LECTURE NOTES

ON

HUMAN COMPUTER INTERACTION (HCI)

(CSC 413)

400LEVEL COMPUTER SCIENCE

RHEMA UNIVERSITY (TAKE OFF SITE) ABA

Onuoha Oju
ojupro@gmail.com

HUMAN COMPUTER INTERACTION (HCI)

Introduction

Human Computer Interface (HCI) was previously known as the man-machine studies or man-machine interaction. It deals with the design, execution and assessment of computer systems and related phenomenon that are for human use. It is also known as CHI (Computer Human Interface) or MMI (Man Machine Interaction). It is concerned with design, evaluation, and implementation. It is used to provide a user-friendly environment. Human uses digital devices to perform various activities. HCI is to design systems in such a way that make them efficient, stable, usable and attainable. Lack of communication can result in poor designed user interfaces. It provides ways to reduce design time through various task models. HCI can be used in all disciplines wherever there is a possibility of computer installation.

Definitions

"Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."

Human-computer interaction is the study, planning, and design of how people and computers work together so that a person's needs are satisfied in the most effective way.

HCI (human-computer interaction) is the study of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings. As its name implies, HCI consists of three parts: the user, the computer itself, and the ways they work together (interaction).

User: By "user", we may mean an individual user, a group of users working together. An appreciation of the way people's sensory systems (sight, hearing, touch) relay information is vital. Also, different users form different conceptions or mental models about their interactions and have different ways of learning and keeping knowledge and. In addition, cultural and national differences play a part.

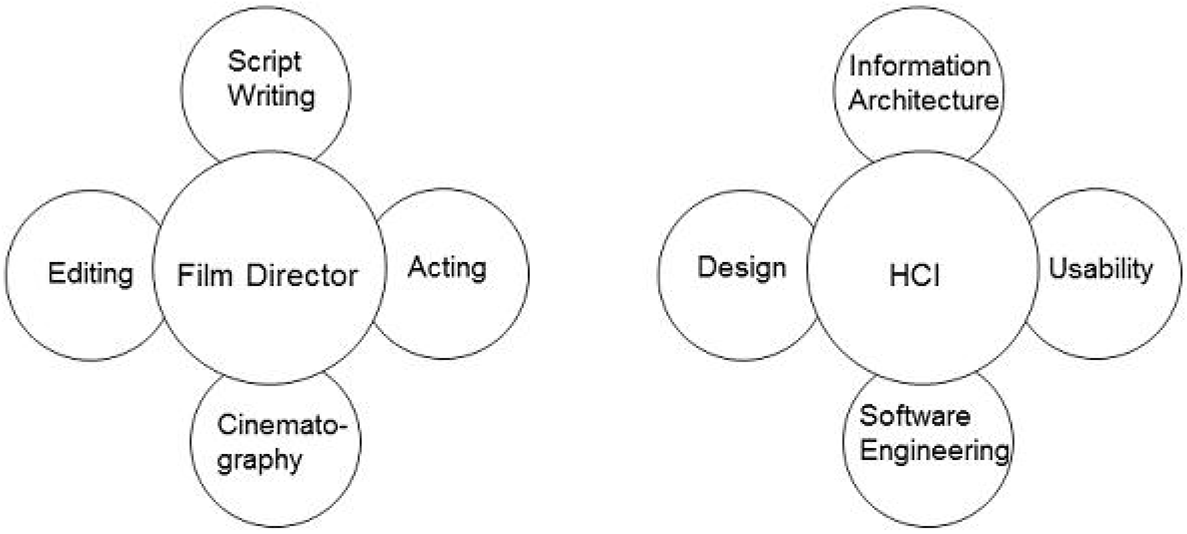
Computer: When we talk about the computer, we're referring to any technology ranging from desktop computers, to large scale computer systems. For example, if we were discussing the design of a Website, then the Website itself would be referred to as "the computer". Devices such as mobile phones or VCRs can also be considered to be “computers”.

Interaction: There are obvious differences between humans and machines. In spite of these, HCI attempts to ensure that they both get on with each other and interact successfully. In order to achieve a usable system, you need to apply what you know about humans and computers, and consult with likely users throughout the design process. In real systems, the schedule and the budget are important, and it is vital to find a balance between what would be ideal for the users and what is feasible in reality.

- **HCI Analogy**

Let us take a known analogy that can be understood by everyone. A film director is a person who with his/her experience can work on script writing, acting, editing, and cinematography. He/She can be considered as the only person accountable for all the creative phases of the film.

Similarly, HCI can be considered as the film director whose job is part creative and part technical. An HCI designer have substantial understanding of all areas of designing. The following diagram depicts the analogy −



Historical Evolution

From the initial computers performing batch processing to the user-centric design, there were several milestones which are mentioned below −

•Early computer (e.g. ENIAC, 1946) − Improvement in the H/W technology brought massive increase in computing power. People started thinking on innovative ideas.

•Visual Display Unit (1950s) − SAGE (semi-automatic ground environment), an air defense system of the USA used the earliest version of VDU.

•Development of the Sketchpad (1962) − Ivan Sutherland developed Sketchpad and proved that computer can be used for more than data processing.

•Douglas Engelbart introduced the idea of programming toolkits (1963) − Smaller systems created larger systems and components.

•Introduction of Word Processor, Mouse (1968) − Design of NLS (oNLine System).

•Introduction of personal computer Dynabook (1970s) − Developed smalltalk at Xerox PARC.

•Windows and WIMP interfaces − Simultaneous jobs at one desktop, switching between work and screens, sequential interaction.

•1980‘s are the booming phase for HCI. Some of the market leaders like Apple and Microsoft plays a crucial role for the modern development of HCI. GUI (Graphic User Interface) application was created that was easy to use, understand and visualize.

•XEROX STAR was released in 1981. It had mouse driven graphical user interface and built-in ethernet network and protocol. It also had laser printer. This was considered far ahead of its time. Two years later in 1983 Apple Lisa was released, it offered document-centered graphical interface based on the metaphor of desktop.
•In 1984 first Macintosh was release and it was revolutionary. It had good graphic user interface and a variety of fonts that makes your document more appealing to readers.
•Direct Manipulation introduced by Ben Shneiderman (1982) − First used in Apple Mac PC (1984) that reduced the chances for syntactic errors.
•In 1990‘s internet starts it's journey. Communication between people become very easy through social networking like Email. The World Wide Web(WWW) was created by Tim Berners-Lee. It is way for people to share information.
•In 2000 mobile, laptop, tablet was a buzz word in this period. These gadget provides more flexibility to user. User can connect with anyone at any place. Smart phones comes into picture. User don't need any mouse or pointing device to select anything. They can use their fingers to interact with device. It provides more features like built-in music player, camera, weather forecast, Internet, GPS, games, video conferencing and many more.
•In 2006 NINTENDO released Wii. It was famous for it's rear remote controller a handheld pointing device that detects movements in 3D. It enables users to simulate real world sports and activities through different games. This paved the way for gaming consoles like XBOX
•Windows 10 is a series of operating systems developed by Microsoft released in 2015. It made user experience more consistent between different classes of device. The rising popularity and availability of laptops and computer systems, Microsoft made windows 10 adaptable into different systems.
•VR oculus rift was a revolution in virtual reality. It was launched in 2016. The rift is primarily a gaming device. However it is also capable of viewing conventional movies and videos from inside the virtual cinema environment. It is increasingly used in universities and schools as an educational tool.
•Ubiquitous Computing − Currently the most active research area in HCI. Sensor based/context aware computing also known as pervasive computing.

The Goals of HCI

The goals of HCI are to produce usable and safe systems, as well as functional systems. In order to produce computer systems with good usability, developers must attempt to:

- understand the factors that determine how people use technology
- develop tools and techniques to enable building suitable systems
- achieve efficient, effective, and safe interaction
- put people first

Underlying the whole theme of HCI is the belief that people using a computer system should come first. Their needs, capabilities and preferences for conducting various tasks should direct developers in the way that they design systems. People should not have to change the way that they use a system in order to fit in with it. Instead, the system should be designed to match their requirements.

- Usability

Usability is one of the key concepts in HCI. It is concerned with making systems easy to learn and use. A usable system is:

- easy to learn
- easy to remember how to use
- effective to use
- efficient to use
- safe to use
- enjoyable to use

Why is usability important?

Many everyday systems and products seem to be designed with little regard to usability. This leads to frustration, wasted time and errors. This list contains examples of interactive products:
mobile phone, computer, personal organizer, remote control, soft drink machine, coffee machine, ATM, ticket machine, library information system, the web, photocopier, watch, printer, stereo, calculator, videogame etc.

Factors in HCI

There are a large number of factors which should be considered in the analysis and design of a system using HCI principles. Many of these factors interact with each other, making the analysis even more complex. The main factors are listed below:

- Organisation Factors

Training, job design, politics, roles, work organisation

- Environmental Factors

Noise, heating, lighting, ventilation, Health and Safety Factors

- The User

Cognitive processes and capabilities
Motivation, enjoyment, satisfaction, personality, experience

- Comfort Factors

Seating, equipment, layout.

- User Interface

Input devices, output devices, dialogue structures, use of colour, icons, commands, navigation, graphics, natural language, user support, multimedia,

- Task Factors

Easy, complex, novel, task allocation, monitoring, skills

- Constraints

Cost, timescales, budgets, staff, equipment, buildings

- System Functionality

Hardware, software, application

- Productivity Factors

Increase output, increase quality, decrease costs, decrease errors, increase innovation

Disciplines contributing to HCI

The field of HCI covers a wide range of topics, and its development has relied on contributions from many disciplines. Some of the main disciplines which have contributed to HCI are:

- Computer Science

o technology
o software design, development & maintenance
o User Interface Management Systems (UIMS) & User Interface Development Environments (UIDE)
o prototyping tools
o graphics

- Cognitive Psychology

o information processing
o capabilities
o limitations
o cooperative working
o performance prediction

- Social Psychology

o social & organizational structures

- Ergonomics/Human Factors

o hardware design
o display readability

- Linguistics

o natural language interfaces

- Artificial Intelligence

o intelligent software

- Philosophy, Sociology & Anthropology

o Computer supported cooperative work (CSCW)

- Engineering & Design

o graphic design
o engineering principles

Use Cases of HCI

•Smart home: Smart homes refers to home amenities that have been fitted with communication technology enabling some degree of automation or remote control. It includes control of air conditioning, heating and lighting through voice activated commands or mobile app. Home security systems are also fitted with communication technology to alert the residents in case of burglary.

•Biometric Sensors: Biometric sensors are the use of human biometrics in various technological applications. It can be used in access controls for example granting access to a computer network or security system.

•Autonomous vehicle: An autonomous vehicle is one that can drive itself. Tesla is a company which pioneered the engineering of autonomous driving vehicles. It has advanced autopilot technology which allows real time navigation updates.
•Virtual assistants: Another innovation in this era is the intelligent virtual assistant or intelligent personal assistant. It is a software agent that can perform task or services or an individual based on commands or questions. These virtual assistants can interpret human speech and respond via voices.
•Smart phones for Visual Disabilities: There are some features present in smart phones that make the life of people with disabilities easy. Voiceover is a screen reader which basically means that your phone will talk out loud and tell you what's on the screen. User can control it with certain touch gestures. There are some other features also like Magnification.

Application of HCI in different domains
It includes the design and development of application. These applications include desktop application, websites and mobile apps. These applications are used in different domains it includes healthcare, banking, education, networking and many more.

- **Health care**

Patients have so many options now a days. They can buy medicines online and book appointments with doctor just with the help of mobile application. Augmented Reality (AR) and Virtual Reality (VR) are now transforming surgical process, previously it was very risky. Now doctor can use 3D animations to visualize the process. It can be used to train new surgeons.

- **Education**

Now students can understand any concept more easily. There are so many resources available on internet now a days. Class room teaching are now very interesting with the help of smart classes. AR/VR can really help students to visualize any concept very easily. Students have option to study online. During COVID-19 students couldn't able to go outside their home. In this situation they have option to study online.

- **Banking**

Now common people don't need to wait in long queues of bank. They can get banking solution right at their home using Net banking or Mobile banking. These applications also provide user a secure environment to avoid cybercrimes.

- **Networking**

Networking is very easy now a days. It includes social media networking and business networking. Now it is very easy for us to connect and share thoughts with anyone. It streamlines the process of finding jobs.

The Basic Concepts of Human Computer Interaction

Guidelines in HCI

- Shneiderman's Eight Golden Rules

Ben Shneiderman, an American computer scientist consolidated some implicit facts about designing and came up with the following eight general guidelines −
•Strive for Consistency.
•Cater to Universal Usability.
•Offer Informative feedback.
•Design Dialogs to yield closure.
•Prevent Errors.
•Permit easy reversal of actions.
•Support internal locus of control.
•Reduce short term memory load.

These guidelines are beneficial for normal designers as well as interface designers. Using these eight guidelines, it is possible to differentiate a good interface design from a bad one. These are beneficial in experimental assessment of identifying better GUIs.

- **Norman's Seven Principles**

To assess the interaction between human and computers, Donald Norman in 1988 proposed seven principles. He proposed the seven stages that can be used to transform difficult tasks. Following are the seven principles of Norman −
•Use both knowledge in world & knowledge in the head.
•Simplify task structures.
•Make things visible.
•Get the mapping right (User mental model = Conceptual model = Designed model).
•Convert constrains into advantages (Physical constraints, Cultural constraints, Technological constraints).
•Design for Error.
•When all else fails − Standardize.

- **Heuristic Evaluation**

Heuristics evaluation is a methodical procedure to check user interface for usability problems. Once a usability problem is detected in design, they are attended as an integral part of constant design processes. Heuristic evaluation method includes some usability principles such as Nielsen's ten Usability principles.
Nielsen's Ten Heuristic Principles
•Visibility of system status.
•Match between system and real world.
•User control and freedom.
•Consistency and standards.
•Error prevention.
•Recognition rather than Recall.
•Flexibility and efficiency of use.
•Aesthetic and minimalist design.
•Help, diagnosis and recovery from errors.

•Documentation and Help

The above mentioned ten principles of Nielsen serve as a checklist in evaluating and explaining problems for the heuristic evaluator while auditing an interface or a product.

Interface Design Guidelines
Some more important HCI design guidelines are presented in this section. General interaction, information display, and data entry are three categories of HCI design guidelines that are explained below.

- General Interaction

Guidelines for general interaction are comprehensive advices that focus on general instructions such as −
•Be consistent.
•Offer significant feedback.
•Ask for authentication of any non-trivial critical action.
•Authorize easy reversal of most actions.
•Lessen the amount of information that must be remembered in between actions.
•Seek competence in dialogue, motion and thought.
•Excuse mistakes.
•Classify activities by function and establish screen geography accordingly.
•Deliver help services that are context sensitive.
•Use simple action verbs or short verb phrases to name commands.

- Information Display

Information provided by the HCI should not be incomplete or unclear or else the application will not meet the requirements of the user. To provide better display, the following guidelines are prepared −
•Exhibit only that information that is applicable to the present context.
•Don't burden the user with data, use a presentation layout that allows rapid integration of information.
•Use standard labels, standard abbreviations and probable colors.
•Permit the user to maintain visual context.
•Generate meaningful error messages.
•Use upper and lower case, indentation and text grouping to aid in understanding.
•Use windows (if available) to classify different types of information.
•Use analog displays to characterize information that is more easily integrated with this form of representation.
•Consider the available geography of the display screen and use it efficiently.

- Data Entry

The following guidelines focus on data entry that is another important aspect of HCI −
•Reduce the number of input actions required of the user.
•Uphold steadiness between information display and data input.

•Let the user customize the input.
•Interaction should be flexible but also tuned to the user's favored mode of input.
•Disable commands that are unsuitable in the context of current actions.
•Allow the user to control the interactive flow.
•Offer help to assist with all input actions.
•Remove "mickey mouse" input.

Interactive System Design

The design and usability of systems leave an effect on the quality of people’s relationship to technology. Web applications, games, embedded devices, etc., are all a part of this system, which has become an integral part of our lives. Let us now discuss on some major components of this system.

- **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.

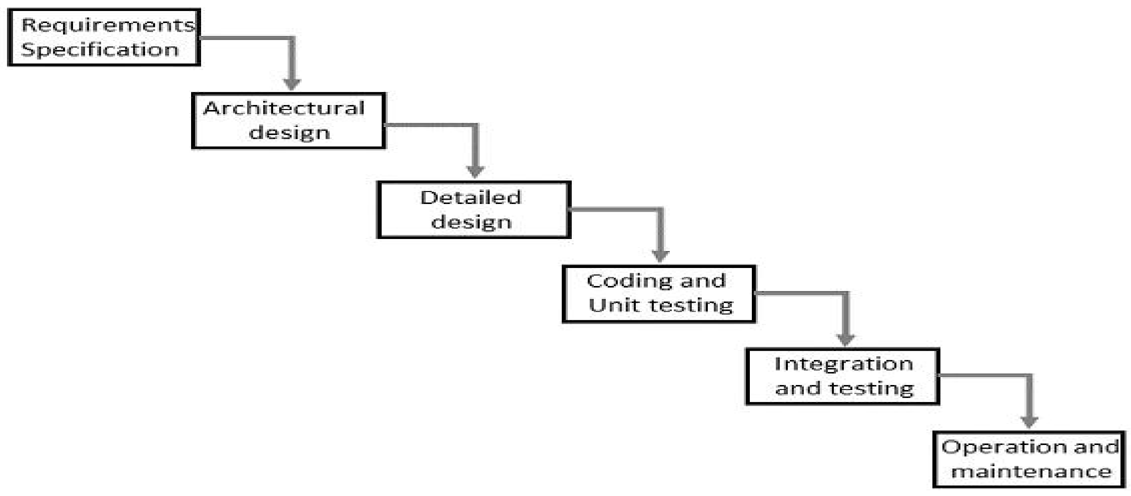
- **Software Tools**

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.

HCI and Software Engineering

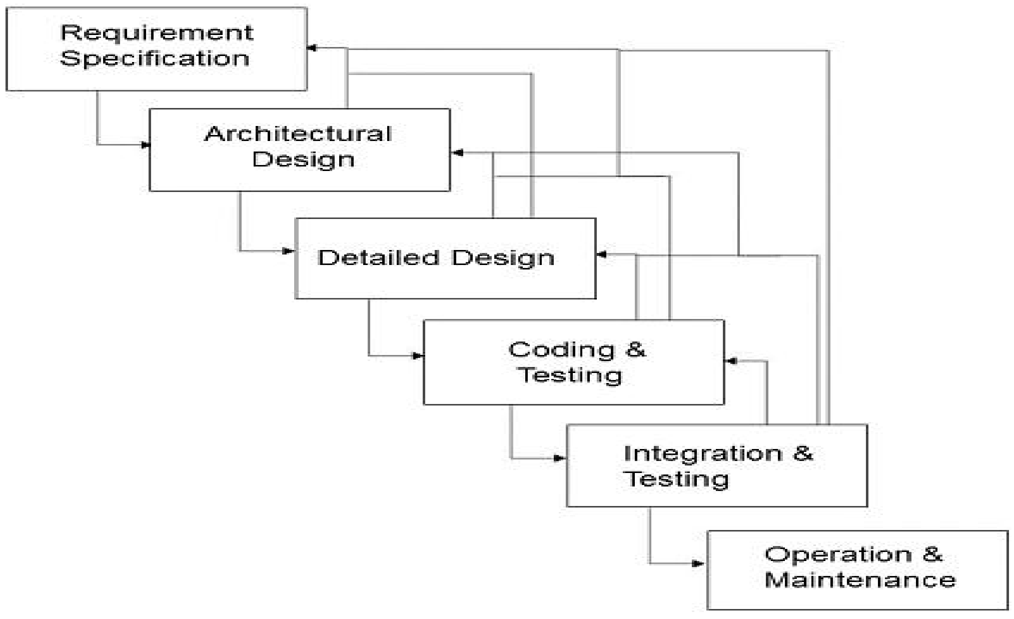
Software engineering is the study of designing, development and preservation of software. It comes in contact with HCI to make the man and machine interaction more vibrant and interactive. Let us see the following model in software engineering for interactive designing.

- **The Waterfall Method**



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.

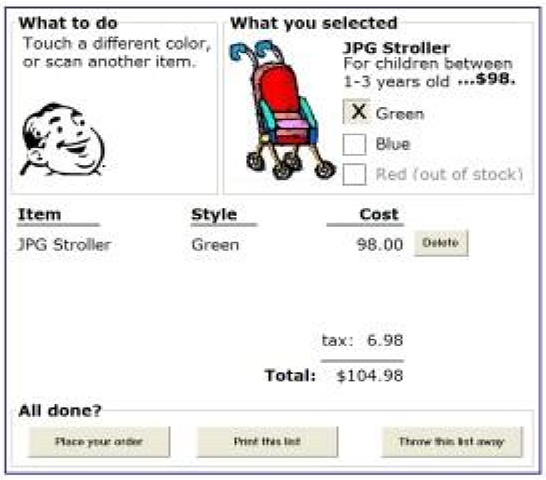
- **Interactive System Design**



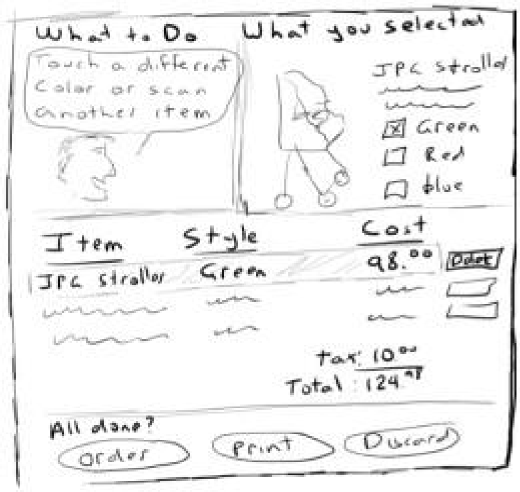
The interactive system design shows that every phase depends on each other to serve the purpose of designing and product creation. It is a continuous process as there is so much to know and users keep changing all the time. An interactive system designer should recognize this diversity.

- **Prototyping**

Prototyping is another type of software engineering model 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

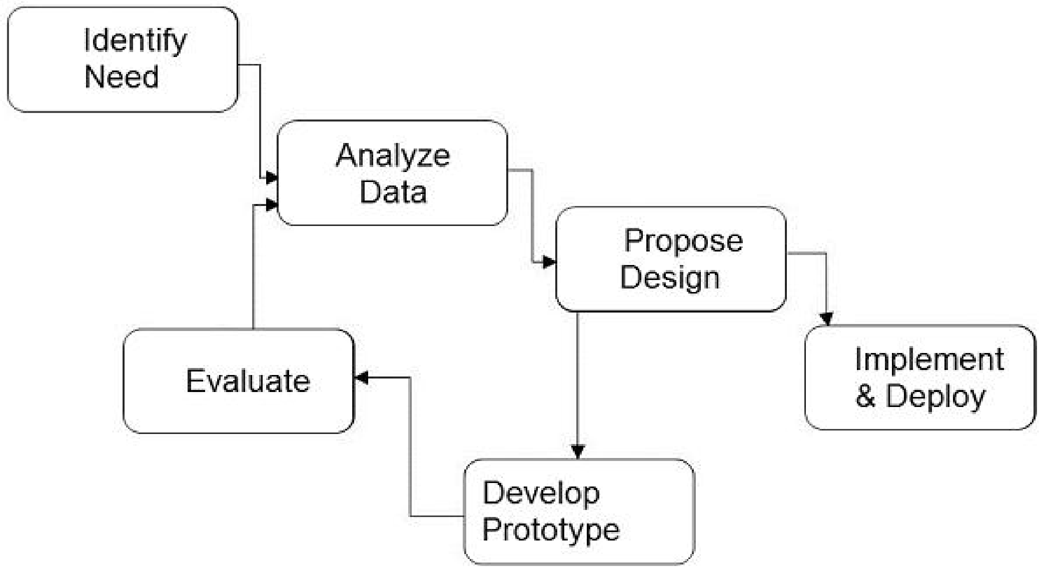
The above diagram can be considered as a Low Fidelity Prototype as it uses manual procedures like sketching in a paper.
A Medium Fidelity Prototype involves some but not all procedures of the system. E.g., first screen of a GUI.
Finally, a Hi Fidelity Prototype simulates all the functionalities of the system in a design. This prototype requires, time, money and work force.

- **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.

- **Interactive System Design Life Cycle (ISLC)**

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



- **Interactive Devices**

Several interactive devices are used for the human computer interaction. Some of them are known tools and some are recently developed or are a concept to be developed in the future. In this chapter, we will discuss on some new and old interactive devices.

- Touch Screen

The touch screen concept was prophesized decades ago, however the platform was acquired recently. Today there are many devices that use touch screen. After vigilant selection of these devices, developers customize their touch screen experiences.
The cheapest and relatively easy way of manufacturing touch screens are the ones using electrodes and a voltage association. Other than the hardware differences, software alone can bring major differences from one touch device to another, even when the same hardware is used.
Along with the innovative designs and new hardware and software, touch screens are likely to grow in a big way in the future. A further development can be made by making a sync between the touch and other devices.
In HCI, touch screen can be considered as a new interactive device.

- Gesture Recognition

Gesture recognition is a subject in language technology that has the objective of understanding human movement via mathematical procedures. Hand gesture recognition is currently the field of focus. This technology is future based.
This new technology magnitudes an advanced association between human and computer where no mechanical devices are used. This new interactive device might terminate the old devices like keyboards and is also heavy on new devices like touch screens.

- Speech Recognition

The technology of transcribing spoken phrases into written text is Speech Recognition. Such technologies can be used in advanced control of many devices such as switching on and off

the electrical appliances. Only certain commands are required to be recognized for a complete transcription. However, this cannot be beneficial for big vocabularies.
This HCI device help the user in hands free movement and keep the instruction-based technology up to date with the users.

- Keyboard

A keyboard can be considered as a primitive device known to all of us today. Keyboard uses an organization of keys/buttons that serves as a mechanical device for a computer. Each key in a keyboard corresponds to a single written symbol or character.
This is the most effective and ancient interactive device between man and machine that has given ideas to develop many more interactive devices as well as has made advancements in itself such as soft screen keyboards for computers and mobile phones.

- Response Time

Response time is the time taken by a device to respond to a request. The request can be anything from a database query to loading a web page. The response time is the sum of the service time and wait time. Transmission time becomes a part of the response time when the response has to travel over a network.
In modern HCI devices, there are several applications installed and most of them function simultaneously or as per the user's usage. This makes a busier response time. All of that increase in the response time is caused by increase in the wait time. The wait time is due to the running of the requests and the queue of requests following it.
So, it is significant that the response time of a device is faster for which advanced processors are used in modern devices.

- **Design Process & Task Analysis**

HCI Design
HCI design is considered as a problem-solving process that has components like planned usage, target area, resources, cost, and viability. It decides on the requirement of product similarities to balance trade-offs.
The following points are the four basic activities of interaction design −

- Identifying requirements
- Building alternative designs
- Developing interactive versions of the designs
- Evaluating designs

Three principles for user-centered approach are −

- Early focus on users and tasks
- Empirical Measurement
- Iterative Design

- **Design Methodologies**

Various methodologies have materialized since the inception that outline the techniques for human–computer interaction. Following are few design methodologies −

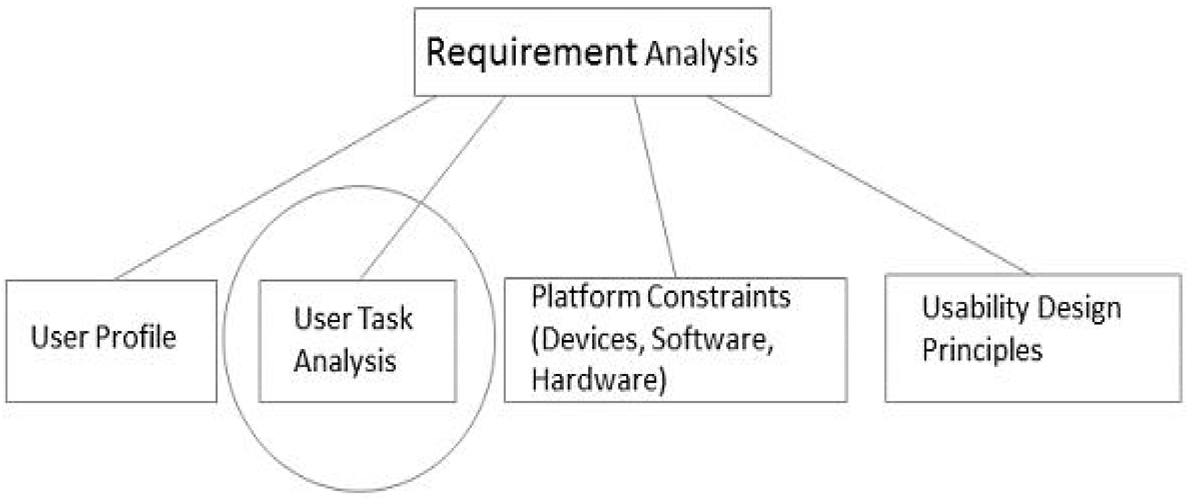
- Activity Theory − This is an HCI method that describes the framework where the human-computer interactions take place. Activity theory provides reasoning, analytical tools and interaction designs.
- User-Centered Design − It provides users the center-stage in designing where they get the opportunity to work with designers and technical practitioners.
- Principles of User Interface Design − Tolerance, simplicity, visibility, affordance, consistency, structure and feedback are the seven principles used in interface designing.
- Value Sensitive Design − This method is used for developing technology and includes three types of studies − conceptual, empirical and technical.
  - Conceptual investigations works towards understanding the values of the investors who use technology.
  - Empirical investigations are qualitative or quantitative design research studies that shows the designer's understanding of the users' values.
  - Technical investigations contain the use of technologies and designs in the conceptual and empirical investigations.

- **Participatory Design**

Participatory design process involves all stakeholders in the design process, so that the end result meets the needs they are desiring. This design is used in various areas such as software design, architecture, landscape architecture, product design, sustainability, graphic design, planning, urban design, and even medicine.
Participatory design is not a style, but focus on processes and procedures of designing. It is seen as a way of removing design accountability and origination by designers.

- **Task Analysis**

Task Analysis plays an important part in User Requirements Analysis.


Requirement Analysis
User Profile
User Task Analysis
Platform Constraints (Devices, Software, Hardware)
Usability Design Principles

Task analysis is the procedure to learn the users and abstract frameworks, the patterns used in workflows, and the chronological implementation of interaction with the GUI. It analyzes the ways in which the user partitions the tasks and sequence them.

- **What is a TASK?**

Human actions that contributes to a useful objective, aiming at the system, is a task. Task analysis defines performance of users, not computers.

- Hierarchical Task Analysis

Hierarchical Task Analysis is the procedure of disintegrating tasks into subtasks that could be analyzed using the logical sequence for execution. This would help in achieving the goal in the best possible way.
"A hierarchy is an organization of elements that, according to prerequisite relationships, describes the path of experiences a learner must take to achieve any single behavior that appears higher in the hierarchy. (Seels & Glasgow, 1990, p. 94)".

- Techniques for Analysis
    - Task decomposition − Splitting tasks into sub-tasks and in sequence.
    - Knowledge-based techniques − Any instructions that users need to know.

'User' is always the beginning point for a task.

- Ethnography − Observation of users' behavior in the use context.
- Protocol analysis − Observation and documentation of actions of the user. This is achieved by authenticating the user's thinking. The user is made to think aloud so that the user's mental logic can be understood.

- **Engineering Task Models**

Unlike Hierarchical Task Analysis, Engineering Task Models can be specified formally and are more useful.

- Characteristics of Engineering Task Models
    - Engineering task models have flexible notations, which describes the possible activities clearly.
    - They have organized approaches to support the requirement, analysis, and use of task models in the design.
    - They support the recycle of in-condition design solutions to problems that happen throughout applications.
    - Finally, they let the automatic tools accessible to support the different phases of the design cycle.

- **Concur Task Tree (CTT)**

CTT is an engineering methodology used for modeling a task and consists of tasks and operators. Operators in CTT are used to portray chronological associations between tasks. Following are the key features of a CTT −

- Focus on actions that users wish to accomplish.
- Hierarchical structure.
- Graphical syntax.
- Rich set of sequential operators.

- **Dialog Design**

A dialog is the construction of interaction between two or more beings or systems. In HCI, a dialog is studied at three levels −

- Lexical − Shape of icons, actual keys pressed, etc., are dealt at this level.
- Syntactic − The order of inputs and outputs in an interaction are described at this level.
- Semantic − At this level, the effect of dialog on the internal application/data is taken care of.

- **Dialog Representation**

To represent dialogs, we need formal techniques that serves two purposes −

- It helps in understanding the proposed design in a better way.
- It helps in analyzing dialogs to identify usability issues. E.g., Questions such as "does the design actually support undo?" can be answered.

- **Introduction to Formalism**

There are many formalism techniques that we can use to signify dialogs. In this chapter, we will discuss on three of these formalism techniques, which are −

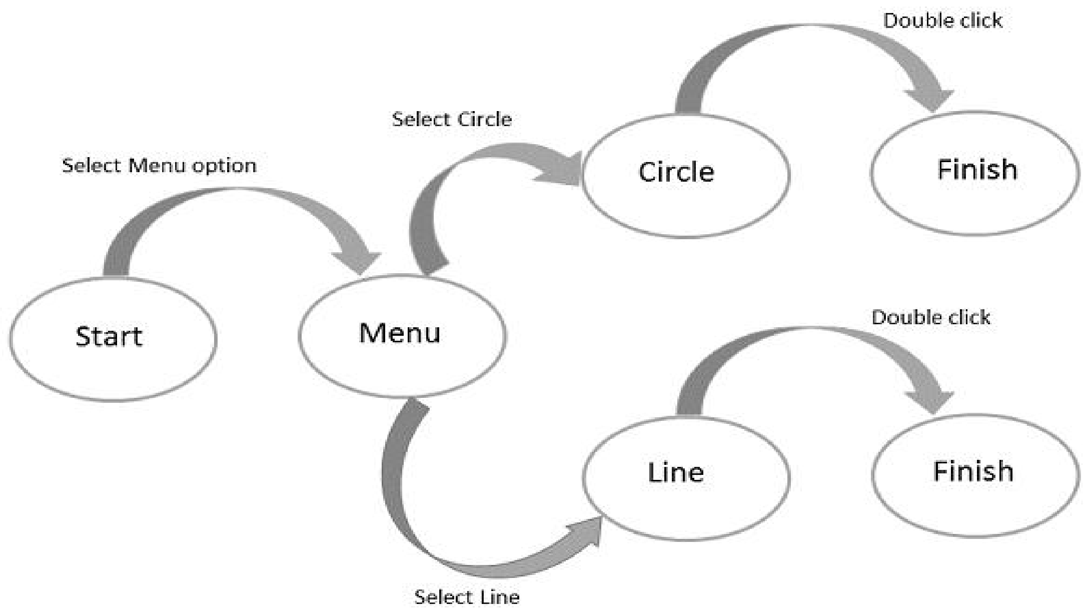
- The state transition networks (STN)
- The statecharts
- The classical Petri nets

- **State Transition Network (STN)**

STNs are the most spontaneous, which knows that a dialog fundamentally denotes to a progression from one state of the system to the next.

The syntax of an STN consists of the following two entities −

- Circles − A circle refers to a state of the system, which is branded by giving a name to the state.
- Arcs − The circles are connected with arcs that refers to the action/event resulting in the transition from the state where the arc initiates, to the state where it ends.



STN Diagram

- **StateCharts**

StateCharts represent complex reactive systems that extends Finite State Machines (FSM), handle concurrency, and adds memory to FSM. It also simplifies complex system representations. StateCharts has the following states −

Active state − The present state of the underlying FSM.

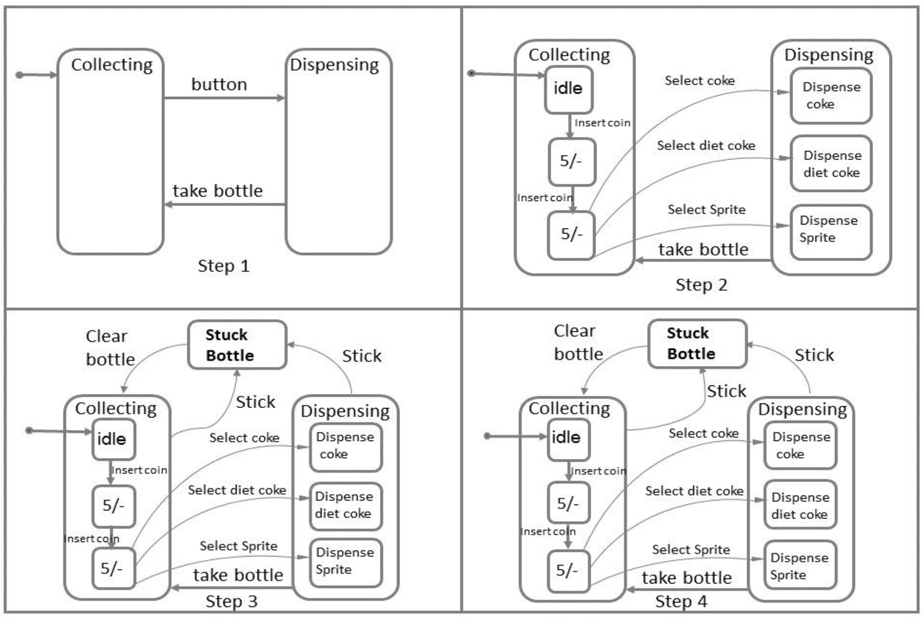
Basic states − These are individual states and are not composed of other states.

Super states − These states are composed of other states.

Illustration

For each basic state b, the super state containing b is called the ancestor state. A super state is called OR super state if exactly one of its sub states is active, whenever it is active.

Let us see the StateChart Construction of a machine that dispense bottles on inserting coins.



The above diagram explains the entire procedure of a bottle dispensing machine. On pressing the button after inserting coin, the machine will toggle between bottle filling and dispensing modes. When a required request bottle is available, it dispenses the bottle. In the background, another procedure runs where any stuck bottle will be cleared. The 'H' symbol in Step 4, indicates that a procedure is added to History for future access.

- **Petri Nets**

Petri Net is a simple model of active behavior, which has four behavior elements such as − places, transitions, arcs and tokens. Petri Nets provide a graphical explanation for easy understanding.

  - Place − This element is used to symbolize passive elements of the reactive system. A place is represented by a circle.
  - Transition − This element is used to symbolize active elements of the reactive system. Transitions are represented by squares/rectangles.
  - Arc − This element is used to represent causal relations. Arc is represented by arrows.
  - Token − This element is subject to change. Tokens are represented by small filled circles.

- **Visual Thinking**

Visual materials have assisted in the communication process since ages in form of paintings, sketches, maps, diagrams, photographs, etc. In today's world, with the invention of technology and its further growth, new potentials are offered for visual information such as thinking and reasoning. As per studies, the command of visual thinking in human-computer interaction (HCI) design is still not discovered completely. So, let us learn the theories that support visual thinking in sense-making activities in HCI design.

An initial terminology for talking about visual thinking was discovered that included concepts such as visual immediacy, visual impetus, visual impedance, and visual metaphors, analogies and associations, in the context of information design for the web.
As such, this design process became well suited as a logical and collaborative method during the design process. Let us discuss in brief the concepts individually.

- **Visual Immediacy**

It is a reasoning process that helps in understanding of information in the visual representation. The term is chosen to highlight its time related quality, which also serves as an indicator of how well the reasoning has been facilitated by the design.

- **Visual Impetus**

Visual impetus is defined as a stimulus that aims at the increase in engagement in the contextual aspects of the representation.

- **Visual Impedance**

It is perceived as the opposite of visual immediacy as it is a hindrance in the design of the representation. In relation to reasoning, impedance can be expressed as a slower cognition.

- **Visual Metaphors, Association, Analogy, Abduction and Blending**
    - When a visual demonstration is used to understand an idea in terms of another familiar idea it is called a visual metaphor.
    - Visual analogy and conceptual blending are similar to metaphors. Analogy can be defined as an implication from one particular to another. Conceptual blending can be defined as combination of elements and vital relations from varied situations.

The HCI design can be highly benefited with the use of above-mentioned concepts. The concepts are pragmatic in supporting the use of visual procedures in HCI, as well as in the design processes.

- **Direct Manipulation Programming**

Direct manipulation has been acclaimed as a good form of interface design, and are well received by users. Such processes use many sources to get the input and finally convert them into an output as desired by the user using inbuilt tools and programs.
"Directness" has been considered as a phenomenon that contributes majorly to the manipulation programming. It has the following two aspects.

- Distance
- Direct Engagement

- Distance

Distance is an interface that decides the gulfs between a user's goal and the level of explanation delivered by the systems, with which the user deals. These are referred to as the Gulf of Execution and the Gulf of Evaluation.

- The Gulf of Execution

The Gulf of Execution defines the gap/gulf between a user's goal and the device to implement that goal. One of the principal objectives of Usability is to diminish this gap by removing barriers and follow steps to minimize the user's distraction from the intended task that would prevent the flow of the work.

- The Gulf of Evaluation

The Gulf of Evaluation is the representation of expectations that the user has interpreted from the system in a design. As per Donald Norman, The gulf is small when the system provides information about its state in a form that is easy to get, is easy to interpret, and matches the way the person thinks of the system.

- Direct Engagement

It is described as a programming where the design directly takes care of the controls of the objects presented by the user and makes a system less difficult to use.

The scrutiny of the execution and evaluation process illuminates the efforts in using a system. It also gives the ways to minimize the mental effort required to use a system.

- Problems with Direct Manipulation
    - Even though the immediacy of response and the conversion of objectives to actions has made some tasks easy, all tasks should not be done easily. For example, a repetitive operation is probably best done via a script and not through immediacy.
    - Direct manipulation interfaces finds it hard to manage variables, or illustration of discrete elements from a class of elements.
    - Direct manipulation interfaces may not be accurate as the dependency is on the user rather than on the system.
    - An important problem with direct manipulation interfaces is that it directly supports the techniques, the user thinks.

- **Item Presentation Sequence**

In HCI, the presentation sequence can be planned according to the task or application requirements. The natural sequence of items in the menu should be taken care of. Main factors in presentation sequence are –

- Time
- Numeric ordering
- Physical properties

A designer must select one of the following prospects when there are no task-related arrangements –

- Alphabetic sequence of terms
- Grouping of related items
- Most frequently used items first
- Most important items first

- **Menu Layout**
    - Menus should be organized using task semantics.
    - Broad-shallow should be preferred to narrow-deep.
    - Positions should be shown by graphics, numbers or titles.
    - Subtrees should use items as titles.
    - Items should be grouped meaningfully.
    - Items should be sequenced meaningfully.
    - Brief items should be used.
    - Consistent grammar, layout and technology should be used.
    - Type ahead, jump ahead, or other shortcuts should be allowed.
    - Jumps to previous and main menu should be allowed.
    - Online help should be considered.

Guidelines for consistency should be defined for the following components −

- Titles
- Item placement
- Instructions
- Error messages
- Status reports

- **Form Fill-in Dialog Boxes**

Appropriate for multiple entry of data fields −

- Complete information should be visible to the user.
- The display should resemble familiar paper forms.
- Some instructions should be given for different types of entries.

Users must be familiar with −

- Keyboards
- Use of TAB key or mouse to move the cursor
- Error correction methods
- Field-label meanings
- Permissible field contents
- Use of the ENTER and/or RETURN key.

- **Form Fill-in Design Guidelines −**
    - Title should be meaningful.
    - Instructions should be comprehensible.
    - Fields should be logically grouped and sequenced.
    - The form should be visually appealing.
    - Familiar field labels should be provided.
    - Consistent terminology and abbreviations should be used.
    - Convenient cursor movement should be available.
    - Error correction for individual characters and entire field's facility should be present.
    - Error prevention.
    - Error messages for unacceptable values should be populated.
    - Optional fields should be clearly marked.
    - Explanatory messages for fields should be available.
    - Completion signal should populate.

- **Information Search & Visualization**

Database Query

A database query is the principal mechanism to retrieve information from a database. It consists of predefined format of database questions. Many database management systems use the Structured Query Language (SQL) standard query format.

Example

```
SELECT DOCUMENT#
FROM JOURNAL-DB
WHERE (DATE >= 2004 AND DATE <= 2008)
AND (LANGUAGE = ENGLISH OR FRENCH)
AND (PUBLISHER = ASIST OR HFES OR ACM)
```

Users perform better and have better contentment when they can view and control the search. The database query has thus provided substantial amount of help in the human computer interface.

The following points are the five-phase frameworks that clarifies user interfaces for textual search −

- Formulation − expressing the search
- Initiation of action − launching the search
- Review of results − reading messages and outcomes
- Refinement − formulating the next step
- Use − compiling or disseminating insight

Multimedia Document Searches

Following are the major multimedia document search categories.

- **Image Search**

Preforming an image search in common search engines is not an easy thing to do. However there are sites where image search can be done by entering the image of your choice. Mostly, simple drawing tools are used to build templates to search with. For complex searches such as fingerprint matching, special software are developed where the user can search the machine for the predefined data of distinct features.

- **Map Search**

Map search is another form of multimedia search where the online maps are retrieved through mobile devices and search engines. Though a structured database solution is required for complex searches such as searches with longitude/latitude. With the advanced database options, we can retrieve maps for every possible aspect such as cities, states, countries, world maps, weather sheets, directions, etc.

- **Design/Diagram Searches**

Some design packages support the search of designs or diagrams as well. E.g., diagrams, blueprints, newspapers, etc.

- **Sound Search**

Sound search can also be done easily through audio search of the database. Though user should clearly speak the words or phrases for search.

- **Video Search**

New projects such as Infomedia helps in retrieving video searches. They provide an overview of the videos or segmentations of frames from the video.

- **Animation Search**

The frequency of animation search has increased with the popularity of Flash. Now it is possible to search for specific animations such as a moving boat.

- **Information Visualization**

Information visualization is the interactive visual illustrations of conceptual data that strengthen human understanding. It has emerged from the research in human-computer interaction and is applied as a critical component in varied fields. It allows users to see, discover, and understand huge amounts of information at once.
Information visualization is also an assumption structure, which is typically followed by formal examination such as statistical hypothesis testing.

- Advanced Filtering

Following are the advanced filtering procedures −

- Filtering with complex Boolean queries
- Automatic filtering
- Dynamic queries
- Faceted metadata search
- Query by example
- Implicit search
- Collaborative filtering
- Multilingual searches
- Visual field specification

- Hypertext and Hypermedia

Hypertext can be defined as the text that has references to hyperlinks with immediate access. Any text that provides a reference to another text can be understood as two nodes of information with the reference forming the link. In hypertext, all the links are active and when clicked, opens something new.
Hypermedia on the other hand, is an information medium that holds different types of media, such as, video, CD, and so forth, as well as hyperlinks.
Hence, both hypertext and hypermedia refer to a system of linked information. A text may refer to links, which may also have visuals or media. So hypertext can be used as a generic term to denote a document, which may in fact be distributed across several media.

- **Object Action Interface Model for Website Design**

Object Action Interface (OAI), can be considered as the next step of the Graphical User Interface (GUI). This model focusses on the priority of the object over the actions.

- OAI Model

The OAI model allows the user to perform action on the object. First the object is selected and then the action is performed on the object. Finally, the outcome is shown to the user. In this model, the user does not have to worry about the complexity of any syntactical actions.

The object–action model provides an advantage to the user as they gain a sense of control due to the direct involvement in the design process. The computer serves as a medium to signify different tools.

- **Object Oriented Programming**

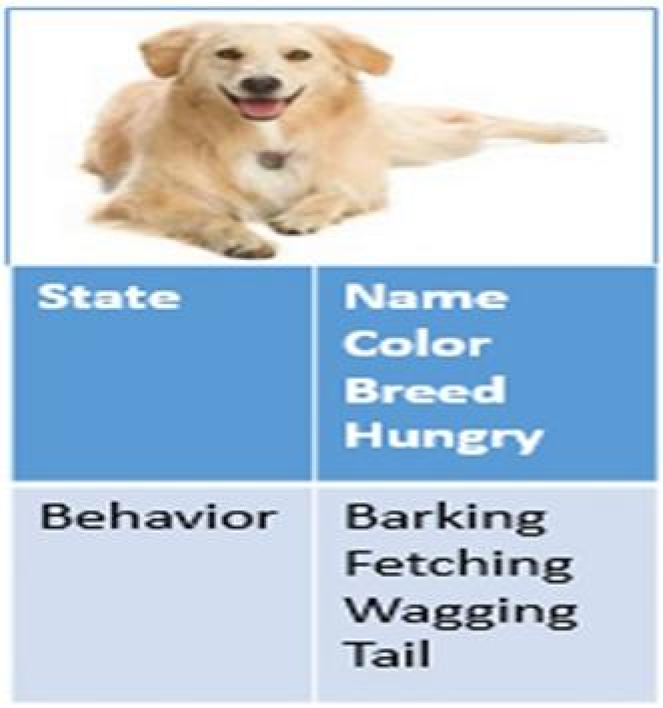
Object Oriented Programming Paradigm (OOPP)

The Object-Oriented programming paradigm plays an important role in human computer interface. It has different components that takes real world objects and performs actions on them, making live interactions between man and the machine. Following are the components of OOPP −

  - This paradigm describes a real-life system where interactions are among real objects.
  - It models applications as a group of related objects that interact with each other.
  - The programming entity is modeled as a class that signifies the collection of related real-world objects.
  - Programming starts with the concept of real world objects and classes.
  - Application is divided into numerous packages.
  - A package is a collection of classes.
  - A class is an encapsulated group of similar real world objects.

- **Objects**

Real-world objects share two characteristics − They all have state and behavior. Let us see the following pictorial example to understand Objects.



State	Name Color Breed Hungry
Behavior	Barking Fetching Wagging Tail

In the above diagram, the object ‘Dog’ has both state and behavior.

An object stores its information in attributes and discloses its behavior through methods. Let us now discuss in brief the different components of object-oriented programming.

- Data Encapsulation

Hiding the implementation details of the class from the user through an object's methods is known as data encapsulation. In object oriented programming, it binds the code and the data together and keeps them safe from outside interference.

- Public Interface

The point where the software entities interact with each other either in a single computer or in a network is known as pubic interface. This help in data security. Other objects can change the state of an object in an interaction by using only those methods that are exposed to the outer world through a public interface.

- Class

A class is a group of objects that has mutual methods. It can be considered as the blueprint using which objects are created. Classes being passive do not communicate with each other but are used to instantiate objects that interact with each other.

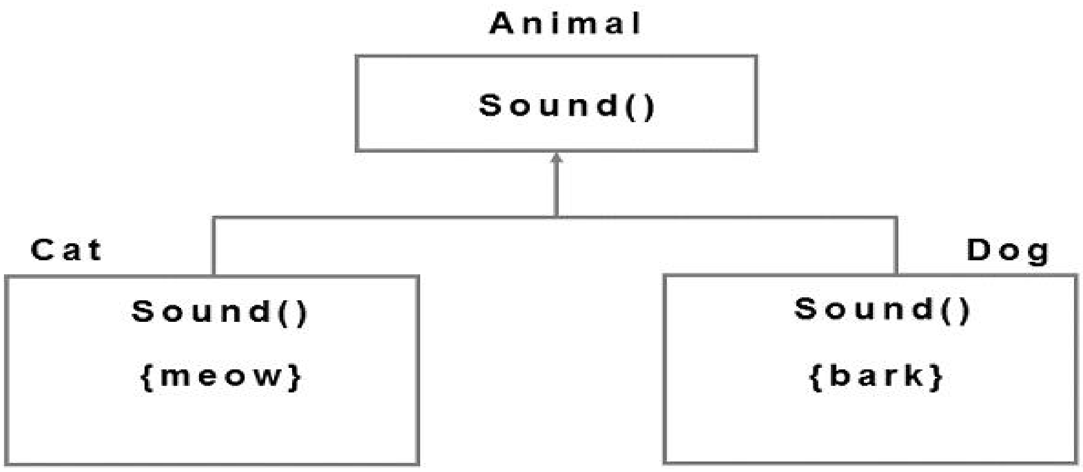
- Inheritance

Inheritance as in general terms is the process of acquiring properties. In OOP one object inherit the properties of another object.

- Polymorphism

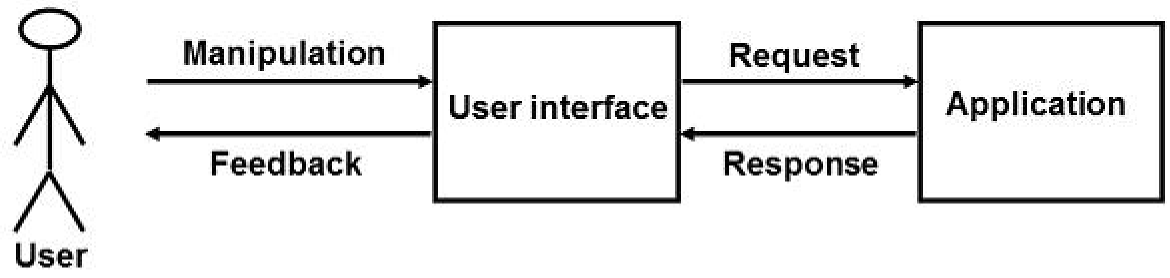
Polymorphism is the process of using same method name by multiple classes and redefines methods for the derived classes.

Example



- **Object Oriented Modeling of User Interface Design**

Object oriented interface unites users with the real-world manipulating software objects for designing purpose. Let us see the diagram.

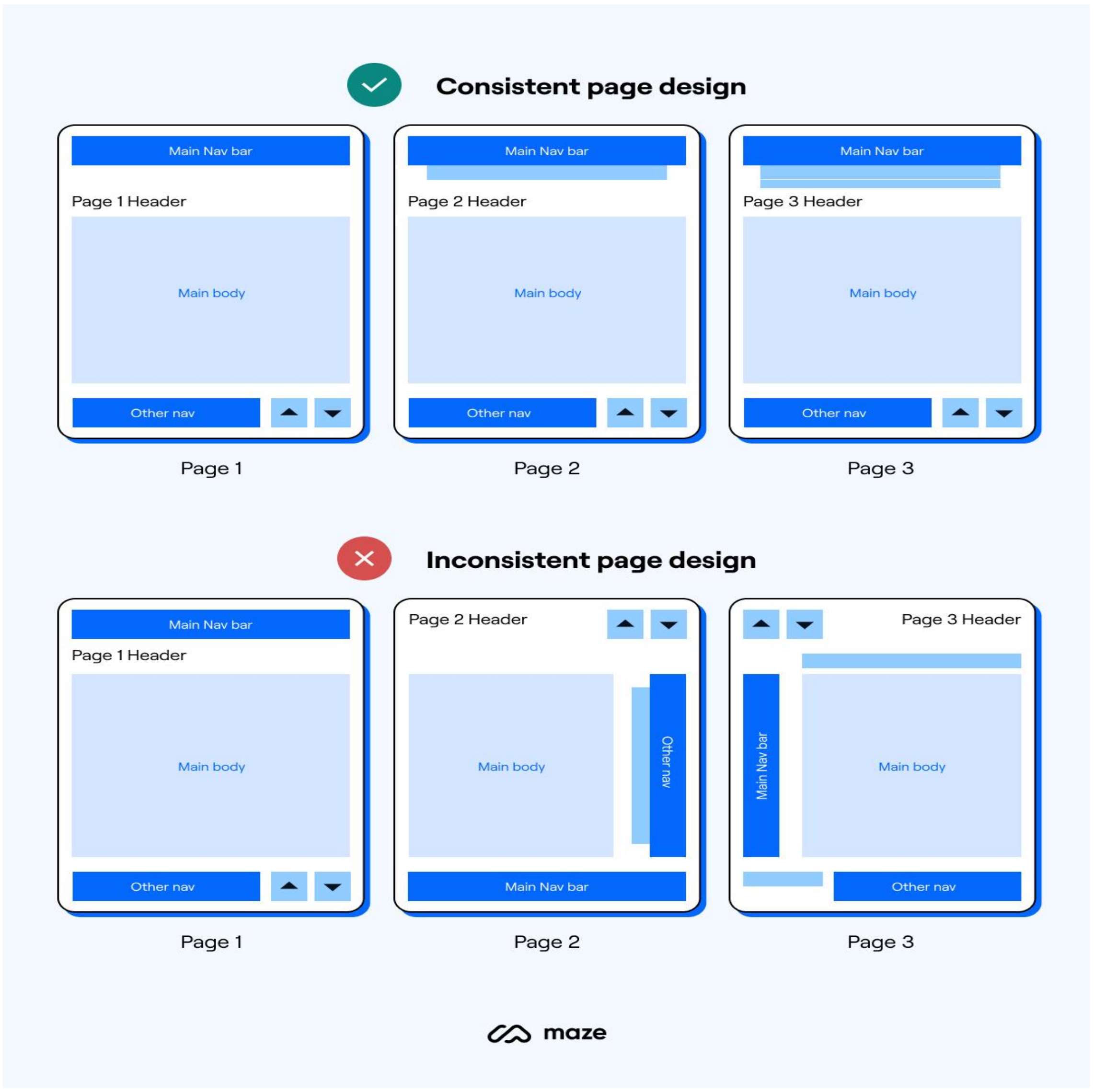


Interface design strive to make successful accomplishment of user’s goals with the help of interaction tasks and manipulation.
While creating the OOM for interface design, first of all analysis of user requirements is done. The design specifies the structure and components required for each dialogue. After that, interfaces are developed and tested against the Use Case. Example − Personal banking application.
The sequence of processes documented for every Use Case are then analyzed for key objects. This results into an object model. Key objects are called analysis objects and any diagram showing relationships between these objects is called object diagram.

- **Principles and Patterns of User Interface Design**

User Interface Design is a platform where the effective interaction of the user with the system can be seen. Its main aim is to enhance the appearance of the product, the quality of technology used and the usability of the product. It refers to the software or the hardware of the system which the user can see and also the various ways or commands to control or use the product.
It focuses on the looks or how the app or software is looking. Attributes like theme, animations, colors, etc constitute the user interface. A user more often judges a system by its interface rather than its functionality. An interface that is designed poorly or not leads to improper use of the system by the user. Many users in business systems interact with the systems using a graphical user interface. Even text-based interfaces are also used. Therefore, it is very important to consider the principles and patterns while designing a User Interface.

Golden rules of user interface design
In his book Designing the User Interface, Ben Shneiderman describes eight golden rules of UI design. Let’s look at them in detail:
1. Make user interfaces consistent
Mike Gilfillan, Technical Lead Developer at Edge of the Web says, “Consistency is key - multiple acolors, fonts and styles can create confusion, whilst consistency creates familiarity.”
Consistent UI means using similar design patterns, identical terminology in prompts, homogenous menus and screens, and consistent commands throughout the interface.



2. Allow users to navigate easily via shortcuts

Expert users, or users who frequent your website or use your product regularly, need shortcuts to move quickly through the interface.

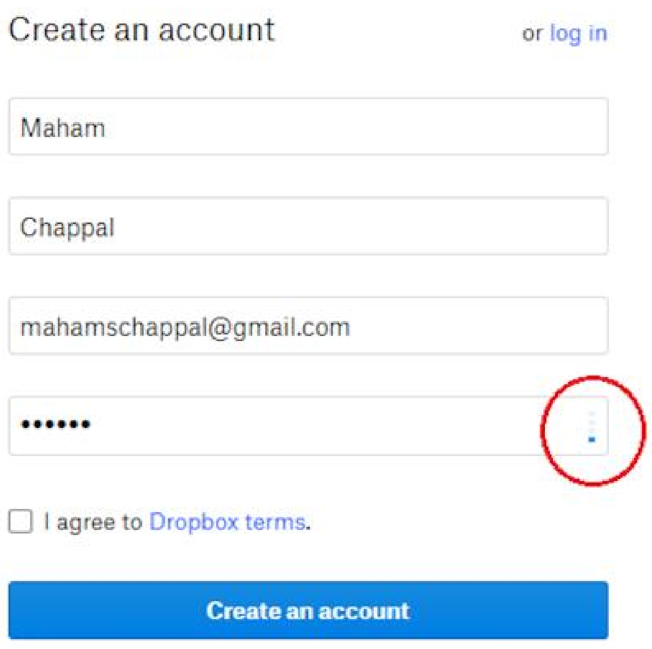
Just like how most Windows users make use of the shortcut CTRL + C to quickly copy text and CTRL + V to paste it, you need to make navigation and operating user interfaces easy through shortcuts.

3. Provide informative feedback

Provide feedback through readable UI copy for all user moves. Ben Shneiderman explains: “For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial.”

For example, when users are asked to create a password, your UI should offer information on how strong it should be by either giving an example of a strong password or using symbols that demonstrate how strong the user’s password currently is.

A great example is DropBox’s signup form. It shows the strength of a user’s password through a sleek bar. When I added just my name in the password field, it signaled how weak my password was through that one bar.


Create an account

or log in

Maham

Chappal

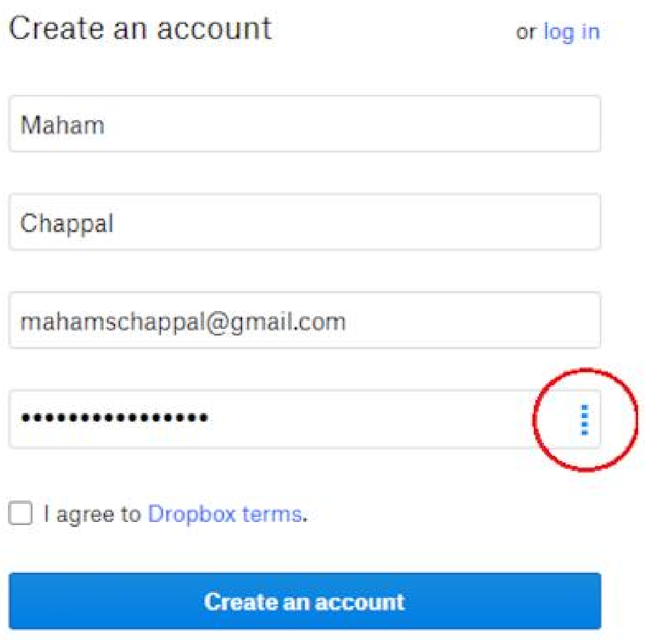
mahamschappal@gmail.com

••••••

I agree to Dropbox terms.

Creating an account on Dropbox

So I created a significantly difficult password with multiple numbers and symbols:


Create an account

or log in

Maham

Chappal

mahamschappal@gmail.com

•••••••••••••••••

I agree to Dropbox terms.

Creating an account on Dropbox

4. Design dialog to yield closure

According to Shneiderman, “Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group of actions gives users the satisfaction of accomplishment, a sense of relief, a signal to drop contingency plans from their minds, and an indicator to prepare for the next group of actions.”

A great example is taking users to a Thank You page with a summary after they complete an order, informing them it's confirmed.

5. Prevent error as much as possible

Make the UI as easy to use as possible by preventing serious user errors. So from greying out menu items that are not available to preventing users from typing in alphabets in fields that ask for phone numbers, try to prevent error as much as possible.

This is also where usability testing comes in. As a designer, you need to ensure everything works as intended by testing the design with users before launching. It not only helps to test the functionality and usability of your product, but also helps you to understand your target audience's needs better.

However, occasionally errors happen. So if a user has made an error, make sure to offer them a clear explanation to understand the error and an easy solution to solve it.

6. Allow users ways to reverse their actions easily

Offer users easy and obvious ways to reverse their steps when they've taken a wrong step.

Shneiderman explains: "This feature relieves anxiety, since users know that errors can be undone, and encourages exploration of unfamiliar options."

Say someone has accidentally added the wrong information in a multi-page form, allow them to go back to that page and rectify their mistake easily without having to start all over again.

7. Support internal locus of control

"Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. They don't want surprises or changes in familiar behavior, and they are annoyed by tedious data-entry sequences, difficulty in obtaining necessary information, and inability to produce their desired result," says Shneiderman.

A great example of keeping users in control is when someone is about to exit Microsoft Office and is asked by the system if they're sure they want to exit without saving their work. Not only does this make the user feel in control, but it also ensures that in case of an accidental exit, their work is not lost.

8. Minimize memory load

A key rule for making user interfaces easy for people to use is minimizing cognitive load. Cognitive load (or memory load) can reduce a user's capacity to perform important tasks, so it's critical that computers take over the burden of memory from them as much as they can.

For instance, don't make users re-enter personal information every time they're buying from your website or add their email address and name every time they log into your website.

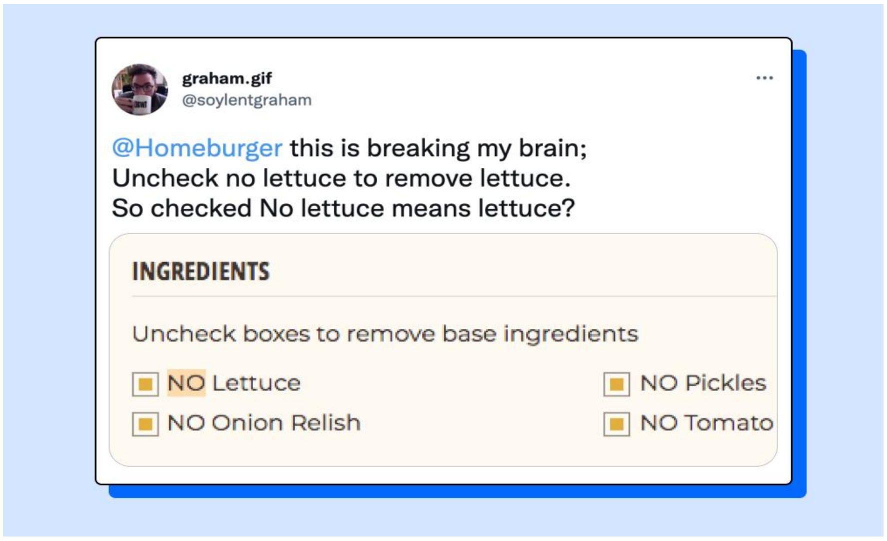
While designing, always choose recognition over recall to enable users to complete their tasks quickly and without hassle.

- **User Interface Design Principles:**

When designing new interfaces or updating existing products, keeping in mind UI design principles is imperative. There are some principles that should be kept in mind while designing the User Interface. These principles are given below:

  - **Clarity**

From recognizing interactive and static elements to making navigation intuitive, clarity is an essential part of a great UI design.


graham.gif
@soylentgraham

@Homeburger this is breaking my brain;
Uncheck no lettuce to remove lettuce.
So checked No lettuce means lettuce?

INGREDIENTS

Uncheck boxes to remove base ingredients

▣ NO Lettuce	▣ NO Pickles
▣ NO Onion Relish	▣ NO Tomato

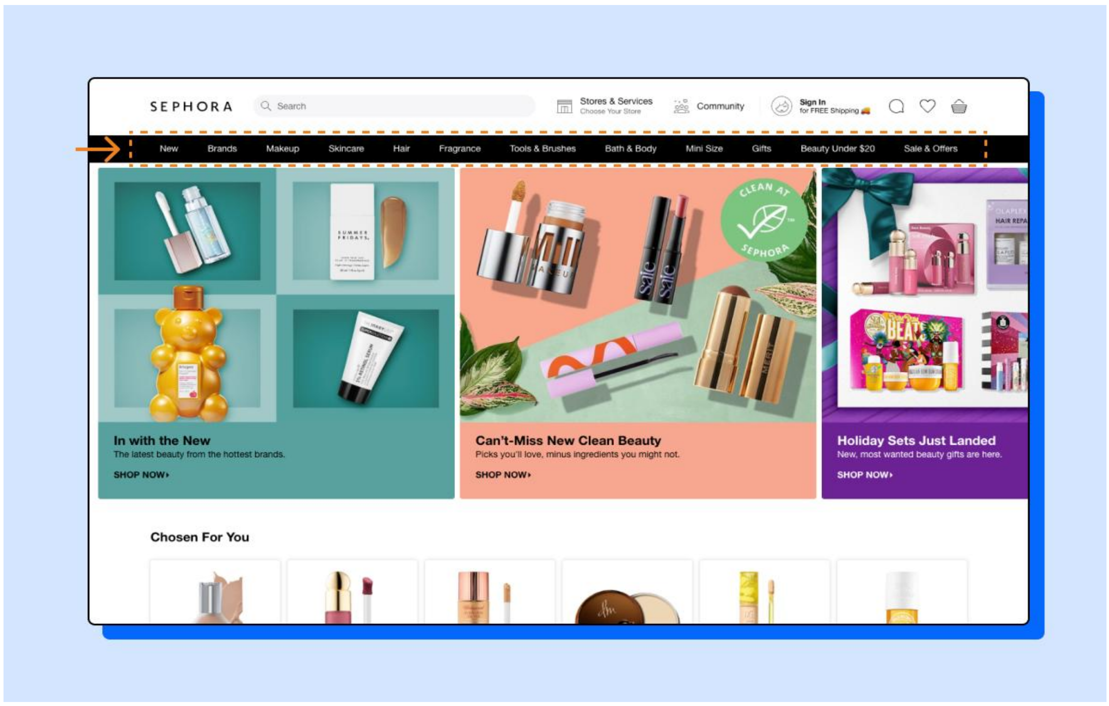
So when you create your product, ask yourself the following questions:

- Is your navigation intuitive? Are users directed and encouraged to move from one page to the next with ease?
- Have you used highly visible buttons that prompt users into clicking them?
- Is the purpose of each element on your product, website or application, clear and easy to understand?

You only have eight seconds to grab a user's attention, so make sure you don't waste those seconds by creating confusion and chaos.

- **Familiarity**

Do you automatically look for the menu at the top of the homepage of a new website?
Why? Because that's where we perceive the menu to be.



Displaying the website menu at the top is a great way of incorporating the design principle of familiarity like Sephora does here

The best interfaces are familiar for users.

Usability, i.e. how easily a user interacts with a product or a website, is closely related to familiarity. Users depend on elements and interfaces acting in a way that's familiar to their digital experience.

Jacob's Law states that "Users spend most of their time on other sites. This means that users prefer your site to work the same way as all the other sites they already know."

Not only should you leverage established UI design principles and rules (like Ben Shneiderman's golden rules of UI design described above) to incorporate familiarity in your design, but also ensure that all elements are in sync throughout the interface.

There are several benefits of incorporating the UI design principle of familiarity in your product:

- Increases user retention
- The more familiar the user is with your interface and more easily they can use your website or app, the more they'll come back to it.
- And since research shows that acquiring a new customer costs 6x to 7x times more than to retain an old one, it's vital that you create a seamless user experience by leveraging familiarity.
- Easier for UI designers
- It's easier for UI designers to incorporate tried and tested interface design solutions than create new ones from scratch.
- Reduces the learning curve for users
- The less time visitors have to spend understanding how the user interface works, the faster they can start using your product or service.
- It also reduces the chances of them exiting your website and moving on to the next website because yours had a steep learning curve.

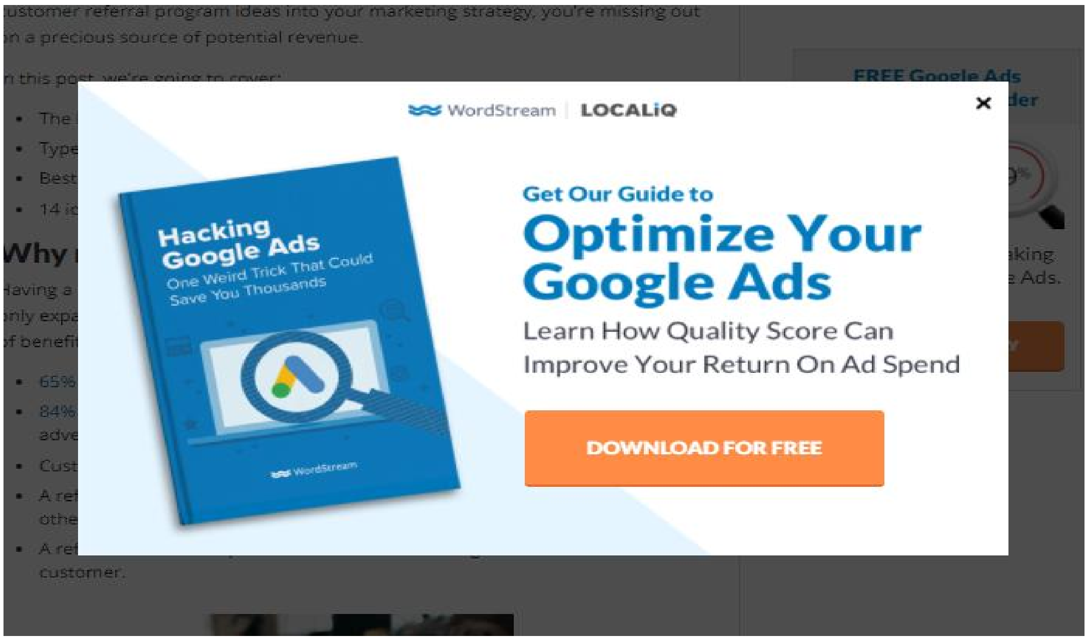
- **User Control**

Place users in control of the interface. Jakob Nielsen explains why this is important: "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."

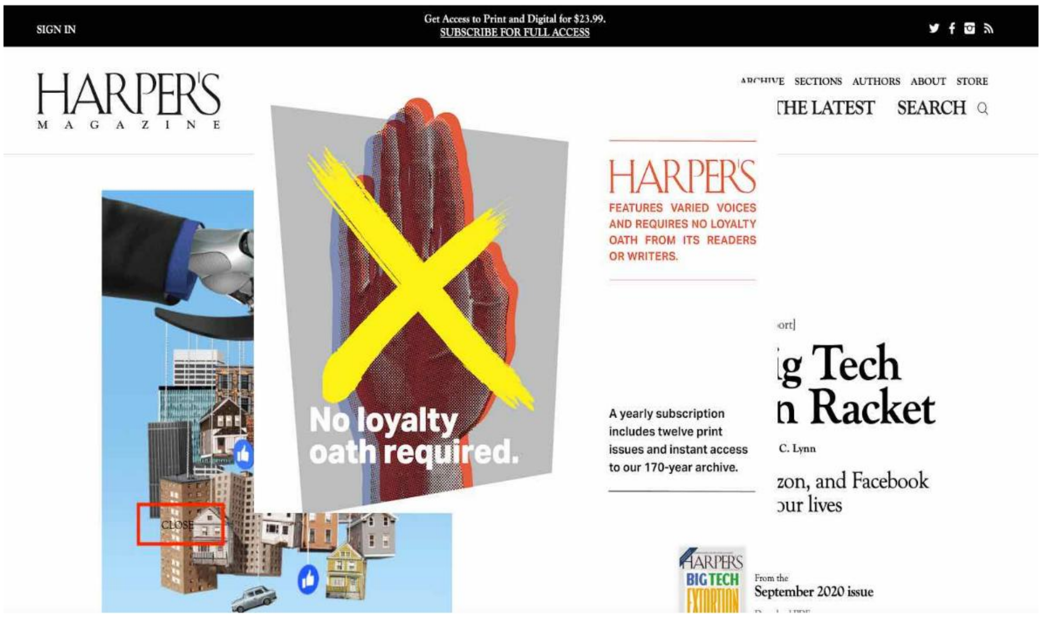
This basically means giving users different options to go back a step when they feel they've made a mistake. This is closely related to Ben Shneiderman's golden UI design rule, 'Supporting internal locus of control', and 'Allowing users to reverse their actions easily'.

As an example, whenever you're creating forms, allow users to click the <Back> button and go back to the page they were last on. Don't take them all the way back to the homepage or the start of the form.

Similarly, when using overlays on your website, make sure the exit button {x} is clear. Otherwise, users might click the back button on the browser and go back two steps instead of just exiting the overlay.
See how the small x button is clear and instantly noticeable here.


WordStream | LOCALiQ
×
Hacking Google Ads
One Weird Trick That Could Save You Thousands
Get Our Guide to
Optimize Your Google Ads
Learn How Quality Score Can Improve Your Return On Ad Spend
DOWNLOAD FOR FREE

Making the exit button on an overlay visible is a great way of placing users in control of the interface.
But, can you find the exit button on this screen?


SIGN IN
Get Access to Print and Digital for $23.99.
SUBSCRIBE FOR FULL ACCESS
HARPER'S MAGAZINE
ARCHIVE SECTIONS AUTHORS ABOUT STORE
THE LATEST SEARCH
No loyalty oath required.
HARPER'S FEATURES VARIED VOICES AND REQUIRES NO LOYALTY OATH FROM ITS READERS OR WRITERS.
A yearly subscription includes twelve print issues and instant access to our 170-year archive.
CLOSE
g Tech
n Racket
C. Lynn
zon, and Facebook
our lives
From the September 2020 issue

An invisible exit button on an overlay can easily confuse and frustrate users

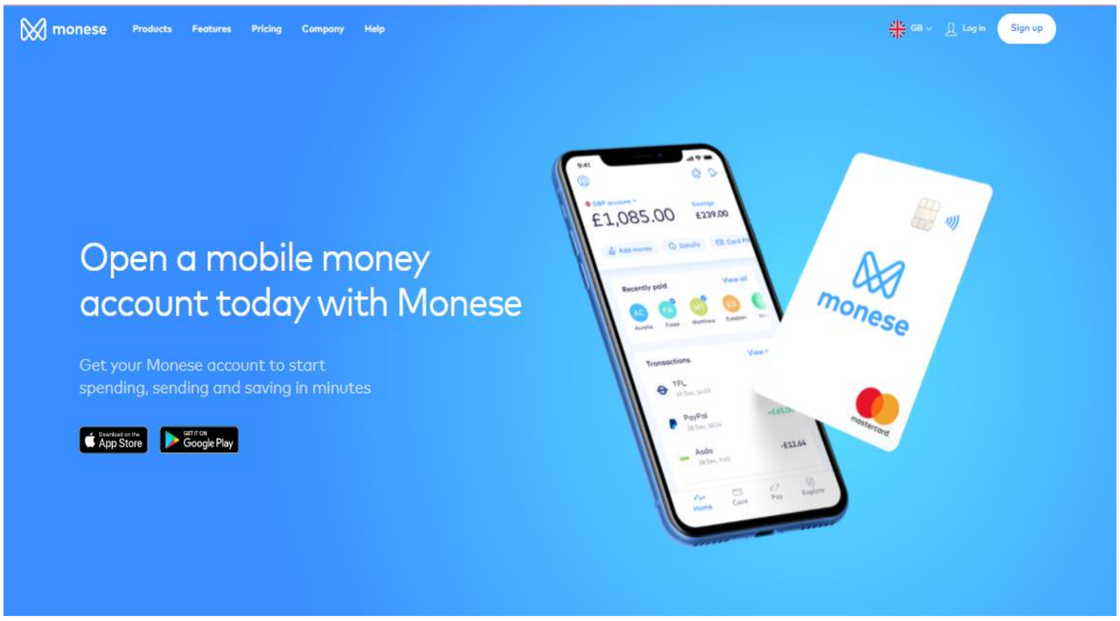
- **Hierarchy**

Strong visual hierarchy is a core design principle of a successful user interface. It consists of arranging visual elements in a way that explains the level of importance of each element and

guides users to take the desired action. As a designer, your job is to organize UI design elements in a way that makes it easy for users to navigate within your product.
As Pascal Potvin, design lead at IBM, says, “[...] Visual hierarchy creates an order of importance to visual elements in order to direct a user’s attention and make information easier to consume.”
It ensures that users see the most important information first, then the next, and so on, and is established through various elements. Some of them include:

- **Color**

The first and one of the most important elements of establishing visual hierarchy is color. Bright colors stand out the most and can be used in muted color schemes to direct users to take a certain action.
Monese and Mailchimp both make great use of colour in their designs
.


monese
Products
Features
Pricing
Company
Help
GB
Log in
Sign up
Open a mobile money account today with Monese
Get your Monese account to start spending, sending and saving in minutes
App Store
Google Play
£1,085.00
monese
mastercard

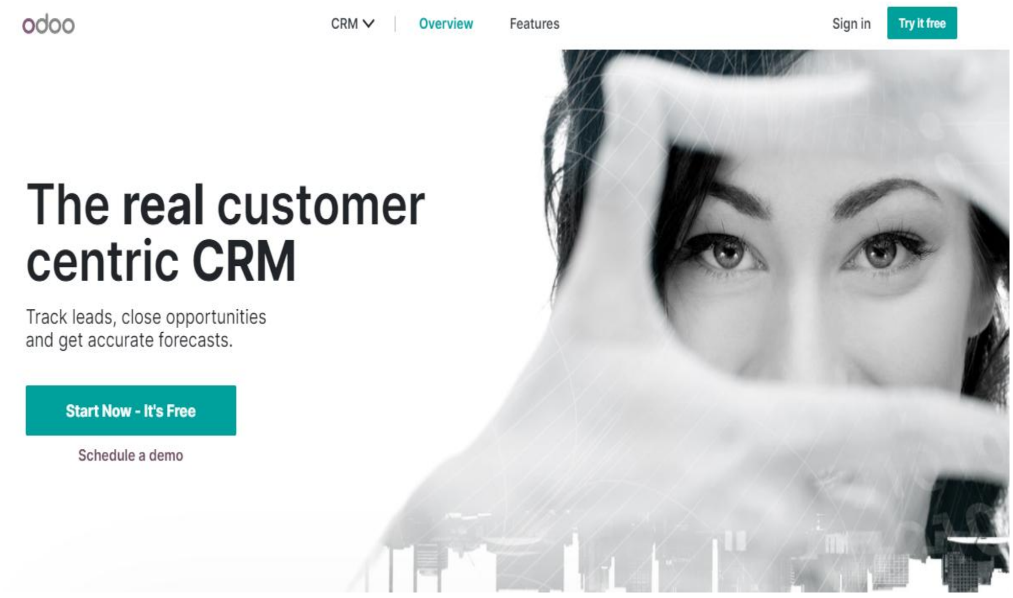
Colour blocking is an excellent way of highlighting certain UI elements

- **Size**

Size matters a lot in UI design, especially when establishing visual hierarchy. The bigger the element is, the more visible it is. Smaller elements are usually those of less importance. So, as a designer, try to make important things (like headlines or CTAs) bigger and bolder.

- **Fonts**

Play around with different sizes, weights and styles of fonts to establish visual hierarchy. This is exactly what Odoo does here.



The real customer centric CRM

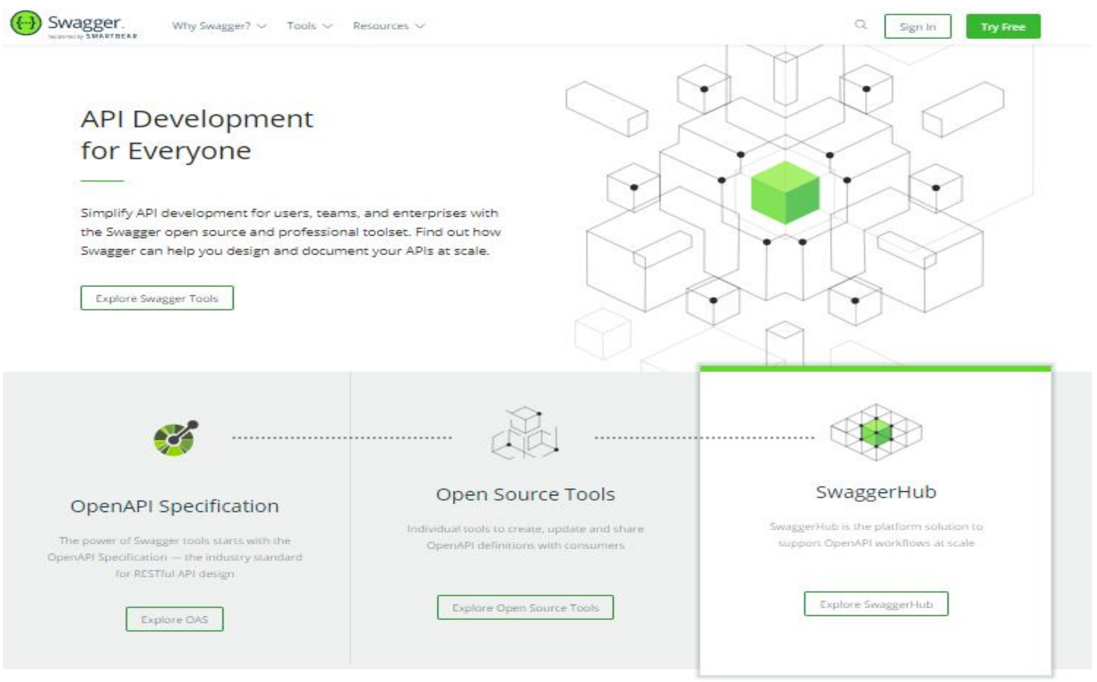
Track leads, close opportunities and get accurate forecasts.

Odoo using different sized fonts to create visual hierarchy

Notice how the main headline, which aptly explains what it's all about, is of the largest font size and two main words ('real' and 'CRM') are written in bold to emphasize its message. And while the CTA is of smaller font size, it's highlighted through a coloured box to ensure that the audience's eyes are drawn to it immediately after reading the headline.

- **Negative space**

Give your elements some breathing room - don't cram all the objects and elements together on the screen. Negative space makes important elements pop and stand out. Swaggar makes great use of negative space on their website's main page.



API Development for Everyone

Simplify API development for users, teams, and enterprises with the Swagger open source and professional toolset. Find out how Swagger can help you design and document your APIs at scale.

OpenAPI Specification

The power of Swagger tools starts with the OpenAPI Specification — the industry standard for RESTful API design

Open Source Tools

Individual tools to create, update and share OpenAPI definitions with consumers

SwaggerHub

SwaggerHub is the platform solution to support OpenAPI workflows at scale

Swagger uses negative space in a great manner to highlight its various features and main message

As Nick Kampouris, User Experience & User Interface Designer, advises, *"Social distance your elements, text blocks and everything on your darn screen. Space is everyone's friend. Use the Law of Proximity to help users visually navigate your pages. This is as much a part of your design than everything else so please mind the gap."*

- **Flexibility**

"Flexibility doesn't just follow a linear path, it's about knowing your customers and giving flexibility for different customer intents," explains Brooke Cowling, Co-founder & CMO of Digital of Things.

So from designing solutions that work great in all situations (from your grand dad's old computer to your son's latest iPad), to using shortcuts that speed up interaction for users, flexibility is a key principle of user interface design.

You not only need to make your user interface learnable for new users, but it should also have accelerators that help expert users speed up their processes. From novices to experts, ensure your product is flexible and efficient for all kinds of users.

Some examples of a flexible UI design include:

- Shortcuts for performing frequently used steps with a single click
- Advanced search features
- Incorporating filter bars
- And think multi-modal.

A multimodal user interface allows users to interact with the system through multiple modes, such as speech, text touch, vision. As a designer, you need to be mindful of all these modes. Also, design the user interface keeping in mind that your audience might use it via mobile, tablet, laptop, or a rusty old computer.

"Many users depend mostly or entirely on mobile access; optimize the UI for the strengths of each device. Be mindful that users' experiences may span across devices and modes of interaction (phone, email, mobile device, desktop, in-person) - design experiences across those modes to support their needs," shares Jon Fukuda, Co-founder and Principal at Limina, and a design and UX industry leader.

- **Accessibility**

Designing your website for all users is essential. According to the WHO, around 285 million people are visually impaired, between 110 million and 190 million adults have significant mobility difficulties and 360 million people worldwide have disabling hearing loss.

Jesse Hausler, Principal Accessibility Specialist at Salesforce, writes, "Accessibility will not force you to make a product that is ugly, boring, or cluttered. It will introduce a set of constraints to incorporate as you consider your design."

When designing products and websites, make sure that you meet the requirements of WCAG (Web Content Accessibility Guidelines).

Some easy ways to incorporate the principle of accessibility while designing products are:

- Use the WebAim Color Contrast checker to create strong color contrasts
- Ensure all your images have corresponding alt attributes

- Ask yourself if users can easily navigate your website via the tab key on your keyboard

- **User Interface Design Pattern**

User Interface Design Pattern is very important in today's world. Nowadays, it is very common and very essential to include User Interface Design Pattern while designing a program or software. These design patterns help in solving the problem as this design pattern serves as a problem solver in many conditions. Some examples are shown in the table given below:

Pattern	Meaning
Progress Indicator	In many programs, we will see this pattern during installation so that the user can feel some process or operation in process behind the screen. It is used to reassure the user that the System is not hung during installation.
Wizard	This pattern is generally used to indicate that the task is completed through various windows. It is used to indicate the user that the installation is complete.
Shopping Cart	This is generally used in online-shopping sites like amazon, flipkart, etc. It is used to provide the list of items to the user that can be purchased.

- **Widget toolkit (GUI Toolkits)**

A widget toolkit, widget library, GUI toolkit, or UX library is a library or a collection of libraries containing a set of graphical control elements (called widgets) used to construct the graphical user interface (GUI) of programs.

Most widget toolkits additionally include their own rendering engine. This engine can be specific to a certain operating system or windowing system or contain back-ends to interface with multiple ones and also with rendering APIs such as OpenGL, OpenVG, or EGL. The look and feel of the graphical control elements can be hard-coded or decoupled, allowing the graphical control elements to be themed/skinned.

Some toolkits may be used from other languages by employing language bindings. Graphical user interface builders such as e.g. Glade Interface Designer facilitate the authoring of GUIs in a WYSIWYG manner employing a user interface markup language such as in this case GtkBuilder.

The GUI of a program is commonly constructed in a cascading manner, with graphical control elements being added directly to on top of one another.

Most widget toolkits use event-driven programming as a model for interaction. The toolkit handles user events, for example when the user clicks on a button. When an event is detected, it is passed on to the application where it is dealt with. The design of those toolkits has been criticized for promoting an oversimplified model of event-action, leading programmers to create error-prone, difficult to extend and excessively complex application code. Finite state machines and hierarchical state machines have been proposed as high-level models to represent the interactive state changes for reactive programs.

- **List of widget (GUI) toolkits**

This is the list of widget toolkits (also known as GUI frameworks), used to construct the graphical user interface (GUI) of programs, organized by their relationships with various operating systems.

Low-level widget toolkits[edit]

Integrated in the operating system

- macOS uses Cocoa. Mac OS 9 and macOS used to use Carbon for 32-bit applications.
- The Windows API used in Microsoft Windows. Microsoft had the graphics functions integrated in the kernel until 2006
- The Haiku operating system uses an extended and modernised version of the Be API that was used by its predecessor BeOS. Haiku Inc. is expected to drop binary and source compatibility with BeOS at some future time, which will result in a Haiku API.

As a separate layer on top of the operating system

- The X Window System contains primitive building blocks, called Xt or "Intrinsics", but they are mostly only used by older toolkits such as: OLIT, Motif and Xaw. Most contemporary toolkits, such as GTK or Qt, bypass them and use Xlib or XCB directly.
- The Amiga OS Intuition was formerly present in the Amiga Kickstart ROM and integrated itself with a medium-high level widget library which invoked the Workbench Amiga native GUI. Since Amiga OS 2.0, Intuition.library became disk based and object oriented. Also Workbench.library and Icon.library became disk based, and could be replaced with similar third-party solutions.
- Since 2005, Microsoft has taken the graphics system out of Windows' kernel.

High-level widget toolkits[edit]

Widget toolkit comparison table

Toolkit name	Windows	macOS	Unix-like	Programming language	License
AWT	cross-platform			Java	
CEGUI	Yes	Yes	Yes	C++	MIT

Widget toolkit comparison table

Toolkit name	Windows	macOS	Unix-like	Programming language	License
Cocoa	Partial	Yes	No	Objective-C	Proprietary
Elementary	Yes	Yes	Yes	C	LGPL, BSD
FLTK	Yes	Yes	Yes	C++	LGPL
Fox toolkit	Yes	No	Yes	C++	LGPL
Fyne	cross-platform			Go	BSD
GNUstep	Yes	Yes	Yes	Objective-C	LGPL
GTK	Yes	Yes	Yes	C	LGPL
Kivy	cross-platform			Python	MIT
LCL	Yes	Yes	Yes	Object Pascal (Free Pascal)	LGPL
IUP	Yes	No	Yes	C	MIT
Juce	cross-platform			C++	GPL, proprietary
LessTif	No	No	Yes	C	LGPL
Motif	No	No	Yes	C	LGPL

Widget toolkit comparison table					
Toolkit name	Windows	macOS	Unix-like	Programming language	License
MFC	Yes	No	No	C++	Proprietary
Nana C++	Yes	No	Yes	C++	Boost license
OWL (superseded by VCL)	Yes	No	No	C++ (Borland C++)	Proprietary
Pivot (WTK)	cross-platform			Java	Apache License
Qt	cross-platform			C++	LGPL, proprietary
Rogue Wave Views	Yes	No	Yes	C++	proprietary
Shoes (GUI toolkit)	cross-platform			Ruby	MIT
Swing	cross-platform			Java	
Tk	Yes	Yes	Yes	C	BSD
TnFOX	Yes	Yes	Yes	C++	LGPL
U++	cross-platform			C++	BSD
VCL (supersedes	Yes	No	No	Object	Proprietary

Widget toolkit comparison table

Toolkit name	Windows	macOS	Unix-like	Programming language	License
OWL)	🟩	🟥	🟥	Pascal (Delphi)	
WTL	Yes	No	No	C++	Microsoft Public License
wxWidgets	cross-platform			C++	WxWindows license

OS dependent

On Amiga

- BOOPSI (Basic Object Oriented Programming System for Intuition) was introduced with OS 2.0 and enhanced Intuition with a system of classes in which every class represents a single widget or describes an interface event. This led to an evolution in which third-party developers each realised their own personal systems of classes.
- MUI: object-oriented GUI toolkit and the official toolkit for MorphOS.
- ReAction: object-oriented GUI toolkit and the official toolkit for AmigaOS.
- Zune (GUI toolkit) is an open source clone of MUI and the official toolkit for AROS.

On macOS[edit]

- Cocoa - used in macOS (see also Aqua). As a result of macOS' OPENSTEP lineage, Cocoa also supports Windows, although it is not publicly advertised as such. It is generally unavailable for use by third-party developers.[citation needed] An outdated and feature-limited open-source subset of Cocoa exists within the WebKit project, however; it is used to render Aqua natively in Safari (web browser) for Windows.[citation needed] Apple's iTunes, which supports both GDI and WPF, includes a mostly complete binary version of the framework as "Apple Application Support".[citation needed]
- Carbon - the deprecated framework used in macOS to port “classic” Mac applications and software to the macOS.
- MacApp, the framework for the Classic Mac OS by Apple.
- PowerPlant, the framework for the Classic Mac OS by Metrowerks.

On Microsoft Windows

- The Microsoft Foundation Classes (MFC), a C++ wrapper around the Windows API.
- The Windows Template Library (WTL), a template-based extension to ATL and a replacement of MFC
- The Object Windows Library (OWL), Borland's alternative to MFC.

- The Visual Component Library (VCL) is Embarcadero's toolkit used in C++Builder and Delphi. It wraps the native Windows controls, providing object-oriented classes and visual design, although also allowing access to the underlying handles and other WinAPI details if required. It was originally implemented as a successor to OWL, skipping the OWL/MFC style of UI creation, which by the mid-nineties was a dated design model.
- Windows Forms (WinForms) is Microsoft's .NET set of classes that handle GUI controls. In the cross-platform Mono implementation, it is an independent toolkit, implemented entirely in managed code (not wrapping the Windows API, which doesn't exist on other platforms).[4] WinForms' design closely mimics that of the VCL.
- The Windows Presentation Foundation (WPF) is the graphical subsystem of the .NET Framework 3.0. User interfaces can be created in WPF using any of the CLR languages (e.g. C#) or with the XML-based language XAML. Microsoft Expression Blend is a visual GUI builder for WPF.
- The Windows UI Library (WinUI) is the graphical subsystem of universal apps. User interfaces can be created in WinUI using C++ or any of the .NET languages (e.g., C#) or with the XML-based language XAML. Microsoft Expression Blend is a visual GUI builder that supports WinUI.

On Unix, under the X Window System

Note that the X Window System was originally primarily for Unix-like operating systems, but it now runs on Microsoft Windows as well using, for example, Cygwin, so some or all of these toolkits can also be used under Windows.

- Motif used in the Common Desktop Environment.
- LessTif, an open source (LGPL) version of Motif.
- MoOLIT, a bridge between the look-and-feel of OPEN LOOK and Motif
- OLIT, an Xt-based OPEN LOOK intrinsics toolkit
- Xaw, the Project Athena widget set for the X Window System.
- XView, a SunView compatible OPEN LOOK toolkit

Cross-platform

Based on C (including bindings to other languages)

- Elementary, open source (LGPL), a part of the Enlightenment Foundation Libraries, a fast, stable, and scalable library that can be used to create both rich and fast applications that can be used on anything from every day desktop computers to small PDA's and set-top boxes.
- GTK, open source (LGPL), primarily for the X Window System, ported to and emulated under other platforms; used in the GNOME, Rox, LXDE and Xfce desktop environments. The Windows port has support for native widgets.
- IUP, open source (MIT), a minimalist GUI toolkit in ANSI C for Windows, UNIX and Linux.
- Tk, open source (BSD-style), a widget set accessed from Tcl and other high-level script languages (interfaced in Python as Tkinter).
- XForms, the Forms Library for X
- XVT, Extensible Virtual Toolkit

Based on C++ (including bindings to other languages)

- CEGUI, open source (MIT License), cross-platform widget toolkit designed for game development, but also usable for applications and tool development. Supports multiple renderers and optional libraries.
- FLTK, open source (LGPL), cross-platform toolkit designed to be small and fast.
- FOX toolkit, open source (LGPL), cross-platform toolkit.
- GLUI, a very small toolkit written with the GLUT library.
- gtkmm, C++ interface for GTK
- Juce provides GUI and widget set with the same look and feel in Microsoft Windows, X Windows Systems, macOS and Android. Rendering can be based on OpenGL.
- Qt, proprietary and open source (GPL, LGPL) available under Unix and Linux (with X11 or Wayland), Windows (Desktop, CE and Phone 8), macOS, iOS, Android, BlackBerry 10 and embedded Linux; used in the KDE, Trinity, LXQt, and Lumina desktop environment, it's also used in Ubuntu's Unity shell.
- Rogue Wave Views (formerly ILOG Views) provides GUI and graphic library for Windows and the main X11 platforms.
- TnFOX, open source (LGPL), a portability toolkit.
- U++ is an Open-source application framework bundled with an IDE (BSD license), mainly created for Win32 and unix like operating system (X11) but now works with almost any operating systems.
- wxWidgets (formerly wxWindows), open source (relaxed LGPL), abstract toolkits across several platforms for C++, Python, Perl, Ruby and Haskell.
- Zinc Application Framework, cross-platform widget toolkit.

Based on Python

- Tkinter, open source (BSD) is a Python binding to the Tk GUI toolkit. Tkinter is included with standard GNU/Linux, Microsoft Windows and macOS installs of Python.
- Kivy, open source (MIT) is a modern library for rapid development of applications that make use of innovative user interfaces, such as multi-touch apps. Fully written in Python with additional speed ups in Cython.
- PySide, open source (LGPL) is a Python binding of the cross-platform GUI toolkit Qt developed by The Qt Company, as part of the Qt for Python project.
- PyQt, open source (GPL and commercial) is another Python binding of the cross-platform GUI toolkit Qt developed by Riverbank Computing.
- PyGTK, open source (LGPL) is a set of Python wrappers for the GTK graphical user interface library.
- wxPython, open source (wxWindows License) is a wrapper for the cross-platform GUI API wxWidgets for the Python programming language.
- Pyjs, open source (Apache License 2.0) is a rich web application framework for developing client-side web and desktop applications, it is a port of Google Web Toolkit (GWT) from Java.

Based on OpenGL

- Clutter (LGPL) (in C) is an open source software library for creating fast, visually rich and animated graphical user interfaces.

Based on Flash

- Adobe Flash allows creating widgets running in most web browsers and in several mobile phones.
- Adobe Flex provides high-level widgets for building web user interfaces. Flash widgets can be used in Flex.
- Flash and Flex widgets will run without a web browser in the Adobe AIR runtime environment.

Based on Go[edit]

- Fyne, open source (BSD) is inspired by the principles of Material Design to create applications that look and behave consistently across Windows, macOS, Linux, BSD, Android and iOS.

Based on XML

- GladeXML with GTK
- XAML with Silverlight or Moonlight
- XUL

Based on JavaScript[edit]

Main article: JavaScript library

General

- jQuery UI
- MooTools
- Qooxdoo Could be understood as Qt for the Web
- Script.aculo.us

RIAs

- Adobe AIR
- Dojo Toolkit
- Sencha (formerly Ext JS)
- Telerik Kendo UI
- Webix
- WinJS
- React

Full-stack framework

- Echo3
- SproutCore
- Telerik UI for ASP/PHP/JSP/Silverlight
- Vaadin - Java
- ZK - A Java Web framework for building rich Ajax and mobile applications

Resource-based

- Google Web Toolkit (GWT)

- Pyjs
- FBML Facebook Markup Language

No longer developed

- YUI (Yahoo! User Interface Library)

Based on SVG[edit]

- Raphaël is a JavaScript toolkit for SVG interfaces and animations

Based on C#[edit]

- Gtk#, C# wrappers around the underlying GTK and GNOME libraries, written in C and available on Linux, MacOS and Windows.
- QtSharp, C# wrappers around the Qt widget toolkit, which is itself based-on the C++ language.
- Windows Forms. There is an original Microsoft's implementation that is a wrapper around the Windows API and runs on windows, and Mono's alternative implementation that is cross platform.

Based on Java[edit]

- The Abstract Window Toolkit (AWT) is Sun Microsystems' original widget toolkit for Java applications. It typically uses another toolkit on each platform on which it runs.
- Swing is a richer widget toolkit supported since J2SE 1.2 as a replacement for AWT widgets. Swing is a lightweight toolkit, meaning it does not rely on native widgets.
- Apache Pivot is an open-source platform for building rich web applications in Java or any JVM-compatible language, and relies on the WTK widget toolkit.
- JavaFX and FXML.
- The Standard Widget Toolkit (SWT) is a native widget toolkit for Java that was developed as part of the Eclipse project. SWT uses a standard toolkit for the running platform (such as the Windows API, macOS Cocoa, or GTK) underneath.
- Codename One originally designed as a cross platform mobile toolkit it later expanded to support desktop applications both through JavaSE and via a JavaScript pipeline through browsers
- java-gnome provides bindings to the GTK toolkit and other libraries of the GNOME desktop environment
- Qt Jambi, the official Java binding to Qt from Trolltech. The commercial support and development has stopped[5]

Based on Object Pascal[edit]

- FireMonkey or FMX is a cross-platform widget and graphics library distributed with Delphi and C++Builder since version XE2 in 2011. It has bindings for C++ through C++Builder, and supports Windows, macOS, iOS, Android, and most recently Linux. FireMonkey supports platform-native widgets, such as a native edit control, and custom widgets that are styled to look native on a target operating system. Its graphics are GPU-accelerated and it supports styling, and mixing its own implementation controls with native system controls, which lets apps use native behaviour where it's important (for example, for IME text input.)

- IP Pascal uses a graphics library built on top of standard language constructs. Also unusual for being a procedural toolkit that is cross-platform (no callbacks or other tricks), and is completely upward compatible with standard serial input and output paradigms. Completely standard programs with serial output can be run and extended with graphical constructs.
- Lazarus LCL (for Pascal, Object Pascal and Delphi via Free Pascal compiler), a class library wrapping GTK+ 1.2–2.x, and the Windows API (Carbon, Windows CE and Qt4 support are all in development).
- fpGUI is created with the Free Pascal compiler. It doesn't rely on any large 3rdParty libraries and currently runs on Linux, Windows, Windows CE, and Mac (via X11). A Carbon (macOS) port is underway.
- CLX (Component Library for Cross-platform) was used with Borland's (now Embarcadero's) Delphi, C++ Builder, and Kylix, for producing cross-platform applications between Windows and Linux. It was based on Qt, wrapped in such a way that its programming interface was similar to that of the VCL toolkit. It is no longer maintained and distributed, and has been replaced with FireMonkey, a newer toolkit also supporting more platforms, since 2011.

Based on Objective-C[edit]

- GNUstep
- Cocoa

Based on Dart[edit]

- Flutter (software) is an open-source and cross platform framework created by Google.

Based on Swift[edit]

- Cocoa Touch is a framework created by Apple to build applications for iOS, iPadOS and tvOS.

Based on Ruby[edit]

- Shoes (GUI toolkit) is a cross platform framework for graphical user interface development.