https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-hci/

Summary

The Rise of HCI

- 1980's
- Apple introduced the Macintosh in 1984
- much more accessible
- making communication simpler
- designing, implementing, and evaluating interactive interfaces

HCI begins

- video game
- word processors
- numerical units

Key components of HCI

- User
- Goal-oriented task
- Interface
- Context

Importance of HCI

1. HCI in daily lives
2. Industry
3. Accessible to disabled
4. An integral part of software success
5. Useful for untrained communities

Examples of HCI

1. IoT technology
2. Eye-tracking technology
3. Speech recognition technology
4. AR/VR technology
5. Cloud computing

Goals of HCI

Usability

- How to use it
- Safe
- Efficient
- Effective
- Utility
- Enjoyable

User Experience

- Disable Traits
- Undesirable Traits

What Is HCI (Human-Computer Interaction)?

Meaning, Importance, Examples, and Goals

Human-computer interaction (HCI) targets the design and implementation of interactive technology.

Human-computer interaction (HCI) is defined as the field of study that focuses on optimizing how users and computers interact by designing interactive computer interfaces that satisfy users' needs. This article explains the fundamentals of HCI, its goals, importance, and examples.

What Is HCI?

Human-computer interaction (HCI) is the field of study that focuses on optimizing how users and computers interact by designing interactive computer interfaces that satisfy users' needs. It is a multidisciplinary subject covering computer science, behavioral sciences, cognitive science, ergonomics, psychology, and design principles.

The emergence of HCI dates back to the 1980s, when personal computing was on the rise. It was when desktop computers started appearing in households and corporate offices. HCI's journey began with video games, word processors, and numerical units.

However, with the advent of the internet and the explosion of mobile and diversified technologies such as voice-based and Internet of Things (IoT), computing became omnipresent and omnipotent. Technological competence further led to the evolution of user interactions. Consequently, the need for developing a tool that would make such man-machine interactions more human-like grew significantly. This established HCI as a technology, bringing different fields such as cognitive engineering, linguistics, neuroscience, and others under its realm.

Today, HCI focuses on designing, implementing, and evaluating interactive interfaces that enhance user experience using computing devices. This includes user interface design, user-centered design, and user experience design.

Human Computer Interaction (HCI) – to satisfy users, mainly designs and interface. How easy your program is and how appealing it is to the user.

- *User Interface (UI)* – aesthetic, colors, buttons, and designs.
- *User Experience (UX)* – how the user feels about your program, user's satisfaction, and user's interaction to your program from first to last.

Importance of HCI

HCI is crucial in designing intuitive interfaces that people with different abilities and expertise usually access. Most importantly, human-computer interaction is helpful for communities lacking knowledge and formal training on interacting with specific computing systems.

With efficient HCI designs, users need not consider the intricacies and complexities of using the computing system. User-friendly interfaces ensure that user interactions are clear, precise, and natural.

Let's understand the importance of HCI in our day-to-day lives:

1. HCI in daily lives

Today, technology has penetrated our routine lives and has impacted our daily activities. To experience HCI technology, one need not own or use a smartphone or computer. When people use an ATM, food dispensing machine, or snack vending machine, they inevitably come in contact with HCI. This is because HCI plays a vital role in designing the interfaces of such systems that make them usable and efficient.

2. Industry

Industries that use computing technology for day-to-day activities tend to consider HCI a necessary business-driving force. Efficiently designed systems ensure that employees are comfortable using the systems for their everyday work. With HCI, systems are easy to handle, even for untrained staff.

HCI is critical for designing safety systems such as those used in air traffic control (ATC) or power plants. The aim of HCI, in such cases, is to make sure that the system is accessible to any non-expert individual who can handle safety-critical situations if the need arises.

3. Accessible to disabled

The primary objective of HCI is to design systems that make them accessible, usable, efficient, and safe for anyone and everyone. This implies that people with a wide range of capabilities, expertise, and knowledge can easily use HCI-designed systems. It also encompasses people with disabilities. HCI tends to rely on user-centered techniques and methods to make systems usable for people with disabilities.

4. An integral part of software success

HCI is an integral part of software development companies that develop software for end-users. Such companies use HCI techniques to develop software products to make them usable. Since the product is finally consumed by the end-user, following HCI methods is crucial as the product's sales depend on its usability.

5. Useful for untrained communities

Today, user manuals for general computer systems are a rarity. Very few advanced and complex computing systems provide user manuals. In general, users expect the systems to be user-friendly and enable them to access the system within a few minutes of interacting with it. Here, HCI is an effective tool that designers can use to design easy-to-use interfaces. HCI principles also ensure that the systems have obvious interfaces and do not require special training to be used. Hence, HCI makes computing systems suitable for an untrained community.

Key components of HCI

Fundamentally, HCI is made up of four key components:

1. The user

The user component refers to an individual or a group of individuals that participate in a common task. HCI studies users' needs, goals, and interaction patterns. It analyzes various parameters such as users' cognitive capabilities, emotions, and experiences to provide them with a seamless experience while interacting with computing systems.

2. The goal-oriented task

A user operates a computer system with an objective or goal in mind. The computer provides a digital representation of objects to accomplish this goal. For example, booking an airline for a destination could be a task for an aviation website. In such goal-oriented scenarios, one should consider the following aspects for a better user experience:

- The complexity of the task that the user intends to accomplish
- Knowledge and skills necessary to interact with the digital object
- Time required to carry out the task

3. The interface

The interface is a crucial HCI component that can enhance the overall user interaction experience. Various interface-related aspects must be considered, such as interaction type (touch, click, gesture, or voice), screen resolution, display size, or even color contrast. Users can adjust these depending on the user's needs and requirements.

For example, consider a user visiting a website on a smartphone. In such a case, the mobile version of the website should only display important information that allows the user to navigate through the site easily. Moreover, the text size should be appropriately adjusted so that the user is in a position to read it on the mobile device. Such design optimization boosts user experience as it makes them feel comfortable while accessing the site on a mobile phone.

4. The context

HCI is not only about providing better communication between users and computers but also about factoring in the context and environment in which the system is accessed. For example, while designing a smartphone app, designers need to evaluate how the app will visually appear in different lighting conditions (during day or night) or how it will perform when there is a poor network connection. Such aspects can have a significant impact on the end-user experience.

Thus, HCI is a result of continuous testing and refinement of interface designs that can affect the context of use for the users.

Examples of HCI

1. IoT technology
2. Eye-tracking technology
3. Speech recognition technology
4. AR/VR technology
5. Cloud computing

Goals of HCI

The principal objective of HCI is to develop functional systems that are usable, safe, and efficient for end-users. The developer community can achieve this goal by fulfilling the following criteria:

- Have sound knowledge of how users use computing systems
- Design methods, techniques, and tools that allow users to access systems based on their needs
- Adjust, test, refine, validate, and ensure that users achieve effective communication or interaction with the systems
- Always give priority to end-users and lay the robust foundation of HCI

To realize the above points, developers must focus on two relevant areas: usability and user experience. Let's look at each category in detail:

1. Usability

Usability is key to HCI as it ensures that users of all types can quickly learn and use computing systems. A practical and usable HCI system has the following characteristics:

- **How to use it**: This should be easy to learn and remember for new and infrequent users to learn and remember. For example, operating systems with a user-friendly interface are easier to understand than DOS operating systems that use a command-line interface.
- **Safe**: A safe system safeguards users from undesirable and dangerous situations. This may refer to users making mistakes and errors while using the system that may lead to severe consequences. Users can resolve this through HCI practices. For example, systems can be designed to prevent users from activating specific keys or buttons accidentally. Another example could be to provide recovery plans once the user commits mistakes. This may give users the confidence to explore the system or interface further.
- **Efficient**: An efficient system defines how good the system is and whether it accomplishes the tasks that it is supposed to. Moreover, it illustrates how the system provides the necessary support to users to complete their tasks.
- **Effective**: A practical system provides high-quality performance. It describes whether the system can achieve the desired goals.
- **Utility**: Utility refers to the various functionalities and tools provided by the system to complete the intended task. For example, a sound utility system offers an integrated development environment (IDE) that provides intermittent help to programmers or users through suggestions.
- **Enjoyable**: Users find the computing system enjoyable to use when the interface is less complex to interpret and understand.

2. User experience

User experience is a subjective trait that focuses on how users feel about the computing system when interacting with it. Here, user feelings are studied individually so that developers and support teams can target particular users to evoke positive feelings while using the system.

HCI systems classify user interaction patterns into the following categories and further refine the system based on the detected pattern:

- **Desirable traits** – satisfying, enjoyable, motivating, or surprising
- **Undesirable traits** – Frustrating, unpleasant, or annoying

Interactive System Design

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiS-_GHws-EAxXDxTgGHYdGCKgQFnoECBgQAQ&url=https%3A%2F%2Fhci-lecture.de%2FHCI%2Ftopics%2Fintro%2F02-TermiBasics%2FtermiBasics20200413.pptx&usg=AOvVaw1trjs2t2uIsMBBxMwRPEn8&opi=89978449

Interactive Systems

Definition and Examples

- An **Interactive System** is a computational system that allows users to interact in real-time. Interactions receive instant feedback visible to the user.
- **Interactive computing** is used in a similar way, with a focus that is less on the systems aspect.

Examples

- **Graphical user interfaces**, such as Windows 10 or MacOS
- **Mobile devices**, such as an Android phone
- gaming consoles, such as Xbox with Kinect or Nintendo Switch
- **Ticket vending machines**, such as the DB-ticket machine
- **Command line interfaces**, such as an SSH console

https://www.tutorialspoint.com/human_computer_interface/interactive_system_design.htm

Concept of Usability Engineering

Usability Engineering is a method in the progress of software and systems, which includes user contribution from the inception of the process and assures the effectiveness of the product through the use of a usability requirement and metrics.

It thus refers to the Usability Function features of the entire process of abstracting, implementing & testing hardware and software products. Requirements gathering stage to installation, marketing and testing of products, all fall in this process.

Goals of Usability Engineering

- **Effective to use** – Functional
- **Efficient to use** – Efficient
- **Error free in use** – Safe
- **Easy to use** – Friendly
- **Enjoyable in use** – Delightful Experience

Usability

Usability has three components – effectiveness, efficiency and satisfaction, using which, users accomplish their goals in particular environments. Let us look in brief about these components.

- **Effectiveness** – The completeness with which users achieve their goals.
- **Efficiency** – The competence used in using the resources to effectively achieve the goals.
- **Satisfaction** – The ease of the work system to its users.

Usability Study

The methodical study on the interaction between people, products, and environment based on experimental assessment. _Example: Psychology, Behavioral Science, etc._

- **Usability Testing**
  The scientific evaluation of the stated usability parameters as per the user's requirements, competences, prospects, safety and satisfaction is known as usability testing.
- **Acceptance Testing**
  Acceptance testing also known as User Acceptance Testing (UAT), is a testing procedure that is performed by the users as a final checkpoint before signing off from a vendor. Let us take an example of the handheld barcode scanner.

```Let us assume that a supermarket has bought barcode scanners from a vendor. The supermarket gathers a team of counter employees and make them test the device in a mock store setting. By this procedure, the users would determine if the product is acceptable for their needs. It is required that the user acceptance testing "pass" before they receive the final product from the vendor.```

https://www.usability.gov/how-to-and-tools/methods/usability-testing.html

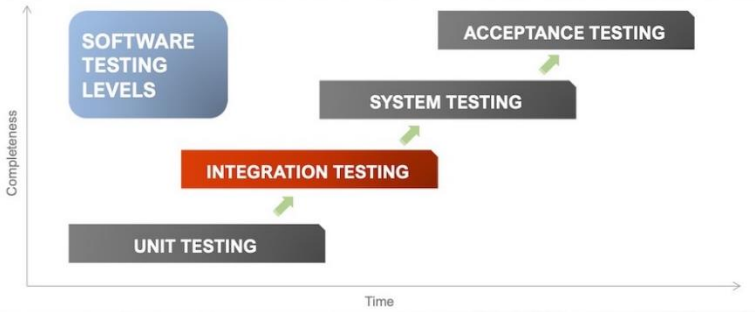
To run an effective usability test, you need to develop a solid test plan, recruit participants, and then analyze and report your findings.

Benefits of Usability Testing

Usability testing lets the design and development teams identify problems before they are coded. The earlier issues are identified and fixed, the less expensive the fixes will be in terms of both staff time and possible impact to the schedule. During a usability test, you will:

- Learn if participants are able to complete specified tasks successfully and
- Identify how long it takes to complete specified tasks
- Find out how satisfied participants are with your Web site or other product
- Identify changes required to improve user performance and satisfaction
- And analyze the performance to see if it meets your usability objectives

https://www.inrhythm.com/guide-to-integration-testing/



system now is perform

Each Testing Level Details

Unit testing: *> Proposal, Languge, Flowchart, Algorithims, IDE etc to make the program*
A Unit is a smallest testable portion of system or application which can be compiled, liked, loaded, and executed. This kind of testing helps to test each module separately.

The aim is to test each part of the software by separating it. It checks that component are fulfilling functionalities or not. This kind of testing is performed by developers.

Integration testing:
Integration means combining. For Example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.

Integrating testing checks the data flow from one module to other modules. This kind of testing is performed by testers.

System Testing:
System testing is performed on a complete, integrated system. It allows checking system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing.

System testing most often the final test to verify that the system meets the specification. It evaluates both functional and non-functional need for the testing.

Acceptance testing*: > Checking on prof > gets grade*
Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.
during development on systems

https://www.tutorialspoint.com/human_computer_interface/interactive_system_design.htm

Software Tools

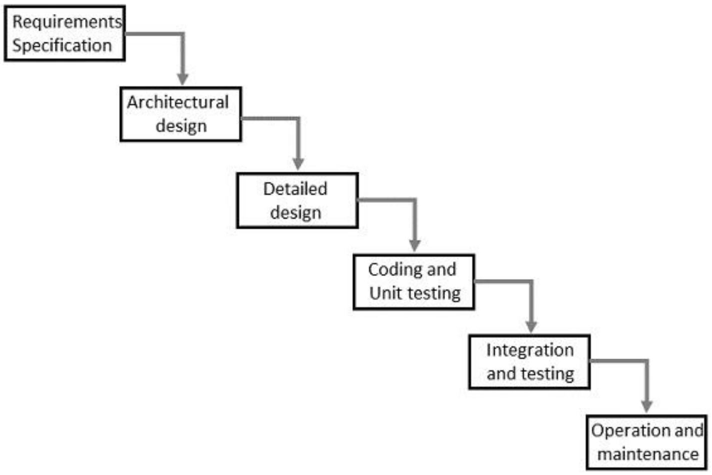
A software tool is a programmatic software used to create, maintain, or otherwise support other programs and applications. Some of the commonly used software tools in HCI are as follows –

- **Specification Methods** – The methods used to specify the GUI. Even though these are lengthy and ambiguous methods, they are easy to understand.
- **Grammars** – Written Instructions or Expressions that a program would understand. They provide confirmations for completeness and correctness.
- **Transition Diagram** – Set of nodes and links that can be displayed in text, link frequency, state diagram, etc. They are difficult in evaluating usability, visibility, modularity and synchronization.
- **Statecharts** – Chart methods developed for simultaneous user activities and external actions. They provide link-specification with interface building tools.
- **Interface Building Tools** – Design methods that help in designing command languages, data-entry structures, and widgets.
- **Interface Mockup Tools** – Tools to develop a quick sketch of GUI. E.g., Microsoft Visio, Visual Studio .Net, etc.
- **Software Engineering Tools** – Extensive programming tools to provide user interface management system.
- **Evaluation Tools** – Tools to evaluate the correctness and completeness of programs.

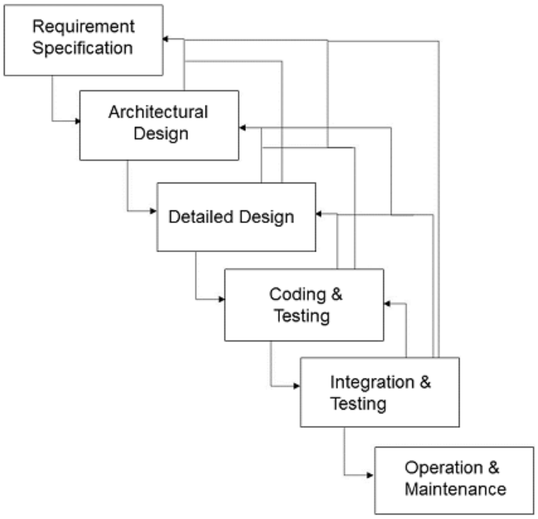
Software engineering is the study of designing, development and preservation of software. It comes in contact with HCI to make the man and machine interaction more vibrant and interactive.

Let us see the following model in software engineering for interactive designing.

The Waterfall Method



Interactive System Design



The uni-directional movement of the waterfall model of Software Engineering shows that every phase depends on the preceding phase and not vice-versa. However, this model is not suitable for the interactive system design.

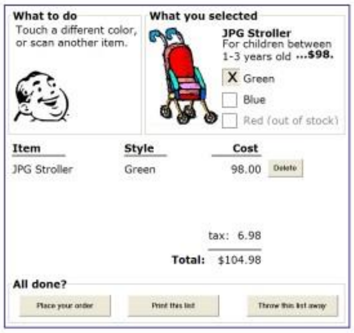
The interactive system design shows that every phase depends on each other to serve the purpose of designing and product creation. It is a continuous process as there is so much to know and users keep changing all the time. An interactive system designer should recognize this diversity.

Prototyping

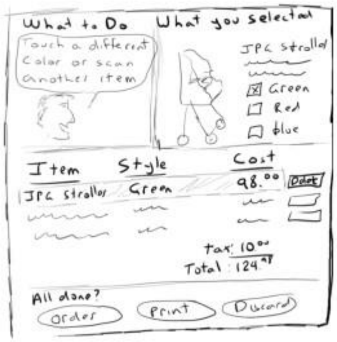
Prototyping is another type of software engineering models that can have a complete range of functionalities of the projected system.

In HCI, prototyping is a trial and partial design that helps users in testing design ideas without executing a complete system.

Example of a prototype can be Sketches. Sketches of interactive design can later be produced into graphical interface. See the following diagram.



Interface of a proposed system



A sketch of the interface

3 types of Prototyping

- **Low Fidelity Prototype**
- **Medium Fidelity Prototype**
- **Hi Fidelity Prototype**

The above diagram can be considered as a **Low Fidelity Prototype** as it uses manual procedures like sketching in a paper.

A **Medium Fidelity Prototype** involves some but not all procedures of the system. E.g., first screen of a GUI.

Finally, a **Hi Fidelity Prototype** simulates all the functionalities of the system in a design. This prototype requires, time, money and work force.

ORAL MIDTERM EXAMINATION SECOND WEEK March 14

Next Week March 14, 2024 [10-1pm]

https://www.tutorialspoint.com/human_computer_interface/interactive_system_design.htm:~:text=User%20Centered%20Design%20(UCD),user%20centered%20design%20or%20UCD.

User Centered Design (UCD)

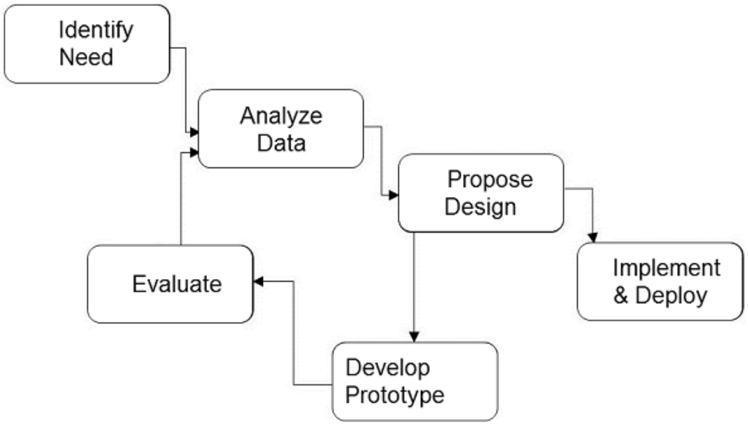
The process of collecting feedback from users to improve the design is known as user centered design or UCD.

UCD Drawbacks

- Passive user involvement.
- User's perception about the new interface may be inappropriate.
- Designers may ask incorrect questions to users.

The stages in the following diagram are repeated until the solution is reached.

Diagram



https://www.pling.org.uk/cs/doi.html

Before we can continue, we need to consider what we mean by the term users. We can consider three levels of users:

"3 Types of Users"

- **Primary users** - regular/frequent users of the system
- **Secondary users** - occasional users of the system, those who use the system occasionally, or only through an intermediary
- **Tertiary users** - those affected by the system introduction, or who will influence purchase of the system

Key Principles of Interactive Design

- **Learnability** - how easy the system is to use
- **Memorability** - how easy is the system to remember how to use
- **Consistency** - to what extent are similar tasks conducted in similar ways within the system (this will contribute to learnability)
- **Visibility** - to what extent does the system make it clear what you can/should do next
- **Constraints** - design the system so that users won't make mistakes and won't be led into frustrating/irritating dead ends
- **Feedback** - provide good feedback on the status of the system, what has happened, what is happening, what will happen

Interaction Styles

We need to get computers and complex systems to do things - somehow we have to "tell" them what to do. In turn, they have to give us information - the status of the system, what to do next, what's wrong, how to fix it.

One metaphor for this is to consider interacting with these systems as a dialogue - you tell/ask the system to do something, it tells you things back; not too dissimilar to having a conversation with a human being. Another metaphor is to consider these systems as objects that you do things with and interact with (for example, putting waste in a waste paper bin), or as navigating through a space and going places (the web).

Command Line Dialogue

This style is not widely used, but is useful to understand it to consider more recent and long-term developments. This is the first interaction style that appeared on PCs, taking over from mainframe systems.

The conversational metaphor applies here, where you're talking to the computer through your keyboard and it reacts. However, the language you speak in must be correct to the last dot and in the correct order, much like speaking a foreign language. The system doesn't give you any clues on what to do, so you must remember (or use a crib sheet), the syntax of the language. These commands can get quite long and complex, especially when passing lots of options and (in UNIX), piping one command to the other.

You also get limited feedback about what is happening, a command such as rf may return you directly to the command line, after deleting 0 or 100 files. The later versions of DOS took the feedback step too far, however, asking for a confirmation of every file by default. This is programmed by the interaction designer, and they have to remember to do this and get the level of interaction right.

If you do get the command correct, however, this can be a very efficient way of operating a complex system. A short, but powerful, language allows you to acheive a great deal and people are willing to invest the time to learn this language to get the most efficient use of the system.

However, although computers are good at dealing with cryptic strings, complex syntax and an exact reproduction of the syntax every time, humans aren't. This interaction system stemmed from the fact that processing power was expensive, so humans had to adapt to the way computers needed to interact, not vice versa. This is no longer the case.

Menu Interaction Style

Although the command line style was good for experts, it wasn't for novice of infrequent users, so an interaction style was developed which is almost the complete opposite of command line dialogue in terms of strengths and weaknesses - menus.

Menus are simple, as not much needs to be remembered as the options are there on screen and the physical interface corresponds directly to the options available. Feedback is immediate - selecting an option will either take you to another screen of menus or to perform the selected task.

Feedback is natural and built in -you can see whether you are going the right way, because either something relevant occurs - much like handling objects gives you instant built in feedback.

Selections from objects (such as radio buttons) can be thought of as a menu, even though the selection method is different.

Menu systems should be usable without any prior knowledge or memory of previous use, as it leads you through the interaction. This is excellent for novice and infrequent users (hence their use in public terminals of many varieties). However, a complex menu structure can complicate matters where particular features will be found (e.g., IVR systems)

Expert and frequent users get irritated by having to move through the same menu structure every time they do a particular task (hence shortcut keys in applications such as Word), and menu systems also present a lack of flexibility - you can only do what menu options are present for you, so menu driven systems only work where there is a fairly limited number of options at any point. Items also need to be logically grouped.

Form Fill-In Interaction Style

A form is provided for the user to fill in. Perhaps not the most revolutionary or interesting of interaction styles, but it is based on another real world metaphor - filling out a paper form. This also illustrates another problem with metaphors, not everything transfers well from one medium to another.

Forms present lots of little problems:

- **How many characters can be typed in the field** - this is often not clear, and there is no inherent indication of this (needs to be explicitely built in by the interaction designer) - this violates the feedback principle.
- **What format is supposed to be used** - particularly for things like dates
- **Lack of information** - what if all information known by the form isn't available immediately. The design may not let you process beyond the dialogue box

However, this system does have strengths. If the system is well-designed, any novice can use them and it has a strong analogy to paper forms, and the user can be led easily through the process. However, they are easy to design badly, and hence confuse and irritate the user.

Direct Manipulation

The heart of modern **graphical user interfaces** (GUIs, sometimes referred to as **WIMP - Windows, Icons, Mouse/Menus, Pointers) or "point and click"** interfaces is direct manipulation (note, GUI and WIMP themselves are not interaction styles).

The idea of direct manipulation is that you have objects (often called widgets) in the interface (icons, buttons, windows, scrollbars, etc) and you manipulate those like real objects.

One of the main advantages of direct manipulation is that you get (almost) immediate natural feedback on the consequences of your action by the change in the object and its context in the interface. Also, with the widgets being onscreen, you are provided with clues as to what you can do (doesn't help much with items buried in menu hierarchies). The nature of widgets should tell you something about what can be done with them (affordance), and with widgets, many more objects can be presented simultaneously (more than menus).

However, interface actions don't always have real world metaphors you can build on (e.g., executing a file) and information rich interactions don't always tend to work well with the object metaphor and direct manipulation becomes tedious for repetitive tasks, such as dragging, copying, etc...

Natural Language Interfaces

The ultimate goal of the conversational metaphor is for us to have a dialogue with the computer in our own natural language. This might be written, but spoken seems easier. Voice in/voice out systems have been developed, where the computer recognises what you say (using speech recognition technology) and produces a response, spoken via text-to-speech (TTS).

This kind of system should be "walk up and use", just speak your own language and this should suit both novices and expert users. This isn't yet technologically feasible, as the AI is largely fake and can only recognise key words. If speech recognition fails, it is easy to get into nasty loops, where it is not clear as to how you get out again (this is the feedback principle). There is also the problem of different accents and recognising older voices, and privacy and security issues are introduced.

ORAL MIDTERM EXAMINATION SECOND WEEK March 14

Next Week March 14, 2024 [10-1pm]

Prepare for oral like a bitch