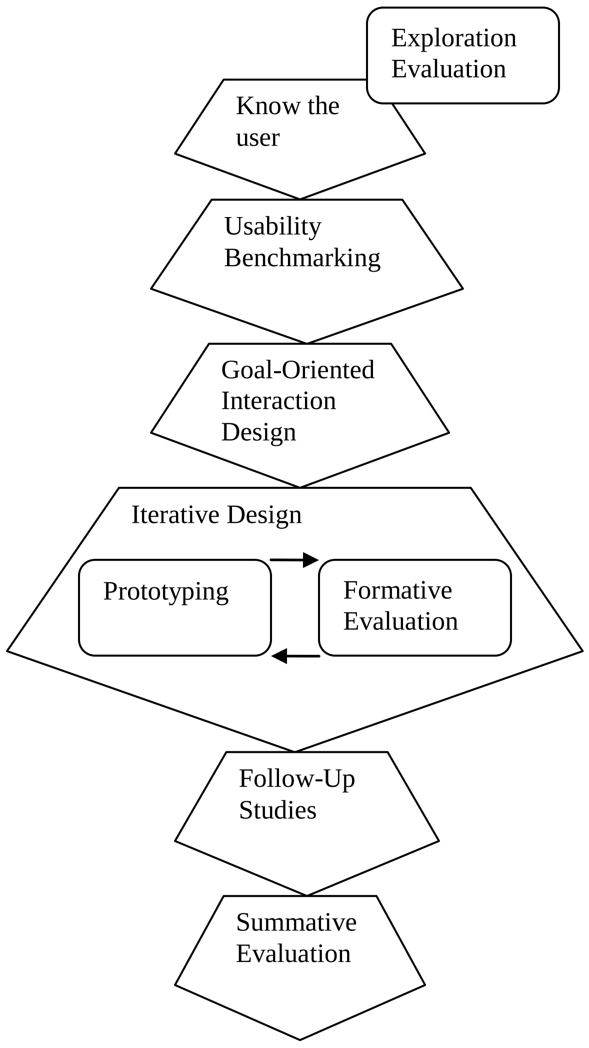


Figure: The usability engineering lifecycle.

1. Know the User

2. Usability Benchmarking

3. Goal-Oriented Interaction Design

4. Iterative Design:

(a) Prototyping

(b) Formative Usability Evaluation (Inspection and/or Testing)

5. Summative Usability Evaluation

6. Follow-up Studies

Know the User

- Qualitative research: observation of users and interviews.
- Exploratory evaluation: which software is used, how is it used, and what is it used for.
- Draw up a user profile for each (potential) class of user, based on behavioural and demographic variables.
- Identify user goals and attitudes.
- Analyse workflow and context of work.
- Draw up a set of typical user scenarios.

Usability Benchmarking

- Analyse competing products or interfaces heuristically and empirically.
- Set measurable usability targets for your own interface.

Interaction Design

Goal-oriented initial design of interface.

Iterative Design

"Design, Test, Redesign."

Build and evaluate prototype interface, then:

- Severity ratings of usability problems discovered.
- Fix problems!new version of interface.
- Capture design rationale: record reasons why changes were made.
- Evaluate new version of interface

Building Prototypes

- Verbal description.
- Paper prototype.
- Working prototype.
- Implementation of final design.

Formative and Summative Usability Evaluation

The usability evaluation methods are described according to who performs them:

- Usability inspection methods
- Usability testing methods

Follow-Up Studies

Important usability data can be gathered after the release of a product for the next version:

- Specific field studies (interviews, questionnaires, observation).
- Standard marketing studies (what people are saying in the newsgroups and mailing lists, reviews and tests in magazines, etc.).
- Instrumented versions of software log data.
- Analyse user complaints to hotline, modification requests, bug reports.

Planning Usability Activities
1. Prioritise activities.
2. Write down explicit plan for each activity.
3. Subject plan to independent review (e.g. colleague from different project).
4. Perform pilot activity with about 10% of total resources, then revise plan for remaining 90%.

Know the User
Qualitative research is used to determine user characteristics, goals, and context of use.

Classifying Users
Users can be classified according to their:

- experience
- educational level
- age
- amount of prior training, etc.

know the user

- Minimal Computer Experience
- Extensive Computer Experience
- Ignorant about Domain Knowledge
- Expert User of System
- Novice User of System

Categories of User Experience
User experience can be thought of along three dimensions: experience of computers in general, understanding of the task domain, and expertise in using the specific system.

Learning Curves

- Some systems are designed to focus on learnability.
- Others emphasise efficiency for proficient users.
- Some support both ease of learning and an "expert mode" (for example rich menus and dialogues plus a command/scripting language)

Most Users are Perpetual Intermediates
The experience level of people using computer software tends, like most population distributions, to follow the classical statistical bell curve (normal distribution).
In terms of using a software interface, the bell curve represents a snapshot in time:

- Beginners do not remain beginners for long.
- The difficulty of maintaining a high level of expertise means that experts fade over time.
- Most users gravitate over time towards intermediacy.
- Most users are neither beginners nor experts: they are perpetual intermediates.

Time, Efficiency

- Focus on Expert User
- Focus on Novice User
- Learning curves for hypothetical systems focusing on the novice user (easy to learn, but less efficient to use) and the expert user (harder to learn, but then highly efficient).

Research the Frames of Reference

Conduct interviews with:

- Project staff (managers, programmers, marketing people) who are in charge of developing the software.
- Subject matter and domain experts.
- Customers (the purchaser of the product, not necessarily the same as the end user).to determine values, expectations, issues, and constraints.

Interviewing Project Staff

- One-on-one interviews.
- Try to discover:
  - vision of the product.
  - budget and schedule.
  - technical constraints.
  - perceptions of who users might be.

Interviewing Subject Matter Experts (SMEs)

- Often hired externally by project manager.
- Provide knowledge of complex domains, regulations, industry best practice.
- Often lean towards expert user perspective (rather than intermediate).

Interviewing Customers

- Customers are the people who make the decision to purchase.
- For consumer products, customers are often the same as users.
- For business settings, customers are rarely actually the users of a product.
- Try to discover the customer’s:
  - goals in purchasing the product
  - frustrations with current solutions
  - decision process for purchasing
  - role in installation and maintenance

Research the End User

The actual users of a product should always be the main focus of the design effort.

- Most people are incapable of accurately assessing their own behaviour [Pinker, 1999].
- Rather than talk to users about how they think they behave, it is better to observe their behaviour first-hand.
- And then ask clarifying questions in the context of use.

Ethnographic Interviews

This is a combination of immersive observation and directed interview techniques.

- Observe the user using their current tools in their normal environment.
- Interviewer assumes the role of an apprentice learning from the master craftsman (user).
- Alternate between observation of work and discussion of its structure and details.

Identifying Candidate Users

Designers must capture the range of user behaviours regarding a product.

- What sorts of people might use this product?
- How might their needs vary?
- What ranges of behaviour might be involved?
- Which kinds of environment might be involved?

Try to interview some people from each different group.

Conducting an Ethnographic Interview

- In actual workplace/environment.
- 45-60 minutes.
- No third parties (supervisors or clients).
- Focus on understanding:
    - Overall goals
    - Current tasks
    - Constraints and exceptions
    - Problems needing solution (where does it hurt?)
    - Broader context
    - Domain issues
    - Vocabulary
- Ask permission to take a few photographs of the user and their workplace (for creating personas).

Usability Benchmarking

Usability benchmarking:

- how usable is the competition?
- how much better should your interface be?
- what is your likely return on investment?

Competitive Analysis

Competitive analysis of competing systems:

- Determine the current state of the art and decide how far to raise the bar.
- Analyse competing products or interfaces heuristically or empirically.
- "Intelligent borrowing" of ideas from other systems.

Set Usability Targets

Decide in advance on usability metrics and desired level of measurable usability (usability targets). For example:

- The current system exhibits 4.5 errors per hour on average for an experienced user. The target for the new version is less than 3 errors per hour.
- From competitive analysis, on the main competing web site, novice users take 8 mins. and 21 secs. on average to book a flight. The target for our new web site is 6 mins.

Return on Investment

Estimate return on investment (ROI) by performing a financial impact analysis:

- Compare potential savings based on loaded cost of users to the estimated cost of the usability effort.
- Jakob Nielsen concludes [Nielsen, 2003] that current best practices call for devoting about 10% of a project's budget to usability.

Goal-Oriented Interaction Design

Designing software based on an understanding of human goals.

- A goal is a final purpose or aim, an objective.
- Tasks are particular ways of accomplishing a goal.

Programmers versus users

Programmer	User
Wants control, will accept some complexity.	Wants simplicity, will accept less control.
Wants to understand, will accept some failure.	Wants success, will accept less understanding.
Concerned with all possible cases, will accept advance preparation.	Concerned with probable cases, will accept occasional setbacks.

Interaction Design versus Interface Design

- Interface design suggests an interface between code on one side and users on the other side, passing messages between them. It gives programmers a licence to code as they please, because the interface will be slapped on when they are done.
- Interaction design refers to function, behaviour, and final presentation.
- Programmers are good at designing the inside of software
- Interaction designers should design the outside.

Personal and Corporate Goals

Personal and corporate goals are different.

- Both are the highest expression of goals for their respective owners (both must be taken into account).
- But people are doing the work, and their personal goals will always take precedence (although they are rarely discussed, precisely because they are personal).

The Interaction Design Process

1. Interview users
2. Create personas
3. Define their goals
4. Create concrete scenarios
5. Move to a design solution

The Design Team

Two designers in core team:

- Designer: generates ideas, leads the process.
- Design Communicator: articulates half-formed ideas, writes design specifications.

Creating Personas

From the insight you gained in your interviews, you now invent user archetypes to represent the main user groups of your product. In other words, you make up pretend users and design for them.

What is a Persona?

A persona is a prototypical user:

- An imaginary, but very specific, example of a particular type of user.
- Cars are designed to appeal to different kinds of drivers with different needs and goals.
- Not “real”, but hypothetical.

A persona is used to role-play through an interface design and check if the design would meet the needs of such a user.

A Good Persona

- A good persona is not “average”, but typical and believable.
- If the set of users interviewed were somehow plotted according to their characteristics as a cloud of points, the best ones to base personas on would be the ones around the edges.
- If our design satisfies the hard cases around the edges, the ones in the middle should be able to use the interface as well.

Define the Persona Precisely

- Specify a name, age, face, and quirky, believable detail.
- For faces, use stock photos from CD-ROM or the internet, or photographs taken during user interviews.
- It is more important to define the persona in great and specific detail, so that it cannot wiggle under the pressure of development, than that the persona be exactly the right one.

Defining Goals for each Persona

Goals and personas co-exist. A persona exists to achieve his goals, a goal exists to give meaning to a persona. Define the goals of each persona.

Defining Scenarios for each Persona

A scenario is a precise description of a persona using an interface to achieve a goal.

- Daily Use Scenarios: the primary actions the user will perform. These need the most robust design.
- Necessary Use Scenarios: More occasional, infrequent actions, which are necessary from time to time.
- Edge Case Scenarios: Loved by programmers, these can largely be ignored during the design process.

As the design progresses, play act the personas through the scenarios to test the validity of the design.

Moving to a Design Solution

Parallel Design

- If time and resources allow, explore design alternatives.
- Have several design teams work independently, then compare draft designs.

PROTOTYPING

- Prototyping is a way of developing prototypes
- Prototyping is used to Perform usability evaluation and get feedback as early as possible in the design cycle by building and evaluating prototypes.

What is a prototype?
In interaction design it can be (among other things):

- a series of screen sketches
- a storyboard, i.e. a cartoon-like series of scenes
- a Power point slide show
- a video simulating the use of a system
- a cardboard mock-up
- a piece of software with limited functionality written in the target language or in another language

Why prototype?

- Evaluation and feedback are central to interaction design
- Stakeholders can see, hold, interact with a prototype more easily than a document or a drawing
- Team members can communicate effectively
- You can test out ideas for yourself
- It encourages reflection: very important aspect of design
- Prototypes answer questions, and support designers in choosing between alternatives

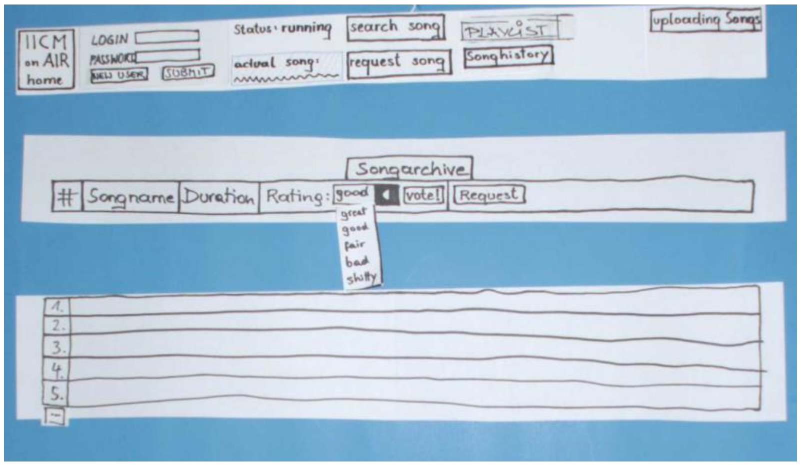
What to prototype?

- Technical issues
- Work flow, task design
- Screen layouts and information display
- Difficult, controversial, critical areas

Types of prototypes in increasing order of complexity:

- Verbal Prototypes: textual description of choices and results.
- Paper Prototypes:
    - Low-Fidelity: hand-drawn sketches.
    - High-Fidelity: more elaborate printouts.
- Interactive Sketches: interactive composition of hand-drawn sketches.
- Working Prototypes: interactive, skeleton implementation.

Finally, throw prototypes away and implement final design.



Paper prototype of IICM on Air.

Verbal Prototype

Simple textual description of choices and results.

Low-Fidelity Paper Prototypes

- Paper prototypes simulate screen and dialogue elements on paper.
- First hand-drawn sketches (lo-fi), later perhaps more elaborate printouts (hi-fi).
- Early usability feedback with throwaway designs: maximum feedback for minimum effort!

High-Fidelity Paper Prototypes

- Elaborate screen designs created with drawing editors such as Adobe Illustrator or Corel Draw.
- Printed out in colour.
- The often look too much like a finished design, and not enough like a prototype.
- Users tend to comment on the choice of fonts and colours, rather than the flow through the application.

Interactive Sketches

- Scan in hand-drawn interface sketches.
- Assemble interactive prototype with clickable elements (say with Macromedia Director).
- Retains throwaway, casual look to encourage criticism and discussion.

Working Prototypes

- Simple algorithms: ignore special cases.
- Fake data: similar data, images instead of video, etc.
- Wizard of Oz: human expert operating behind the scenes to simulate interface responses.
- Prototyping tools: e.g. Director, ToolBook.
- UIMS (User Interface Management Systems): interactive interface builders such as Visual Basic.

Dimensions of Working Prototypes

Working prototypes cut down on either the number of features, or the depth of functionality of features:

- Vertical Prototype: in-depth functionality for a few selected features.
- Horizontal Prototype: full interface features, but no underlying functionality.
- Scenario Prototype: only features and functionality along the specific scenarios or paths through the interface which are to be evaluated.

Implementation

Implement final design.
Competitive analysis of software components:

- Use existing interface framework as far as possible (Motif, MS-Windows, Java Swing) – saves a lot of work.
- Use existing components and applications rather than re-inventing the wheel.

Usability Inspection Methods

Inspection of interface design using heuristic methods (based on analysis and judgement rather than experiment).

1. Heuristic Evaluation: A small team of evaluators inspects an interface using a small checklist of general principles and produces an aggregate list of potential problems.
2. Guideline Checking: An evaluator checks an interface against a detailed list of specific guidelines and produces a list of deviations from the guidelines.
3. Cognitive Walkthrough: A small team walks through a typical task in the mind set of a novice user and produces a success or failure story at each step along the correct path. (analyses learnability)
4. Guideline Scoring: An evaluator scores an interface against a detailed list of specific guidelines and produces a total score representing the degree to which an interface follows the guidelines.
5. Action Analysis: An evaluator produces an estimate of the time an expert user will take to complete a given task, by breaking the task down into ever smaller steps and then summing up the atomic action times. (analyses efficiency)

Heuristic Evaluation

- First described in [Nielsen and Molich, 1990].
- Small team of evaluators (usually usability specialists) systematically checks interface design against small set of recognised usability principles (the “heuristics”).

Usability Heuristics

1. Visibility of System Status [Feedback]
   The system should always keep users informed about what is going on, through appropriate feedback within reasonable time. For example: busy cursor [1–10s], progress indicator [>10s].
2. Match Between System and the RealWorld [Speak the Users’ Language]
   The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order. Match users’ mental model. Beware of misleading metaphors.
3. User Control and Freedom [Clearly Marked Exits]
   Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. Consistency and Standards [Consistency]
   Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. Error Prevention
   Even better than good error messages is a careful design which prevents a problem from occurring in the first place. For example: select file from menu rather than typing in name, confirmation before dangerous actions, beware of modes, avoid similar command names.
6. Recognition rather than Recall
   Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate. Users’ short-term memory is limited. Provide examples, default values, easily retrievable instructions.
7. Flexibility and Efficiency of Use [Accelerators]
   Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. For example: abbreviations, command keys, type-ahead, edit and reissue previous commands, menu of most recently used files, macros.
8. Aesthetic and Minimalist Design [Minimalist Design]
   Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility. “Less is more”
9. Help Users Recognise, Diagnose, and Recover from Errors [Good Error Messages]

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution. Phrase error messages defensively, never blame the user. Multilevel messages. Link to help system.

10. Help and Documentation
    Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Merits and demerits of Heuristic Evaluation

Merits

- cheap
- intuitive
- usable early in development process
- finds many problems
- finds both major and minor problems

Demerits

- may miss domain-specific problems

Guideline Checking

- Checks guidelines on specific advice about usability characteristics of an interface.
- An evaluator checks an interface against a detailed list of specific guidelines and produces a list of deviations from the guidelines.
- Whereas heuristic evaluation employs 10 broad principles, guideline checking often involves dozens (or hundreds) of more specific individual items on a checklist.

Example Sets of Guidelines

- Sidney Smith and Jane Mosier; Design Guidelines for Designing User Interface Software;

Merits and demerits of Guideline Checking

Merits

- cheap
- intuitive
- usable early in development process

Demerits

- time-consuming
- can be intimidating – often hundreds or thousands of specific guidelines.

Cognitive Walkthrough

This is a task-oriented walkthrough of interface, imagining novice users' thoughts and actions. It focuses explicitly on learnability.

- Design may be mock-up or working prototype.
- Analogous to structured walkthrough in software engineering.
- Based on cognitive model (CE+) of human exploratory learning.

Exploratory Learning

Rather than read manual or attend course, users often prefer to learn new system by "trial and error"! exploratory learning [Carroll and Rosson, 1987]:

- Start with rough idea of task to be accomplished.
- Explore interface and select most appropriate action.
- Monitor interface reactions.
- Determine what action to take next.

The CE+ Model of Exploratory Learning

Based on psychological studies, the CE+ model describes exploratory learning behaviour in terms of 3 components:

- Problem-Solving Component
  User chooses among alternative actions based on similarity between the expected consequences of an action and the current goal. After executing selected action, user evaluates system response and decides whether progress is being made toward the goal. A mismatch results in an undo.
- Learning Component
  When above evaluation process leads to positive decision, the action taken is stored in long-term memory as a rule.
- Execution Component
  User first attempts to fire applicable rule matching current context. If none found, problem-solving component is invoked.

Cognitive Walkthrough Preparation

a) Identify user population.
b) Define suite of representative tasks.
c) Describe or implement interface or prototype.
d) Specify correct action sequence(s) for each task.

Cognitive Walkthrough Steps

For each action in solution path, construct credible "success" or "failure" story about why user would or would not select correct action.

Critique the story to make sure it is believable, according to four criteria:

a) Will the user be trying to achieve the right effect?
   What is users' goal – will they want to select this action?
b) Will the user know that the correct action is available?
   Is control (button, menu, switch, triple-click, etc.) for action apparent (visible)?
c) Will the user know that the correct action will achieve the desired effect? Once users find control, will they recognise that it is the correct control to produce the desired effect?
d) If the correct action is taken, will the user see that things are going ok? After correct action, will users realise progress has been made towards the goal (feedback)?

Note that CW always tracks the correct action sequence. Once the user deviates from the correct path their further progress is no longer considered.

Group Walkthrough

- Performed by mixed team of analysts (designers, engineers, usability specialist).
- Capture critical information on three group displays (flip charts, overheads):

1. User knowledge (prior to and after action).
2. Credible success or failure story.
3. Side issues and design changes.

Merits and demerits of Cognitive Walkthrough
Merits and demerits

- finds task-oriented problems
- helps define users' goals and assumptions
- usable early in development process

Demerits

- time-consuming
- some training required
- needs task definition methodology
- applies only to ease of learning problems

Guideline Scoring

- The interface is scored according to its conformance against a weighted list of specific guidelines.
- A total score is produced, representing the degree to which an interface follows the guidelines.

Merits and demerits of Guideline Scoring
Merits

- cheap
- intuitive

Demerits

- must select and weight guidelines
- guidelines or weightings often domain-dependent

Action Analysis

This a quantitative analysis of actions to predict time skilled user requires to complete tasks, based on time estimates for typical interface actions. It focuses on performance of skilled user (efficiency).

Two flavours (levels of detail):

a) Formal or "Keystroke-Level"
b) Informal or "Back-of-the-Envelope"

Keystroke-Level Analysis

Described by [Card et al., 1983].

- Developed from GOMS (Goals, Operators, Methods, Selection) modeling.
- Extremely detailed, may often predict task duration to within 20%, but very tedious to carry out.
- Used to estimate performance of high-use systems (e.g. telephone operator workstations).

Procedure for Keystroke-Level Analysis

- Break down tasks hierarchically into subtasks until reach fraction of second level of detail.
- Use average values for action times (determined through extensive published research) to predict expected performance for particular task. See Table below.
- The analysts do not measure action times themselves, but refer to published tables.

	Action	Time
Physical Movements	One keystroke	0.28
	Point with mouse	1.5
	Move hand to mouse or function key	0.3
Visual Perception	Respond to brief light	0.1
	Recognise 6-letter word	0.34
	Move eyes to new location on screen	0.23
Mental Actions	Retrieve one item from long-term memory	1.2
	Learn one step of a procedure	25
	Execute a mental step	0.075
	Choose among methods	1.2

Table: Average Times for typical keystroke-level actions, in seconds. From [Olson and Olson, 1990], and cited by [Lewis and Rieman, 1993].

Back-of-the-Envelope Action Analysis

From [Lewis and Rieman, 1993]. "Back-of-the-Envelope" uses the analogy of sketching out a rough analysis on the back side of an envelope while somewhere away from your desk ("Milchm¨adchenrechnung" in German).

- List actions required to complete a task (as before), but in much less detail – at level of explaining to a user:

"Select Save from the File menu"
"Edit the file name"
"Confirm by pressing OK"

- At this level of analysis, every action takes at least 2 to 3 seconds (videotape a few users doing random tasks if you do not believe it takes this long!).

- Allows quick estimation of expected performance of interface for particular task.

Merits and demerits of Action Analysis
Merits

- predicts efficiency of interface before building it

Demerits

- time-consuming
- some training required

Objects-Actions Interface Model

There exists two basic interaction models for any given system :

- Object-Action model : The user first selects an object and then selects the action to be performed on the selected object
- Action-Object model : The user first selects an action to be performed and then selects the objects on which this action will be performed.

Scope/Application

- There has been always distinction between those two models in several domains starting long ago by compiler designers.
- These designers always wanted to distinguish between;
  - **syntax, which is the different tokens parsed from a source code file,**

    - **and the semantics, which are the actual operations invoked by that text.**
- Different areas use the syntactic-semantic model of human behavior, such as programming, database manipulation facilities and direct manipulation (Shneiderman 1980,1981)
- The OAI model is also in harmony with the software engineering model of Object Oriented programming model that has been popular in the past decades.

Example

- To distinguish the difference between the two models, consider the course of interaction of a user using a command based system (UNIX) versus using a direct manipulation GUI environment (WINDOWS).
- The task is to copy a file from a directory (Unix) or folder (Windows) to another location
- In the command based system which is based on the Action-Object model, the user starts by specifying the action to be performed, which is "copy" on our case, next he specifies the objects on which this action will be performed, which are the file name and the destination folder
- In contrast, a user on a GUI based environment like windows chooses the object by clicking on the file name, and then performs the copy action by dragging the file from the current folder to the new folder
- Which is synonymous with real life action?

Principles
Syntactic knowledge:

- This is the information necessary to be maintained and memorized by a user to be able to use a certain system efficiently.
- For example a user of a certain programming language can use it efficiently only when he knows a good deal of commands and syntax specific to this language.
- This kind of knowledge has the following drawbacks:
- Syntactic knowledge is system dependent. It's hard to apply previous system knowledge to the new one.
- This goes from using different keyboard layout to the different syntax necessary to perform each task
    - The reason behind that is that the user builds a mental model of a system while interacting with it.
    - Shifting to a new system, supposedly similar in functionality, the user tries to apply the previous model to that system and faces frustration when the results he gets are not the ones he is expecting.

The OAI model:

- Designing an OAI model starts with examining and understanding the tasks to be performed by the system.
- The domain of tasks include the universe of objects within which the user works to accomplish a certain goal as well as the domain of all possible actions performed by the user.
- Once these tasks objects and actions are agreed upon, the designer starts by creating an isomorphic representation of the corresponding interface objects and actions.

- The interface actions are usually performed by pointing device or keyboard and hence have to be visual to the user so that the later can decompose his plan into steps of actions such as pointing, clicking, dragging, etc...
- It is easy to conclude that people learn these issues independently from the underlying implementation on a certain system.
- We note that a user has to be first proficient in the task domain before using an interface to accomplish those real-world tasks

Task hierarchies of objects and actions:

For the designer:
The following steps are recommended (Shneiderman) in order to build correct tasks hierarchies by designers for a system:

1. Know about the users and their tasks (Interviewing users, reading workbooks and taking training sessions)
2. Generate hierarchies of tasks and objects to model the users' tasks
3. Design interface objects and actions that metaphorically map to the real world universe

Interface hierarchies of objects and actions:

- Similar to the task domain, the interface domain contains hierarchies of objects and tasks at different levels

Interface Objects:

- Users interacting with a computer get to understand some high level concepts relevant to that system. As an example, they learn that computer stores information, that these information are stored in files contained within a hierarchy of directories, and that each file has its own attributes like name, size, date, etc

Interface Actions:

- These are also hierarchies of lower levels actions. A high level plan is to create a text file might involve mid-level actions such as creating a file, inserting text and saving that file.
- The mid-level action of saving a file the file can be decomposed into lower level actions such as storing the file with a backup copy and may be applying the access control rights.
- Further lower level actions might involve choosing the name of the file, the location folder to be saved in, dealing with errors such as space shortage, and so on...

For the user:

- There are several ways users learn interface objects and actions such as demonstrations, sessions, or trial and error sessions. When these objects and actions have logical structure that can be related to other familiar task objects and actions, this knowledge becomes stable in the user's memory.

For the designer:

- The OAI model helps a designer to understand the complex processes that a user has to perform in order to successfully use an interface to perform a certain task. Designers model the interface actions and objects based on familiar example and then fine tune these models to fit the task and the user.

Limitations and challenges:

- It's hard to apply a series of actions, that you already performed on an object, to another set of objects (what command line users call "batching") ..
- It is hard to integrate batching techniques in visual environments since those techniques rely mainly on the concept of applying a sequence of actions on some objects, some of which exist only as intermediate results from an action in the same sequence
- It's also hard in the object-action model to perform pipelining .
- In a pipeline, outputs of previous action-object command can be fed to another action and so on .
- Constructing a pipeline in Unix is simple and proven very efficient ,again because of the nature of the model used.
- In an objects-actions model environment, it is hard to construct pipelines and the way people accomplish such tasks is creating intermediate objects and applying subsequent actions on these objects.
- One last challenge appears when users start to become proficient using a certain system, and their performance curve start to raise.
- At a certain time, in a visual environment, and for certain natures of tasks, objects-action model will perform poorly.
- One of the examples is programmers using a visual tool to design an object oriented program by visually drawing classes, objects and relationships.
- A skilled programmer is much faster typing than using a mouse. shifting hands between the keyboard and mouse (to draw an object and name it) introduces certain delay

GOMS

What is GOMS?

- A family of user interface modeling techniques
- Goals, Operators, Methods, and Selection rules
- Input: detailed description of UI and task(s)
- Output: various qualitative and quantitative measures

What GOMS can model

- Task must be goal-directed
  - Some activities are more goal-directed than others
  - Even creative activities contain goal-directed tasks
- Task must a routine cognitive skill - as opposed to problem solving as in Cognitive Walkthrough

- Serial and parallel tasks

GOMS Output

- Functionality coverage and consistency
    - Does UI contain needed functions?
    - Are similar tasks performed similarly? (NGOMSL only?)
- Operator sequence
    - In what order are individual operations done?
    - Abstraction of operations may vary among models
- Execution time
    - By expert
    - Very good rank ordering
    - Absolute accuracy ~10-20%
- Procedure learning time
    - Accurate for relative comparison only
    - Does not include time for learning domain knowledge
- Error recovery

A**pplications of GOMS analysis**

- Compare UI designs
- Profiling
- Sensitivity and parametric analysis
- Building a help system
    - GOMS modelling makes user tasks and goals explicit
    - Can suggest questions users will ask and the answers

Advantages of GOMS

- Gives several qualitative and quantitative measures
- Model explains **why** the results are what they are
- Less work than user study
- Easy to modify when interface is revised
- Research ongoing for tools to aid modeling process

Disadvantages of GOMS

- Not as easy as heuristic analysis, guidelines, or cognitive walkthrough
- Only works for goal-directed tasks
- Assumes tasks are performed by expert users
- Evaluator must pick users' tasks/goals
- Does not address several important UI issues, such as
    - readability of text
    - memorability of icons, commands
- Does not address social or organizational impact

HCI Interface Design

Initial Design

Copy Interaction Techniques From Other Systems

- look at what other applications are doing
- become familiar with leading applications
- learn WHY things are done the way they are.
  - e.g. why tool palettes and not pull down menus

HCI principles on Good Design

Shneiderman's Principles of Human-Computer Interface Design:

- Recognize Diversity - In order to recognize diversity, you, the designer, must take into account the type of user frequenting your system, ranging from novice user, knowledgeable but intermittent user and expert frequent user.
- Each type of user expects the screen layout to accommodate their desires, novices needing extensive help, experts wanting to get where they want to go as quickly as possible.
- Accommodating both styles on the same page can be quite challenging.
- You can address the differences in users by including both menu or icon choices as well as commands (i.e. Command or Control P for Print as well as an icon or menu entry), or providing an option for both full descriptive menus and single letter commands.

You Should Use the Eight Golden Rules of Interface Design:

1. Strive for consistency consistent sequences of actions should be required in similar situations
   - identical terminology should be used in prompts, menus, and help screens
   - consistent color, layout, capitalization, fonts, and so on should be employed throughout.
2. Enable frequent users to use shortcuts
   - to increase the pace of interaction use abbreviations, special keys, hidden commands, and macros
3. Offer informative feedback
   - for every user action, the system should respond in some way (in web design, this can be accomplished by DHTML - for example, a button will make a clicking sound or change color when clicked to show the user something has happened)
4. **Design dialogs to yield closure**
   - Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions shows the user their activity has completed successfully
5. **Offer error prevention and simple error handling**
   - design the form so that users cannot make a serious error; for example, prefer menu selection to form fill-in and do not allow alphabetic characters in numeric entry fields
   - if users make an error, instructions should be written to detect the error and offer simple, constructive, and specific instructions for recovery

- segment long forms and send sections separately so that the user is not penalized by having to fill the form in again - but make sure you inform the user that multiple sections are coming up

6. **Permit easy reversal of actions**
   - Allow a user reverse any action he/she has taken. This facilitates faster recovery from certain errors

7. **Support internal locus of control**
   - Experienced users want to be in charge. Surprising system actions, tedious sequences of data entries, inability or difficulty in obtaining necessary information, and inability to produce the action desired all build anxiety and dissatisfaction
8. **Reduce short-term memory load**
   - A famous study suggests that humans can store only 7 (plus or minus 2) pieces of information in their short term memory. You can reduce short term memory load by designing screens where options are clearly visible, or using pull-down menus and icons
9. **Prevent Errors** - prevent errors whenever possible. Steps can be taken to design so that errors are less likely to occur, using methods such as organizing screens and menus functionally, designing screens to be distinctive and making it difficult for users to commit irreversible actions. Expect users to make errors, try to anticipate where they will go wrong and design with those actions in mind.

Norman's Research

- Has contributed extensively to the field of human-computer interface design This psychologist has taken insights from the field of industrial product design and applied them to the design of user interfaces.
- According to Norman, design should:
  - use both knowledge in the world and knowledge in the head. Knowledge in the world is overt - we don't have to overload our short term memory by having to remember too many things (icons, buttons and menus provide us with knowledge in the world - we don't have to remember the command for printing, it's there in front of us).
  - On the other hand, while knowledge in the head may be harder to retrieve and involves learning, it is more efficient for tasks which are used over and over again (providing a command key sequence like Control P for Print is an example of this).
- "make it easy to determine what actions are possible at any moment (make use of constraints)“ For example:
  - well-designed things can only be put together certain ways (the trapezoidal SCSI cable is an example of good design - can only plug it in one way)
  - menus only display the actions which can be carried out at that time (other options are dimmed).
- "Make things visible, including the conceptual model of the system, the alternative actions and the results of actions“
  - You can provide an overview map of your site so that your user can design their own mental map of how things work.

- " Make it easy to evaluate the current state of the system“
  - You can do that by providing feedback in the form of messages or flashing buttons.
- "Follow natural mappings between intentions and the required actions, between actions and the resulting effect; and between the information that is visible and the interpretation of the system state" For example:
  - It should be obvious what the function of a button or menu is - use conventions already established for the web, don't try to design something which changes what people are familiar with.
  - The underlined phrase on a web page is a well-known clue that a link is present. From past experience, users understand that clicking on an underlined phrase should take them somewhere else.
- In other words, make sure that
  - (1) the user can figure out what to do, and
  - (2) the user can tell what is going on.

Conceptual design: from requirements to design

- Transform user requirements/needs into a conceptual model
- “a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended”
- Don’t move to a solution too quickly. Iterate, iterate, iterate
- Consider alternatives: prototyping helps

Three perspectives for a conceptual model

Which interaction mode?

- How the user invokes actions
- Activity-based: instructing, conversing, manipulating and navigating, exploring and browsing.
- Object-based: structured around real-world objects

Which interaction paradigm?

- desktop paradigm, with WIMP interface (windows, icons, menus and pointers),
- ubiquitous computing
- pervasive computing
- wearable computing
- mobile devices and so on.

Is there a suitable metaphor?

- Interface metaphors combine familiar knowledge with new knowledge in a way that will help the user understand the product.

- Three steps: understand functionality, identify potential problem areas, generate metaphors
- Evaluate metaphors:
- ▪ How much structure does it provide?
- ▪ How much is relevant to the problem?
- ▪ Is it easy to represent?
- ▪ Will the audience understand it?
- ▪ How extensible is it?

Expanding the conceptual model

- What functions will the product perform?

What will the product do and what will the human do (task allocation)?

- How are the functions related to each other?

sequential or parallel?
categorisations, e.g. all actions related to telephone memory storage

- What information needs to be available?

What data is required to perform the task?
How is this data to be transformed by the system?

Physical design: getting concrete

- Considers more concrete, detailed issues of designing the interface
- Iteration between physical and conceptual design
- Guidelines for physical design

Nielsen's heuristics
Cooper's About Face 2.0
Styles guides: commercial, corporate
decide 'look and feel' for you
widgets prescribed, e.g. icons, toolbar

Cooper's Excise Traps

- Don't force the user to go to another window to perform a function that affects this window
- Don't force the user to remember where he put things in the hierarchical file system
- Don't force the user to resize windows unnecessarily
- Don't force the user to move windows
- Don't force the user to reenter personal settings
- Don't force the user the fill fields to satisfy some arbitrary measure of completeness
- Don't force the user to ask permission to make changes.
- Don't ask the user to confirm his actions.

- Don't let the user's actions result in an error.
- Different kinds of widget (dialog boxes, toolbars, icons, menus etc)

menu design
icon design
screen design
information display

Icon design

- Good icon design is difficult
- Meaning of icons is cultural and context sensitive
- Some tips:

always draw on existing traditions or standards
concrete objects or things are easier to represent than actions

- From clip art, what do these mean to you?

Screen design

Two aspects:

- How to split across screens

moving around within and between screens
how much interaction per screen?

- Individual screen design

white space: balance between enough information/interaction and clarity
grouping items together: separation with boxes? lines? colors?

Screen design: splitting functions across screens

- Task analysis as a starting point
- Each screen contains a single simple step?
- Frustration if too many simple screens
- Keep information available: multiple screens open at once

Individual screen design

- Draw user attention to salient point, e.g. colour, motion, boxing
- Animation is very powerful but can be distracting
- Good organization helps: grouping, physical proximity
- Trade off between sparse population and overcrowding

Organization of Screen Elements

- **Balance**
- Symmetry

- Regularity
- Predictability
- Sequentiality
- Economy
- Unity
- Proportion
- Simplicity
- Groupings

Balance

- Equal weight of screen elements
    - Left to right, top to bottom

Symmetry

- Replicate elements left and right of the center line

Regularity

- Create standard and consistent spacing on horizontal and vertical alignment points

Predictability

- Put things in predictable locations on the screen
- User expects title & menu bar on top of screen
- Visual scene needs to be completely processed - objects not in expected places

Sequentiality

- Guide the eye through the task in an obvious way
    - The Eye is attracted to:
        - bright elements over less bright
        - Isolated elements over grouped
        - graphics before text
        - color before monochrome
        - saturated vs. less saturated colors
        - dark areas before light
        - big vs. small elements
        - unusual shapes over usual ones

Economy

- Use as few styles, fonts, colors, display techniques, dialog styles, etc., as possible

Unity

- Make items appear as a unified whole (for visual coherence)
  - Use similar shapes, sizes, or colors
  - Leave less space between screen elements than at the margin of the screen

Proportion

- Create groupings of data or text by using aesthetically pleasing proportions

Simplicity

- Minimize the number of aligned points
  - Use only a few columns to display screen elements
- Combine elements to minimize the number of screen objects
  - Within limits of clarity

Groupings

- Use visual arrangements to provide functional groupings of screen elements
  - Align elements in a group
  - Evenly space elements in a group
  - Provide separation between groups
- Use additional group elements sparingly
  - color & borders add complexity

Simple Grouping

- Similar elements aligned vertically
- Vertical distance between similar objects small

Menu design

- How long is the menu to be?
- In what order will the items appear?
- How is the menu to be structured, e.g. when to use sub-menus, dialog boxes?
- What categories will be used to group menu items?
- How will division into groups be denoted, e.g. different colors, dividing lines?
- How many menus will there be?
- What terminology to use? (results of requirements activities will indicate this)
- How will any physical constraints be accommodated, e.g. mobile phone?

Menus

- Structured access to system's functionality
    - The designer predefines the structure.
        - Comprehensible and natural to use
    - The computer displays options.
        - display options and accept inputs
    - The user makes choices.
        - Task-oriented

Menu Structure

- Match menu structure to user tasks
    - Not system data or internal structure
- Provide a "main menu"
    - Where in Windows?
- Allow for customization
    - Not all users work the same way

Depth vs. Breadth

- Breadth yields:
    - Fewer steps, shorter access times
    - Fewer paths to get lost in
    - Easier learning, since relationships are visible
- Disadvantages
    - More crowded
    - May confuse similar choices

- Depth yields:
    - Less crowding on menus
    - Fewer choices to scan
    - Easier to hide unavailable choices
    - Similar choices unlikely to be presented together
- Disadvantages
    - More steps, clicks, choices
    - Can't see relationships between choices
    - Can get lost
    - Higher error rates

D vs. B Design Guidelines

- Fewer levels is usually better
- Breadth within a level

    - 4 - 8 choices without grouping
    - 9 or more with grouping
    - Larger numbers in special cases
        - Expert user; simple choices; logical grouping

Menu Display

- Permanent display
    - Frequent use
    - Critical options
    - Screen availability
- On-demand display
    - Infrequent use
    - Expert users

Menu Ordering

- By natural order
- For small lists by:
    - Sequence of occurrence
    - Frequency of occurrence
    - Importance
- Alphabetic order for:
    - Long lists
    - Short lists with no obvious patter or frequency
- Separate destructive choices
- Maintain consistent ordering

Menu Choices

- Present text choices vertically
    - Left-justify text
- Provide choice descriptors for complex systems
    - Look ahead
        - Shows next lower menu when cursor passes over choice
    - Micro-help
        - Brief menu description in pop-up or status bar

Wording Menu Choices

- Clear, common meaning
    - Vocabulary of the user
- Single words or very short phrases

- Grammatical consistency
    - Key word first
    - Parallel construction

Arranging Menu Items

- Use Columns
    - Top-to-bottom reading
    - Left justify descriptions
- For horizontal menus
    - Left-to-right reading
- Intent Indicators
    - Use arrow or triangle to indicate cascading menus
    - Use elipsis (…) to indicate option resulting in a window
    - Use **NO** indicator for items resulting in an action

Line Separators

- Separate vertical groupings with solid lines
- Separate subgroupings with dashed lines
- Left-justify lines under the 1st letter of the description
- Right-justify under the last character of the longest description

Keyboard Accelerator

- Provide keyboard accelerator
    - Expert users
    - Motor skill problems
    - Cramped or grimy places
        - Factory floor
        - Portables
- Keyboard equivalents
    - "Access keys"
- May be a function key or combination of keys
    - Function key easier to learn than modifier + key
    - No more than 2 keys together in any case
    - Use "+" sign to indicate multiple keys required
- Don't use accelerators for pop-ups or cascaded menus

Choice Selection

- Provide defaults if sensible
    - Last or most frequent choice
- Highlight current choice
- Indicate unavailable choices

    - "Grayed out"
    - Removed from menu
        - General rule: do NOT remove choices
- Make choices large enough to select

- Keyboard selection
    - Up & Down Arrows move cursor up and down
    - Right & Left Arrows move cursor left & right for horizontal menus
- Selection/Execution
    - Provide separate selection and execution actions
    - Highlight selected choice or modify cursor shape
    - Allow alternative techniques for selection & execution if possible

Mark Toggles

- Indicate item **state**
- Good for showing an item is selected over a long period of time
- Use as a reminder that the item is selected or active
- Indicator goes to the left of the item

- Advantages
    - Shorter menus
    - Decreased clutter
    - Quicker access
    - Opposite action visible
- Disadvantages
    - Hidden opposite action
    - Command only

Screen-based Controls

- Widgets
    - elements of screen displays
    - interaction toolkits
    - ready-made interaction objects
    - predefined behaviors
    - customizable properties

Functions of Widgets

- Selecting options and commands
- Entering and editing data values

- Displaying data

Kinds of Widgets

- Buttons
- Text entry/read-only
- Selection
- Combination entry/selection
- Specialized or custom
- Presentation

Design issues

- Labels and graphics
- Layout and organization
- Activation

Labels and Graphics

- Use labels and captions
  - Use standard names when appropriate
  - Use regular system font
  - Clearly tie the text to the control
- Maintain consistent heights and widths
  - Use common shapes (mostly rectangles)
- Pick icons that map to the actions
  - Supplement icons with text descriptions

Layout and Organization

- Provide adequate spacing
- Limit the number of controls on one screen
- Keep related controls together
  - Use visual enclosure of groups where appropriate

Aligning list boxes

- Align list boxes vertically rather than horizontally.
- Horizontally aligned list boxes are more difficult for the user to use, as the controls cannot be scanned easily

Activation

- Provide keyboard equivalents
  - Control activation

    - Movement among controls within a screen
- Provide feedback for actions
- Gray out unavailable choices

Buttons

- Initiates an action
    - to activate a command (an alternate to menu choice or command line entry).
    - to display another window or menu selection
- Always visible
    - provides convenient access to frequently-used commands
    - standard shapes and screen location for similar commands.
    - Logical organization

- Types
    - Command buttons -- text as labels
    - Bar buttons (menu buttons) -- graphics and/or text as labels
    - Radio buttons

Button Design Issues

- Labels
- Shapes and Graphics
- Location and layout
- Organization
- Activation

Buttons – labels

- Use standard button labels when available
- Provide meaningful action description
- Use regular system font
    - unless for some special purposes
- Center the label text
- Provide consistency across all screens

Buttons --shape and graphics

- Use rectangular shape whatever possible
- Maintain consistent button heights and widths
- Design graphics/icons that have natural mapping to the actions
- Enhanced graphics with text description

Buttons – Organization

- Maintain consistency in button locations across screens and windows
- provide adequate spacing between buttons and other screen controls
- Restrict the number of buttons on one screen
- Follow standards
- Keep related buttons together

Buttons – Activation

- Consider different actions for
  - mouse enter/exit .. mouse down/up/
- Consider keyboard equivalents for actions
- Provide feedback for actions
  - highlight the button when the button is selected
- Gray out unavailable choices.

Text Entry/Read-Only Controls

- Text boxes
  - Editable/read-only (fields vs. labels)
  - single line/multiple lines
  - fixed size/resizable
  - fixed length/variable lengths
  - visual box/non-visual box
  - scrollable /non-scrollable
- Properties
  - background/foreground colors
  - sizes/fonts/styles of text
  - alignments

Text Box Design

- Provide descriptive caption
- Logical arrangement of multiple fields
- Consider the cursor movement from one field to another.
- Provide large enough boxes for fixed-length data
- Select reasonable fonts/sizes/colors
- Design highlight to attract attention

Selection Controls

- Present all options or choices on the screen
  - Radio Buttons

  - Check Boxes
  - Palettes
  - List Boxes
  - Combo Boxes
    - Drop-Down/Pop-Up
- Single Selection/Multiple Selection?

Selection Design
- Choice Description
  - Meaningful and clear description for the value or effects of the choice
  - Use single line of text whenever possible
- Organization
  - Meaningful order of choices
  - Consider adding a enclosure box
- Activation
  - Provide visual feedback
  - Provide default values

List Box or Combo Box?
- List box
  - unlimited number of choices
  - possible multiple choices
  - consumes screen space
  - can be set to different size
  - easy to see the choices
- Combo box
  - unlimited number of choices
  - highlight the selection
  - conserves screen space
  - Extra step to display all the choices

Other Controls
- Scroll Bars
- Sliders
- Toggle Switches
- Tab pages
  - Contain tabbed divider pages

Presentation Controls

- Provide additional information to screen elements
  - Tooltips
    - a small popup window attached to an object
    - shows only when the mouse moves over the object
  - Static Text Fields -- labels
  - Group Boxes
    - Combined controls in one box
  - Progress Indicators