UNIT - I

Objective:

The intention of this subject is to learn the ways of designing user-friendly interfaces or interactions. Considering which, we will learn the following −

- Ways to design and assess interactive systems.
- Ways to reduce design time through cognitive system and task models.
- Procedures and *heuristics* for interactive system design.

Introduction

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.

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.

Definition of HCI

Definition according to ACM SIGCHI (Association for Computing Machinery- Special Interest Group on Computer–Human Interaction) founded in 1982

“**Human-computer interaction is a discipline concerned with the design, implementation and evaluation of interactive computing systems for human use and with the study of major phenomenon surrounding them”**

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.

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 in the table below:

Organization Factors: Training, job design, politics, roles, work organization

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 color, 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

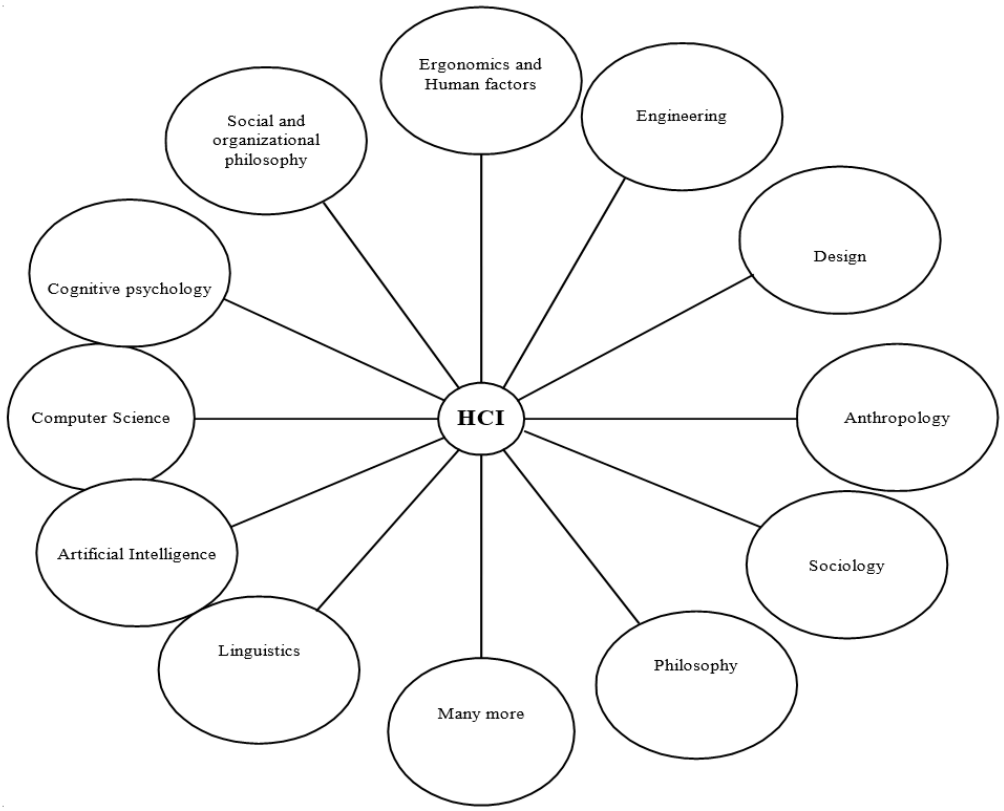
Productivity Factors: Increase output, increase quality, decrease costs, decrease errors, increase innovation

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

Cognitive Psychology: information processing, capabilities, limitations, cooperative working, performance prediction



Social Psychology: social & organizational structures

Ergonomics/Human Factors: hardware design, display readability

Linguistics: natural language interfaces

Artificial Intelligence: intelligent software

Philosophy, Sociology & Anthropology: Computer supported cooperative work (CSCW)

Engineering & Design: graphic design, engineering principles

What HCI Is and Why It Is Important

Human–computer interaction (HCI) is a cross-disciplinary area (e.g., engineering, psychology, ergonomics, and design) that deals with the theory, design, implementation, and evaluation of the ways that humans use and interact with computing devices. *Interaction* is a concept to be distinguished from another similar term, *interface*. Roughly speaking, interaction refers to an abstract model by which humans interact with the computing device for a given task, and an interface is a choice of technical realization (hardware or software) of such a given interaction model. Thus, the letter *I* in HCI refers to both interaction and interface, encompassing the abstract model and the technological methodology (Figure 1.1).

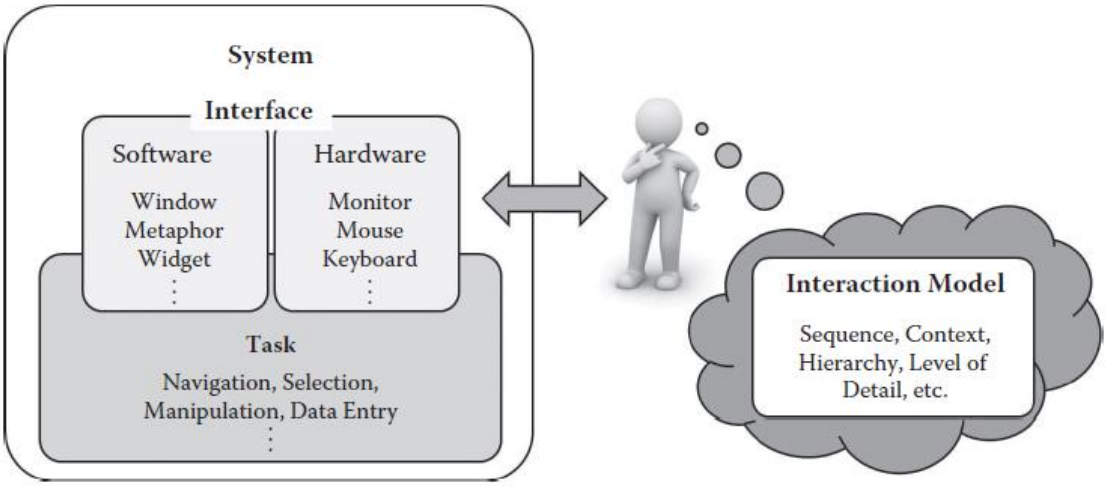


Figure 1.1 The distinguishing concepts of interaction (model) and interface.

The early focus of HCI has been in how to design interaction and implement interfaces for high usability. The term *high usability* means that the resulting interfaces are easy to use, efficient for the task, ensure safety, and lead to a correct completion of the task. Usable and efficient interaction with the computing device in turn translates to higher productivity.

Principles of HCI

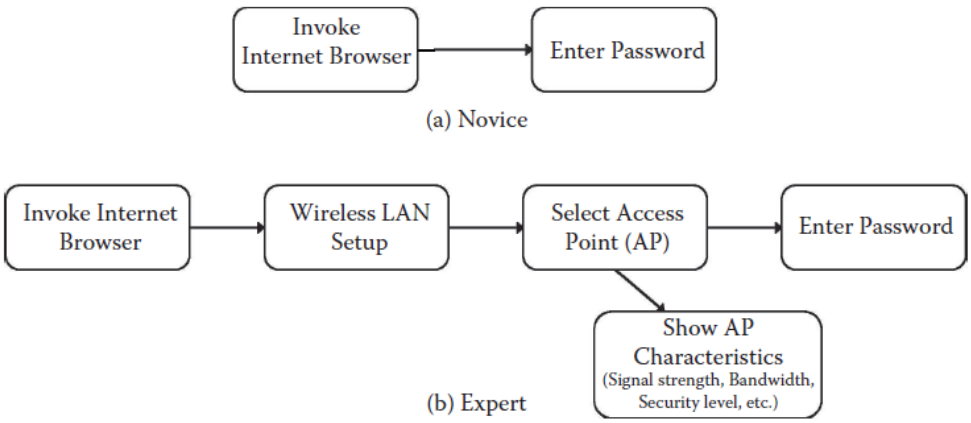
Despite its importance, good HCI design is generally difficult, mainly because it is a multi-objective task that involves simultaneous consideration of many things, such as the types of users, characteristics of the tasks, capabilities and cost of the devices, lack of objective or exact quantitative evaluation measures, and changing technologies, to name just a few. A considerable knowledge in many different fields is required. Over the relatively young history of HCI, researchers and developers in the field have accumulated and established basic principles for good HCI design in hopes of achieving some of the main objectives (as a whole) that were laid out in the previous section. These HCI principles are general, fundamental, and commonsensical, applicable to almost any HCI design situation. Here, we provide a short review of **the main HCI principles.**

"Know Thy User"

The foremost creed in HCI is to devise interaction and interfaces around the target users. This principle simply states that the interaction and interface should cater to the needs and capabilities of the target user of the system in design. However, as easy as this sounds, it is more often the case that the HCI designers and implementers proceed without a full understanding of the user, for example, by just guessing and pretending to know and be able to predict how the representative user might respond to one's design. Ideally, comprehensive information (e.g., age, gender, education level, social status, computing experience, cultural background) about the representative target user should be collected and analyzed to determine their probable preferences, tendencies, capabilities (physical and mental), and skill levels. Such information can be used to properly model interaction and pick the right interface solution for the target users.

Understand the Task

Another almost-commonsensical principle is to base HCI design on the understanding of the task. The term *task* refers to the job to be accomplished by the user through the use of the interactive system. In fact, understanding the task at hand is closely related to the interaction modeling and user analysis. It really boils down to identifying the sequence and structure of subtasks at an abstraction level appropriate for the typical user within the larger application context. Take the subtask (for a larger application) for "changing the Wi-Fi connection access point" for a smartphone. For an expert user experienced in computer networks, the task might be modeled with detailed steps, asking the user to select from a pool of available nearby access points based on their characteristics such as the signal strength, bandwidth, security level, and so forth. On the other hand, for a casual user, the subtask might only involve entering a password for the automatically selected access point (Figure).



Reduce Memory Load

Designing interaction with as little memory load as possible is a principle that also has a theoretical basis. Humans are certainly more efficient in carrying out tasks that require less memory burden, long or short term. Keeping the user's short-term memory load light is of particular importance with regard to the interface's role as a quick and easy guidance to the completion of the task. The capacity of the human's short-term memory (STM) is about 5–9 chunks of information (or items meaningful with respect to the task), famously known as the "magic number". Light memory burden also leads to less erroneous behavior. This fact is well applied to interface design, for instance, in keeping the number of menu items or depth to less than this amount to maintain good user awareness of the ongoing task or in providing reminders and status information continuously throughout the interaction.

Strive for Consistency

In the longer term, one way to unburden the memory load is to keep consistency. This applies to (a) both within an application and across different applications and (b) both the interaction model and interface implementation. For instance, the user is likely to get confused and exhibit erroneous responses if the same subtask is involved, at different times, for different interaction steps or interface methods. Note that the exact same subtasks may appear across different applications as well. Aside from being able to remember what to do, consistency and familiarity also lead to higher acceptability and preference. One way the Microsoft Windows®–based applications maintain their competitiveness is by promoting consistent and familiar interfaces.

Remind Users and Refresh Their Memory

Any significant task will involve the use of memory, so another good strategy is to employ interfaces that give continuous reminders of important information and thereby refresh the user's memory. The human memory dissipates information quite quickly, and this is especially true when switching tasks in multitasking situations (which is a very prevalent form of interaction these days). In fact, research shows that our brain internally rehearses information encoding during multitasking. Even a single task may proceed in different contextual spans. For instance, in an online shopping application, one might cycle through the entry of different types of information: item selection, delivery options, address, credit card number, number of items, etc. To maintain the user's awareness of the situation and further elicit correct responses, informative, momentary, or continuous feedback will refresh the user's memory and help the user complete the task easily.

One particular type of informative feedback (aside from the current status) is the reaffirmation of the user action to signal the closure of a larger process. An example might be not only explicitly confirming the safe receipt of a credit card number, but also signaling that the book order is complete (and "closed"). Such a closure will bring satisfaction by matching the user's mental picture of the ongoing interactive process (Figure 1.10).

Figure 1.10 Reaffirming the user's action (i.e., credit card number correctly and securely entered) and a larger interactive process (i.e., the book purchase is complete).

Prevent Errors/Reversal of Action

While supporting a quick completion of the task is important, error-free operation is equally important. As such, the interaction and interface should be designed to avoid confusion and mental overload. Naturally, all of the aforementioned principles apply here. In addition, one effective technique is to present or solicit only the relevant information/action as required at a given time. Inactive menu items are good examples of such a technique. Also, having the system require the user to choose from possibilities (e.g., menu system) is generally a safer approach than to rely on recall (e.g., direct text input) (Figure 1.11).

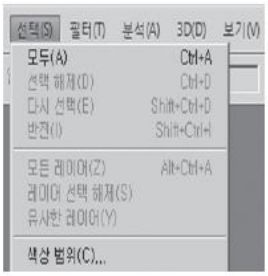


Figure 1.11 Preventing errors by presenting only the relevant information at a given time (inactive menu items) and making selections rather than enforcing recall or full manual input specification.

Despite employing some of the principles and techniques described here, there is always a chance that the user will make mistakes. Thus, a very obvious but easy-to-forget feature is to allow an easy reversal of action. This puts the user into a comfortable state and increases user satisfaction as well (Figure 1.12).

Figure 1.12 Making the user comfortable by always allowing an easy reversal of action.

Naturalness

The final major HCI principle is to favor “natural” interaction and interfaces. *Naturalness* refers to a trait that is reflective of various operations in our everyday life. For instance, a perfect HCI may one day be realized when a natural language–based conversational interface is possible, because this is the prevalent way that humans communicate. However, it can be tricky to directly translate real-life styles and modes of interaction to and for interaction with a computer. Perhaps a better approach is to model interaction “metaphorically” to the real-life counterpart, extracting the conceptual and abstract essence of the task. For instance, Figure 1.13 shows an interface called the ARCBall for rotating an object in 3-D space using a mouse (2-D device). In order to rotate, the selected object is overlaid with and enclosed by a transparent sphere, and the user drags on the surface of the sphere to rotate the object inside. One might consider this rotation technique to be metaphoric because it abstracts the interaction object into the shape of a sphere, the most rotational object we know.

A natural or metaphoric interface (assuming that the metaphor is not contrived) will also have *affordance*, a property (or additional cues) that appeals to our innate perception and cognition, thus making it so intuitive that the interface would require almost no learning. In the example of the ARCBall, the spherical shape of the rotator GUI may be regarded to exhibit a high level of affordance, requiring no explanation as to how to rotate the object.

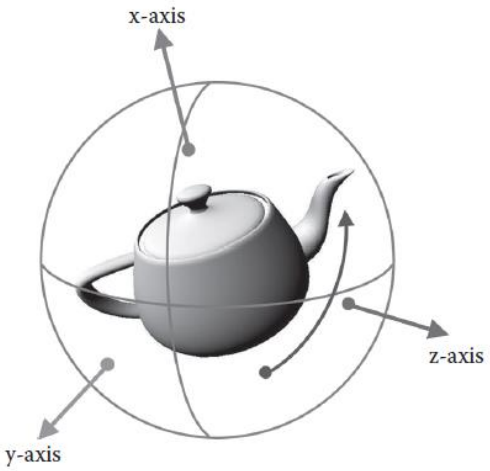


Figure 1.13 ARCBall: 3-D object rotation by using the sphere metaphor. It is also very intuitive with a high level of affordance. (From Shoemake, K., *Graphics Interface,* 92, 151–156, 1992 [7].)

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.
- **The idea of metaphor** − Xerox star and alto were the first systems to use the concept of metaphors, which led to spontaneity of the interface.

- **Direct Manipulation introduced by Ben Shneiderman (1982) −** First used in Apple Mac PC (1984) that reduced the chances for syntactic errors.
- **Vannevar Bush introduced Hypertext (1945) −** To denote the non-linear structure of text.
- **Multimodality** (late 1980s).
- **Computer Supported Cooperative Work (1990's) −** Computer mediated communication.
- **WWW (1989) −** The first graphical browser (Mosaic) came in 1993.
- **Ubiquitous Computing −** Currently the most active research area in HCI. Sensor based/context aware computing also known as pervasive computing.

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¦.

How many are actually easy, effortless, and enjoyable to use? For example, a photocopier might have buttons like these on its control panel.

Imagine that you just put your document into the photocopier and set the photocopier to make 15 copies, sorted and stapled. Then you push the big button with the "C" to start making your copies.
What do you think will happen?
(a) The photocopier makes the copies correctly.
(b) The photocopier settings are cleared and no copies are made.
If you selected (b) you are right! The "C" stands for clear, not copy. The copy button is actually the button on the left with the "line in a diamond" symbol. This symbol is widely used on photocopiers, but is of little help to someone who is unfamiliar with this.

Usability Engineering:

> "UE is an approach to the development of software and systems which involves user participation from the outset and guarantees the usefulness of the product through the use of a **usability specification** and **metrics**."

UE thus refers to the USABILITY FUNCTION aspects of the entire **process** of conceptualizing, executing & testing products (both hardware as well as software), from requirements gathering stage to installation / marketing & testing of their use.

Definition of usability

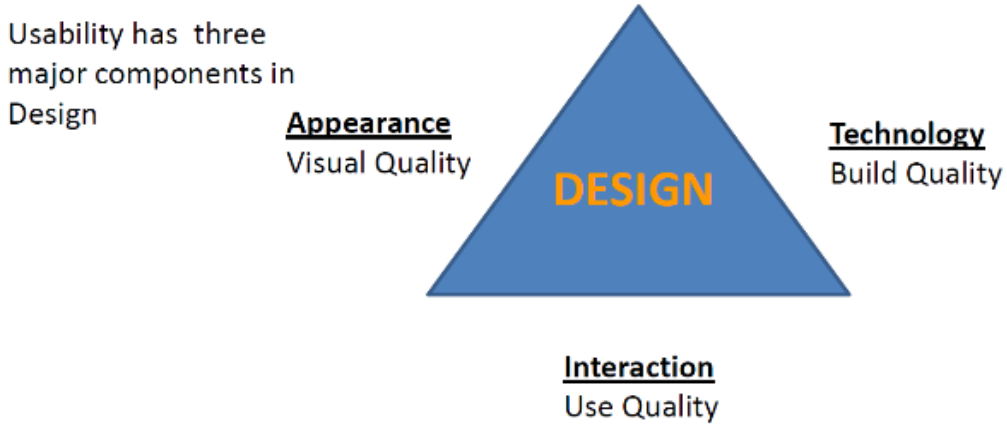
- Usability is the effectiveness, efficiency and satisfaction with which users achieve specific goals in particular environments; where
  - **Effectiveness** is the accuracy and completeness with which specified users can achieve specified goals in particular environments;
  - **Efficiency** is the resources expended in relation to the accuracy and completeness of goals achieved; and
  - **Satisfaction** is the comfort (experience) and acceptability of the work system to its users and other people affected by its use.

User's Definition of Usability

USABILITY: The ability of a User to Use the product/ system / environment as desired
Usability Engineering: The 'affordance' offered by a product that makes it useable.

- **Usability does not happen by itself. It has to be "engineered" into the product.**
- **Usability is related to Human performance:** Capabilities, Limits, and Consequences.

Usability is conceptualized into the product by DESIGN



- '**Usability'** is the measure of the *quality* of a User's experience when interacting with a product or system.
- **'Usability Engineering'** is the processes of deriving, specifying, measuring, constructing and evaluating usability features into products and systems**.**
- **Usability Study** is the systematic analysis based on heuristics and/or experimental evaluation of the interaction between people and the products including the environment of use. *Psychology/ Cognitive Sc/ Behavioral Sc.*
- **Usability Testing** is the scientific verification of the specified usability parameters with respect to the User's needs, capabilities, expectations, safety & satisfaction.
  Area where Usability is applied:
  - Usability as applied to Product Design
  - Usability as applied to Human Computer Interaction
  - Usability as applied to Human Environment Interaction
  - Usability as applied to Systems (including Engineering systems)

The UE lifecycle

SYSTEM LIFE CYCLE						
FEASIBILITY		REQUIREMENTS		DESIGN	IMPLEMENT	RELEASE
USER REQs	CONTEXT OF USE	FUNCTIONAL	TECHNICAL	PROTOTYPE	USEABILITY TESTING	FEEDBACK

Design Stages:

Task	Information produced
Knowing the user	User characteristics, User background
Knowing the task	User's current task, Task analysis
User requirements	User requirements specification
Setting usability goals	Usability specification
Design process	Design Specification
HCI Guidelines & heuristic analysis	Feedback for design iteration
Prototyping	Prototype for user testing

Evaluation with users	Feedback for freezing design
Redesign and evaluate with users	Finished product
Evaluate with users and report	Feedback on product for future systems

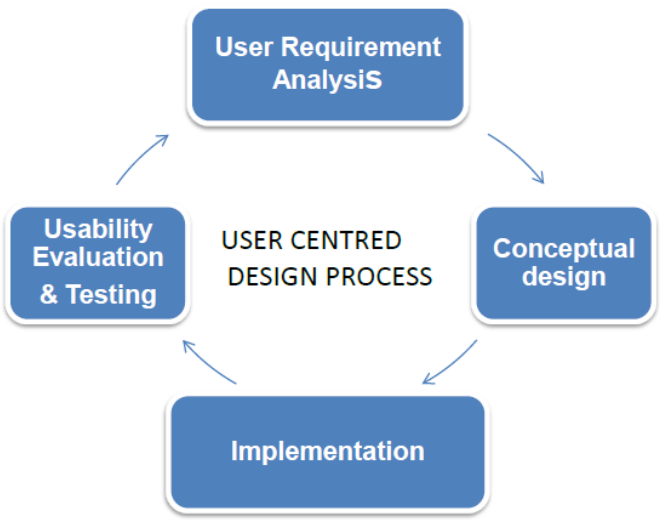
The goals of Usability Engineering

5 Es

Effective to use - Functional
Efficient to use - Efficient
Error free in use - Safe
Easy to use - Friendly
Enjoyable in use - Pleasurable Experience

Achieves 5 times Enhancement in engineering value.

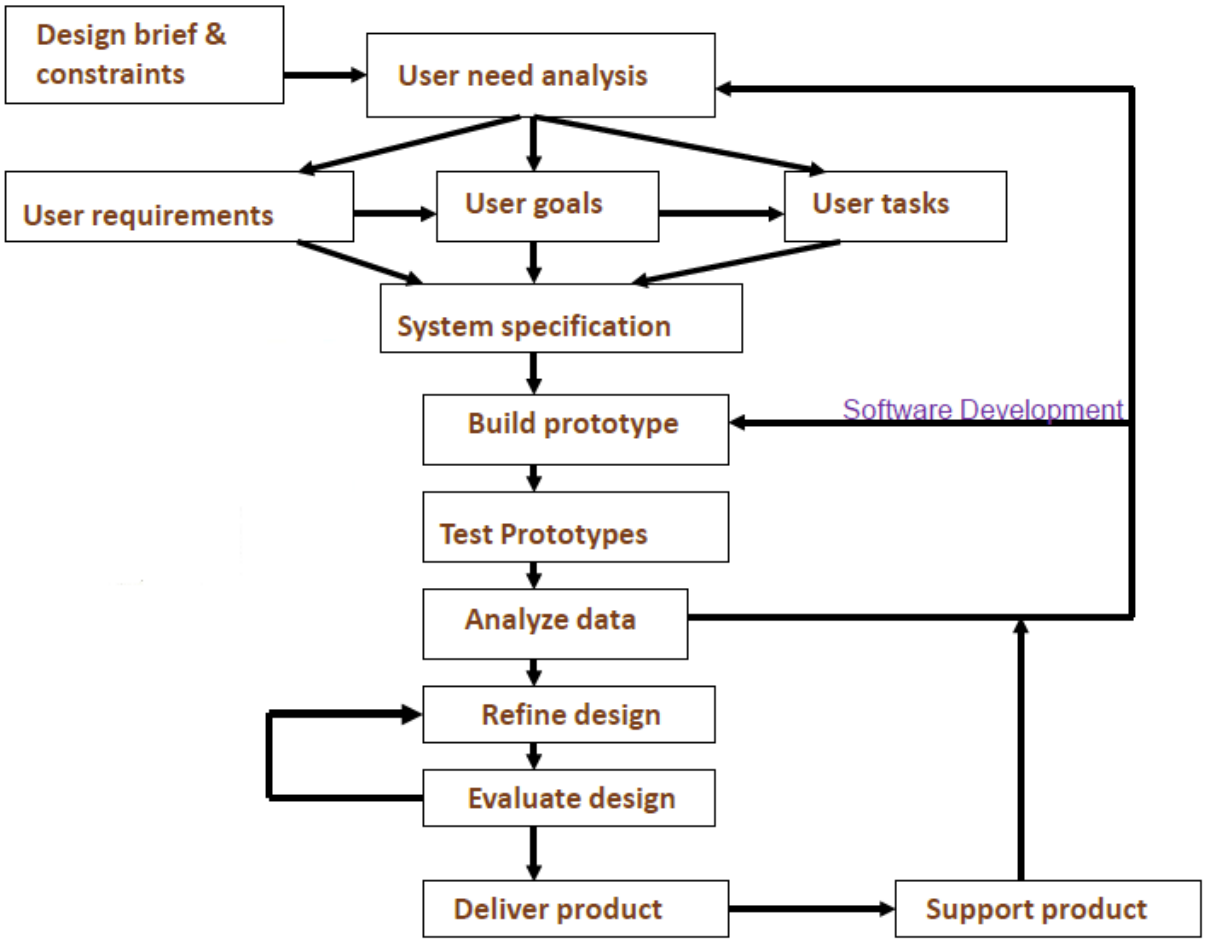
UE is based on a **User- Centered Design (UCD)** approach to analysis and design. It concentrates on those aspects of products & services that have a bearing on their effective, efficient & pleasurable USE by humans.



"Human-centered design is an approach to interactive system development that focuses specifically on making systems usable. It is a multi-disciplinary activity."

The UCD Methodology.

User centered design processes: UCD



Definition of UE & other related fields

HCI: 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. *by ACM - Association for Computing Machinery.*

Human Factors & Ergonomics: Stress on human physical issues (physiology) and on optimizing work processes.

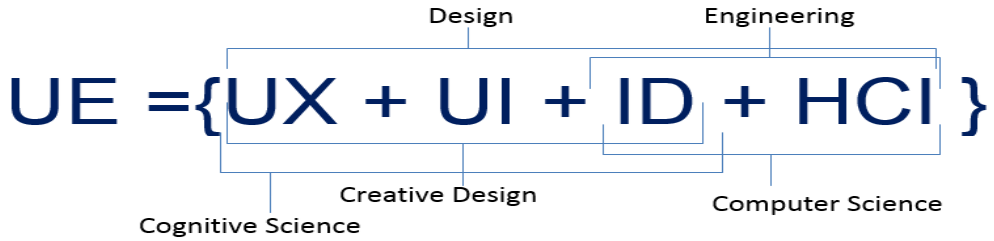
User Interface Design: Focuses on interface layer assuming all deeper functions are fixed.

HCD- Human Centered Design: Approaches to software engineering with user focus at all stages of software design

ID – Interaction Design: wider scope in terms of devices beyond computers. More emphasis on cognitive & experiential factors.

UE- Usability engineering focuses on design & implementation processes. It is essentially research & design based activity.

Relationship between UE & Human Computer Interaction; Interaction Design; Experience Design; GUI Design



Where,

UX = User Experience

UI = User Interface

ID = Interaction design

HCI= Human Computer Interaction

UE = Usability Engineering

UE vs Software Engineering

Key difference

"In most cases of the design and development of commercial software, usability is not dealt with at the same level as other aspects of SE,

e.g.

- Clear **usability objectives** are not set; and
- Resources for appropriate activities are not given priority by project management)."

Usability Testing & UE – the difference

Usability engineering: Methodical approach to producing user interface + Experience+ function + aesthetics.

A way to deliver a product that works

Usability Testing

– Part of process of UE

– Real users performing real tasks.

Usability Testing

- Analytical evaluation:
  - By simulating *how* the user's activity will be performed.
  - Heuristic evaluation measures design against a list of usability factors.

- Empirical evaluation:
  - By building and testing a *prototype*
  - Formal usability testing tests a component of the design under controlled conditions - actual users.
  - Formal usability testing requires a usability laboratory.

Cost-justifying usability

- **$1 spent on usability = $10 saved (Nielsen, 1993).** Rs. 50 spent saves Rs 500 worth of trouble shooting due to poor design
- Mobile / Tablet / Device companies now are heavily investing in UE. as the value adder as well as product differentiator.
- They do not consider 'cost' as a constraining factor as far as UE is concerned.

The UE processes is based on four fundamental axioms of Design

- User is the only constant entity of an artificially created system.
- User is the starting point of all design
- User is the final datum of reference for all design decisions
- User is the measure of all things.

PRINCIPLES TO SUPPORT USABILITY

The principles we present are first divided into three main categories:

- **Learnability** – the ease with which new users can begin effective interaction and achieve maximal performance.
- **Flexibility** – the multiplicity of ways in which the user and system exchange information.
- **Robustness** – the level of support provided to the user in determining successful achievement and assessment of goals.

In the following, we will subdivide these main categories into more specific principles that support them.

Learnability

Learnability revolves around the features of an interactive system that allow novice users to understand how to use it at first and then how to attain a maximal level of performance. Learnability makes use of various factor to familiarize a user to a new system:

- **Predictability** - It makes use of the user past knowledge of interacting with a similar system to ease the new system interaction. For example the transition from windows 7 to windows 8 had a rough path because of how much the system was changed. The start button as everyone knows ended up sending the user to the tiles windows instead of the expected menu. The notion of predictability deals with the user's ability to determine the effect of possible interaction to the operations on the system. Another form of predictability has to do with the user's ability to know which actions can be executed.
- **Synthesizability** - Synthesizability the ability of the user to assess the effect of past operations on the current state. A user expects to see any important change that is occurring while he is interacting with the system. For example, when the user makes a payment he expects to receive a receipt, otherwise he will think that the transaction failed or that he is been scammed. The synthesizability relies on

the principle of honesty of the user interface to provide an observable and informative account of occurring change.

- **Familiarity** - The principle of familiarity is to make use of the new users past experience with other applications. This experience can come from real life situation interaction to interaction with other computer system. For example a danger alert on any system is red, this immediately warns the user that there is a possible threat since red is used in everyday life as such.
- **Generalization** - Generalizability can be seen as a form of consistency. Users often try to extend their knowledge of specific interaction behavior to situations that are similar but previously unknown. Generalization can occur within a single application or across a variety of applications. For example, Microsoft office software menus act more or less the same way. This concept of generalization is what makes it easy for people to transition from Microsoft Word to Excel.
- **Consistency** - Consistency relates to the similarities in behavior arising from alike situations or alike task objectives. It can be expressed in terms of the form of input terms or output responses with respect to the meaning of actions in the conceptual of the system. For example, a user expects a radio button to allow only one choice of the available options.

Flexibility

Flexibility refers to the diversity of ways in which the user and the system exchange information. This can be achieved through categorized ways, consisting mainly of Dialog Initiative, Multithreading, Task migratability, Substitutivity and Customizability.

- **Dialog initiative** - There is two ways to achieve that. One of them is User pre-emptive. In this communication model, the user is the one to initiate an action on the system. An example, is a tourist using the map system in an airport to obtain direction to his flight fate. The other method is system pre-emptive where it is the system that initiates an action for the user to respond to. For example a system warning where the user can only click on the ok button.
- **Multithreading** - It is the ability to support more than one task of the user system dialog interaction at a time. Concurrent multi-threading allows simultaneous communication of information concerning separate tasks. Interleaved multi-threading allows a temporal overlap between separate tasks, but stipulates that at any given instant the communication is restricted to a single task.
- **Task migratability** - task migratability is the ability to transfer the control for task execution between system and user. A new system need to be able to change from a state to another that provides more advantages. Any system need to be design with the idea that it will need to be changed in the future and if an error occurs revert back to the previous. For example, some task like maintaining an airplane in the correct direction is managed by the computer system but in case of emergency, the pilot can control the direction manually.
- **Substitutivity** - It requires that equivalent values can be substituted for each other. It contributes towards flexibility of the system by letting a user choose which action best suits his needs. For example, a user can double click on an icon to open it or can select it and then press enter on the keyboard. It also need to cater to the presentation of output in different ways. For example, a temperature can be

displayed by a graph if it needs to capture a trend or it can be simply displayed for the user to see at a particular time.

- **Customizability** - It refers to the modifiability of the user interface by the user or the system. The term “adaptability” is used when the interface can be modified by the user and the term “adaptivity” is used when it can be modified by the system. Under adaptability, the principle of lexical customization is applied to limit what the user can change. It can be that the user is only allowed to change the color scheme of the interface.

Robustness

Robustness concerns itself with supporting the user in successfully accomplishing an action with the system and assessment of the action. There are various principles to be applied to support a system robustness: observability, Recoverability, Responsiveness, Task Conformance

- **Observability** - Observability allows the user to evaluate the internal state of the system by means of its perceivable representation at the interface. It is further broken down into five other principles:
    - Browsability: Allows the user to explore the current internal state of the system without modifying it.
    - Default: Assist the user by passive recall, such as suggesting the user possible words based on his text input.
    - Reachability: Allows the user to navigate through observable states.
    - Persistence: This principle deals with the duration of an observable state.
    - task performance: It covers the extent to which a system services support all of the tasks the user wishes to perform and in a way that the user understands them
- **Recoverability** - It is the ability of a system to recover in case of an error. There are two directions in which recovery can occur, forward or backward. Forward error recovery accepts that an error has occurred in the current state and negotiation from that state towards the desired state. This situation is encountered when there is a disk failure and windows OS is trying to repair it. Backward error recovery is attempting to undo the effects of previous interaction in order to return to a state prior to the error occurrence.
- **Responsiveness** - Responsiveness deals with the time needed for the system to communicate with the user. In general, short durations and instantaneous response times are desirable. Moreover, the response time need to be stable. Response time stability covers the invariance of the duration for identical or similar computational resources. For example, the user expects the same time response as clicking on a drop down menu as clicking on a radio button. Variations in response time will impede anticipation of the user.
- **Task conformance** - Task conference ensure that the system allows a user to perform task he needs and in an expecting way. Task completeness covers whether a system can perform all tasks of interest. Task adequacy deals with the user ability to understand the tasks. It is important that a system allows the user to perform any desired task in an application as specified in the system specification prior to the delivery of the system.

HCI and Software Engineering

Placing the design of interactive systems within the established frameworks of software development.
Software engineering: addresses the management and technical issues of the development of software systems. One of the cornerstones of software engineering is the *software life cycle*, which describes the activities that take place from the initial concept formation for a software system up until its eventual phasing out and replacement.

The important point that we would like to draw out is that issues from HCI affecting the usability of interactive systems are relevant within all the activities of the software life cycle. Therefore, software engineering for interactive system design is not simply a matter of adding one more activity that slots in nicely with the existing activities in the life cycle.

The Central Idea
Suppose you are designing a database management system (DBMS): what are your design objectives

- Efficient storage of large databases (storage)
- Efficient way to retrieve results of a query from database (retrieval)
- Allowing the user to access the database (interaction)

Note that this is a scenario where the user interacts with the system (database)
However, the user is a "computer expert", who has "technical knowledge" about the system
Through some query language, the user can access, manipulate and update the database

Example: Tourist information system (Usability Purpose)

- In the background, it is nothing but a database of various tourist-related information
- However, its users may or may not be "computer experts"
- They do not care about what goes on inside
- They just want to "get" the information "easily"
- The term "easily" is very significant
- It means, we need to have an interface and interaction mechanism that do not require any specialized knowledge
- That is, we need a "usable" system
- Design goal of an interactive system: increase usability

Software engineering.

What Happens in Software Engineering
A fundamental feature of software engineering, therefore, is that it provides the structure for applying techniques to develop software systems. The software life cycle is an attempt to identify the activities that occur in software development. These activities must then be ordered in time in any development project and appropriate techniques must be adopted to carry them through.

In the development of a software product, we consider two main parties: the customer who requires the use of the product and the designer who must provide the product.

- **The waterfall model: the simplest and typical way to visualize software design**
- **Design process composed of a series of sub-stages**
- **Each sub-stage follows the previous stage and precedes the next stage (looks like a waterfall).**

Requirements specification

In requirements specification, the designer and customer try to capture a description of *what* the eventual system will be expected to provide. Requirements specification involves eliciting information from the customer about the work environment, or domain, in which the final product will function. Requirements specification begins at the start of product development. Though the requirements are from the customer's perspective, if they are to be met by the software product they must be formulated in a language suitable for implementation. Requirements are usually initially expressed in the native language of the customer.

Architectural design

The requirements specification concentrates on what the system is supposed to do. The next activities concentrate on *how* the system provides the services expected from it. The first activity is a high-level decomposition of the system into components that can either be brought in from existing software products or be developed from scratch independently. An architectural design performs this decomposition. It is not only concerned with the functional decomposition of the system, determining which components provide which services. It must also describe the interdependencies between separate components and the sharing of resources that will arise between components.

The Waterfall Model

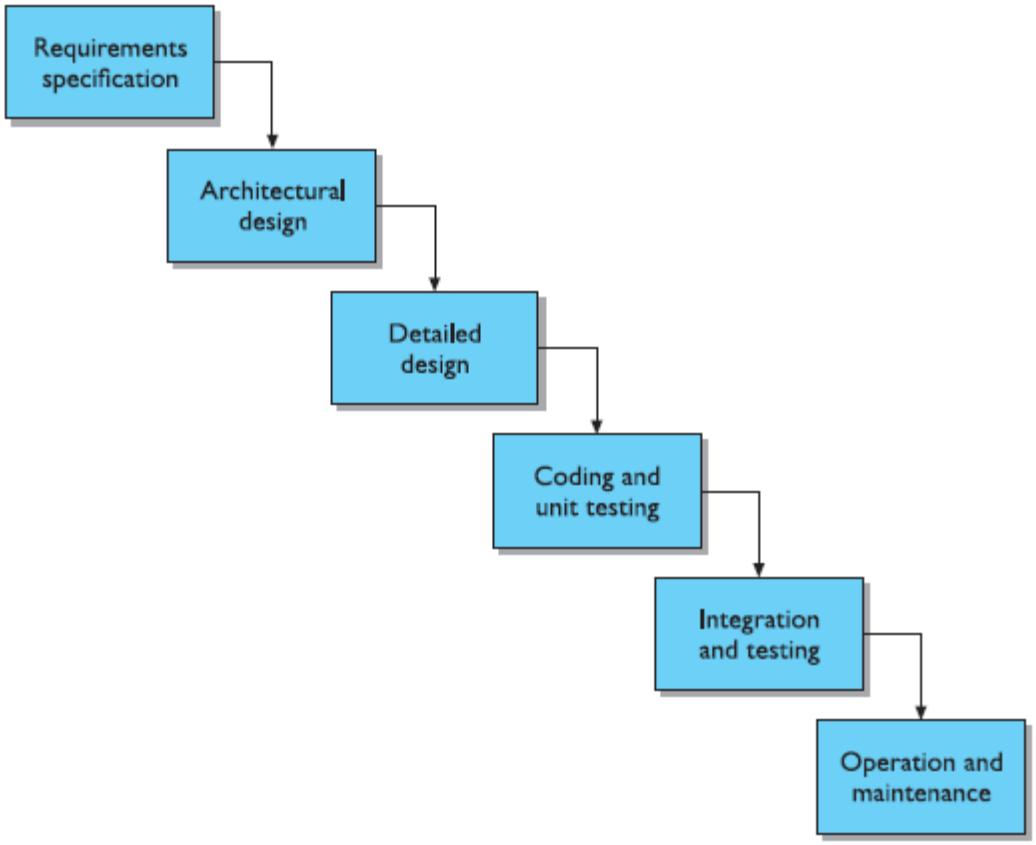


Figure: The activities in the waterfall model of the software life cycle

Detailed design

The architectural design provides a decomposition of the system description that allows for isolated development of separate components which will later be integrated. The detailed design is a refinement of the component description provided by the architectural design. The behavior implied by the higher-level description must be preserved in the more detailed description.

Coding and unit testing

With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and testing

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Maintenance

After product release, all work on the system is considered under the category of maintenance, until such time as a new version of the product demands a total redesign or the product is phased out entirely. Maintenance involves the correction of errors in the system which are discovered after release and the revision of the system services to satisfy requirements that were not realized during previous development. Therefore, maintenance provides feedback to all of the other activities in the life cycle.

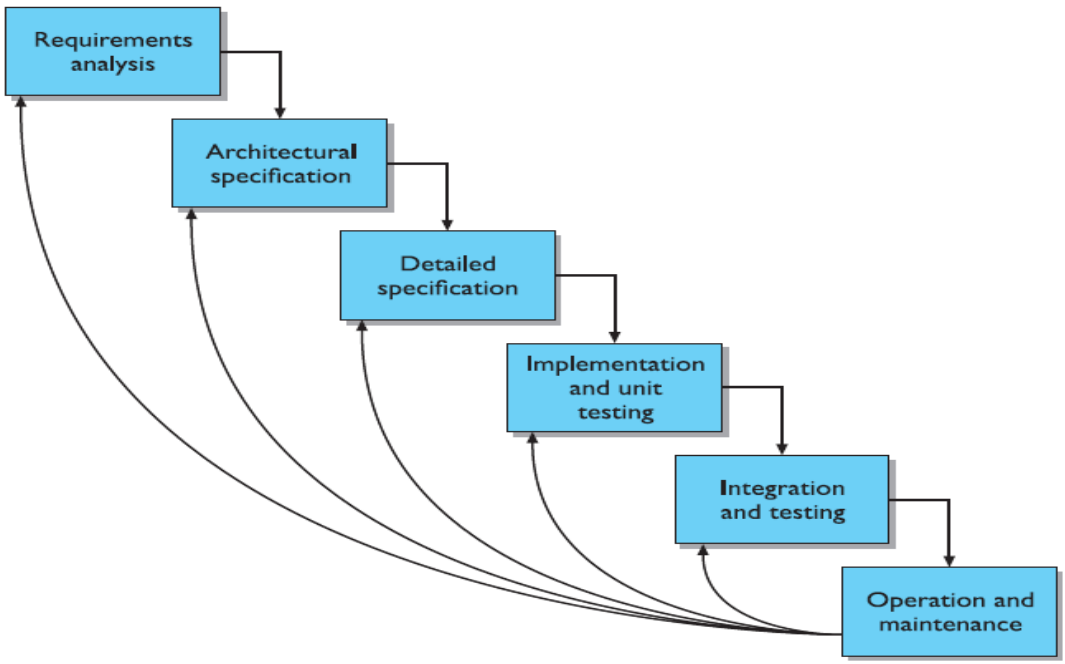


Figure Feedback from maintenance activity to other design activities

Interactive systems and the software life cycle

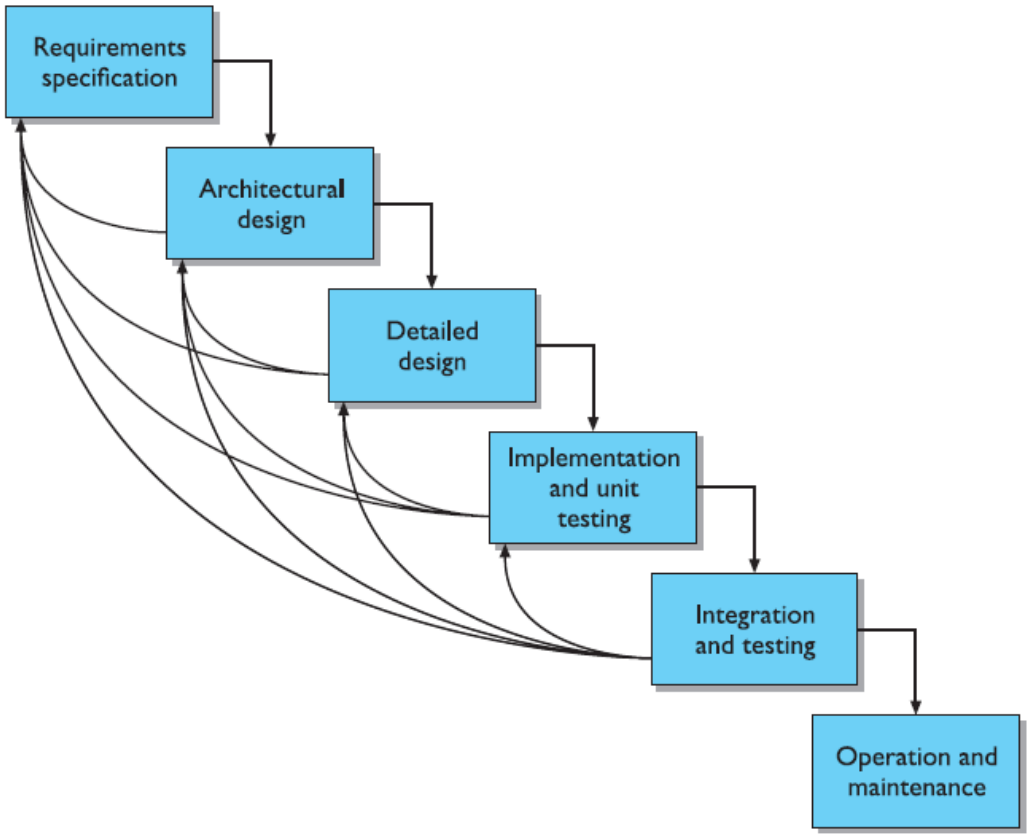


Figure Representing iteration in the waterfall model

Interactive System Design

- We are trying to design “for” the user not for a programmer’s convenience or expert’s use
- We should consider
  - Abilities and needs of the users
  - Their usage context
  - Their work setting
- In other words, we have to “know the user”

Need: Know the User

- A never ending process because there is so much to know and because the users keep changing
- An interactive system designer should consider the human factors that characterize users
- This knowledge cannot be captured at once

These factors (user characteristics) vary with

- Age, gender, physical and cognitive abilities, personality
- Education, cultural or ethnic background
- Training, motivation, goals

The Human Factors

- **Perception**: our ability to perceive our surroundings
  –Can be visual, auditory or haptic (touch)
- **Cognition**: the way we process the perceived information in our "mind" and take decisions
- **Motor action:** this is the mechanism through which we interact with the surrounding
  –Example: hand movement, eyeball movement, speech

An interactive system designer should recognize this diversity

Need: Recognize Diversity

- **Systems used by several communities of users**
  –No single design can satisfy all users and situations
- **Designer faces real challenge to cater to the need of each community**
  –Designers must characterize users and situations as precisely and completely as possible

A Generic User Characterization

- **Novice or first time users**
  –Know nothing about the task or interface concepts
  –Often anxious about the computer and its functionality
- **Knowledgeable or intermediate users**
  –Have stable task concepts and broad interface concepts
- **Expert users**
  –Thoroughly familiar with the task and interface concepts
  –Want to get their job done quickly

Design involves acquiring new knowledge and using it to refine design in continuous cycle (till some "acceptable" design is found)

–The reason for so many "feedbacks" in the waterfall model

User Centered Design (UCD)

The design process, where designer collects feedback about the design from users and use this to refine design, is known as "user centered design" or UCD

UCD is based on understanding the domain of work or play in which people are engaged and in which they interact with computers

Assumptions

Result of a good design is a satisfied user

Process of design is a collaboration between designers and user.

Design evolves and adapts to users' changing concerns, and the process produces a specification as an important by product

The user and designer are in constant communication during the entire process

UCD Drawbacks

- **In UCD, user involvement is "passive"**

–The designer elicits feedback from user (through interviews, informal discussions etc)

–Prepares specification on the basis of user response

–Take feedback on the design and makes refinements

- **Problems with "passive" involvement of user**

–User intuition about a new interface may not be correct (feedback not reliable)

–The interview process itself may not be formulated properly (designer asks wrong questions)

–It is not possible for the designer to identify all possible issues to take feedback from users, as the designer's knowledge about the user may not be complete.

Participatory Design

• **Solution:** make (representative) users a part of the design team such a design process, where end users are part of the design team, is known as "participatory design".

Participatory Design: Key Points

Users are first-class members of the design team

– As opposed to their passive involvement in UCD

Users are considered subject experts

–Know all about their work context

Iterative design process

–All design stages are subject to revision

Interactive System Design Life Cycle (ISLC)

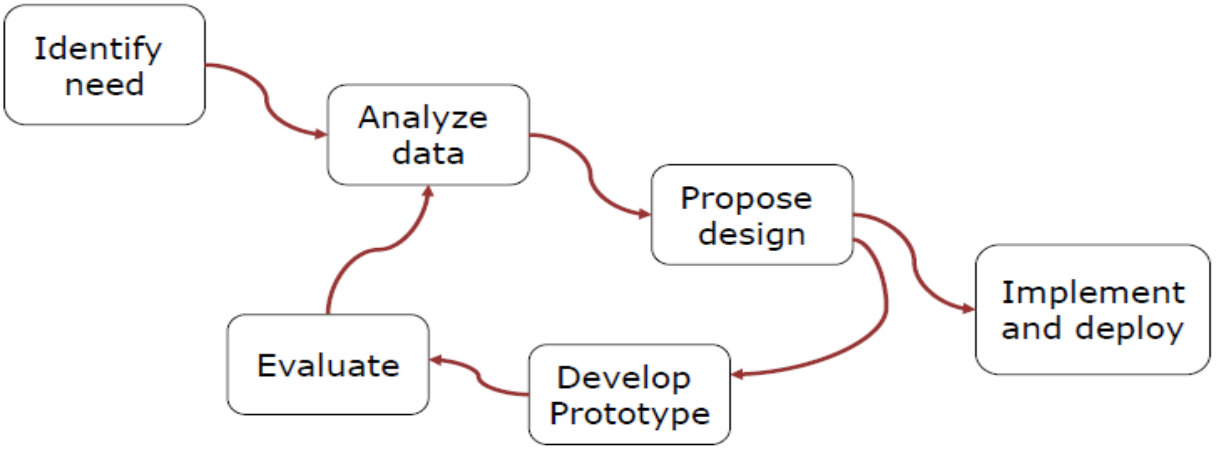
Key stages

–Know the user, propose design, evaluate design by users and refine design

Iterative design process

–The above stages are iterated till an acceptable (determined from user feedback) solution is obtained

ISLC: A Consolidated Diagram



Life Cycle Stage: Identify Need

What is wanted –identify users and their need. The first stage is establishing what exactly is needed. As a precursor to this it is usually necessary to find out what is currently happening. For example, how do people currently watch movies? What sort of personal appliances do they currently use? There are a number of techniques used for this in HCI: interviewing people, videotaping them, looking at the documents and objects that they work with, observing them directly.

Designers make use of one or more methods to identify the requirements such methods include

–Interview (structured, semi-structured, unstructured)

–Contextual inquiry (Contextual inquiry is a semi-structured interview method to obtain information about the context of use, where users are first asked a set of standard questions and then observed and questioned while they work in their own environments)

–Cultural probes

–Ethnography (the scientific description of peoples and cultures with their customs, habits, and mutual differences)

–User models

Life Cycle Stage: Analyze Data

The results of observation and interview need to be ordered in some way to bring out key issues and communicate with later stages of design. To capture how people carry out the various tasks that are part of their work and life. These techniques can be used both to represent the situation as it is and also the desired situation.

Types of analysis are performed

- **Scenario analysis:** analyze data collected from the user on one or more usage scenario of the system
- **Task analysis**: analyze tasks required to be carried out by the user to operate the system
- **System level task analysis**: analysis of external tasks required to operate the system
- **Cognitive task analysis**: analysis of tasks performed in the mind of the user

Life Cycle Stage: Propose Design

Design proposal arrived at from the analysis of collected data There are numerous rules, guidelines and design principles that can be used to help with this and how to design taking into account many different kinds of user. We need to record our design choices in some way and there are various notations and methods to do this, including those used to record the existing situation. Simple notations for designing navigation within a system and some basic heuristics to guide the design of that navigation and will look more closely at the layout of individual screens. It is at this stage also where input from theoretical work is most helpful, including cognitive models, organizational issues and understanding communication.

Life Cycle Stage: Develop Prototype

Implement a prototype of the design for collecting user feedback. Produce a developmental version to check that designers ideas meet customers' requirements, and to try out novel concepts to see if they work, and so on. Prototypes are not necessarily functional, particularly if they are trying out new interface/interaction ideas - they can be mocked up, perhaps first on paper, then on the machine, before ever being attached to pieces of code.

A spectrum of techniques is used in developing prototypes

–Paper prototype (one extreme)

–Complete software (other extreme)

Life Cycle Stage: Evaluate Design

•Evaluation of the design by users

•In the initial design phase, evaluation is done on prototypes

–Cost effective and easier to perform

–Suitable for iterative design process where the evaluation is performed many times

•Evaluation of the design by users

Prototypes and near-working systems, in alpha or beta release, should be carefully evaluated to see if they meet the clients requirements and are easy, intuitive and sensible to use. It is often the case that prospective users are very different to the actual designers and so find certain things particularly difficult with the current system, and the aim of evaluating the system at this stage is to catch these errors.

The full system is typically evaluated at the end
–Full system evaluation is costly in terms of money, manpower, time and effort
–Hence, typically done once or a limited number of times
Several evaluation methods are available
–Checklist/guideline based evaluation
Heuristic evaluation, cognitive walkthrough
–Model-based evaluation: employs models (of the system or user or hybrid) for evaluation
Hybrid models are essentially models that combines the features of both the system and the user
–Empirical evaluation –evaluate with real users
Involve implementation of the system with full functionalities

GUI Design & Aesthetics

GUI: Graphic User Interface

The interface through which a user operates a program or an application or an device

Consists of individual or group of ICONS, buttons, scroll bars, menus, widgets, boxes, status lamps, labels, instructions, and visuals etc.-arranged on the screen in a pattern that is visually pleasing as well as ergonomically useable.

Very important and critical component in facilitating user interaction with the software & hardware inside the device / product.

GUI determines the Usability Index of the product as a whole. Gives the product an identity, personality & character.

Requirements of a GUI Should be

FUNCTIONAL:

Useable -Easy to operate; locate what is required & where it is required on the screen; and do what is expected of it –without need for learning or training

AESTHETIC:

Pleasing to the eye; Highest Visual Quality; Identifiable; Distinct; Recognizable, Recallable

COMMUNICABLE:

Express what it represents; how it is to be operated; Unambiguous; Meaningful; Culturally & Contextually compatible

In GUI Design Aesthetics is about

Sensory + Empirical + Taste + Judgment

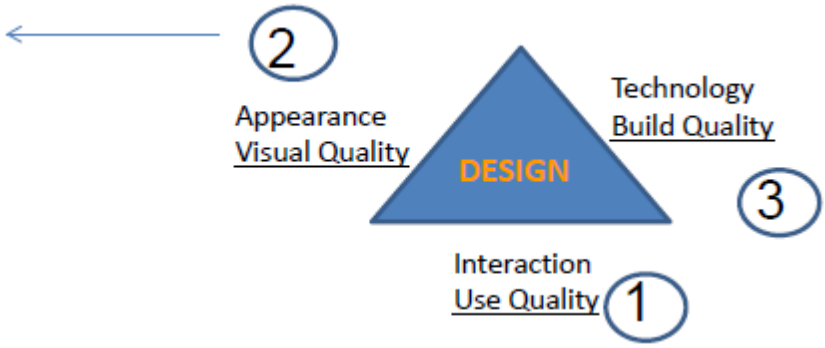
The Philosophical argument of aesthetics shown below is incorporated into Interfaces through Graphic designing

Simplicity + Infinity + Eternity + Serenity = Beauty



Aesthetics is both Art as well as Mathematics. It is both rational as well as emotional at the same time.

Aesthetics is a medium for User Experience



Aesthetics (Look & Feel) + Communication + Use ability = Total UI Experience

Role of Aesthetics –often misunderstood & underestimated

•Aesthetics is not mere beautification.

•It has as much to do with FUNCTION as with beauty

•Aesthetics is not the surface characteristics of a GUI It is not decoration. It is not cosmetic

•A 'good looking' GUI needs also -to function

-to communicate

-to express

-to instruct

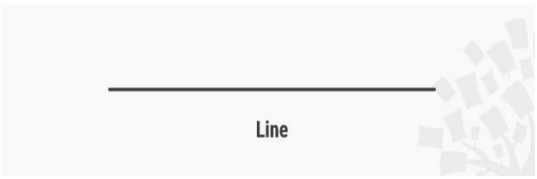
-to perform

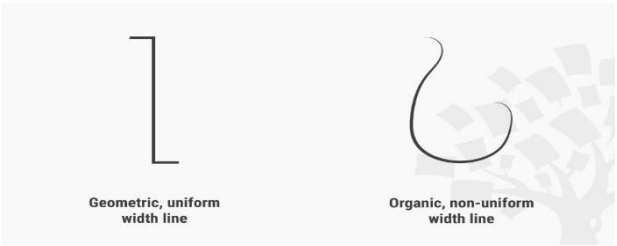
There are elements & principles of good aesthetic configuration

ELEMENTS OF DESIGN:

Line

Lines are strokes connecting two points, and the most basic element of visual design. We can use them to create shapes, and when we repeat them, we can form patterns that create textures.

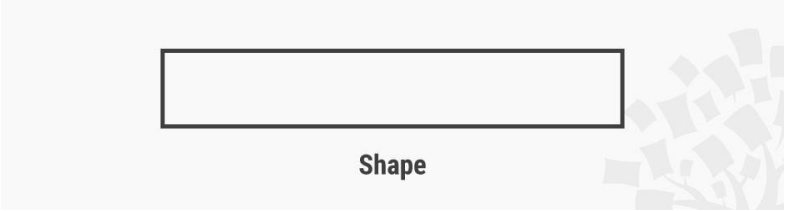




Although simple, lines can possess a large variety of properties that allow us to convey a range of expressions. For example, lines can be thick or thin, straight or curved, have uniform width or taper off, be geometric (i.e., look like they are drawn by a ruler or compass) or organic (i.e., look like they are drawn by hand).

Shape

Shapes are self-contained areas, usually formed by lines (although they may also be formed by using a different color, value or texture). A shape has two dimensions: length and width.



We tend to identify objects by their basic shapes, and only focus on the details (such as lines, values, colors and textures) on closer inspection. For this reason, shapes are crucial elements that we designers use for quick and effective communication.

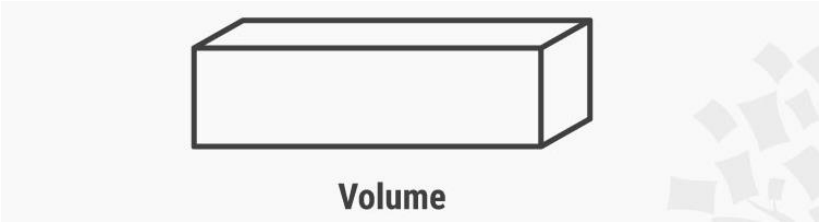
Negative/White Space

Negative space (also known as white space) is the empty area around a (positive) shape. The relation between the shape and the space is called *figure/ground*, where the shape is the figure and the area around the shape is the ground. We should be aware that when designing positive shapes, we are also designing negative spaces at the same time. **Negative space** is just as important as the positive shape itself — because it helps to define the boundaries of the positive space and brings balance to a composition.

Some designs make use of negative space to create interesting visual effects. For example, the famous World Wide Fund for Nature (WWF) logo makes use of the confusion between positive shape and negative space to create the image of a panda.

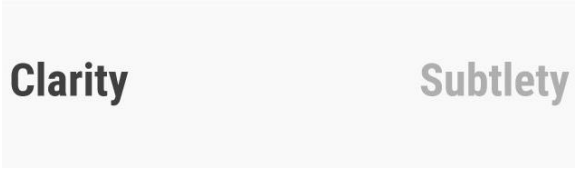
Volume

Volume applies to visuals that are three-dimensional and have length, width and depth. We rarely use volume in visual design, because most digital products end up being viewed on a 2D screen, although some apps and websites do use 3D models and graphics. (Technically, though, 3D images viewed on a 2D screen are still 2D images.)



Value

Value, quite simply, describes light and dark.



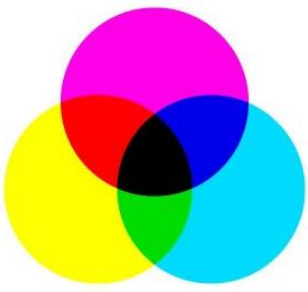
A design with a high contrast of values (i.e., one which makes use of light and dark values) creates a sense of *clarity*, while a design with similar values creates a sense of *subtlety*. We can also use value to simulate volume in 2D, for instance, by using lighter values where the light hits the object and darker values for shadows.

Color

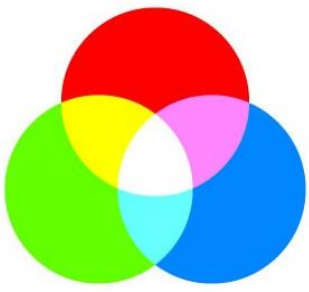
Color is an element of light. Color theory is a branch of design focused on the mixing and usage of different colors in design and art. In color theory, an important distinction exists between colors that mix subtractively and colors that mix additively.

In paint, colors mix *subtractively* because the pigments in paints absorb light. When different pigments are mixed together, the mixture absorbs a wider range of light, resulting in a darker color. A subtractive mix of cyan, magenta and yellow will result in a black color. A subtractive mix of colors in paint and print produces the CMYK (i.e., **C**yan, **M**agenta, **Y**ellow and blac**k**) color system.

In digital design, where the product shows up on a screen, colors mix *additively*, since the screen emits light and colors add to one another accordingly. When different colors are mixed together on a screen, the mixture emits a wider range of light, resulting in a lighter color. An additive mix of red, blue and green colors on screens will produce white light. An additive mix of colors on digital screens produces the RGB (i.e., **R**ed, **G**reen, **B**lue) color system.



Subtractive mix
(in print)

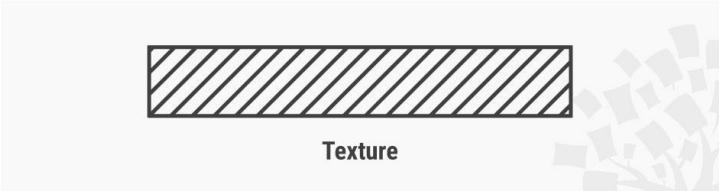


Additive mix
(on screen)

We use colors in visual design to convey emotions in and add variety and interest to our designs, separate distinct areas of a page, and differentiate our work from the competition.

Texture

Texture is the surface quality of an object.



As a designer, you can work with two types of textures: *tactile* textures, where you can feel the texture, and *implied* textures, where you can only see — i.e., not feel — the texture. Most visual designers will work with implied textures, since screens (at least as far as the state of the art had pushed them by the mid-2010s) are unable to produce tactile textures.

Color

Color is the result of light reflecting back from an object to our eyes. The color that our eyes perceive is determined by the pigment of the object itself. Color theory and the color wheel are often referred to when studying color combinations in visual design. Color is often deemed to be an important element of design as it is a universal language which presents the countless possibilities of visual communication.

Graphic Designers use metrics to specify colors.

Hue: refers to the names of the primary colors. (Red, green and blue).

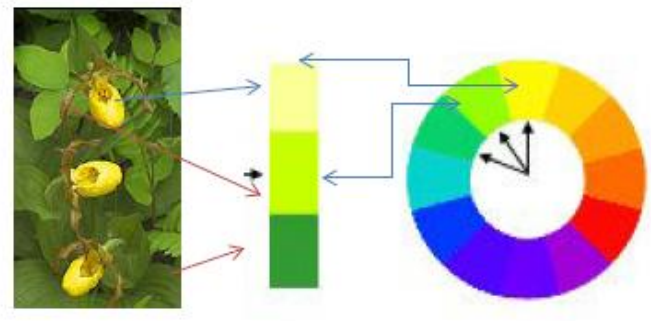
Value: lightness and darkness of the hue –

Shade: amount of white or black added.

Intensity: the purity or saturation of the color

Monochromatic: use of one color where only the value of the color changes

Analogous colors: colors that are adjacent to each other on the color wheel, e.g. yellow and green are analogous.



Limitations of Technology

The Visible spectrum consists of billions of colors, a computer monitor can display millions, a high quality printer is only capable of producing thousands, and older computer systems may be limited to 216 cross-platform colors.

The Psychology of Colors

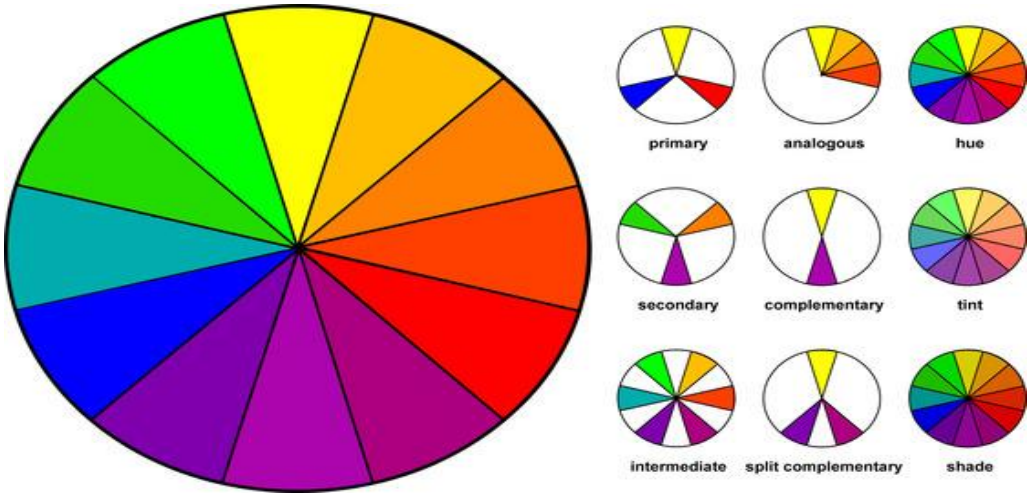
WARM colors include: yellows, red and orange we associate these with blood, sun and fire.

COOL colors include: violet, blue and green because of our association with sky, water.

Color Theme

Choices of color given to the User. Simple Pick & Chose does not confuse the user with 100s of colors to choose from.

A set of colors are carefully decided upon by a designer which form a ‘theme’ .All screens will have visual elements from the theme.



Point

A Point is basically the beginning of “something” in “nothing”. It forces the mind to think upon its position and gives something to build upon in both imagination and space. Some abstract points in a group can provoke humanagination to link it with familiar shapes or forms

Pattern

Many textures appear to repeat the same motif. When a motif is repeated over and over again in a surface, it results in a pattern. Patterns are frequently used in fashion design or textile design, where motifs are repeated to create decorative patterns on fabric or other textile materials. Patterns are also used in architectural design, where decorative structural elements such as windows, columns, or pediments, are incorporated into building design.

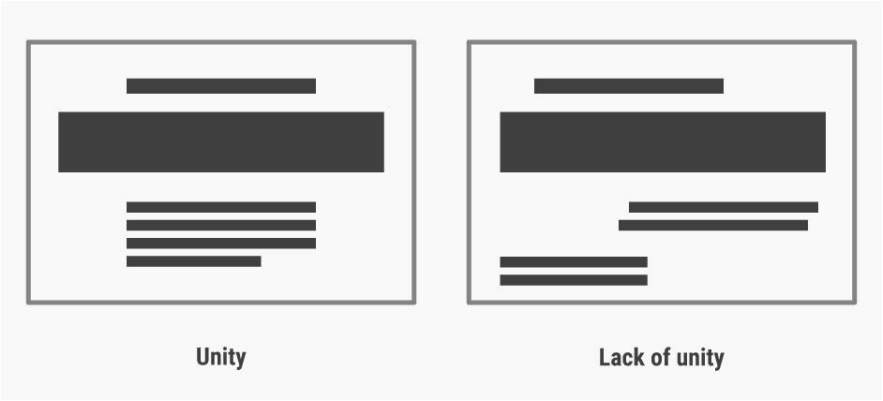
Form

In visual design, form is described as the way an artist arranges elements in the entirety of a composition. It may also be described as any three-dimensional object. Form can be measured, from top to bottom (height), side to side (width), and from back to front (depth). Form is also defined by light and dark. It can be defined by the presence of shadows on surfaces or faces of an object. There are two types of form, geometric (artificial) and natural (organic form). Form may be created by the combining of two or more shapes. It may be enhanced by tone, texture or color. It can be illustrated or constructed.

PRINCIPLES OF DESIGN

Unity

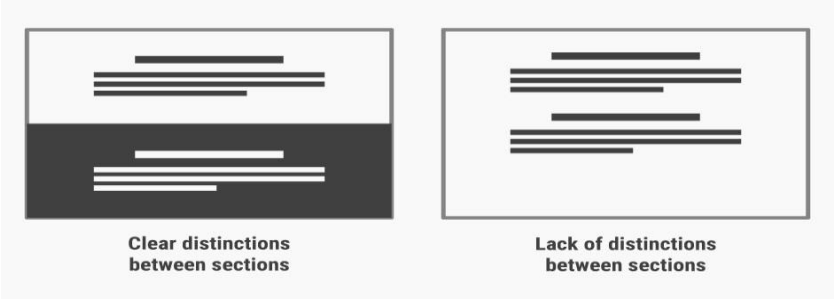
Unity has to do with creating a sense of harmony between all elements in a page. A page with elements that are visually or conceptually arranged together will likely create a sense of unity.



When we're designing websites, we can make use of a grid for achieving a sense of unity, since elements organised in a grid will follow an orderly arrangement. We do need, however, to introduce some *variety* in our work in order to strike a balance between a boring and a chaotic design.

Gestalt

Gestalt refers to our tendency to perceive the *sum* of all parts as opposed to the individual elements. The human eye and brain perceive a unified shape in a different way to the way they perceive the individual parts of such shapes. In particular, we tend to perceive the overall shape of an object first, before perceiving the details (lines, textures, etc.) of the object.

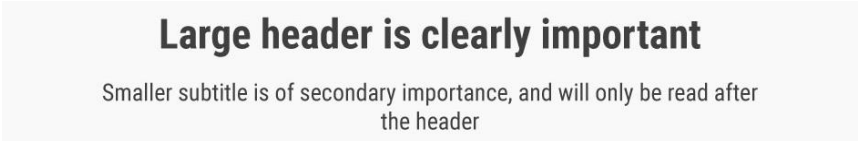


Gestalt is the reason that we can see a square, circle and triangle even though the lines are not complete. We see the whole formed by the dotted lines first, before perceiving the separate dotted lines in each of the images.

Gestalt is important, for instance, in making separate sections of a website distinct by increasing the white space between them. As designers, we should make sure that the parts of a website we group together by using gestalt principles i.e., if they are close to one another, have the same shape, and/or are similarly sized are indeed conceptually grouped together. "Accidentally" grouping elements which are not conceptually similar will result in confused users.

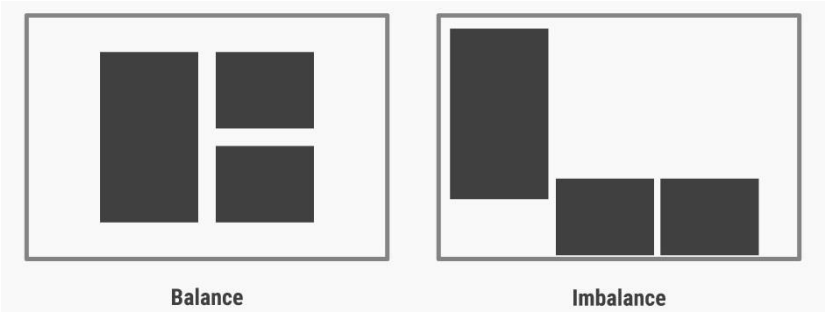
Hierarchy

Hierarchy shows the *difference in importance* of the elements in a design. Colour and size are the most common ways we can create hierarchy — for instance, by highlighting a primary button, or using larger fonts for headings. Items that appear at the top of a page or app also tend to be viewed as having a higher hierarchy than those appearing below.



Balance

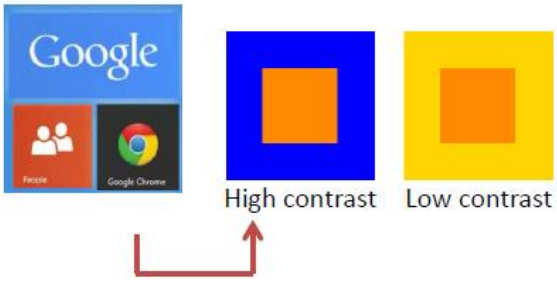
Balance is the principle governing how we distribute the elements of a design *evenly*. Balanced designs tend to appear calm, stable and natural, while imbalanced designs make us feel uneasy.



Balance can be achieved by having symmetry in the design (for instance, having a webpage with centralised text and images). However, you can also achieve balance *without* symmetry — perhaps unsurprisingly, this is known as asymmetrical balance. We achieve asymmetrical balance when we arrange differently sized elements in a way that results in unity. We can imagine a centre point of the design and distribute the elements in a way that creates balance.

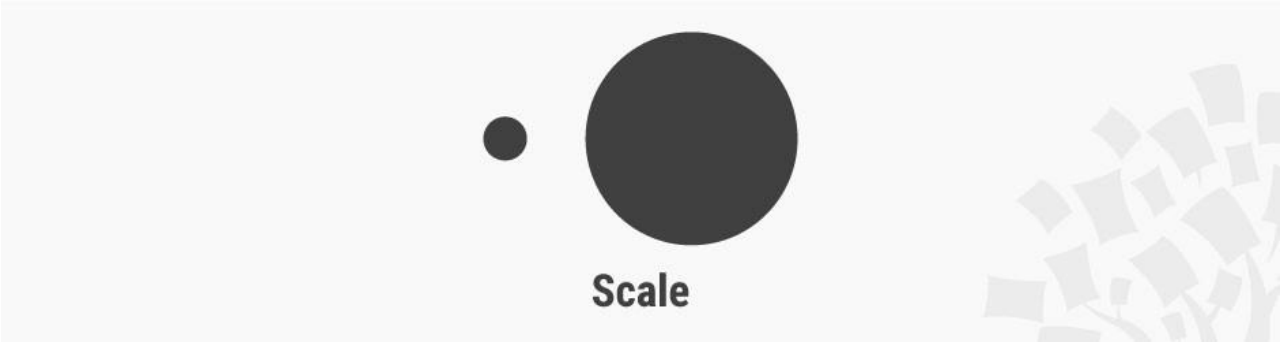
Contrast

We use contrast to make an element *stand out* by manipulating differences in colour, value, size and other factors. For instance, as designers (be it in logo design, UI design, etc.), we often use the colour red to make certain elements stand out. In iOS, red often appears in the “Delete” action to signify that an (often) irreversible action is about to occur. On the other hand, green is often something we use (at least in Western design) in positive actions such as “Go” and “Accept” — thus highlighting that we cannot ignore the cultural meaning of colours when designing for contrast. If you’re designing for a client in a far-off land, learn about and adjust your work to conform to the cultural considerations. For instance, ask yourself, “Is their red lucky or angry?” or “Is their black businesslike or funerary?”



Scale

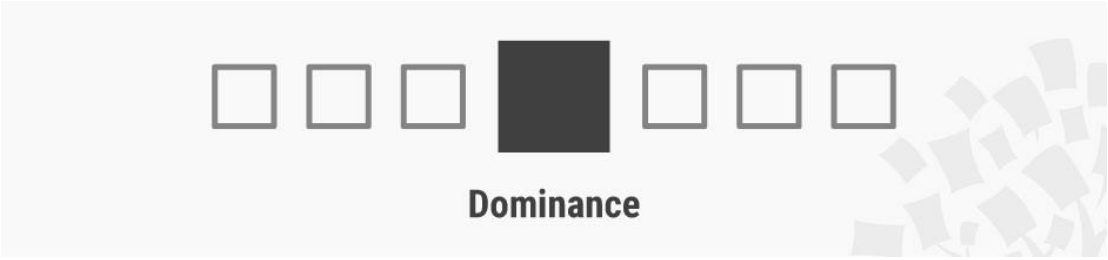
Scale describes the *relative sizes* of the elements in a design. By using scale to make an element larger than others appearing with it, you can *emphasis* that element. Not only can you make an element stand out this way—you can also use scale to create a sense of *depth* (since nearer objects appear larger to the human eye). Exaggerated scales of images also add a certain level of interest and drama to them.



Scale can be used to create a hierarchy for and add emphasis to certain elements on a design.

Dominance

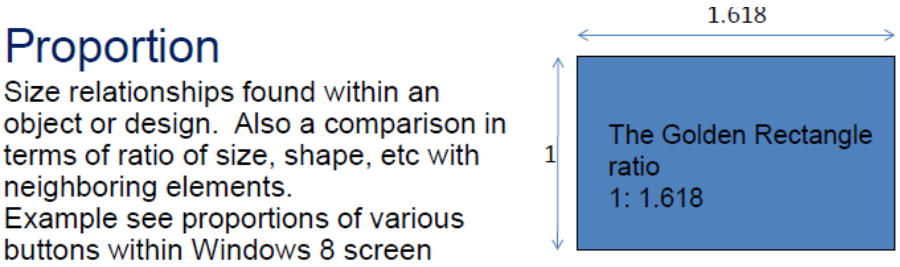
Dominance creates *focus* on a *single* element. We can use colour, shape, contrast, scale, and/or positioning to achieve this. For instance, most websites have a main “hero” image, which uses dominance to appeal to users, drawing them to it naturally.



Dominance can be established by using positioning, shape and colour, among many other factors.

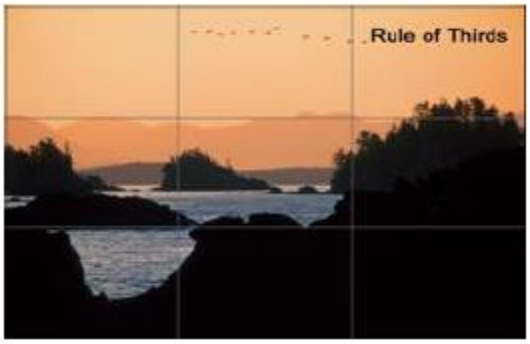
When working in visual design, we should ensure that we use dominance while still maintaining the *unity* and *balance* of websites — if not, the design would potentially produce a disorienting experience for users.

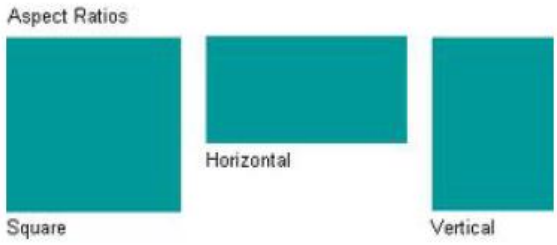
Proportion:



Proportion & Rule of thirds division of a screen

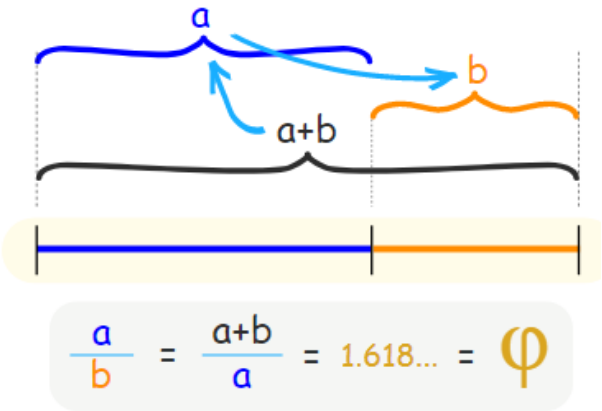
Proportion refers to the size relationship of visual elements to each other and to the whole picture. One of the reasons proportion is often considered important in composition is that viewers respond to it emotionally.



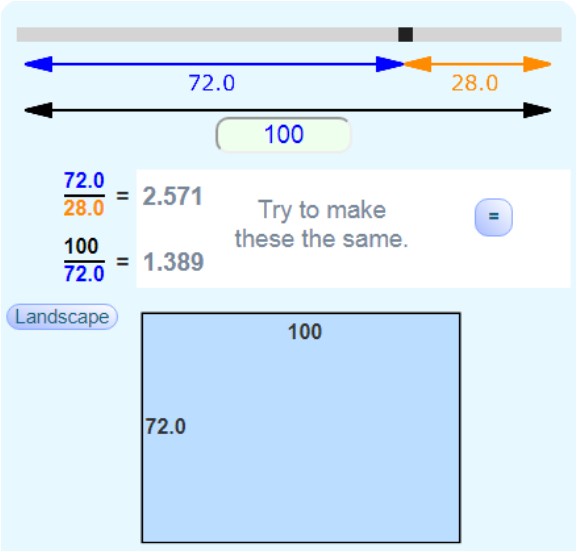


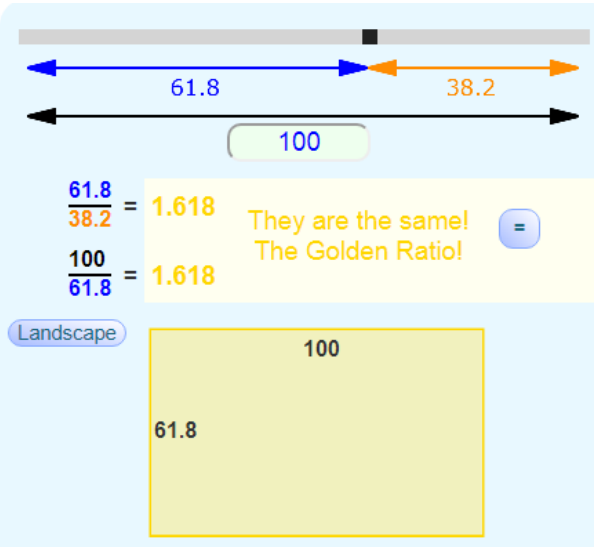
The **rule of thirds** is a "rule of thumb" or guideline which applies to the process of composing visual images such as designs, films, paintings, and photographs. The guideline proposes that an image should be imagined as divided into nine equal parts by two equally spaced horizontal lines and two equally spaced vertical lines, and that important compositional elements should be placed along these lines or their intersections.

We find the golden ratio when we divide a line into two parts so that: the long part divided by the short part ***is also equal to*** the whole length divided by the long part



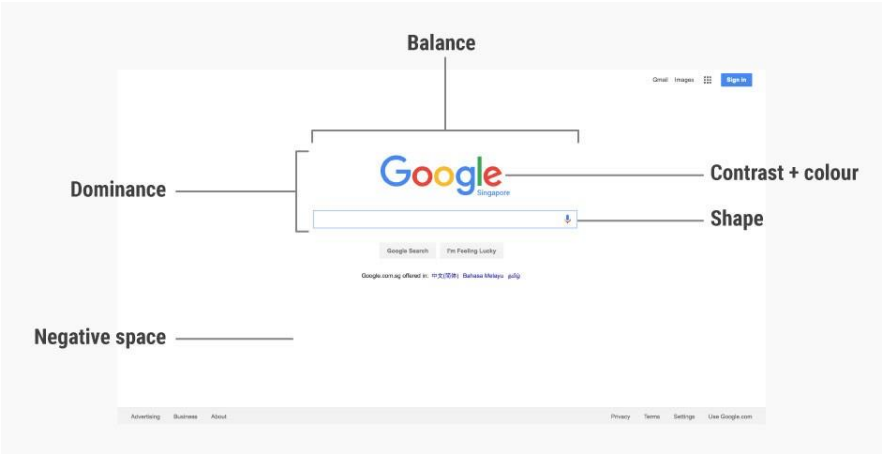
Some artists and architects believe the Golden Ratio makes the most pleasing and beautiful shape.





Examples to highlight design elements and principles
Google's homepage

Google's homepage is one of the most visited webpages in the world. The raw simplicity of the page is partly why it is so well designed, but here are other factors that make this page work superbly:



- **Dominance**: The large Google logo and search box gives it dominance, making it the core (and to most, sole) focus of the entire page.
- **Contrast (and colour)**: Google's logo uses bright (mostly primary) colours, and these mix well, forming a visually pleasing logo. The logo also has sufficient contrast against a white background, making it stand out on the page.
- **Shape**: The search box uses a rectangular shape to delineate the search field, making it very usable.
- **Negative space**: Google's homepage is predominantly made out of negative space, which makes the search box (the main function of the page) the centre of attention. The negative space also works well for the page, as it acts like a blank sheet of paper before users type in their search terms.
- **Balance**: The page is almost vertically symmetrical, resulting in a sense of balance that is very pleasing and calm to look at.

The Clustering Principle:

Organizing the screen into visually separate blocks of similar controls, preferably with a title for each block. Modern WIMP (Windows-Icons-Menus-Pointer) systems are a natural expression of the Clustering Principle.

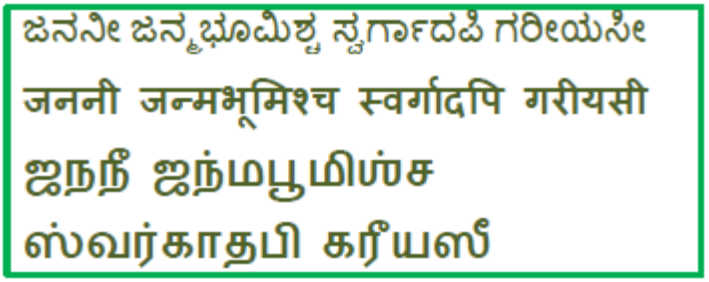
Information on a screen which is not catagorised into some order (right hand screen in the above figure) can be confusing. GRIDS are therefore used to not only to align & please aesthetically but also catagorise UI elements according to functions.

Type size and font, for example: the Reduced Clutter Principle would suggest that one or two type styles are sufficient.

Avoid fancy fonts totally

Safe Fonts

- Arial, Helvetica, sans-serif
- Courier New, Courier, mono
- Verdana, Arial, Helvetica, sans-serif
- Geneva, Arial, Helvetica, sans-serif

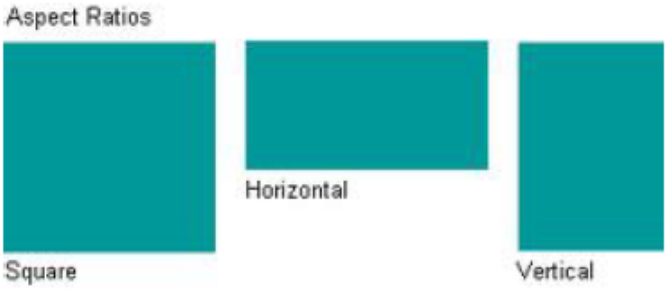


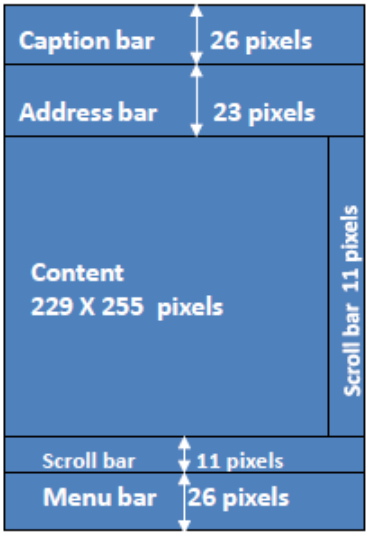
Weight of font matters

BOLD –some times, results in poor smudged readability on mobile screens -even on AMOLED

Regional Fonts still have unresolved problems when used in low resolution & small displays screens.

Screen resolution & aesthetics





25 to 30% of the area is taken by buttons, etc. Therefore only about 229 X 255 is effectively available from a 320 X240 display.

Case Study 1 : Windows GUI

Aesthetic and minimalist design:

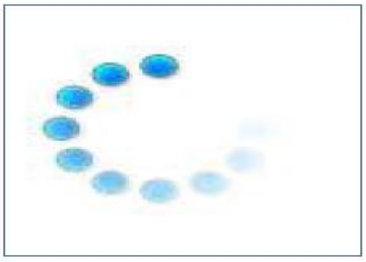
The system is not cluttered with excessive use of icons and buttons. Tabs are used to separate different functionalities. A simple rectangle composition arrangement is used to model information.

Recognition rather than recall: The use of colour schemes and icons act to denote functionalities . Example ‘ Head Phone icon”. This design feature promotes recognition of rather than recall of system functionalities.

Case Study 2 : Icon Design

Two simple icons communicating an activity in progress.

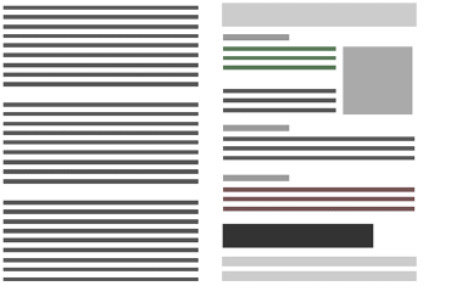
Both the icons are graphically simple, do the function of informing the status & are not complicated to understand. They use gradient in colorus (monochrome) to depict time progress through animation. The circular form express the abstract concept of time. The state of ‘please wait’ is expressed in a pleasant peaceful unhurried manner. In terms of construction, the icons do not take expensive screen real estate; need very less computing memory; are amiable to both pixel as well as vector graphics. The icon has achieved this by employing aesthetic principles in their form, colour, shape, configuration, motion & composition –all of them put together holistically resulting in a simple ‘design’.



Graphic Design –Website Layout

HCI-Designers besides being Engineers are Artists in the sense that they have to become sensitive to the visual language and master the use of visual elements in accordance to Principles.

Graphic Design Case Study 3



A case study of a website’s visual quality.

The principles of Cognitive Science –Gestalt laws govern aesthetics.

Aesthetics

Ordered Grid -

Rows and Columns

Good composition –

Position with respect to area

Visual Balance –

Symmetry / Asymmetry

Low visual noise –

No clutter or crowding

Color & Graphics –

Simple plain light reflective /absorbing colors with no fancy labels.

•Aesthetics is a specialized discipline.

•It has as much science & technology in it as much as Art.

•It is qualitative as well as quantitative judgment.

•Creative Designers are best equipped to decide on aesthetics as they are trained professionally.

Prototyping Techniques

Prototyping

Prototypes are experimental and incomplete designs which are cheaply and fast developed. Prototyping, which is the process of developing prototypes, is an integral part of iterative user-centered design because it enables designers to try out their ideas with users and to gather feedback.

The main purpose of prototyping is to involve the users in testing design ideas and get their feedback in the early stage of development, thus to reduce the time and cost. It provides an efficient and effective way to refine and optimize interfaces through discussion, exploration, testing and iterative revision. Early evaluation can be based on faster and cheaper prototypes before the start of a full-scale implementation. The prototypes can be changed many times until a

better understanding of the user interface design has been achieved with the joint efforts of both the designers and the users.

- A prototype is essentially a model of the system
- The prototype (model) can have limited or full range of functionalities of the proposed system
- A widely used technique in engineering where novel products are tested by testing a prototype
- Prototypes can be “throw away” (e.g., scale models which are thrown away after they serve their purpose) or can go into commercial use.
- In software development prototypes can be:

Paper-based: likely to be thrown away after use

Software-based: can support few or all functionalities of the proposed system. May develop into full-scale final product

What to Prototype?

Any aspect of the design that needs to be evaluated

–Work flow

–Task design

–Screen layout

–Difficult, controversial, critical areas

Prototypes in HCI

In HCI, prototypes take many forms

–A storyboard (cartoon-like series of screen sketches)

–A power point slide slow

–A video simulating the use of a system

– Cardboard mock-up

–A piece of software with limited functionality

–Even a lump of wood

We can categorize all these different forms of prototypes in three groups

–Low fidelity prototypes

–Medium fidelity prototypes

–High fidelity prototypes

Low Fidelity Prototypes

Low-fidelity prototypes are quickly constructed to depict concepts, design alternatives, and screen layouts, rather than to model the user interaction with a system. Low-fidelity prototypes provide limited or no functionality. They are intended to demonstrate the general look and the feel of the interface, but not the detail how the application operates. They are created to communicate and exchange ideas with the users, but not to serve as a basis for coding and testing. A facilitator who knows the application thoroughly is generally needed to demonstrate the prototype to the users.

•Basically paper mock-up of the interface look, feel, and functionality

–Quick and cheap to prepare and modify

•Purpose

–Brainstorm competing designs

–Elicit user reaction (including any suggestions for modifications)

Low Fidelity Prototypes: Sketches

Interface of a proposed system

A sketch of the interface

•In a sketch, the outward appearance of the intended system is drawn

–Typically a crude approximation of the final appearance

•Such crude approximation helps people concentrate on high level concepts

–But difficult to visualize interaction (dialog's progression)

Low Fidelity Prototypes: Storyboarding

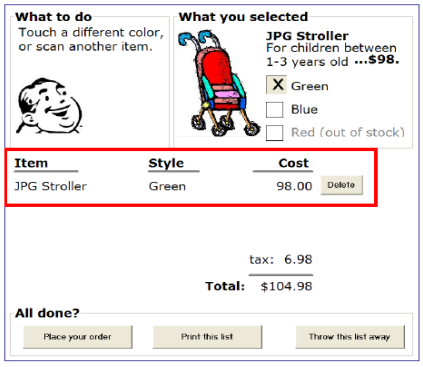
•Scenario-based prototyping

•Scenarios are scripts of particular usage of the system

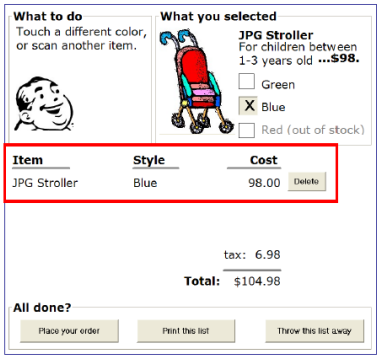
•The following (four) slides show an example storyboarding of a scenario of stroller-buying using an e-commerce interface

Initial screen. Shows the layout of interface options.

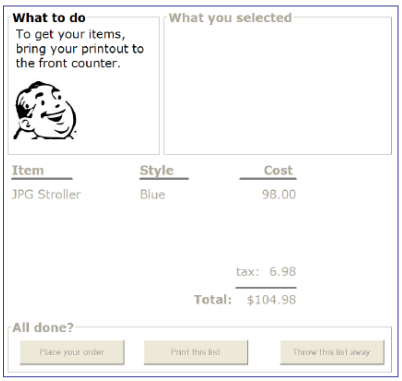
Once a stroller is selected by the customer, its tag is scanned with a hand-held scanner. The details of the stroller is displayed on the interface if the scanning is successful. Also, the option buttons become active after a successful scan.



However, the customer can choose a different product at this stage and the same procedure is followed. For example, the customer may choose a stroller with different color.



Once the customer finalizes a product, a bill is generated and displayed on the interface. The option buttons become inactive again.



Here, a series of sketches of the key frames during an interaction is drawn

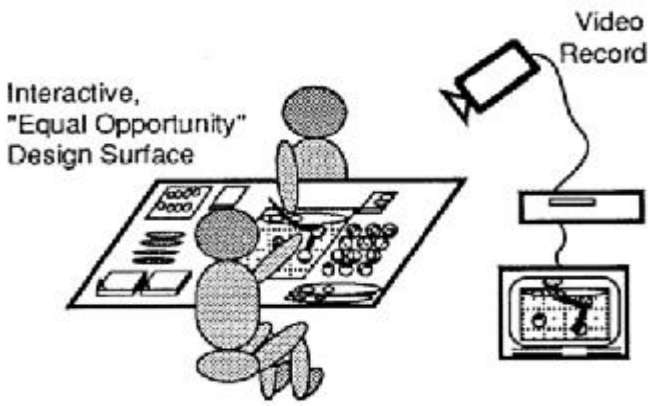
- Typically drawn for one or more typical interaction scenarios
- Captures the interface appearance during specific instances of the interaction
- Helps user evaluate the interaction (dialog) unlike sketches

Low Fidelity Prototypes: Pictiv

•Pictiv stands for "plastic interface for collaborative technology initiatives through video exploration".

- Basically, using readily available materials to prototype designs
  - Sticky notes are primarily used (with plastic overlays)
  - Represent different interface elements such as icons, menus, windows etc. by varying sticky note sizes
- Interaction demonstrated by manipulating sticky notes
  - Easy to build new interfaces "on the fly"
- Interaction (sticky note manipulation) is videotaped for later analysis

PICTIVE was begun in reaction to software rapid prototyping environments, in which developers have a disproportionate design impact, but non-computer users are relatively disempowered by the complexity of current software prototyping environments. The PICTIVE techniques were designed to be used by people who were not necessarily programming professionals, so all participants have equal opportunity to contribute their ideas.



The apparatus for PICTIVE includes video camera and a collection of design objects. The design objects fall into two categories. The first category is simple office materials, including pens, high-lighters, papers, Post-It, stickers. Labels and paper clips -- all in a range of bright colors. The second category is materials prepared by the developer, such as menu bars, dialogue boxes, special icons for the project and so on.

The procedures of PICTIVE are as follows.

First, both the users and the developers are asked to prepare a "homework". Typically, the users are asked to think about task scenarios, for instance, "what you would like the system to do for you and what steps are required to finish the job". The developers need to construct a preliminary set of system components for the users to manipulate based on prior discussions with the users.

Second, during the PICTIVE session, both the users and developers manipulate the design objects on a design surface, where the designs are put together as multiple layers of sticky notes and overlays that can be changed by simple colored pens. Each participant brings her or his expertise. The resulting design is not driven by any single participant, but represents a synthesis of the different participants' different views. The users' work scenarios are explored in detail, leading to a discussion of how the developers' technology can be applied to meet the users' human needs.

A coordinator may be needed to keep the group on track. A video camera is focused on the design objects and to record the voices of the design team as they manipulate those objects. The video record of the design session will serve as a dynamic presentation of the design. Figure 6 illustrates a scene in the PICTIVE session.

Medium Fidelity Prototypes

•Prototypes built using computers

–More powerful than low fidelity prototypes

–Simulates some but not all functionalities of the system

–More engaging for end users as the user can get better feeling of the system

–Can be helpful in testing more subtle design issues

Medium Fidelity Prototypes: Scenarios

Computer are more useful (than drawing on paper as in storyboarding) to implement scenarios

–Provide many useful tools (e.g., power point slides, animation)

–More engaging to end-users (and easier to elicit better response) compared to hand-drawn story-boarding

Broadly of two types

Vertical prototyping: Vertical prototyping cuts down on the number of features, so that the result is a narrow system that includes in-depth functionality, buy only for a few selected features. Vertical prototypes allow users to perform and test some real tasks.

–**Example:** working of a single menu item in full

Horizontal prototyping: Horizontal prototyping reduces the level of functionality so that the result is a surface layer that includes the entire user interface to a full-featured system without underlying functionality.
Horizontal prototypes allow users to feel the entire interface, even though they cannot perform any real tasks.

The main advantages of horizontal prototypes are that they can be implemented fast with the use of prototyping and screen design tools, and they can be used to assess the interface as a whole.

–**Example**: first screen of an interface (showing layout)

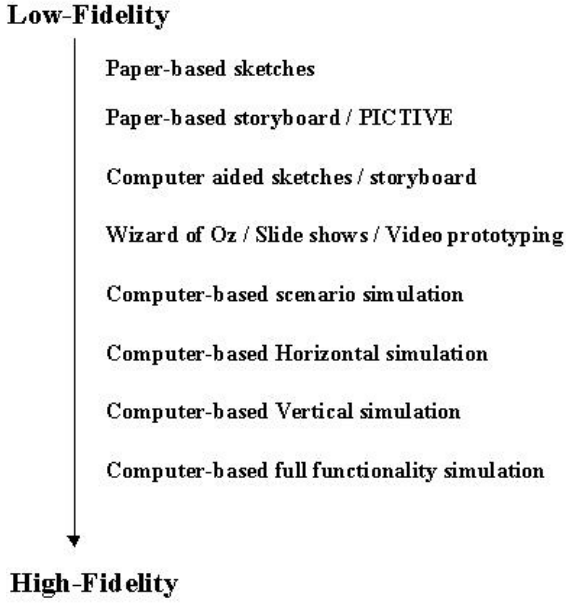
High Fidelity Prototypes

High fidelity is highly-functional and interactive prototyping, which is quite close to the final product, with lots of functionality functions and details included. So, this is often used in the later usability evaluation to discover the potential issues that may exist in the workflow, interactivity and so on.

Typically a software implementation of the design with full or most of the functionalities

–Requires money, manpower, time and effort

–Typically done at the end for final user evaluations



Transition of Prototyping Techniques:

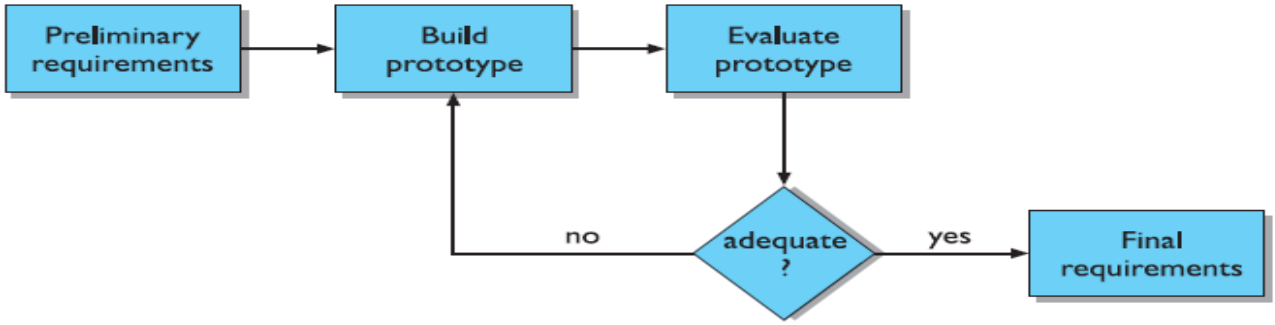
Prototype and Final Product

•Prototypes are designed and used in either of the following ways:

–**Throw-away: prototypes** are used only to elicit user reaction. Once their purpose is served, they are thrown away.

–Typically done with low and some medium fidelity prototypes

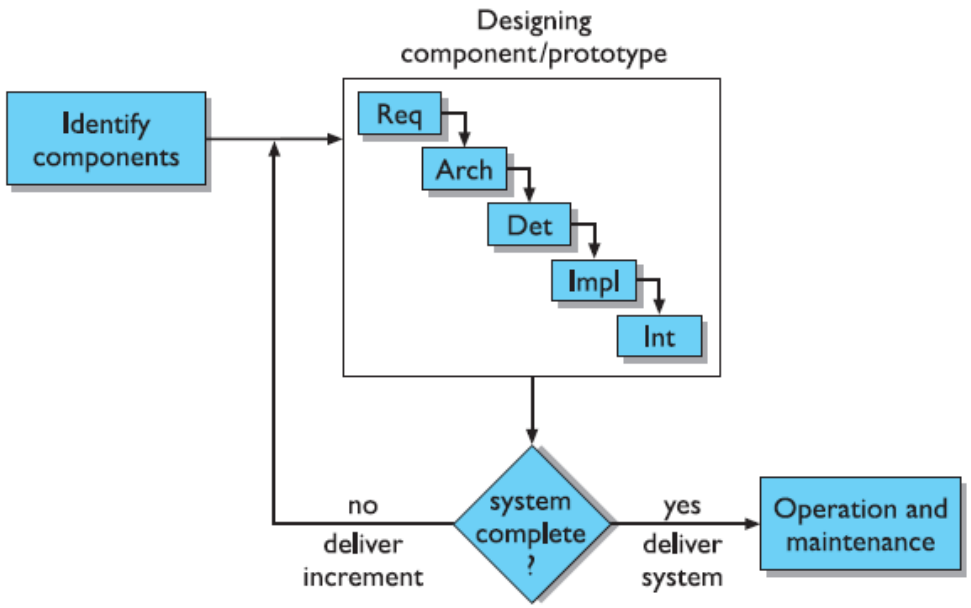
The prototype is built and tested. The design knowledge gained from this exercise is used to build the final product, but the actual prototype is discarded. **Figure depicts the procedure in using throw-away prototypes** to arrive at a final requirements specification in order for the rest of the design process to proceed.



–**Incremental:** Product is built as separate components (modules). After each component is prototyped and tested, it is added to the final system

–Typically done with medium and hi fidelity prototypes

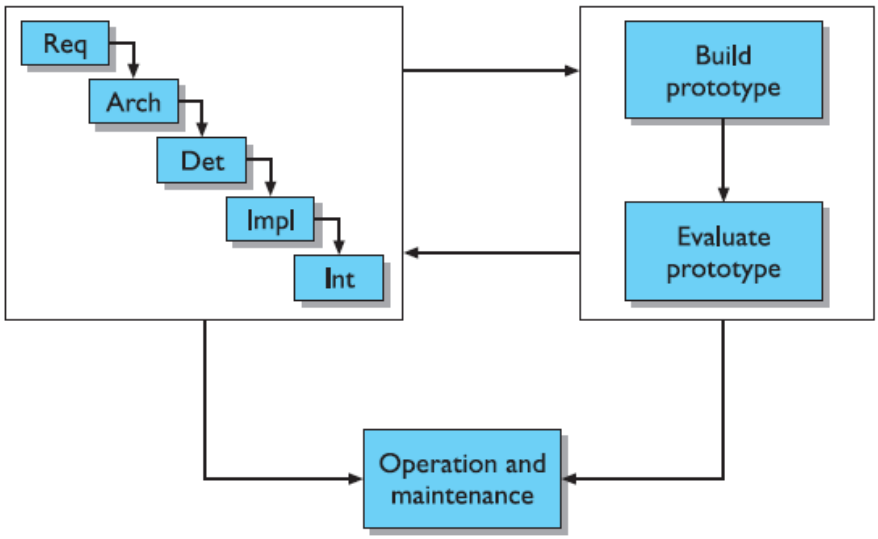
Incremental The final product is built as separate components, one at a time. There is one overall design for the final system, but it is partitioned into independent and smaller components. The final product is then released as a series of products, each subsequent release including one more component. This is depicted in.



–**Evolutionary:** A single prototype is refined and altered after testing, iteratively, which ultimately "evolve" to the final product

–Typically done with hi fidelity prototypes

Evolutionary Here the prototype is not discarded and serves as the basis for the next iteration of design. In this case, the actual system is seen as evolving from a very limited initial version to its final release, as depicted in Figure. Evolutionary prototyping also fits in well with the modifications which must be made to the system that arise during the operation and maintenance activity in the life cycle.



Prototyping Tools

•For (computer-based) medium and hi fidelity prototype developed, several tools are available

–**Drawing tools**, such as Adobe Photoshop, MS Visio can be used to develop sketch/storyboards.

–**Presentation software**, such as MS Power Point with integrated drawing support are also suitable for low fidelity prototypes.

–Media tools, such as Adobe flash can be to develop storyboards. Scene transition is achieved by simple user inputs such as key press or mouse clicks.

–Interface builders, such as VB, Java Swing with their widget libraries are useful for implementing screen layouts easily (horizontal prototyping). The interface builders also supports rapid implementation of vertical prototyping through programming with their extensive software libraries.

The Wizard of Oz Technique

•A technique to test a system that does not exist

•First used to test a system by IBM called the listening typewriter (1984)

–Listening typewriter was much like modern day voice recognition systems. User inputs text by uttering the text in front of a microphone. The voice is taken as input by the computer, which then identifies the text from it.

•Implementing voice recognition system is too complex and time consuming

•Before the developers embark on the process, they need to check if the "idea" is alright; otherwise the money and effort spent in developing the system would be wasted

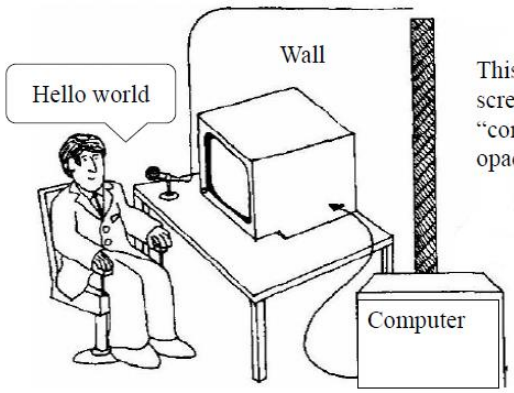
•Wizard of oz provides a mechanism to test the idea without implementing the system

•Suppose a user is asked to evaluate the listening typewriter

•He is asked to sit in front of a computer screen

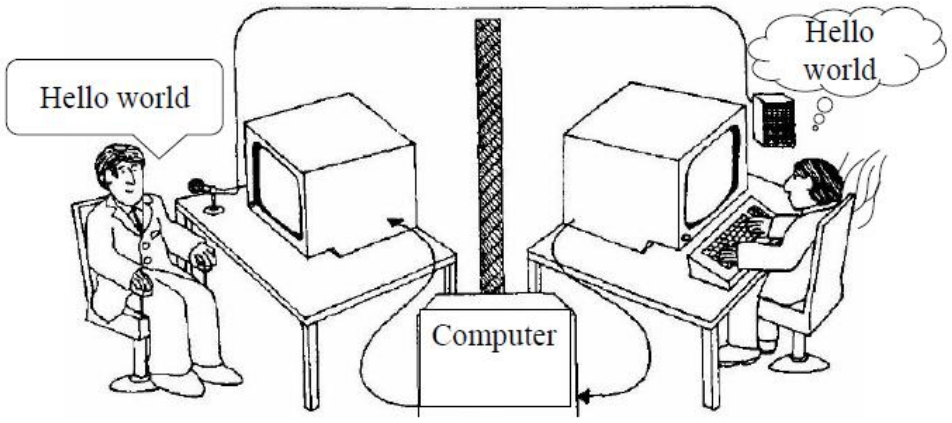
•A microphone is placed in front of him

•He is told that "whatever he speaks in front of the microphone will be displayed on the screen"



This is what the user sees: a screen, a microphone and the "computer" in front of a opaque wall.

This is what happens behind the wall. A typist (the wizard) listen to the utterance of the user, types it, which is then displayed on the user's screen. The user thinks the computer is doing everything, since the existence of the wizard is unknown to him.



•Human ‘wizard’ simulates system response

–Interprets user input

–Controls computer to simulate appropriate output

–Uses real or mock interface

–Wizard is typically hidden from the user; however, sometimes the user is informed about the wizard’s presence

•The technique is very useful for

–Simulating complex vertical functionalities of a system

–Testing futuristic ideas