UNIT I

FOUNDATIONS OF HCI

The Human: I/O channels – Memory – Reasoning and problem solving; The computer: Devices Memory – processing and networks; Interaction: Models – frameworks – Ergonomics – styles elements – interactivity- Paradigms.

HCI (human-computer interaction)

Study of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings

- Information i/o
  visual, auditory, haptic, movement
- Information stored in memory
  sensory, short-term, long-term
- Information processed and applied
  reasoning, problem solving, skill, error

The Human

I/O channels

❖ A person's interaction with the ***outside world occurs through information being received and sent: input and output***.
❖ In an interaction with a computer the ***user receives information that is output by the computer, and responds by providing input to the computer the user's output*** become the computer's input and vice versa.
❖ A particular channel ***may have a primary role as input or output in the interaction***, it is more than likely that it is also used in the other role.

For example, sight may be used ***primarily in receiving information from the computer, but it can also be used to provide information to the computer***, for example by fixating on a particular screen point when using an eye gaze system.

❖ ***Input in the human*** occurs mainly ***through the senses and output through the motor control of the effectors***.
Five senses: sight, hearing, touch, taste and smell
Taste and smell do not currently play a significant role in HCI
❖ Similarly there are a ***number of effectors, including the limbs, fingers, eyes, head and vocal system***.

In the interaction with the computer, ***the fingers play the primary role***, through typing or mouse control, with some use of voice, and eye, head and body position.

Imagine using a personal computer with a mouse and a keyboard.The application you are using has a graphical interface, with menus, icons and windows.

In your interaction with this system you receive Information primarily by sight, from what appears on the screen.

However, you may also receive information by ear: For example, the computer may `beep' at you if you make a mistake or to draw attention to something, or there may be a voice commentary in a multimedia presentation

Vision

Human vision ***Highly complex activity with range of physical and perceptual limitations***, yet it is the primary source of information for the average person. We can roughly **divide visual perception into two stages.**

- ***the physical reception of the stimulus from outside world,***
- ***the processing and interpretation of that stimulus***

The human eye

- ***Vision begins with light***.
  The eye ismechanism for ***receiving light and Transforming it into electrical energy*.**
- Light is ***reflected from objects in the world and their image is focused upside down on the back of the eye***.

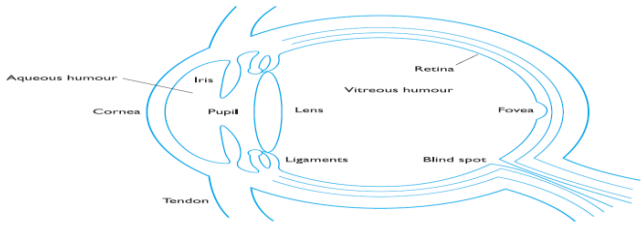


Figure 1.1 The human eye

- The ***receptors in the eye transform it into electrical signals, which are passed to brain***

The cornea and lens at the front of eye focus the light into a sharp image on the back of the eye, the retina. The retina is light sensitive and contains two types of photoreceptor: rods and cones.

Rods

⌘ ***Highly sensitive to light*** and therefore allow us to see under a low level of illumination.
⌘ It is ***unable to resolve fine detail and are subject to light saturation***.
⌘ It is the ***reason for the temporary blindness*** we get when moving from a darkened room into sunlight: the rods have been active and are saturated by the sudden light.

Cones

⌘ Cones are the ***second type of receptor in the eye***.
⌘ They are ***less sensitive to light than the rods and can therefore tolerate more light***.
⌘ There are three types of cone, each sensitive to a different wavelength of light.
⌘ This allows color vision. The eye has ***approximately 6 million cones***, mainly concentrated on the fovea

Fovea

⌘ Fovea is a ***small area of the retina on which images are fixated***.

Blind spot

⌘ Blind spot is also ***situated at retina***.
⌘ Although the retina is ***mainly covered with photoreceptors there is one blind spot where the optic nerve enter the eye***.
⌘ The blind spot has ***no rods or cones***, yet our visual system compensates for this so that in normal circumstances we are unaware of it.

Nerve cells

⌘ The retina also has ***specialized nerve cells called ganglion cells***.
⌘ There are two types:
⌘ X-cells: These are ***concentrated in the fovea and are responsible for the early detection of pattern.***
⌘ Y-cells : These are more ***widely distributed in the retina*** and are responsible for the early detection of movement.

Visual perception

⌘ Understanding the basic construction of the eye goes some way to explaining the physical mechanism of vision but visual perception is more than this.
⌘ The information ***received by the visual apparatus must be filtered and passed to processing elements which allow us to recognize coherent scenes, disambiguate relative distances and differentiate color.***
⌘ Let us see how we perceive size and depth, brightness and color, each of which is crucial to the design of effective visual interfaces.

Perceiving size and depth

Imagine you are standing on a hilltop. Beside you on the summit you can see rocks, sheep and a small tree.

⌘ We can identify similar objects regardless of the fact that they appear to us to be vastly different sizes. In fact, we can use this information to judge distance. So how does the eye perceive size, depth and relative distances?
⌘ To understand this we must consider how the image appears on the retina.
⌘ As we mentioned, ***reflected light from the object forms an upside-down image on the retina. The size of that image is specified as visual angle.***
Figure1.2 illustrates how the visual angle is calculated

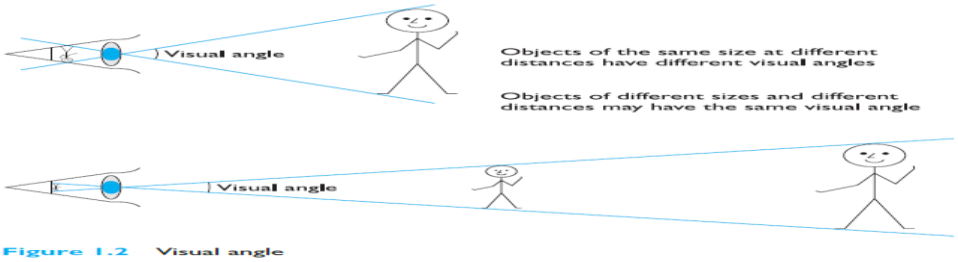


Figure 1.2 Visual angle

If were to draw a line from the top of the object to a central point on the front of the eye and a second line from the bottom of the object to the same point, the visual angle of the object is the angle between these two lines.

Visual angle is affected by both the size of the object and its distance from the eye. Therefore if two objects are at the same distance, the larger one will have the larger visual angle.

The ***visual angle indicates how much of the field of view is taken by the object*.** The visual angle measurement is given in either degrees or minutes of arc, where 1 degree is equivalent to 60 minutes of arc, and 1 minute of arc to 60 seconds of arc.

Visual acuity is the ability of a ***person to perceive fine detail***. A number of measurements have been established to test visual acuity, most of which are included in standard eye tests.

For example, a person with normal vision can detect a single line if it has a visual angle of 0.5 seconds of arc. Spaces between lines can be detected at 30 seconds to 1 minute of visual arc. These represent the limits of human visual acuity.

Perceiving brightness

A second step of ***visual perception is the perception of brightness***.

Brightness

A subjective reaction to level of light. It is affected by ***luminance,*** which is the amount of light is emitted by an object. The luminance of an object is dependent on the amount of light falling on the object's surface and its reflective properties.

Contrast is related to luminance:

It is a function of the luminance of an object and the luminance of its background. In dim lighting, the rods predominate vision. Since there are fewer rods on the fovea, object in low lighting can be seen easily when fixated upon, and are more visible in peripheral vision. In normal lighting, the cones take over.

Perceiving color:

A third factor that we need to consider is perception of color.***Color is usually regarded as being made up of three components: hue, intensity and saturation***.

Hue is determined by the ***spectral wavelength of the light***. Blues have short wavelengths, greens medium and reds long. Approximately 150 different hues can be discriminated by the average person.

Intensity is the ***brightness of the color***, and ***saturation*** is the amount of ***whiteness in the color***. By varying these two, we can perceive in the region of ***7 million different colors.*** The eye perceives color because the cones are sensitive to light of different wavelengths. There are three different types of cone, each sensitive to a different color (blue, green and red).

Color vision is best in the fovea, and worst at the periphery where rods predominate. It should also be noted that only 3–4% of the fovea is occupied by cones which are sensitive to blue light, making blue acuity lower.

The capabilities and limitations of visual processing

Visual processing involves the transformation and interpretation of a complete image, from the light that is thrown onto the retina.

For example, if we know that an ***object is a particular size, we will perceive it as that size no matter how far it is from us***.

Visual processing compensates for the movement of the image on the retina which occurs as we move around and as the object which we see moves.

Although the retinal image is moving, the image that we perceive is stable. Similarly, color and brightness of objects are perceived as constant, in spite of changes in luminance

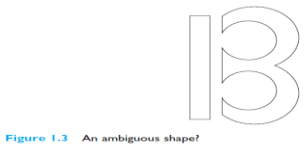


Figure 1.3 An ambiguous shape?

Figure 1.4 ABC

Figure 1.5 12 13 14

Figure 1.6 The Muller–Lyer illusion – which line is longer?

Consider Figure1.6 Which line is longer? Most people when presented with this will say that the top line is longer than the bottom. In fact, the two lines are the same length. This may be due to a false application of the law of size constancy: the top line appears like a concave edge, the bottom like a convex edge. The former therefore seems further away than the latter and is therefore scaled to appear larger.

A similar illusion is the ***Ponzo illusion*** (Figure1.7). Here the ***top line appears longer, owing to the distance effect, although both lines are the same length.*** These illusions demonstrate that our perception of size is not completely reliable

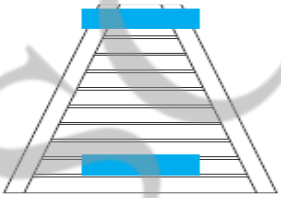


Figure 1.7 The Ponzo illusion – are these the same size?

The quick brown
fox jumps over the
the lazy dog.

Figure 1.8 Is this text correct?

Reading

There are ***several stages in the reading process***.

1. First, the ***visual pattern of the word on the page is perceived***.
2. It is then ***decoded with reference to an internal representation of language.***
3. The final stages of language processing include ***syntactic and semantic analysis and operate on phrases or sentences.***

During reading, ***the eye makes jerky movements called saccades followed by fixations.***

Perception occurs ***during the fixation periods, which account for approximately 94% of the time elapsed.***

The eye moves backwards over the text as well as forwards, in what are known as regressions.

If the text is complex there will be more regressions.

Adults read approximately 250 words a minute.

It is unlikely that ***words are scanned serially, character by character, since experiments have shown that words can be recognized as quickly as single characters***.

Instead***, familiar words are recognized using word shape***.

The speed at which text can be read is a measure of its legibility. Experiments have shown that standard font sizes of 9 to 12 points are equally legible, given proportional spacing between lines.

Similarly line lengths of between 2.3 and 5.2 inches (58 and 132 mm) are equally legible.

This is thought to be due to a number of factors including a longer line length, fewer words to a page,

The Human ear

- **Physical apparatus:**
  - outer ear – protects inner and amplifies sound
  - middle ear – transmits sound waves as vibrations to inner ear
  - inner ear – chemical transmitters are released and cause impulses in auditory nerve

- **Sound**
  - pitch–sound frequency
  - loudness –amplitude
  - timbre–type or quality

Part	function
OUTER EAR Pinna	collects and directs sound waves into the ear canal.
ear canal / auditory canal	transmits sound waves to the eardrum.
Eardrum	vibrates and transmits sound waves to the ossicles.
MIDDLE EAR Ossicles	intensify the vibrations of the sound waves by 22 times before transmitting to the oval window.
Eustachian tube	balances the air pressure at both side of the eardrum.
oval window	transmits sound vibrations from the middle ear to the inner ear.
INNER EAR Cochlea	transforms sound vibrations into impulses.
semicircular canals	balance the body position.
auditory nerves	send messages to the brain which interprets the messages as sound.

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

Touch

The third and last of the senses that we will consider is touch or haptic perception.

- Touch provides us with vital information about our environment.
- It tells us when we touch something hot or cold, and can therefore act as a warning.
- It also provides us with feedback when we attempt to lift an object,

For example. Consider the act of picking up a glass of water If we could only see the glass and not feel when our hand made contact with it or feel its shape, the speed and accuracy of the action would be reduced. This is the experience of users of certain virtual reality games: they can see the computer-generated objects which they need to manipulate but they have no physical sensation of touching them.

- Provides important feedback about environment.
- May be key sense for someone who is visually impaired.
- Stimulus received via receptors in the skin:
  thermoreceptors – heat and cold
  nociceptors – pain
  mechanoreceptors – pressure (some instant, some continuous)

Rapidly adapting mechanoreceptors respond to immediate pressure as the skin is indented. These receptors also react more quickly with increased pressure.

However, they stop responding if continuous pressure is applied. Slowly adapting mechanoreceptors respond to continuously applied pressure.

Movement

- A simple action such as hitting a button in response to a question involves a number of processing stages.
- The stimulus (of the question) is received through the sensory receptors and transmitted to the brain.
- The question is processed and a valid response generated. The brain then tells the appropriate muscles to respond.
- Each of these stages takes time, which can be roughly divided into reaction time and movement time.
- Movement time is dependent largely on the physical characteristics of the subjects: their age and fitness, for example.
- Reaction time varies according to the sensory channel through which the stimulus is received.

  A person can react to an auditory signal in approximately 150 ms, to a visual signal in 200 ms and to pain in 700 ms.

Factors such as skill or practice can reduce reaction time, and fatigue can increase it.

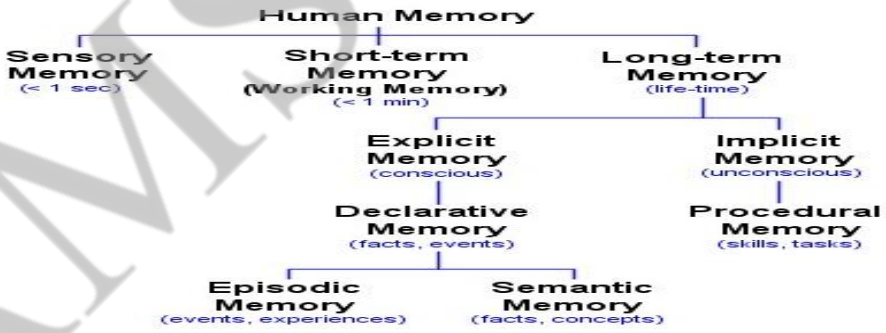
A second measure of motor skill is accuracy. Speed and accuracy of movement are important considerations in the design of interactive systems, primarily in terms of the time taken to move to a particular target on a screen. The target may be a button, a menu item or an icon, for example.

The time taken to hit a target is a function of the size of the target and the distance that has to be moved. This is formalized in Fitts' law of this formula, which have varying constants, but they are all very similar.

One common form is **Movement time = a + b log2(distance/size + 1)**

where a and b are empirically determined constants.

HUMAN MEMORY



Short-term memory

- Short-term memory or working memory acts as a 'scratch-pad' for temporary recall of information.
- It is used to store information which is only required fleetingly. For example, calculate the multiplication 35 × 6 in your head. The chances are that you will have done this calculation in stages, perhaps 5 × 6 and then 30 × 6 and added the results; or you may have used the fact that 6 = 2 × 3 and calculated 2 × 35 = 70 followed by 3 × 70.

To perform calculations such as this we need to store the intermediate stages for use later. Or consider reading. In order to comprehend this sentence you need to hold in your mind the beginning of the sentence as you read the rest. Both of these tasks use short-term memory.

➢ Short-term memory can be accessed rapidly, in the order of 70 ms. However, it also decays rapidly, meaning that information can only be held there temporarily, in the order of 200 ms.

Short-term memory also has a limited capacity. There are two basic methods for measuring memory capacity. The first involves determining the length of a sequence which can be remembered in order. The second allows items to be freely recalled in any order. Using the first measure, the average person can remember 7 ±2 digits.

Long-term memory

- If short-term memory is our working memory or 'scratch-pad', long-term memory is our main resource.
- Here we store factual information, experiential knowledge, procedural rules of behavior – in fact, everything that we 'know'. It differs from short-term memory in a number of significant ways.

First, it has a huge, if not unlimited, capacity.

Secondly, it has a relatively slow access time of approximately a tenth of a second. Thirdly, forgetting occurs more slowly in long-term memory, if at all. These distinctions provide further evidence of a memory structure with several parts. Long-term memory is intended for the long-term storage of information. Information is placed there from working memory through rehearsal. Unlike working memory there is little decay: long-term recall after minutes is the same as that after hours or days.

Long-term memory structure

Two types of long-term memory: **Episodic memory and semantic memory.**

❖ ***Episodic memory represents our memory of events and experiences in a serial form***. It is from this memory that we can reconstruct the actual events that took place at a given point in our lives.

❖ ***Semantic memory, on the other hand, is a structured record of facts, concepts and skills that we have acquired.*** The information in semantic memory is derived from that in our episodic memory, such that we can learn new facts or concepts from our experiences. Semantic memory is structured in some way to allow access to information, representation of relationships between pieces of information, and inference. ***One model for the way in which semantic memory is structured is as a network. Items are associated to each other in classes, and may inherit attributes from parent classes. This model is known as a semantic network.***

- Semantic networks ***represent the associations and relationships between single items in memory***. However, they do not allow us to model the representation of more complex objects or events, which are perhaps composed of a number of items or activities. ***Structured representations such as frames and scripts organize information into data structures.*** *Slots* in these structures allow attribute values to be added. Frame slots may contain default, fixed or variable information.

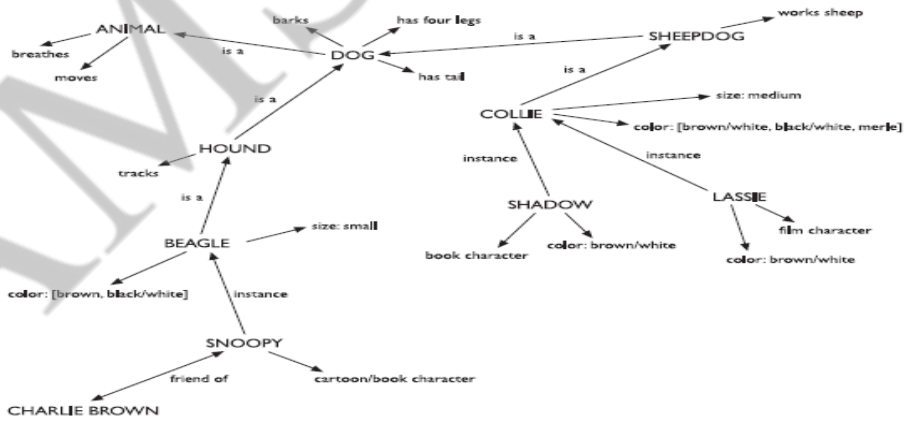


Figure 1.11 Long-term memory may store information in a semantic network

A **frame is instantiated when the slots are filled with appropriate values. Frames and scripts can be linked together in networks to represent hierarchical structured knowledge.** Returning to the 'dog' domain, a frame-based representation of the knowledge may look something like Figure 1.12

Figure 1.12 A frame-based representation of knowledge

Scripts attempt to model the representation of stereotypical knowledge about situations.

Consider the following sentence:

John took his dog to the surgery. After seeing the vet, he left.

From our knowledge of the activities of dog owners and vets, we may fill in a substantial amount of detail. The animal was ill. The vet examined and treated the animal. John paid for the treatment before leaving.

A script represents ***this default or stereotypical information,*** allowing us to interpret partial descriptions or cues fully. A script comprises a number of elements, which, like slots, can be filled with appropriate information:

Entry conditions Conditions that must be satisfied for the script to be activated.
Result Conditions that will be true after the script is terminated.
Props Objects involved in the events described in the script.
Roles Actions performed by particular participants.
Scenes The sequences of events that occur.
Tracks A variation on the general pattern representing an alternative scenario.

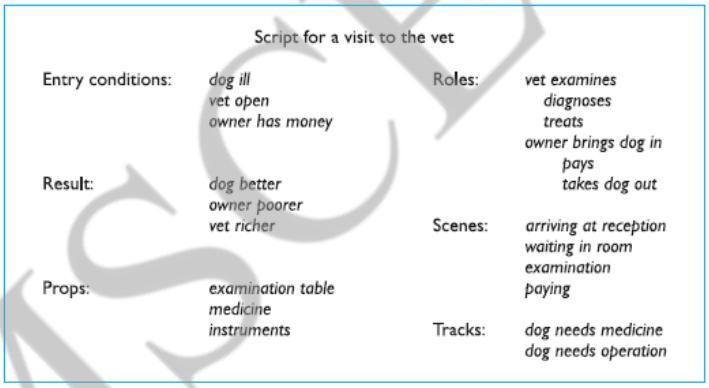


Figure 1.13 A script for visiting the vet

An example script for going to the vet is shown in Figure 1.13

A final type of knowledge representation which we hold in memory is the representation of procedural knowledge, our knowledge of how to do something. A common model for this is the production system. Condition–action rules are stored in long-term memory. Information coming into short-term memory can match a condition in one of these rules and result in the action being executed. For example, a pair of production rules might be

IF dog is wagging tail
THEN pat dog
IF dog is growling
THEN run away

1.6.5 LONG-TERM MEMORY PROCESSES

There are three main activities related to long-term memory**:**

- ***storage or remembering of information***
- ***forgetting***
- ***information retrieval***.

We shall consider each of these in turn.

First, how does information get into long-term memory and how can we improve this process? Information from short-term memory is stored in long-term memory by ***rehearsal***. The repeated exposure to a stimulus or the rehearsal of a piece of information transfers it into long-term memory.

This process can be optimized in a number of ways. Ebbinghaus performed numerous experiments on memory, using himself as a subject . In these experiments he tested his ability to learn and repeat nonsense syllables, comparing his recall minutes, hours and days after the learning process.

He discovered that the ***amount learned was directly proportional to the amount of time spent learnin***g. This is known as the ***total time hypothesis***. However, experiments by Baddeley andothers suggest that learning time is most effective if it is distributed over time.

For example, in an experiment in which Post Office workers were taught to type ,those whose training period was divided into weekly sessions of one hour performed better than those who spent two or four hours a week learning (although the former obviously took more weeks to complete their training). ***This is known as the distribution of practice effect.***

However, repetition is not enough to learn information well. If information is not meaningful it is more difficult to remember. This is illustrated by the fact that it is more difficult to remember a set of words representing concepts than a set of words representing objects.

First try to remember the words in list A and test yourself.

List A: Faith Age Cold Tenet Quiet Logic Idea Value Past Large
Now try list B.
List B: Boat Tree Cat Child Rug Plate Church Gun Flame Head

The second list was probably easier to remember than the first since you could visualize the objects in the second list. if information is meaningful and familiar, it can be related to existing structures and more easily incorporated into memory.

There are two main theories of forgetting

- ***Decay***
- ***Interference***

1. ***The first theory suggests that the information held in long-term memory may eventually be forgotten. E***bbinghaus concluded from his experiments with nonsense syllables that information in memory decayed logarithmically, that is that it was lost rapidly to begin with, and then more slowly. ***Jost's law, which follows from this, states that if two memory traces are equally strong at a given time the older one will be more durable.***

2. *The* ***second theory is that information is lost from memory through interference. If we acquire new information it causes the loss of old information. This is termed retroactive interference.***

A common example of this is the fact that if you change telephone numbers, learning your new number makes it more difficult to remember your old number. This is because the new association masks the old. However***, sometimes the old memory trace breaks through and interferes with new information. This is called proactive inhibition***.

First, proactive inhibition demonstrates the recovery of old information even after it has been 'lost' by interference. Secondly, there is the 'tip of the tongue' experience, which indicates that some information is present but cannot be satisfactorily accessed. Thirdly, information may not be recalled but may be recognized, or may be recalled only with prompting.

This leads us to the third process of memory: information retrieval. Here we need to distinguish between two types of information retrieval, recall and recognition.

In recall the information is reproduced from memory. In recognition, the presentation of the information provides the knowledge that the information has been seen before. Recognition is the less complex cognitive activity since the information is provided as a cue.

However, recall can be assisted by the provision of retrieval cues, which enable the subject quickly to access the information in memory. One such cue is the use of categories.

In an experiment subjects were asked to recall lists of words, some of which were organized into categories and some of which were randomly organized. The words that were related to a category were easier to recall than the others. Recall is even more successful if subjects are allowed to categorize their own lists of words during learning. For example, consider the following list of words:

child red plane dog friend blood cold tree big angry

Now make up a story that links the words using as vivid imagery as possible. Now try to recall as many of the words as you can. Did you find this easier than the previous experiment where the words were unrelated?

THINKING: REASONING AND PROBLEM SOLVING

Thinking can require different amounts of knowledge. Some thinking activities are very directed and the knowledge required is constrained. Others require vast amounts of knowledge from different domains. For example, performing a subtraction calculation requires a relatively small amount of knowledge, from a constrained domain, whereas understanding newspaper headlines demands knowledge of politics, social structures, public figures and world events.

Reasoning

Reasoning is the process by which we use the knowledge we have to draw conclusions or infer something new about the domain of interest. There are a number of different types of reasoning: deductive, inductive and abductive.

Deductive reasoning

Deductive reasoning derives the logically necessary conclusion from the given premises.

For example,

If it is Friday then she will go to work
It is Friday
Therefore she will go to work.

It is important to note that this is the *logical* conclusion from the premises; it does not necessarily have to correspond to our notion of truth. So, for example,

If it is raining then the ground is dry
It is raining
Therefore the ground is dry.

is a perfectly valid deduction, even though it conflicts with our knowledge of what is true in the world. Deductive reasoning is therefore often misapplied.

Given the premises
Some people are babies
Some babies cry

many people will infer that 'Some people cry'. This is in fact an invalid deduction since we are not told that all babies are people. It is therefore logically possible that the babies who cry are those who are not people. It is at this point, where truth and validity clash, that human deduction is poorest.

We assume a certain amount of shared knowledge in our dealings with each other, which in turn allows us to interpret the inferences and deductions implied by others. If validity rather than truth was preferred, all premises would have to be made explicit

Inductive reasoning

Induction is generalizing from cases **we have seen to infer information about cases we have not seen**.

For example, if every elephant we have ever seen has a trunk, we infer that all elephants have trunks. Of course, this inference is unreliable and cannot be proved to be true; it can only be proved to be false.

We can disprove the inference simply by producing an elephant without a trunk. However, we can never prove it true because, no matter how many elephants with trunks we have seen or are known to exist, the next one we see may be trunkless. The best that we can do is gather evidence to support our inductive inference.

In spite of its unreliability, induction is a useful process, which we use constantly in learning about our environment. We can never see all the elephants that have ever lived or will ever live, but we have certain knowledge about elephants which we are prepared to trust for all practical purposes, which has largely been inferred by induction.

Even if we saw an elephant without a trunk, we would be unlikely to move from our position that 'All elephants have trunks', since we are better at using positive than negative evidence. This is illustrated in an experiment first devised by Wason

You are presented with four cards as in Figure 1.14. Each card has a number on one side and a letter on the other. Which cards would you need to pick up to test the truth of the statement 'If a card has a vowel on one side it has an even number on the other'?

A common response to this (was it yours?) is to check the E and the 4. However, this uses only positive evidence. In fact, to test the truth of the statement we need to check negative evidence: if we can find a card which has an odd number on one side and a vowel on the other we have disproved the statement. We must therefore check E and 7. (It does not matter what is on the other side of the other cards: the statement does not say that all even numbers have vowels, just that all vowels have even numbers.)

Figure 1.14 Wason's cards

Abductive reasoning

Abduction reasons from a fact to the action or state that caused it. This is the method we use to derive explanations for the events we observe.

For example, suppose we know that Sam always drives too fast when she has been drinking.

If we see Sam driving too fast we may infer that she has been drinking.

Of course, this too is unreliable since there may be another reason why she is driving fast: she may have been called to an emergency, for example. In spite of its unreliability, it is clear that people do infer explanations in this way, and hold onto them until they have evidence to support an alternative theory or explanation.

This can lead to problems in using interactive systems. If an event always follows an action, the user will infer that the event is caused by the action unless evidence to the contrary is made available.

Problem solving

If **reasoning is a means of inferring new information from what is already known, problem solving is the process of finding a solution to an unfamiliar task, using the knowledge we have**.

Human problem solving is characterized by the **ability to adapt the information we have to deal with new situations**. However, often solutions seem to be original and creative. There are a number of different views of how people solve problems. The earliest, dating back to the first half of the twentieth century, is the Gestalt view that problem solving involves both reuse of knowledge and insight.

This has been largely superseded but the questions it was trying to address remain and its influence can be seen in later research. A second major theory, proposed in the 1970s by Newell and Simon, was the problem space theory, which takes the view that the mind is a limited information processor. Later variations on this drew on the earlier theory and attempted to reinterpret Gestalt theory in terms of information processing theories.

Gestalt theory

Gestalt psychologists **were answering the claim, made by behaviorists, that problem solving is a matter of reproducing known responses or trial and error. P**roblem solving is both productive and reproductive. ***Reproductive problem solving draws on previous experience as the behaviorists claimed, but productive problem solving involves insight and restructuring of the problem. I***ndeed, reproductive problem solving could be a hindrance to finding a solution, since a person may 'fixate' on the known aspects of the problem and so be unable to see novel interpretations that might lead to a solution

Gestalt psychologists backed up their claims with experimental evidence. Kohler provided evidence of apparent insight being demonstrated by apes, which he observed joining sticks together in order to reach food outside their cages . However, this was difficult to verify since the apes had once been wild and so could have been using previous knowledge.

Other experiments observed human problem-solving behavior. One well-known example of this is Maier's pendulum problem . The problem was this: the subjects were in a room with two pieces of string hanging from the ceiling. Also in the room were other objects including pliers, poles and extensions. The task set was to tie the pieces of string together. However, they were too far apart to catch hold of both at once.

The movement of the string had given insight and allowed the subjects to see the problem in a new way. The experiment also illustrates fixation: subjects were initially unable to see beyond their view of the role or use of a pair of pliers.

Problem space theory

Newell and Simon proposed that problem solving centers on the problem space. ***The problem space comprises problem states, and problem solving involves generating these states using legal state transition operators. The problem has an initial state and a goal state and people use the operators to move from the former to the latter. Such problem spaces may be huge, and so heuristics are employed to select appropriate operators to reach the goal.***

One such heuristic is ***means–ends analysis. In means–ends analysis the initial state is compared with the goal state and an operator chosen to reduce the difference between the two***. For example, imagine you are reorganizing your office and you want to move your desk from the north wall of the room to the window.

Your initial state is that the desk is at the north wall. The goal state is that the desk is by the window. The main difference between these two is the location of your desk. You have a number of operators which you can apply to moving things: you can carry them or push them or drag them, etc. However, you know that to carry something it must be light and that your desk is heavy. You therefore have a new subgoal: to make the desk light. Your operators for this may involve removing drawers, and so on.

An important feature of Newell and Simon's model is that it operates within the constraints of the human processing system, and so searching the problem space is limited by the capacity of short-term memory, and the speed at which information can be retrieved.

Within the problem space framework, experience allows us to solve problems more easily since we can structure the problem space appropriately and choose operators efficiently.

Newell and Simon's theory, and their General Problem Solver model which is based on it, have largely been applied to problem solving in well-defined domains, for example solving puzzles. These problems may be unfamiliar but the knowledge that is required to solve them is present in the statement of the problem and the expected solution is clear.

In real-world problems finding the knowledge required to solve the problem may be part of the problem, or specifying the goal may be difficult.

Problems such as these require significant domain knowledge: for example, to solve a programming problem you need knowledge of the language and the domain in which the program operates. In this instance specifying the goal clearly may be a significant part of solving the problem.

Analogy in problem solving

A third element of problem solving is the use of analogy. ***ANALOGY is done by mapping knowledge relating to a similar known domain to the new problem – called analogical mapping.***

Similarities between the known domain and the new one are noted and operators from the known domain are transferred to the new one.This process has been investigated using analogous stories. Gick and Holyoak gave subjects the following problem:

A doctor is treating a malignant tumor. In order to destroy it he needs to blast it with high-intensity rays. However, these will also destroy the healthy tissue surrounding the tumor. If he lessens the rays' intensity the tumor will remain. How does he destroy the tumor?

The solution to this problem is to fire low-intensity rays from different directions converging on the tumor.

That way, the healthy tissue receives harmless low intensity rays while the tumor receives the rays combined, making a high-intensity dose. The investigators found that only 10% of subjects reached this solution without help. However, this rose to 80% when they were given this analogous story and told that it may help them:

A general is attacking a fortress. He can't send all his men in together as the roads are mined to explode if large numbers of men cross them. He therefore splits his men into small groups and sends them in on separate roads.

Skill acquisition

All of the problem solving that we have considered so far **has concentrated on handling unfamiliar problems**. However, for much of the time, the problems that we face are not completely new. Instead, we gradually acquire skill in a particular domain area. But how is such skill acquired and what difference does it make to our problem-solving performance? We can gain insight into how skilled behavior works, and how skills are acquired, by considering the difference between novice and expert behavior in given domains

A commonly studied domain is chess playing. It is particularly suitable since it lends itself easily to representation in terms of problem space theory. The initial state is the opening board position; the goal state is one player checkmating the other; operators to move states are legal moves of chess. It is therefore possible to examine skilled behavior within the context of the problem space theory of problem solving.

ACT* identifies three basic levels of skill:

1. The learner uses general-purpose rules which interpret facts about a problem.

This is slow and demanding on memory access.

2. The learner develops rules specific to the task.

3. The rules are tuned to speed up performance.

General mechanisms are provided to account for the transitions between these levels. For example, **proceduralization is a mechanism to move from the first to the second**. It removes the parts of the rule which demand memory access and replaces variables with specific values.

Generalization, on the other hand, is a mechanism which moves from the second level to the third. It generalizes from the specific cases to general properties of those cases. Commonalities between rules are condensed to produce a general-purpose rule.

These are best illustrated by example. Imagine you are learning to cook. Initially you may have a general rule to tell you how long a dish needs to be in the oven, and a number of explicit representations of dishes in memory. You can instantiate the rule by retrieving information from memory.

IF cook[type, ingredients, time]
THEN
cook for: time
cook[casserole, [chicken,carrots,potatoes], 2 hours]
cook[casserole, [beef,dumplings,carrots], 2 hours]
cook[cake, [flour,sugar,butter,eggs], 45 mins]

Gradually your knowledge becomes proceduralized and you have specific rules for

each case:

IF type is casserole
AND ingredients are [chicken,carrots,potatoes]
THEN
cook for: 2 hours
IF type is casserole
AND ingredients are [beef,dumplings,carrots]
THEN
cook for: 2 hours
IF type is cake
AND ingredients are [flour,sugar,butter,eggs]
THEN
cook for: 45 mins

Finally, you may generalize from these rules to produce general-purpose rules, which exploit their commonalities:

IF type is casserole

AND ingredients are ANYTHING

THEN

cook for: 2 hours

The first stage uses knowledge extensively. The second stage relies upon known procedures. The third stage represents skilled behavior. Such behavior may in fact become automatic and as such be difficult to make explicit.

Errors and mental models

Human capability for interpreting and manipulating information is quite impressive. However, we do make mistakes. Some are trivial, resulting in no more than temporary inconvenience or annoyance. Others may be more serious, requiring substantial effort to correct. Occasionally an error may have catastrophic effects, as

we see when 'human error' results in a plane crash or nuclear plant leak.

Why do we make mistakes and can we avoid them?

There are several different types of error.

If a pattern of behavior has become automatic and we change some aspect of it, the more familiar pattern may break through and cause an error. A familiar example of this is where we intend to stop at the shop on the way home from work but in fact drive past. Here, the activity of driving home is the more familiar and overrides the less familiar intention. Other errors result from an incorrect understanding, or model, of a situation or system. People build their own theories to understand the causal behavior of systems.

These have been termed mental models. They have a number of characteristics.

Mental models are often partial: the person does not have a full understanding of the working of the whole system. They are unstable and are subject to change. They can be internally inconsistent, since the person may not have worked through the logical consequences of their beliefs. They are often unscientific and may be based on superstition rather than evidence. Often they are based on an incorrect interpretation of the evidence.

EMOTION

Our emotional response to situations affects how we perform. For example, positive emotions enable us to think more creatively, to solve complex problems, whereas negative emotion pushes us into narrow, focussed thinking. A problem that may be easy to solve when we are relaxed, will become difficult if we are frustrated or afraid.

Psychologists have studied emotional response for decades and there are many theories as to what is happening when we feel an emotion and why such a response occurs.

More than a century ago, William James proposed what has become known as the James–Lange theory (Lange was a contemporary of James whose theories were similar): that emotion was the interpretation of a physiological response, rather than the other way around. So while we may feel that we respond to an emotion, James contended that we respond physiologically to a stimulus and interpret that as emotion:

Common sense says, we lose our fortune, are sorry and weep; we meet a bear, are frightened and run; we are insulted by a rival, are angry and strike. The hypothesis here . . . is that we feel sorry because we cry, angry because we strike, afraid because we tremble.

Schachter and Singer proposed a third interpretation: that **emotion results from a person evaluating physical responses in the light of the whole situation**. So whereas the same physiological response can result from a range of different situations, the emotion that is felt is based on a cognitive evaluation of the circumstance and will depend on what the person attributes this to. So the same physiological response of a pounding heart will be interpreted as excitement if we are in a competition and fear if we find ourselves under attack.

Whatever the exact process, what is clear is that emotion involves both physical and cognitive events. Our body responds biologically to an external stimulus and we interpret that in some way as a particular emotion.

That biological response – known as affect – changes the way we deal with different situations, and this has an impact on the way we interact with computer systems.

As Donald Norman says:

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

A typical computer system

Consider a typical computer setup as shown in Figure 2.1

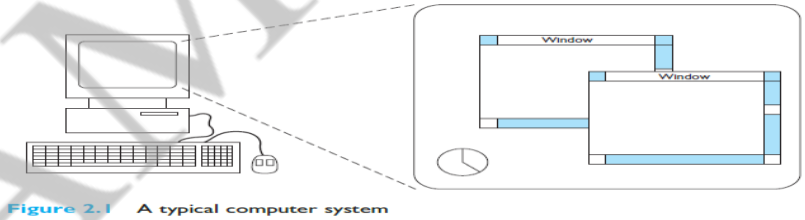


Figure 2.1 A typical computer system

There is the computer 'box' itself, a keyboard, a mouse and a color screen. The screen layout is shown alongside it. Some of this variation is driven by different hardware configurations: desktop use, laptop computers, PDAs (personal digital assistants).

Levels of interaction – batch processing

In the early days of computing, information was entered into the computer in a large mass – batch data entry. There was minimal interaction with the machine: the user would simply dump a pile of punched cards onto a reader, press the start button, and then return a few hours later.

With batch processing the interactions take place over hours or days. In contrast the typical desktop computer system has interactions taking seconds or fractions of a second (or with slow web pages sometimes minutes!).

Richer interaction – everywhere

Information appliances are putting internet access or dedicated systems onto the fridge, microwave and washing machine: to automate shopping, give you email in your kitchen or simply call for maintenance when needed. We carry with us WAP phones and smartcards, have security systems that monitor us and web cams that show our homes to the world. Is Figure 2.1 really the typical computer system or is it really more like Figure 2.2

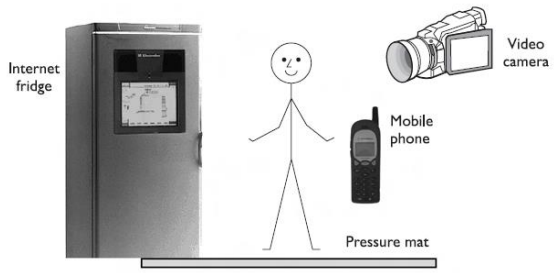


Figure 2.2 A typical computer system? Photo courtesy Electrolux

TEXT ENTRY DEVICES

The most obvious means of text entry is the plain keyboard, but there are several variations on this: different keyboard layouts, 'chord' keyboards that use combinations of fingers to enter letters, and phone key pads.

The alphanumeric keyboard

The keyboard is still one of the most common input devices in use today. It is used for entering textual data and commands. The vast majority of keyboards have a standardized layout, and are known by the first six letters of the top row of alphabetical keys, QWERTY. There are alternative designs which have some advantages over the QWERTY layout, but these have not been able to overcome the vast technological inertia of the QWERTY keyboard. These alternatives are of two forms: 26 key layouts and chord keyboards. A 26 key layout rearranges the order of the alphabetic keys, putting the most commonly used letters under the strongest fingers, or adopting simpler practices. In addition to QWERTY, we will discuss two 26 key layouts, alphabetic and DVORAK, and chord keyboards

Ease of learning – alphabetic keyboard

One of the most obvious layouts to be produced is the alphabetic keyboard, in which the letters are arranged alphabetically across the keyboard. It might be expected that such a layout would make it quicker for untrained typists to use, but this is not the case. Studies have shown that this keyboard is not faster for properly trained typists, as we may expect, since there is no inherent advantage to this layout. And even for novice or occasional users, the alphabetic layout appears to make very little difference to the speed of typing

The QWERTY keyboard

The layout of the digits and letters on a QWERTY keyboard is fixed (see Figure 2.3), but non-alphanumeric keys vary between keyboards. For example, there is a difference between key assignments on British and American keyboards (in particular, above the 3 on the UK keyboard is the pound sign £, whilst on the US keyboard there is a dollar sign $). The standard layout is also subject to variation in the placement of brackets, backslashes and suchlike. In addition different national keyboards include accented letters and the traditional French layout places the main letters in different locations – the top line starts AZERTY. The QWERTY arrangement of keys is not optimal for typing, however. The reason for the layout of the keyboard in this fashion can be traced back to the days of mechanical typewriters. the QWERTY keyboard

remains the dominant layout. There is also a large investment in current keyboards, which would all have to be either replaced at great cost, or phased out, with the subsequent requirement for people to be proficient on both keyboards.

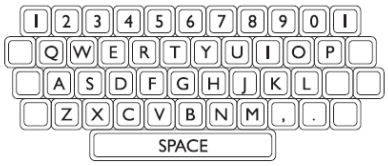


Figure 2.3 The standard QWERTY keyboard

Ergonomics of use – DVORAK keyboard and split designs

The DVORAK keyboard uses a similar layout of keys to the QWERTY system, but assigns the letters to different keys. Based upon an analysis of typing, the keyboard is designed to help people reach faster typing speeds. It is biased towards right-handed people, in that 56% of keystrokes are made with the right hand. The layout of the keys also attempts to ensure that the majority of keystrokes alternate between hands, thereby increasing the potential speed. By keeping the most commonly used keys on the home, or middle, row, 70% of keystrokes are made without the typist having to stretch far, thereby reducing fatigue and increasing keying speed. The layout also aims to minimize the number of keystrokes made with the weak fingers. Many of these requirements are in conflict, and the DVORAK keyboard represents one possible solution. Experiments have shown that there is a speed improvement of between 10 and 15%, coupled with a reduction in user fatigue due to the increased ergonomic layout of the keyboard

Alphabetic keyboard

One of the most obvious layouts to be produced is the alphabetic keyboard, in which the letters are arranged alphabetically across the keyboard. It might be expected that such a layout would make it quicker for untrained typists to use, but this is not the case. Studies have shown that this keyboard is not faster for properly trained typists, as we may expect, since there is no inherent advantage to this layout. And even for novice or occasional users, the alphabetic layout appears to make very little difference to the speed of typing. These keyboards are used in some pocket electronic personal organizers, perhaps because the layout looks simpler to use than the QWERTY one. Also, it dissuades people from attempting to use their touch-typing skills on a very small keyboard and hence avoids criticisms of difficulty of use.

Dvorak Keyboard

Attempts at designing alternative keyboards that are more efficient and quicker to use have produced, among others, the Dvorak and Alphabetic boards. The Dvorak board, first patented in 1932, was designed using the following principles: Layout is arranged on the basis of frequency of usage of letters and the frequency of letter pattern and sequences in the English language. All vowels and the most frequently used consonants are on the second or home row, so that something like 70% of common words are typed on this row alone. Faster operation is made possible by tapping with fingers on alternate hands (particularly the index fingers) rather than by repetitive tapping with one finger and having the majority of keying assigned to one hand, as in the QWERTY keyboard, which favors left-handers. Since the probability of vowels and consonants altering is very high, all vowels are typed with the left hand and frequent home row consonants with the right.

Chord keyboards

Chord keyboards are significantly different from normal alphanumeric keyboards. Only a few keys, four or five, are used (see Figure 2.4) and letters are produced by pressing one or more of the keys at once. For example, in the *Microwriter*, the pattern of multiple keypresses is chosen to reflect the actual letter shape.

Such keyboards have a number of advantages. They are extremely compact: simply reducing the size of a conventional keyboard makes the keys too small and close together, with a correspondingly large increase in the difficulty of using it. The learning time for the keyboard is supposed to be fairly short – of the order of a few

hours – but social resistance is still high. Moreover, they are capable of fast typing speeds in the hands (or rather hand!) of a competent user. Chord keyboards can also be used where only one-handed operation is possible, in cramped and confined

conditions. In particular, courtroom stenographers use a special form of two-handed chord keyboard and associated shorthand to enter text at full spoken speed.

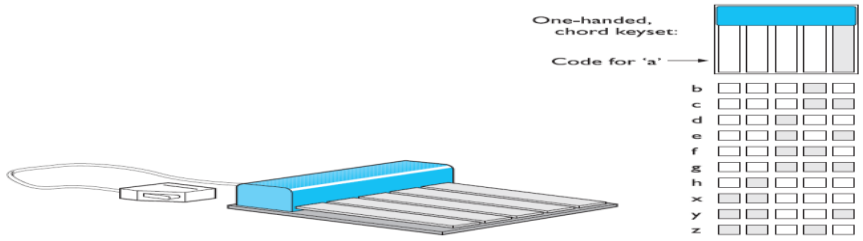


Figure 2.4 A very early chord keyboard (left) and its lettercodes (right)

Typical key mapping:
1 – space, comma, etc. (varies)
2 – a b c
3 – d e f
4 – g h i
5 – j k l
6 – m n o
7 – p q r s
8 – t u v
9 – w x y z
0 – +, &, etc.

Figure 2.5 Mobile phone keypad. Source: Photograph by Alan Dix (Ericsson phone)

Phone pad and T9 entry

With mobile phones being used for SMS text messaging and WAP the phone keypad has become an important form of text input. Unfortunately a phone only has digits 0–9, not a full alphanumeric keyboard. To overcome this for text input the numeric keys are usually pressed several times – Figure 2.5 shows a typical mapping of digits to letters. For example, the 3 key has 'def ' on it. If you press the key once you get a 'd', if you press 3 twice you get an 'e', if you press it three times you get an 'f '.

The main number-to-letter mapping is standard, but punctuation and accented letters differ between phones. Also there needs to be a way for the phone to distinguish, say, the 'dd' from 'e'. On some phones you need to pause for a short period between successive letters using the same key, for others you press an additional key (e.g. '#').

Most phones have at least two *modes* for the numeric buttons: one where the keys mean the digits (for example when entering a phone number) and one where they mean letters (for example when typing an SMS message). Some have additional modes to make entering accented characters easier. Also a special mode or setting is needed for capital letters although many phones use rules to reduce this, for example automatically capitalizing the initial letter in a message and letters following full stops, question marks and exclamation marks.

Handwriting recognition

Handwriting is a common and familiar activity, and is therefore attractive as a method of text entry. If we were able to write as we would when we use paper, but with the computer taking this form of input and converting it to text, we can see that it is an intuitive and simple way of interacting with the computer. However, there are a number of disadvantages with handwriting recognition. Current technology is still fairly inaccurate and so makes a significant number of mistakes in recognizing letters, though it has improved rapidly. Moreover, individual differences in handwriting are enormous, and make the recognition process even more difficult. The most significant information in handwriting is not in the letter shape itself but in the stroke information – the way in which the letter is drawn. This means that devices which support handwriting recognition must capture the stroke information, not just the final character shape. Because of this, online recognition is far easier than reading handwritten text on paper.

Speech recognition

Speech recognition is a promising area of text entry, but it has been promising for a number of years and is still only used in very limited situations. There is a natural enthusiasm for being able to talk to the machine and have it respond to commands, since this form of interaction is one with which we are very familiar. Successful recognition rates of over 97%

have been reported, but since this represents one letter in error in approximately every 30, or one spelling mistake every six or so words, this is stoll unacceptible (*sic*)!

Moreover, since every person speaks differently, the system has to be trained and tuned to each new speaker, or its performance decreases. Strong accents, a cold or emotion can also cause recognition problems, as can background noise. This leads us on to the question of practicality within an office environment: not only may the background level of noise cause errors, but if everyone in an open-plan office were to talk to their machine, the level of noise would dramatically increase, with associated difficulties. Confidentiality would also be harder to maintain.

Despite its problems, speech technology has found niche markets: telephone information systems, access for the disabled, in hands-occupied situations (especially military) and for those suffering RSI.

POSITIONING, POINTING AND DRAWING

THE MOUSE

The mouse has become a major component of the majority of desktop computer systems sold today, and is the little box with the tail connecting it to the machine in our basic computer system picture (Figure2.6).

- It is a small, palm-sized box housing a weighted ball – as the box is moved over the table top, the ball is rolled by the table and so rotates inside the housing. This rotation is detected by small rollers that are in contact with the ball, and these adjust the values of potentiometers.
- The changing values of these potentiometers can be directly related to changes in position of the ball. The potentiometers are aligned in different directions so that they can detect both horizontal and vertical motion.
- The relative motion information is passed to the computer via a wire attached to the box, or in some cases using wireless or infrared, and moves a pointer on the screen, called the *cursor*.

The whole arrangement tends to look rodent-like, with the box acting as the body and the wire as the tail; hence the term 'mouse' The mouse operates in a planar fashion, moving around the desktop, and is an indirect input device, since a transformation is required to map from the horizontal nature of the desktop to the vertical alignment of the screen. Left–right motion is directly mapped, whilst up–down on the screen is achieved by moving the mouse away–towards the user

The mouse was developed around 1964 **by Douglas C. Engelbart**, and a photograph of the first prototype is shown in Figure2.6 This used two wheels that slid across the desktop and transmitted *x–y* coordinates to the computer. The housing was carved in wood, and has been damaged, exposing one of the wheels.

The original design actually offers a few advantages over today's more sleek versions: by tilting it so that only one wheel is in contact with the desk, pure vertical or horizontal motion can be obtained.

Figure 2.6 The first mouse. Photograph courtesy of Douglas Engelbart and Bootstrap Institute

OPTICAL MICE

Optical mice work differently from mechanical mice. A light-emitting diode emits a weak red light from the base of the mouse. This is reflected off a special pad with a metallic grid-like pattern upon which the mouse has to sit, and the fluctuations in reflected intensity as the mouse is moved over the gridlines are recorded by a sensor in the base of the mouse and translated into relative x, y motion.

There have been experiments with a device called the foot mouse. As the name implies, it is a foot-operated device, although more akin to an isometric joystick than a mouse. The cursor is moved by foot pressure on one side or the other of a pad. This allows one to dedicate hands to the keyboard.

Interestingly foot pedals are used heavily in musical instruments including pianos, electric guitars, organs and drums and also in mechanical equipment including cars, cranes, sewing machines and industrial controls

TOUCHPAD

Touch pads ***are touch-sensitive tablets usually around 2–3 inches (50–75 mm) square.*** They were first used extensively in Apple Powerbook portable computers but are now used in many other notebook computers and can be obtained separately to replace the mouse on the desktop. They are operated by stroking a finger over their surface, rather like using a simulated trackball.

Because they are small it may require several strokes to move the cursor across the screen. This can be improved by using acceleration settings in the software linking the trackpad movement to the screen movement. Rather than having a fixed ratio of pad distance to screen distance, this varies with the speed of movement

TRACKBALL AND THUMBWHEEL

The trackball is really just an upside-down mouse! A weighted ball faces upwards and is rotated inside a static housing, the motion being detected in the same way as for a mechanical mouse, and the relative motion of the ball moves the cursor. Because of this, the trackball requires no additional space in which to operate, and is therefore a very compact device. It is an indirect device, and requires separate buttons for selection.

Thumbwheels are different in that they have two orthogonal dials to control the cursor position. Such a device is very cheap, but slow, and it is difficult to manipulate the cursor in any way other than horizontally or vertically. This limitation can sometimes be a useful constraint in the right application. For instance, in CAD the designer is almost always concerned with exact verticals and horizontals, and a device that provides such constraints is very useful, which accounts for the appearance of thumbwheels in CAD systems.

Another successful application for such a device has been in a drawing game such as Etch-a-Sketch in which straight lines can be created on a simple screen, since the predominance of straight lines in simple drawings means that the motion restrictions are an advantage rather than a handicap.

JOYSTICK AND KEYBOARD NIPPLE

The joystick is an indirect input device, taking up very little space. Consisting of a small palm-sized box with a stick or shaped grip sticking up from it, the joystick is a simple device with which movements of the stick cause a corresponding movement of the screen cursor. There are two types of joystick: the absolute and the isometric.

In the absolute joystick, movement is the important characteristic, since the position of the joystick in the base corresponds to the position of the cursor on the screen.

In the isometric joystick, the pressure on the stick corresponds to the velocity of the cursor, and when released, the stick returns to its usual upright centered position.

This type of joystick is also called the velocity-controlled joystick, for obvious reasons. The buttons are usually placed on the top of the stick, or on the front like a trigger. Joysticks are inexpensive and fairly robust, and for this reason they are often found in computer games.

A smaller device but with the same basic characteristics is used on many laptop computers to control the cursor. Some older systems had a variant of this called the keymouse, which was a single key

TOUCH-SENSITIVE SCREENS (TOUCHSCREENS)

Touch screens are another method of allowing the user to point and select objects on the screen, but they are much more direct than the mouse, as they detect the presence of the user's finger, or a stylus, on the screen itself. They work in one of a number of different ways: by the finger (or stylus) interrupting a matrix of light beams, or by capacitance changes on a grid overlaying the screen, or by ultrasonic reflections. Because the user indicates exactly which item is required by pointing to it, no mapping is required and therefore this is a direct device.

The touch screen is very fast, and requires no specialized pointing device. It is especially good for selecting items from menus displayed on the screen. Because the screen acts as an input device as well as an output device, there is no separate

hardware to become damaged or destroyed by dirt; this makes touch screens suitable for use in hostile environments. They are also relatively intuitive to use and have been used successfully as an interface to information systems for the general public.

They suffer from a number of disadvantages, however. Using the finger to point is not always suitable, as it can leave greasy marks on the screen, and, being a fairly blunt instrument, it is quite inaccurate.

Stylus and light pen

An older technology that is used in the same way is the light pen. The pen is connected to the screen by a cable and, in operation, is held to the screen and detects a burst of light from the screen phosphor during the display scan.

The light pen can therefore address individual pixels and so is much more accurate than the touch screen Both stylus and light pen can be used for fine selection and drawing, but both can be tiring to use on upright displays and are harder to take up and put down when used together with a keyboard.

Stylus, light pen and touch screen are all very direct in that the relationship between the device and the thing selected is immediate. In contrast, mouse, touchpad, joystick and trackball all have to map movements on the desk to cursor movement on the screen.

However, the direct devices suffer from the problem that, in use, the act of pointing actually obscures the display, making it harder to use, especially if complex detailed selections or movements are required in rapid succession. This means that screen designs have to take into account where the user's hand will be.

For example, you may want to place menus at the bottom of the screen rather than the top. Also you may want to offer alternative layouts for right-handed and left-handed users.

DIGITIZING TABLET

The digitizing tablet is a more ***specialized device typically used for freehand drawing, but may also be used as a mouse substitute.*** Some highly accurate tablets, usually using a puck (a mouse-like device), are used in special applications such as digitizing information for maps.

The tablet provides positional information by measuring the position of some device on a special pad, or tablet, and can work in a number of ways. The resistive tablet detects point contact between two separated conducting sheets.

It has advantages in that it can be operated without a specialized stylus – a pen or the user's finger is sufficient. The magnetic tablet detects current pulses in a magnetic field using a small loop coil housed in a special pen.

There are also capacitative and electrostatic tablets that work in a similar way. The sonic tablet is similar to the above but requires no special surface. An ultrasonic pulse is emitted by a special pen which is detected by two or more microphones which then triangulate the pen position. This device can be adapted to provide 3D input, if required.

Digitizing tablets are capable of high resolution, and are available in a range of sizes. Sampling rates vary, affecting the resolution of cursor movement, which gets progressively finer as the sampling rate increases. The digitizing tablet can be used to detect relative motion or absolute motion, but is an indirect device since there is a mapping from the plane of operation of the tablet to the screen. It can also be used for text input; if supported by character recognition software, handwriting can be interpreted.

Problems with digitizing tablets are that they require a large amount of desk space, and may be awkward to use if displaced to one side by the keyboard.

EYEGAZE

Some systems require you to wear special glasses or a small head-mounted box, others are built into the screen or sit as a small box below the screen. A low-power laser is shone into the eye and is reflected off the retina. The reflection changes as the angle of the eye alters, and by tracking the reflected beam the eyegaze system can determine the direction in which the eye is looking. The system needs to be calibrated, typically by staring at a series of dots on the screen, but thereafter can be used to move the screen cursor or for other more specialized uses.

Eyegaze is a very fast and accurate device, but the more accurate versions can be expensive. It is fine for selection but not for drawing since the eye does not move in smooth lines. Also in real applications it can be difficult to distinguish deliberately gazing at something and accidentally glancing at it.

CURSOR KEYS AND DISCRETE POSITIONING

All of the devices we have discussed are capable of giving near continuous 2D positioning, with varying degrees of accuracy. For many applications we are only interested in positioning within a sequential list such as a menu or amongst 2D cells as in a spreadsheet. Even for moving within text discrete up/down left/right keys can sometimes be preferable to using a mouse.

Cursor keys are available on most keyboards. Four keys on the keyboard are used to control the cursor, one each for up, down, left and right. There is no standardized layout for the keys. Some layouts are shown in Figure2.7, but the most common now is the inverted 'T'.

Cursor keys used to be more heavily used in character-based systems before windows and mice were the norm. However, when logging into remote machines such as web servers, the interface is often a virtual character-based terminal within a telnet window.

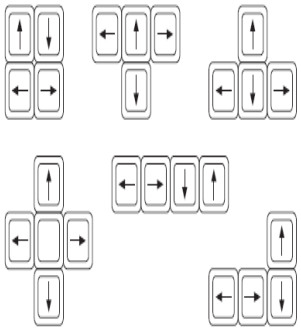


Figure 2.7 Various cursor key layouts

Small devices such as mobile phones, personal entertainment and television remote controls often require discrete control, either dedicated to a particular function such as volume, or for use as general menu selection. Figure2.7 shows examples of these.

The satellite TV remote control has dedicated '+/–' buttons for controlling volume and stepping between channels. It also has a central cursor pad that is used for on-screen menus. The mobile phone has a single central joystick-like device. This can be pushed left/right, up/down to navigate within the small 3 × 3 array of graphical icons as well as select from text menus.

DISPLAY DEVICES

CATHODE RAY TUBE

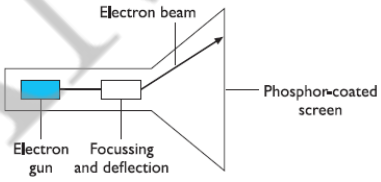


Figure 2.10 CRT screen

The cathode ray tube is the television-like computer screen still most common as we write this, but rapidly being displaced by flat LCD screens. It works in a similar way to a standard television screen.

A stream of electrons is emitted from an electron gun, which is then focused and directed by deflection magnetic fields. As the beam hits the phosphor-coated screen, the phosphor is phosphor- excited by the electrons and glows.

The coated screen electron beam is scanned from left to right, and then flicked back to rescan the next line, from top to bottom. Black and white screens are able to display grayscale by varying the intensity of the electron beam; color is achieved using more complex means.

Three electron guns are used, one each to hit red, green and blue phosphors. Combining these colors can produce many others, including white, when they are all fully on.

These three phosphor dots are focused to make a single point using a shadow mask, which is imprecise and gives color screens a lower resolution than equivalent monochrome screens.

The CRT is a cheap display device and has fast enough response times for rapid animation coupled with a high color capability. Note that animation does not necessarily means little creatures and figures running about on the screen, but refers in a more general sense to the use of motion in displays: moving the cursor, opening windows, indicating processor-intensive calculations, or whatever.

As screen resolution increased, however, the price rises. Because of the electron gun and focusing components behind the screen, CRTs are fairly bulky, though recent innovations have led to flatter displays in which the electron gun is not placed so that it fires directly at the screen, but fires parallel to the screen plane with the resulting beam bent through 90 degrees to his the screen.

HEALTH HAZARDS OF CRT DISPLAYS

Most people who habitually use computers are aware that screens can often cause eyestrain and fatigue; this is usually due to flicker, poor legibility or low contrast. There have also been many concerns relating to the emission of radiation from screens. These can be categorized as follows:

- X-rays which are largely absorbed by the screen (but not at the rear!)
- ultraviolet and infrared radiation from phosphors in insignificant levels
- radio frequency emissions, plus ultrasound (approximately 16 kHz)
- electrostatic field which leaks out through the tube to the user. The intensity is dependent on distance and humidity. This can cause rashes in the user electromagnetic fields (50 Hz to 0.5 MHz) which create induction currents in conductive materials, including the human body.

Two types of effects are attributed to this: in the visual system, a high incidence of cataracts in visual display unit (VDU) operators, and concern over reproductive disorders (miscarriages and birth defects).

Generally, there are a number of common-sense things that can be done to relieve strain and minimize any risk. These include

- not sitting too close to the screen
- not using very small fonts
- not looking at the screen for a long time without a break
- working in well-lit surroundings
- not placing the screen directly in front of a bright window.

Liquid Crystal Display

Liquid Crystal Displays are mostly used in personal organizer or laptop computers. It is a light, flat plastic screen. These displays utilize liquid crystal technology and are smaller, lighter and consume far less power than traditional CRTs. These are also commonly referred to as flat-panel displays. They have no radiation problems associated with them, and are matrix addressable, which means that individual pixels can be accessed without the need for scanning. This different technology can be used to replace the standard screen on a desktop computer, and this is now common. However, the particular characteristics of compactness, lightweight, and low power consumption have meant that these screens have created a large niche in the computer market by monopolizing the notebook and portable computer systems side.

Special displays

There are a number of other display technologies used in niche markets. The one you are most likely to see is the gas plasma display, which is used in large screens The random scan display, also known as the directed beam refresh, or

vector display, works differently from the bitmap display, also known as raster scan, Instead of scanning the whole screen sequentially and horizontally, the random scan draws the lines to be displayed directly. By updating the screen at at least 30 Hz to reduce flicker, the direct drawing of lines at any angle means that jaggies are not created, and higher resolutions are possible, up to 4096 ×

4096 pixels. Color on such displays is achieved using beam penetration technology, and is generally of a poorer quality. Eyestrain and fatigue are still a problem, and these displays are more expensive than raster scan ones, so they are now only used in niche applications.

The direct view storage tube is used extensively as the display for an analog storage oscilloscope, which is probably the only place that these displays are used in any great numbers. They are similar in operation to the random scan CRT but the image is maintained by flood guns which have the advantage of producing a stable display with no flicker. The screen image can be incrementally updated but not selectively erased; removing items has to be done by redrawing the new image on a completely erased screen. The screens have a high resolution, typically about 4096 × 3120 pixels, but suffer from low contrast, low brightness and a difficulty in displaying color.

Large displays and situated displays

Displays are no longer just things you have on your desktop or laptop. In shops and garages large screen adverts assault us from all sides.

There are several types of large screen display. Some use gas plasma technology to create large flat bitmap displays. These behave just like a normal screen except they are big and usually have the HDTV (high definition television) wide screen format which has an aspect ratio of 16:9 instead of the 4:3 on traditional TV and monitors.

Where very large screen areas are required, several smaller screens, either LCD or CRT, can be placed together in a video wall. These can display separate images, or a single TV or computer image can be split up by software or hardware so that each screen displays a portion of the whole and the result is an enormous image. This is the technique often used in large concerts to display the artists or video images during the performance.

The disadvantage of projected displays is that the presenter's shadow can often fall across the screen. Sometimes this is avoided in fixed lecture halls by using back projection. In a small room behind the screen of the lecture theatre there is a projector producing a right/left reversed image. The screen itself is a semi-frosted glass so that the image projected on the back can be seen in the lecture theatre. Because there are limits on how wide an angle the projector can manage without distortion, the size of the image is limited by the depth of the projection room behind, so these are less heavily used than front projection.

As well as for lectures and meetings, display screens can be used in various public places to offer information, link spaces or act as message areas. These are often called *situated displays* as they take their meaning from the location in which they are situated. These may be large screens where several people are expected to view or interact simultaneously, or they may be very small

Digital paper

A new form of 'display' that is still in its infancy is the various forms of digital paper. These are thin flexible materials that can be written to electronically, just like a computer screen, but which keep their contents even when removed from any electrical supply.

Electronics embedded into the material allow each tiny sphere to be rotated to make it black or white. When the electronic signal is removed the ball stays in its last orientation. A different technique has tiny tubes laid side by side. In each tube is light-absorbing liquid and a small reflective sphere. The sphere can be made to move to the top surface or away from it making the pixel white or black. Again the sphere stays in its last position once the electronic signal is removed.

DEVICES FOR VIRTUAL REALITY AND 3D INTERACTION

Virtual reality (VR) systems and various forms of 3D visualize require you to navigate and interact in a three-dimensional space. Sometimes these use the ordinary controls and displays of a desktop computer system, but there are also special devices used both to move and interact with 3D objects and to enable you to see a 3D environment.

Positioning in 3D space

Virtual reality systems present a 3D virtual world. Users need to navigate through these spaces and manipulate the virtual objects they find there. Navigation is not simply a matter of moving to a particular location, but also of choosing a particular orientation. In addition, when you grab an object in real space, you don't simply move it around, but also twist and turn it, for example when opening a door. Thus the move from mice to 3D devices usually involves a change from two degrees of freedom to six degrees of freedom, not just three.

Cockpit and virtual controls

Helicopter and aircraft pilots already have to navigate in real space. Many arcade games and also more serious applications use controls modeled on an aircraft cockpit to 'fly' through virtual space. However, helicopter pilots are very skilled and it takes a lot of practice for users to be able to work easily in such environments. In many PC games and *desktop virtual reality* (where the output is shown on an ordinary computer screen), the controls are themselves virtual. This may be a simulated form of the cockpit controls or more prosaic up/down left/right buttons. The user manipulates these virtual controls using an ordinary mouse (or other 2D device).

The 3D mouse

There are a variety of devices that act as 3D versions of a mouse. Rather than just moving the mouse on a tabletop, you can pick it up, move it in three dimensions, rotate the mouse and tip it forward and backward. The 3D mouse has a full six degrees of freedom as its position can be tracked (three degrees), and also its up/down angle (called *pitch*), its left/right orientation (called *yaw*) and the amount it is twisted about its own axis (called *roll*)

Various sensors are used to track the mouse position and orientation: magnetic coils, ultrasound or even

mechanical joints where the mouse is mounted rather like an angle-poise lamp. With the 3D mouse, and indeed most 3D positioning devices, users may experience strain from having to hold the mouse in the air for a long period. 3D mouse down may even be treated as an action in the virtual environment, that is taking a nose dive.

Dataglove

One of the mainstays of high-end VR systems, the dataglove is a 3D input device. ***Consisting of a lycra glove with optical fibers laid along the fingers, it detects the joint angles of the fingers and thumb.*** As the fingers are bent, the fiber optic cable bends too; increasing bend causes more light to leak from the fiber, and the reduction in intensity is detected by the glove and related to the degree of bend in the joint. Attached to the top of the glove are two sensors that use ultrasound to determine 3D positional information as well as the angle of roll, that is the degree of wrist rotation. Such rich multi-dimensional input is currently a solution in search of a problem, in that most of the applications in use do not require such a comprehensive form of data input, whilst those that do cannot afford it. However, the availability of cheaper versions of the data glove will encourage the development of more complex systems that are able to utilize the full power of the data glove as an input device.

The data glove has the advantage that it is very easy to use, and is potentially very powerful and expressive (it can provide 10 joint angles, plus the 3D spatial information and degree of wrist rotation, 50 times a second). It suffers from extreme expense, and the fact that it is difficult to use in conjunction with a keyboard. The potential for the data glove is vast; gesture recognition and sign language interpretation are two obvious areas that are the focus of active research, whilst less obvious applications are evolving all the time.

Virtual reality helmets

The helmets or goggles worn in some VR systems have two purposes:

(i) they display the 3D world to each eye and
(ii) they allow the user's head position to be tracked.

We will discuss the former later when we consider output devices. The head tracking is used primarily to feed into the output side. As the user's head moves around the user ought to see different parts of the scene. However, some systems also use the user's head direction to determine the direction of movement within the space and even which objects to manipulate (rather like the eyegaze systems)..

Whole-body tracking

The movement of the whole body may be tracked using devices similar to the data glove, or using image-processing techniques. In the latter, white spots are stuck at various points of the user's body and the position of these tracked using two or more cameras, allowing the location of every joint to be mapped

3D displays

Just as the 3D images used in VR have led to new forms of input device, they also require more sophisticated outputs. Desktop VR is delivered using a standard computer screen and a 3D impression is produced by using effects such as shadows, occlusion (where one object covers another) and perspective. This can be very effective and you can even view 3D images over the world wide web using a VRML (virtual reality markup language) enabled browser.

Simulators and VR caves

Because of the problems of delivering a full 3D environment via head-mounted displays, some virtual reality systems work by putting the user within an environment where the virtual world is displayed upon it. The most obvious examples of this are large flight simulators – you go inside a mock-up of an aircraft cockpit and the scenes you would see through the windows are projected onto the virtual windows.

PHYSICAL CONTROLS, SENSORS AND SPECIAL DEVICES

Special displays

Apart from the CRT screen there are a number of visual outputs utilized in complex systems, especially in embedded systems. These can take the form of analog representations of numerical values, such as dials, gauges or lights to signify a certain system state. Flashing light-emitting diodes (LEDs) are used on the back of some computers to signify the processor state, whilst gauges and dials are found in process control systems. Once you start in this mode of thinking, you can contemplate numerous visual outputs that are unrelated to the screen. One visual display that has found a specialized niche is the head-up display that is used in aircraft. The pilot is fully occupied looking forward and finds it difficult to look around the cockpit to get information.

Sound output

Another mode of output that we should consider is that of auditory signals. Often designed to be used in conjunction with screen displays, auditory outputs are poorly understood: we do not yet know how to utilize sound in a sensible way to achieve maximum effect and information transference.

Keyboards can be set to emit a click each time a key is pressed, and this appears to speed up interactive performance. Telephone keypads often sound different tones when the keys are pressed; a noise occurring signifies that the key has been successfully pressed, whilst the actual tone provides some information about the particular key that was pressed

Touch, feel and smell

Our other senses are used less in normal computer applications, but you may have played computer games where the joystick or artificial steering wheel vibrated, perhaps when a car was about to go off the track. In some VR applications, such as the use in medical domains to 'practice' surgical procedures, the *feel* of an instrument moving through different tissue types is very important. The devices used to emulate these procedures have ***force feedback***, giving different amounts of resistance depending on the state of the virtual operation. These various forms of force, resistance and texture that influence our physical senses are **called *haptic* devices**. Haptic devices are not limited to virtual environments, but are used in specialist interfaces in the real world too. Electronic braille displays either have pins that rise or fall to give different patterns, or may involve small vibration pins. Force feedback has been used in the design of in-car controls.

PHYSICAL CONTROLS

A desktop computer has to serve many functions and do has generic keys and controls can be used for a variety of purpose. In contrast, these dedicated controls panes have been designed for a particular device and for a single use. This is why they differ so much.

Usually microwave a flat plastic control panel. The reason is this, the microwave is used in the kitchen whilst cooking, with hands that may be greasy or have food on them. The smooth controls have no gaps where food can accumulate and clog buttons, so it can easily be kept clean an hygienic.

When using the washing machine you are handling dirty clothes, which may be grubby, but not to the same extent, so the smooth easy-clean panel is less important. It has several major settings and the large buttons act both as control and display.

ENVIRONMENT AND BIO SENSING

Although we are not always conscious of them, there are many sensors in our environment--controlling automatic doors, energy saving lights, etc. and devices monitoring our behavior such as security tags in shops. The vision of ubiquitous computing suggests that our world will be filled with such devices

PAPER: PRINTING AND SCANNING

Printing

Older printers had a fixed set of characters available on a print head. These varied from the traditional line printer to golf-ball and daisy-wheel printers. To change a typeface or the size of type meant changing the print head, and was an awkward, and frequently messy, job, but for many years the daisy-wheel printer was the only means of producing high-quality output at an affordable price. However, the drop in the price of laser printers coupled with the availability of other cheap high-quality printers means that daisy-wheels are fast becoming a rarity. All of the popular printing technologies, like screens, build the image on the paper as a series of dots. This enables, in theory, any character set or graphic to be printed, Common types of dot-based printers

Dot-matrix printers

These use an inked ribbon, like a typewriter, but instead of a single character-shaped head striking the paper, a line of *pins* is used, each of which can strike the ribbon and hence dot the paper. Horizontal resolution can be varied by altering the speed of the head across the paper, and vertical resolution can be improved by sending the head twice across the paper at a slightly different position. So, dot-matrix printers can produce fast draft-quality output or slower 'letter'-quality output. They are cheap to run, but could not compete with the quality of jet and laser printers for general office and home printing. They are now only used for bulk printing, or where carbon paper is required for payslips, check printing, etc.)

Ink-jet and bubble-jet printers

These operate by sending tiny blobs of ink from the print head to the paper. The ink is squirted at pressure from an ink-jet, whereas bubble-jets use heat to create a bubble. Both are quite quiet in operation. The ink from the bubble-jet (being a bubble rather than a droplet) dries more quickly than the ink-jet and so is less likely to smear. Both approach laser quality, but the bubble-jet dots tend to be more accurately positioned and of a less broken shape.

***Laser printe*r**

This uses similar technology to a photocopier: 'dots' of electrostatic charge are deposited on a drum, which then picks up toner (black powder). This is then rolled onto the paper and cured by heat. The curing is why laser printed documents come out warm, and the electrostatic charge is why they smell of ozone! In addition, some toner can be highly toxic if inhaled, but this is more a problem for full-time maintenance workers than end-users changing the occasional toner cartridge.

Laser printers give nearly typeset-quality output, with top-end printers used by desktop publishing firms. Indeed, many books are nowadays produced using laser printers.

This resolution is measured in *dots per inch* (dpi). Imagine a sheet of graph paper, and building up an image by putting dots at the intersection of each line. The number of lines per inch in each direction is the resolution in dpi. The most common types of dot-based printers are dot-matrix printers, ink-jet printers and laser printers. These are listed roughly in order of increasing resolution and quality, where dot-matrix printers typically have a resolution of 80–120 dpi rising to about 300–600 dpi for ink-jet printers and 600–2400 dpi for laser printers. By varying the quantity of ink and quality of paper, ink-jet printers can be used to print photo-quality prints from digital photographs. Dot-matrix printers are more often rated in *characters per second* (cps), and typical speeds may be 200 cps for draft and 50 cps for letter-quality print.

Color ink-jet printers are substantially cheaper than even monochrome laser printers. However, the recurrent costs of consumables may easily dominate this initial cost. Both jet and laser printers have special-purpose parts (print cartridges, toner, print drums), which need to be replaced every few thousand sheets; and they must also use high-grade paper. It may be more difficult to find suitable grades of recycled paper for laser printers.

MEMORY

RAM AND SHORT-TERM MEMORY (STM)

At the lowest level of computer memory are the registers on the computer chip, but these have little impact on the user except in so far as they affect the general speed of the computer. Most currently active information is held in silicon-chip random access memory (RAM). Different forms of RAM differ as to their precise access times, power consumption and characteristics. Typical access times are of the order of 10 nanoseconds, that is a hundred-millionth of a second, and information can be accessed at a rate of around 100 Mbytes (million bytes) per second. Typical storage in modern personal computers is between 64 and 256 Mbytes.

Most RAM is volatile, that is its contents are lost when the power is turned off. However, many computers have small amount of non-volatile RAM, which retains its contents, perhaps with the aid of a small battery. This may be used to store setup information in a large computer, but in a pocket organizer will be the whole memory.

Non-volatile RAM is more expensive so is only used where necessary, but with many notebook computers using very low-power static RAM, the divide is shrinking.

DISKS AND LONG-TERM MEMORY (LTM)

For most computer users the LTM consists of disks, possibly with small tapes for backup. The existence of backups, and appropriate software to generate and retrieve them, is an important area for user security. However, we will deal mainly with those forms of storage that impact the interactive computer user.

There are two main kinds of technology used in disks: magnetic disks and optical disks. The most common storage media, floppy disks and hard (or fixed) disks, are coated with magnetic material, like that found on an audio tape, on which the information is stored. Typical capacities of floppy disks lie between 300 kbytes and 1.4 Mbytes, but as they are removable.

Hard disks may store from under 40 Mbytes to several gigabytes (Gbytes), that is several thousand million bytes. Various forms of large removable media are also available, fitting somewhere between floppy disks and removable hard disks, and are especially important for multimedia storage

Optical disks use laser light to read and (sometimes) write the information on the disk. There are various high capacity specialist optical devices, but the most common is the CD-ROM, using the same technology as audio compact discs.

COMPRESSION

Compression techniques can be used to reduce the amount of storage required for text, bitmaps and video. All of these things are highly redundant. Consider text for a moment. In English, we know that if we use the letter ‘q’ then ‘u’ is almost bound to follow. At the level of words, some words like ‘the’ and ‘and’ appear frequently in text in general, and for any particular work one can find other common terms

Compression algorithms take advantage of this redundancy. For example, Huffman encoding gives short codes to frequent words , and runlength encoding represents long runs of the same value by length value pairs. Text can easily be reduced by a factor of five and bitmaps often compress to 1% of their original size.

For video, in addition to compressing each frame, we can take advantage of the fact that successive frames are often similar. We can compute the difference between successive frames and then store only this – compressed, of course. More sophisticated algorithms detect when the camera pans and use this information also. These differencing methods fail when the scene changes, and so the process periodically has to restart and send a new, complete (but compressed) image.

With these reductions it is certainly possible to store low-quality video at 64 kbyte/s; that is, we can store five hours of highly compressed video on our 1 Gbyte hard disk. However, it still makes the humble video cassette look very good value.

Probably the leading edge of video still and photographic compression is fractal compression. Fractals have been popularized by the images of the Mandelbrot set (that swirling pattern of computer-generated colors seen on many T-shirts and posters).

Fractals refer to any image that contains parts which, when suitably scaled, are similar to the whole. If we look at an image, it is possible to find parts which are approximately self-similar, and these parts can be stored as a fractal with only a few numeric parameters.

Fractal compression is especially good for textured features, which cause problems for other compression techniques. The decompression of the image can be performed to any degree of accuracy, from a very rough soft-focus image, toone more detailed than the original.

STORAGE FORMAT AND STANDARDS

The most common data types stored by interactive programs are text and bitmap images, with increasing use of video and audio, and this subsection looks at the ridiculous range of file storage standards. We will consider database retrieval in the next subsection.

The basic standard for text storage is the ASCII (American standard code for information interchange) character codes, which assign to each standard printable character and several control characters an internationally recognized 7 bit code

(decimal values 0–127), which can therefore be stored in an 8 bit byte, or be transmitted as 8 bits including parity. Many systems extend the codes to the values 128–255, including line-drawing characters, mathematical symbols and international letters such as 'æ'.

The most common shared format is rich text format (RTF), which encodes formatting information including style sheets. However, even where an application will import or export RTF, it may represent a cut-down version of the full document style. RTF regards the document as formatted text, that is it concentrates on the appearance. Documents can also be regarded as structured objects: this book has chapters containing sections, subsections . . . paragraphs, sentences, words and characters.

There are ISO standards for document structure and interchange, which in theory could be used for transfer between packages and sites, but these are rarely used in practice. Just as the PostScript language is used to describe the printed page, SGML (standard generalized markup language) can be used to store structured text in a reasonably extensible way. You can define your own structures (the definition itself in SGML), and produce documents according to them. XML (extensible markup language), a lightweight version of SGML, is now used extensively for web-based applications.For bitmap storage the range of formats is seemingly unending.

The stored image needs to record the size of the image, the number of bits per pixel, possibly a color map, as well as the bits of the image itself. In addition, an icon may have a 'hot-spot' for use as a cursor. If you think of all the ways of encoding these features, or leaving them implicit, and then consider all the combinations of these different encodings, you can see why there are problems. And all this before we have even considered the effects of compression! There is, in fact, a whole software industry producing packages that convert from one format to another.

PROCESSING AND NETWORKS

Computers that run interactive programs will process in the order of 100 million instructions per second Effects of finite processor speed.

As we can see, speed of processing can seriously affect the user interface. These effects must be taken into account when designing an interactive system. There are two sorts of faults due to processing speed: those when it is too slow, and those when it is too fast!

This was a functional fault, in that the program did the wrong thing. The system is supposed to draw lines from where the mouse button is depressed to where it is released. However, the program gets it wrong – after realizing the button is down, it does not check the position of the mouse fast enough, and so the user may have moved the mouse before the start position is registered. This is a fault at the implementation stage of the system rather than of the design. But to be fair, the programmer may not be given the right sort of information from lower levels of system software.

A second fault due to slow processing is where, in a sense, the program does the right thing, but the feedback is too slow, leading to strange effects at the interface. In order to avoid faults of the first kind, the system buffers the user input; that is, it remembers keypresses and mouse buttons and movement. Unfortunately, this leads to problems of its own.

One example of this sort of problem is cursor tracking, which happens in character-based text editors. The user is trying to move backwards on the same line to correct an error, and so presses the cursor-left key. The cursor moves and when it is over the correct position, the user releases the key. Unfortunately, the system is behind in responding to the user, and so has a few more cursor-left keys to process – the cursor then overshoots. The user tries to correct this by pressing the cursor-right key, and again overshoots.

A similar problem, icon wars, occurs on window systems. The user clicks the mouse on a menu or icon, and nothing happens; for some reason the machine is busy or slow.

The terms of Interaction

Domain

A domain defines an area of expertise and knowledge in some real-world activity. Some examples of domains are graphic design, authoring and process control in a factory. A domain consists of concepts that highlight its important aspects. In a graphic design domain, some of the important concepts are geometric shapes, a drawing surface and a drawing utensil.

Tasks

Operations to manipulate the concepts of a domain. A *goal* is the desired output from a performed task. For example, one task within the graphic design domain is the construction of a specific geometric shape with particular attributes on the drawing surface. A related goal would be to produce a solid red triangle centered on the canvas. An *intention* is a specific action required to meet the goal.

Task analysis involves the identification of the problem space for the user of an interactive system in terms of the domain, goals, intentions and tasks.

Goal

A *goal* is the desired output from a performed task. For example, one task within the graphic design domain is the construction of a specific geometric shape with particular attributes on the drawing surface. A related goal would be to produce a solid red triangle centered on the canvas. So, goal is ultimate result, ***The execution–evaluation cycle***

Norman's model of interaction is perhaps the most influential in Human–Computer Interaction, possibly because of its closeness to our intuitive understanding of the interaction between human user and computer . The user formulates a plan of action, which is then executed at the computer interface. When the plan, or part of the plan, has been executed, the user observes the computer interface to evaluate the result of the executed plan, and to determine further actions.

The interactive cycle can be divided into two major phases: execution and evaluation. These can then be subdivided into further stages, seven in all. The stages in Norman's model of interaction are as follows:

The interaction

1. Establishing the goal.
2. Forming the intention.
3. Specifying the action sequence.
4. Executing the action.
5. Perceiving the system state.
6. Interpreting the system state.
7. Evaluating the system state with respect to the goals and intentions.

Each stage is, of course, an activity of the user. First the user forms a goal. This is the user's notion of what needs to be done and is framed in terms of the domain, in the task language.

The two major parts, execution and evaluation, of interactive cycle **are further subdivided into seven stages**, where each stage is an activity of the user. Seven stages of action are shown in figure. To understand these we see an example, which was also used by Norman.

Imagine you are sitting reading as evening falls. You decide you need more light; that is you establish the goal to get lighter. Form there you form an intention to switch on the desk lamp, and you specify the actions required to reach over and press the lamp switch. If some one else is closer, the intention may be different-you may ask them to switch on the light for you. Your goal is the same but the intention and actions are different. When you have executed the action you perceive the result, either the light is on or it isn't and you interpret this, based on your knowledge of the world. For example, if the light does not come on you may interpret this as indicating he bulb has blown or the lamp is not plugged into the mains, you will formulate the new state according to the original goals is there is now enough light? It so, the cycle is completed. It not, you may formulate a new intention to switch on the main ceiling light.

Norman also describes the two gulfs, which represent the problems that are caused by some interfaces to their users.

Gulf of execution

Gulf of execution is the difference between the user's formulation of the actions to reach the goal and the actions allowed by the system. If the action allowed by the system correspond to those intended by the user, the interaction will effective. The interface should therefore aim to reduce this gulf of execution.

Gulf of evaluation

The gulf of evaluation is the distance between the physical presentation of the system state and the expectation of the user. If the user can readily evaluate the presentation in terms of his goal, the gulf of evaluation is small. The more effort that is required on the part of the user to interpret the presentation, the less effective the interaction.

The interaction framework

The interaction framework attempts a more realistic description of interaction by including the system explicitly, and breaks it into four main components, as shown in Figure3.2. The nodes represent the four major components in an interactive system – the *System*, the *User*, the *Input* and the *Output* .

Each component has its own language. In addition to the *User*'s task language and the *System*'s core language, which we have already introduced, there are languages for both the *Input* and *Output* components. *Input* and *Output* together form the ***INTERFACE***

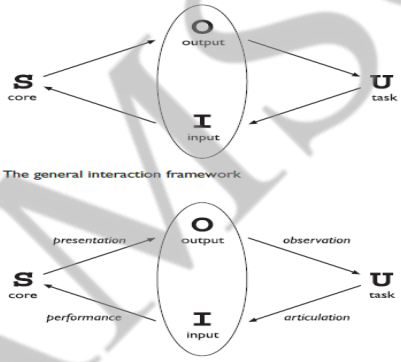


Figure 3.1 The general interaction framework

Figure 3.2 Translations between components

The core language describes computational attributes of the domain relevant to the system state, whereas the task language describes psychological attributes of the domain relevant to the user state. There are also languages for both the input and output components.

Input and output together form the interface. As the interface sits between the user and the system, there are four steps in the interactive cycle, each corresponding to a translation from one component to another, as shown by the labeled arcs in figure3.2.

The user begins the interactive cycle with the formulation of a goal and a task achieves that goal. The only way the user can manipulate the machine is through the input, and so the task must be articulated within the input language, the input language is translated into the core language as operations to be performed by the system. The system then transforms itself as described by the operations; the execution phase of the cycle is complete and the evaluation phase now begins.

The system is in a new state, which must now be communicated to the user. The current values of system attributes are rendered as concepts or features of the output. It is then up to the user to observe the output and assess the results of the interaction relative to the original goal, ending the evaluation phase and, hence, the interactive cycle.

There are four main translations involved in the interaction: articulation, performance, presentation and observation.

The user's formulation of the desired task to achieve some goal needs to be articulated in the input language. The tasks are responses of the user and they need to be translated to stimuli for the input. As pointed out above, this articulation is judged in terms of the coverage from tasks to input and the relative ease with which the translation can be accomplished. The task is phrased in terms of certain psychological attributes that highlight the important features of the domain for the user. If these psychological attributes map clearly onto the input language, then articulation of the task will be made much simpler

Ergonomics

Ergonomics (or human factors) is traditionally the study of the physical characteristic of the interaction: how the controls are designed, the physical environment in which the interaction takes place, and the layout and physical qualities of the screen. A primary focus is on user performance and how the interface enhances or detracts from this. In seeking to evaluate these aspects of the interaction, ergonomics will certainly also touch upon human psychology and system constraints. It is a large and established field, which is closely related to but distinct from HCI.

Physical aspects of Interface are as follow: ·

Arrangement of controls and displays ·

The physical environment ·

Health issues :Use of colors

Arrangement of controls and displays

- **Functional** controls and displays are organized so that those that are functionally related are placed together;
- **Sequential** controls and displays are organized to reflect the order of their use in a typical interaction (this may be especially appropriate in domains where a particular task sequence is enforced, such as aviation);
- **Frequency** controls and displays are organized according to how frequently they are used, with the most commonly used controls being the most easily accessible.

In addition to the organization of the controls and displays in relation to each other, the entire system interface must be arranged appropriately in relation to the user's position. So, for example, the user should be able to reach all controls necessary and view all displays without excessive body movement. Critical displays should be at eye level. Lighting should be arranged to avoid glare and reflection distorting displays.

Health issues

- **Physical position** As we noted in the previous section, users should be able to reach all controls comfortably and see all displays. Users should not be expected to stand for long periods and, if sitting, should be provided with back support. If a particular position for a part of the body is to be adopted for long periods (for example, in typing) support should be provided to allow rest.
- **Temperature** Although most users can adapt to slight changes in temperature without adverse effect, extremes of hot or cold will affect performance and, in excessive cases, health. Experimental studies show that performance deteriorates at high or low temperatures, with users being unable to concentrate efficiently.

- **Lighting** The lighting level will again depend on the work environment. However, adequate lighting should be provided to allow users to see the computer screen without discomfort or eyestrain. The light source should also be positioned to avoid glare affecting the display.
- **Noise** Excessive noise can be harmful to health, causing the user pain, and in acute cases, loss of hearing. Noise levels should be maintained at a comfortable level in the work environment. This does not necessarily mean no noise at all. Noise can be a stimulus to users and can provide needed confirmation of system activity.
- **Time** The time users spend using the system should also be controlled.

The use of color

Colors used in the display should be as distinct as possible and the distinction should not be affected by changes in contrast. Blue should not be used to display critical information. If color is used as an indicator it should not be the only cue: additional coding information should be included.

The colors used should also correspond to common conventions and user expectations. Red, green and yellow are colors frequently associated with stop, go and standby respectively. Therefore, red may be used to indicate emergency and alarms; green, normal activity; and yellow, standby and auxiliary function. These conventions should not be violated without very good cause.

Interaction style

Interaction is communication between computer and human (user). For a successful enjoyable communication interface style has its own importance.

There are a number of common interface styles including

- Command line interface
- Menus
- Natural language Question/answer and query dialog
- Form fills and spreadsheets
- WIMP
- Point and click
- Three-dimensional interfaces

Command line interface

Command line interface was the first interactive dialog style to be commonly used and, in spite of the availability of menu-driven interface, it is still widely used. It provides a means of expressing instructions to the computer directly, using some function keys, single characters, abbreviations or whole-word commands. Command line interface are powerful in that they offer direct access to system functionality, and can be combined to apply a number of tools to the same data. They are also flexible: the command often has a number of options or parameters that will vary its behavior in some way, and it can be applied to many objects at once, making it useful for repetitive tasks.

Menus

In a menu-driven interface, the set of options available to the user is displayed on the screen, and selected using the mouse, or numeric or alphabetic keys. Since the options are visible they are less demanding of the user, relying on recognition rather than recall. However, menu options still need to be meaningful and logically grouped to aid recognition.

Often menus are hierarchically ordered and the option required is not available at the top layer of the hierarchy. The grouping and naming of menu options then provides the only cue for the user to find the required option. Such systems either can be purely text based, with the menu options being presented as numbered choices (see Figure3.8), or may have a graphical component in which the menu appears within a rectangular box and choices are made, perhaps by typing the initial

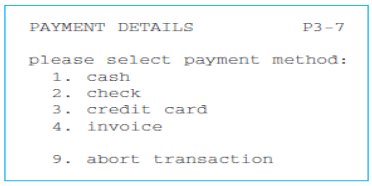


Figure 3.8 Menu-driven interface

letter of the desired selection, or by entering the associated number, or by moving around the menu with the arrow keys.

Natural Language

Perhaps the most attractive means of communicating with computers, at least at first glance, is by natural language. Users unable to remember a command or lost in a hierarchy of menus, may long for the computer that is able to understand instructions expressed in everyday words. Unfortunately, however, the ambiguity of natural language makes it very difficult for a machine to understand.

Question/answer and query dialog

Question and answer dialog is a simple mechanism for providing input to an application in specific domain. The user is asked a series of questions and so is led through the interaction step by step. These interfaces are easy to learn and use, but are limited in functionality and power. As such, they are appropriate for restricted domains and for novice or casual users. Query languages, on the other hand, are used to construct queries to retrieve information from a database. They use natural-language-style phrases, but in fact require specific syntax, as well as knowledge of database structure.

Queries usually require the user to specify an attribute or attributes for which to search the database, as well as the attributes of interest to be displayed. This is straightforward where there is a single attribute, but becomes complex when multiple attributes are involved, particularly of the user is interested in attribute A or attribute B, or attribute A and not attribute B, or where values of attributes are to be compared. Most query language do not provide direct confirmation of what was requested, so that the only validation the user has is the result of the search. The effective use of query languages therefore requires some experience.

Form-fills and spreadsheets

Form-filling interfaces are used primarily for data entry but can be useful in data retrieval applications. The user is presented with a display resembling a paper form, with slots to fill in as shown in figure. Most form-filling interfaces allow easy movement around the form and allow some fields to be left blank. They also require correction facilities, as users may change their minds or make a mistake about the value that belongs in each field. Spreadsheets are sophisticated variation of form filling. The spreadsheet comprises a grid of cells, each of which can contain a value or a formula.

The WIMP Interfaces

Currently many common environments for interactive computing are examples of the WIMP interface style, often simply called windowing systems. WIMP stands for windows, icons, menus, and pointers, and is default interface style for the majority of interactive computer systems in use today, especially in the PC and desktop workstation arena.

Point and Click interface

In most multimedia systems and in web browsers, virtually all actions take only a single click of the mouse button. You may point at a city on a map and when you click a window opens, showing you tourist information about the city. You may point at a word in some text and when you click you see a definition of the word. You may point at a recognizable iconic button and when you click some action is performed.

Three-dimensional interfaces

Three-dimensional interfaces

There is an increasing use of three-dimensional effects in user interfaces. The most obvious example is virtual reality, but VR is only part of a range of 3D techniques available to the interface designer.The simplest technique is where ordinary WIMP elements, buttons, scroll bars, etc, are given a 3D appearance using shading, giving the appearance of being sculpted out of stone.

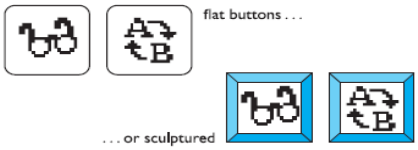


Figure 3.12 Buttons in 3D say 'press me'

The WIMP Interfaces

Windows

Windows are ***areas of the screen that behave as if they were independent terminals in their own right. A window can usually contain text or graphics, and can be moved or resized. More than one window can be on a screen at once, allowing separate tasks to be visible at the same time***. Users can direct their attention to the different windows as they switch from one thread of work to another. If one window overlaps the other, the back window is partially obscured, and then refreshed when exposed again. Overlapping windows can cause problems by obscuring vital information, so windows may also be tiled, when they adjoin but do not overlap each other.

Alternatively, windows may be placed in a cascading fashion, where each new window is placed slightly to the left and below the previous window. In some systems this layout policy is fixed, in others the user can select it.

Usually windows have various things associated with them that increase their usefulness. Scrollbars are one such attachment, allowing the user to move the contents of the window up and down, or from side to side. This makes the window behave as if it were a real window onto a much larger world, where new information is brought into view by manipulating the scrollbars. There is usually a title bar attached to the top of a window, identifying it to the user, and there may be special boxes in the corners of the window to aid resizing, closing, or making as large as possible. Each of these can be seen in the figure

In addition, some systems allow windows within windows. For example, in Microsoft Office applications, such as Excel and Word, each application has its own window and then within this each document has a window. It is often possible to have different layout policies within the different application windows. Icons Windows can be closed and lost forever, or they can be shrunk to some very reduced representation. A small picture is used to represent a closed window, and this representation is known as an icon. By allowing icons, many windows can be available on the screen at the same time, ready to be expanded to their full size by clicking on the icon. Shrinking a window to its icon is known as iconifying the window. When a user temporarily does not want to follow a particular thread of dialog, he can suspend that dialog by iconifying the window containing the dialog. The icon saves space on the screen and serves as a remainder to the user that he can subsequently resume the dialog by opening up the window. Figure shows a few examples of icons used in a typical windowing system (Microsoft).

Icons can also be used to represent other aspects of the system, such as a waste basket for throwing unwanted files into, or various disks, programs or functions, that are accessible to the user. Icon can take many forms: they can be realistic representation of the objects that they stand for, or they can be highly stylized. They can even be arbitrary symbols, but these can be difficult for users to interpret.

Pointers

The Pointer is an important component of the WIMP interface, since the interaction style required by WIMP relies very much on pointing and selecting things such as icons. The mouse provides an input device capable of such tasks, although joysticks and trackballs are other alternatives. The user is presented with a cursor on the screen that is controlled by the input device.

A verity of pointer cursors is shown in figure. The different shape of cursor are often used to distinguish modes, for example the normal pointer cursor maybe an arrow, but change to change to cross-hairs when drawing a line. Cursors are also used to tell the user about system activity, for example a watch or hourglass cursor may be displayed when the system s busy reading a file. Pointer cursors are like icons, being small bitmap images, but in addition all cursors have a hot-spot, the location to which they point

Menus

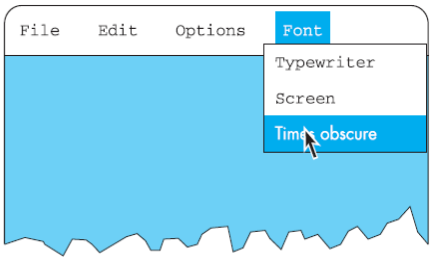


Figure 3.18 Pull-down menu

The last main feature of the windowing system is the menu, an interaction technique that is common across many non-windowing systems as well. A menu presents a choice of operations or services that can be performed by the system at a given time. As we discussed our ability to recall information is inferior to our ability to recognize it from some visual cue. Menus provide information cues in the form of an ordered list of operations that can be scanned. This implies that the names used for the commands in the menu should be meaningful and informative. The pointing device is used to indicate the desired option.

As the pointer moves to the position of a menu item, the item is usually highlighted to indicate that it is the potential candidate for selection. Selection usually requires some additional user action, such as pressing a button on the mouse that controls the pointer cursor on the screen or pressing some special key on the keyboard.

Menus are inefficient when they have too many items, and so cascading menus are utilized, in which item selection opens up another menu adjacent to the item, allowing refinement of the selection. Several layers of cascading menus can be used. The main menu can be visible to the user all the time, as a menu bar and submenus can be pulled down or across from it upon request.

Menu bars are often placed at the top of the screen or at the top of each window. Alternative includes menu bars along one side of the screen, or even placed amongst the windows in the main `desktop' area. Websites use a variety of menu bar locations, including top, bottom and either side of the screen. Alternatively, the main menu can be hidden and upon request it will pop up onto the screen. These pop-up menus are often used to present context- sensitive options, for example allowing one to examine properties of particular on- screen objects.

In some systems they are also used to access more global actions when the mouse is depressed over the screen background. Pull-down menus are dragged down from the title at the top of the screen, by moving the mouse pointer into the title par area and pressing the button. Fall-down menus are similar, except that the menu automatically appears when the mouse pointer enters the title bar, without the user having to press the button. Some menus explicitly asked to go away. Pop up menus appear when a particular region of the screen, may be designated by an icon, is selected, but they only stay as long as the mouse button is depressed.

Another approach to menu selection is to arrange the options in a circular fashion. The pointer appears in the center of the circle, and so there is the same distance to travel to any of the selections. This has the advantages that it is easier to select items, since they can each have a larger target area, and that the selection time for each item is the same, since the pointer is equidistant from them all. However, these pie menus take up more screen space and are therefore less common in interface.

The major problems with menus in general are deciding what items to include and how to group those items. Including too many items makes menus too long or creates too many of them, whereas grouping causes problems in that items that relate to the same topic need to come under the same heading, yet many items could be grouped under more than one heading. In pull-down menus the menu label should be chosen to reflect the function of the menu items, and items grouped within menus by function. These groupings should be consistent across applications so that the user can transfer learning to new applications. Menu items should be ordered in the menu according to importance and frequency of use, and appropriate functionalities should be kept apart to prevent accidental selection of the wrong function, with potentially disastrous consequences.

Keyboard accelerators

Menus often offer keyboard accelerators, key combinations that have the same effect as selecting the menu item. This allows more expert users, familiar with the system, to manipulate things without moving off the keyboard, which is

often faster. The accelerators are often displayed alongside the menu item so that frequent use makes them familiar. Buttons are individual and isolated regions within display that can be selected by the user to invoke specific operations. These regions are referred to as buttons because they are purposely made to resemble the push buttons you would find on a control panel. `Pushing' the button invokes a command, the meaning of which is usually indicated by a textual label or a small icon.

Radio Buttons can also be used to toggle between two states, displaying status information such as whether the current font is italicized or not in a word processor, or selecting options on a web form. Such toggle buttons can be grouped together to allow a user to select one feature form a set of mutually exclusive options, such as the size in points of the current font. These are called radio buttons. Check boxes It a set of options is not mutually exclusive, such as font characteristics like bold, italic and underlining, and then a set of toggle buttons can be used to indicate the on/off status of the options. This type of collection of buttons is sometimes referred to as check boxes.

Toolbars

Many systems have a collection of small buttons, each with icons, placed at the top or side of the window and offering commonly used functions. The function of this toolbar is similar to a menu bar, but as the icons are smaller than the equivalent text more functions can be simultaneously displayed. Sometimes the content of the toolbar is fixed, but often users can customize it, either changing which functions area made available, or choosing which of several predefined toolbars is displayed.

Palettes

In many application programs, instructions can either one of several modes. The defining characteristic of modes is that the interpretation of actions, such as keystrokes or gestures with the mouse, changes as the mode change. For example, using the standard UNIX text editor vi, keystrokes can be interpreted either as operations to insert characters in the document or as operations to perform file manipulation. Problems occur if the user is not aware of the current mode.

Palettes are a mechanism for ***making the set of possible modes and the active mode visible to the user.*** A palette is usually a **collection of icons that are reminiscent of the** purpose of the various modes. An example in a drawing package would be a collection of icons to indicate the pixel color or pattern that is used to fill in objects, much like an artist's palette for paint.

Some systems allow the user to create palettes from menus or toolbars. In the case of pull-down menus, the user may be able `tear off' the menu, turning it into a palette showing the menu items. In the case of toolbars, he may be able to drag the toolbar away from its normal position and place it anywhere on the screen. Tear-off menus are usually those that are heavily graphical anyway, for example line style of color selection in a drawing package.

Dialog boxes

Dialog boxes are information windows used by the system to bring the user's attention to some important information, possibly an error or a warning used to prevent a possible error. Alternatively, they are used to invoke a sub dialog between user and system for a very specific task that will normally be embedded within some larger task. For example, most interactive applications result in the user creating some file that will have to be named and stored within the filing system. When the user or the file and indicate where it is to be located within the filing system. When the save sub dialog is complete, the dialog box will disappear. Just as windows are used to separate the different threads of user-system dialog, so too are dialog boxes used to factor out auxiliary task threads from the main task dialog.

Interaction Paradigms

Paradigms for interaction have for the most part been dependent upon technological advances and their creative application to enhance interaction.

Time sharing

In the 1940s and 1950s, the significant advances in computing consisted of new hardware technologies. Mechanical relays were replaced by vacuum electron tubes. Tubes were replaced by transistors, and transistors by integrated chips, all of which meant that the amount of sheer computing power was increasing by orders of magnitude. By the 1960s it was becoming apparent that the explosion of growth in computing power would be wasted if there were not an equivalent explosion of ideas about how to channel that power. One of the leading advocates of research into human-centered applications of

computer technology was J.C.R Licklider, who became the director of the Information Processing Techniques Office of the US Department of Defense's **Advanced Research Agency (ARPA).**

One of the major contributions to come out of this new emphasis in research was the concept of time-sharing, in which a single computer could support multiple users. Previously, the human was restricted to batch sessions, in which complete jobs were submitted on punched cards or paper tape to an operator who would then run them individually on the computer.

Time-sharing systems of the 1960s made programming a truly interactive venture and brought about a subculture of programmers known as `hackers' single-minded masters of detail who took pleasure in understanding complexity. Though the purpose of the first interactive time-sharing systems was simply to augment the programming capabilities of the early hackers, it marked a significant stage in computer applications for human use.

Video display units

As early as the mid-1950s researchers were experimenting with the possibility of presenting and manipulating information from a computer in the form of images on a video display unit (VDU). These display screens could provide a more suitable medium than a paper printout for presenting vast quantities of strategic information for rapid assimilation. It was not until 1962, however, when a young graduate student at the Massachusetts Institute of Technology (MIT), Ivan Sutherland, astonished the established computer science community with the Sketchpad program, that the capabilities of visual images were realized.

Sketchpad demonstrated two important ideas. First, computers could be used for more than just data processing. They could extend the user's ability to abstract away from some levels of detail, visualizing and manipulating different representations of the same information.

Programming toolkits Dougles Engelbart's ambition since the early 1950s was to use computer technology as a means of complementing human problem-solving activity. Engelbart's idea as a graduate student at the University of California at Berkeley was to use the computer to teach humans. This dream of naïve human users actually learning from a computer was a stark contrast to the prevailing attitude of his contemporaries that computers were purposely complex technology that only the intellectually privileged were capable of manipulating. Personal computing Programming toolkits provide a means for those with substantial computing skills to increase their productivity greatly. But Engelbart's vision was not exclusive to the computer literate.

The decade of the 1970s saw the emergence of computing power aimed at the masses, computer literate or not. One of the first demonstrations that the powerful tools of the hacker could be made accessible to the computer novice was a graphics programming language for children called LOGO. The inventor, Seymen Papert, wanted to develop a language that was easy for children to use.

Window systems and the WIMP interface

With the advent and immense commercial success of personal computing, the emphasis for increasing the usability of computing technology focused on addressing the single user who engaged in a dialog with the computer in order to complete some work. Humans are able to think about more than one thing at a time, and in accomplishing some piece of work, they frequently interrupt their current train of thought to pursue some other related piece of work.

A personal computer system which forces the user to progress in order through all of the tasks needed to achieve some objective, from beginning to end without any diversions, does not correspond to that standard working pattern. If the personal computer is to be an effective dialog partner, to must be as flexible in its ability to change the topic as the human is. But the ability to address the needs of a different user task is not the only requirement.

Computer systems for the most part react to stimuli provided by the user, so they are quite amenable to a wandering dialog initiated by the user. As the ser engages in more than one plan of activity over a stretch of time, it becomes difficult for him to maintain the status of the overlapping threads of activity. Interaction based on windows, icons, menus, and pointers--the WIMP interface--is now commonplace. These interaction devices first appeared in the commercial marketplace in April 1981, when Xerox Corporation introduced the 8010 Star Information System

Direct Manipulation

In the early 1980s as the price of fast and high-quality graphics hardware was steadily decreasing, designers were beginning to see that their products were gaining popularity as their visual content increased.

As long as the user-system command line prompt computing was going to stay within the minority population of the hackers who reveled in the challenge of complexity. In a standard command line interface, the only way to get any feedback on the results of previous interaction is to know that you only have to ask for it and to know how to ask for it. Rapid visual and audio feedback on a high-resolution display screen or through a high-quality sound system makes it possible to provide evaluative information for every executed user action.

Rapid feedback is just one feature of the interaction technique known as direct manipulation. Ben Shneiderman is attributed with coining this phrase in 1982 to describe the appeal of graphics-based interactive systems such as Sketchpad and the Xerox Alto and Star. He highlights the following features of a direct manipulation interface.

- visibility of the objects of interest incremental action at the interface with rapid feedback on all actions
- reversibility of all actions, so that users are encouraged to explore without severe penalties
- syntactic correctness of all actions, so that every user action is a legal
- operation replacement of complex command language with actions to manipulate
- directly the visible object

The first real commercial success which demonstrated the inherent usability of direct manipulation interfaces for the general public was the Macintosh personal computer, introduced by Apple Computer, Inc.

Hypertext

In 1945, Vannevar Bush, then the highest-ranking scientific administrator in the US war effort, published an article entitled `As We May Think' in The Atlantic Monthly. Bush was in charge of over 6000 scientists who had greatly pushed back the frontiers of scientific knowledge during the Second World War. He recognized that a major drawback of these prolific research efforts was that it was becoming increasingly difficult to keep in touch with the growing body of scientific knowledge in the literature.

In his opinion, the greatest advantages of this scientific revolution were to be gained by those individuals who were able to keep abreast of an ever-increasing flow of information. To that end, he described an innovative and futuristic information storage and retrieval apparatus the memex , which was constructed with technology wholly existing in 1945 and aimed at increasing the human capacity to store and retrieve, connected pieces of knowledge by mimicking our ability to create random associative links.

An unsuccessful attempt to create a machine language equivalent of the memex on early 1960s computer hardware led Nelson on a lifelong quest to produce Xanadu, a potentially revolutionary worldwide publishing and information retrieval system based on the idea of interconnected, non-linear text and other media forms.

A traditional paper is read from beginning to end, in a linear fashion. But within that text, there are often ideas or footnotes that urge the reader to digress into richer topic. The linear format for information does not provide much support for this random and associated browsing task. What Bush's memex suggested was to preserve the non- linear browsing structure in the actual documentation. Nelson coined the phrase hypertext in the mid 1960s to reflect this non-linear text structure

Multi-modality

The majority of interactive systems still use the traditional keyboard and a pointing device, such as a mouse, for input and are restricted to a color display screen with some sound capabilities for output. Each of these input and output devices can be considered as communication channels for the system and they correspond to certain human communication channels.

A multi-modal interactive system is a system that relies on the use of multiple human communication channels. Each different channel for the user is referred to as a modality of interaction. In this sense, all interactive systems can be considered multi-model, for human have always used their visual and haptic channels in manipulating a computer. In fact, we often use our audio channel to hear whether the computer is actually running properly. However, genuine multi-modal systems rely to an extent on simultaneous use of multiple communication channels for both input and output. Humans quite naturally process information by simultaneous use of different channels.

Computer-supported cooperative work

Another development in computing in the 1960s was the establishment of the first computer networks, which allowed communication between separate machines.

Personal computing was all about providing individuals with enough computing power so that they were liberated from dumb terminals, which operated on a time- sharing systems. It is interesting to note that as computer networks become widespread, individuals retained their powerful workstations but now wanted to reconnect themselves to the rest of the workstations in their immediate working environment, and even throughout the world. One result of this reconnection was the emergence of collaboration between individuals via the computer called computer supported cooperative work, or CSCW.

The main distinction between CSCW systems and interactive systems designed for a single user is that designer can no longer neglect the society within which any single user operates. CSCW systems are built to allow interaction between humans via the computer and so the needs of the many must be represented in the one product.

World Wide Web

Probably the most significant recent development interactive computing is the World Wide Web, often referred to as just the web, or WWW. The web is built on top of the Internet, and offers an easy to use, predominantly graphical interface to information, hiding the underlying complexities of transmission protocols, addresses and remote access to data.

The Internet is simply a collection of computers, each linked by any sort of data connections, whether it be slow telephone line and modem or high-bandwidth optical connection. The computers of the Internet all communicate using common data transmission protocols and addressing systems. This makes it possible for anyone to read anything from anywhere, in theory, if it conforms to the protocol. The web builds on this with its own layer of network protocol, a standard markup notation for laying out pages of information and a global naming scheme.

Web pages can contain text, color images, movies, sound and, most important, hypertext links to other web pages. Hypermedia documents can therefore be published by anyone who has access to a computer connected to the Internet.

Ubiquitous computing

In the late 1980s, a group of researchers at Xerox PARC led by Mark Weiser, initiated a research program with the goal of moving human-computer interaction away from the desktop and out into our everyday lives. Weiser observed. The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. These words have inspired a new generation of researchers in the area of ubiquitous computing. Another popular term for this emerging paradigm is pervasive computing, first coined by IBM.

The intention is to create a computing infrastructure that permeates our physical environment so much that we do not notice the computer may longer.

A good analogy for the vision of ubiquitous computing is the electric motor. When the electric motor was first introduced, it was large, loud and very noticeable. Today, the average household contains so many electric motors that we hardly ever notice them anymore. Their utility led to ubiquity and, hence, invisibility. Sensor-based and context-aware interaction The yard-scale, foot-scale and inch-scale computers are all still clearly embodied devices with which we interact, whether or not we consider them `computers'. There are an increasing number of proposed and existing technologies that embed computation even deeper, but unobtrusively, into day-to-day life. Weiser's dream was computers anymore', and the term ubiquitous computing encompasses a wide range from mobile devices to more pervasive environments.

Sensor-based and context-aware interaction

The yard-scale, foot-scale and inch-scale computers are all still clearly embodied devices with which we interact, whether or not we consider them `computers'. There are an increasing number of proposed and existing technologies that embed computation even deeper, but unobtrusively, into day-to-day life. Weiser's dream was computers anymore', and the term ubiquitous computing encompasses a wide range from mobile devices to more pervasive environments.

UNIT 2 – DESIGN AND SOFTWARE PROCESS

2.1 INTERACTION DESIGN BASICS:

2.1.1 INTRODUCTION

- design: – what it is, interventions, goals, constraints
- the design process – what happens when
- users – **who they are, what they are like …**
- scenarios– rich stories of design
- navigation– finding your way around a system
- iteration and prototypes

2.1.2 WHAT IS DESIGN?

achieving goals within constraints
goals – purpose-who is it for, why do they
want it constraints- materials, platforms
trade-offs- Choosing goals and constraints

2.1.2.1 THE GOLDEN RULE OF DESIGN

understand your materials - For Human–Computer Interaction the obvious materials are the human and the computer.

That is we must:
understand computers – limitations, capacities, tools, platform
understand people – psychological, social aspects, human error.

2.1.2.2 TO ERR IS HUMAN

People make mistakes. This is not 'human error', an excuse to hide behind in accident reports, it is human nature. Systems should be designed to reduce the likelihood of those mistakes and to minimize the consequences when mistakes happen.
If you design using a physical material, you need to understand how and where failures would occur and strengthen the construction, build in safety features or redundancy.

2.1.3 THE PROCESS OF DESIGN

A system has been designed and built, and only when it proves unusable do they think to ask how to do it right! In other companies usability is seen as equivalent to testing – checking whether people can use it and fixing problems, rather than making sure they can from the beginning. Usability is designed in from the start. View of four main phases plus an iteration loop, focussed on the design of interaction (Figure 2.1).

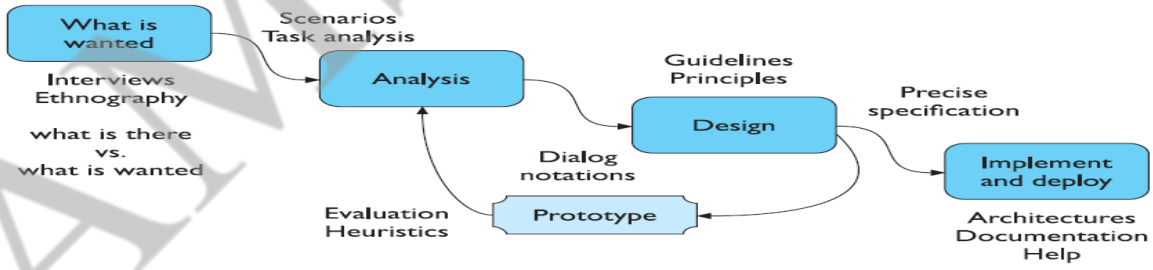


Fig 2.1 Interaction Design process

Requirements – what is wanted. 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.
Used for this in HCI: interviewing people, videotaping them, looking at the documents and objects that they work with, observing them directly.

Analysis 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.

Design There is a central stage when you move from what you want, to how to do it. 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.

Iteration and prototyping Humans are complex and we cannot expect to get designs right first time. We therefore need to evaluate a design to see how well it is working and where there can be improvements. Most user interface design therefore involves some form of prototyping, producing early versions of systems to try out with **real users.**

Implementation and deployment Finally, we need to create it and deploy it. This will involve writing code, perhaps making hardware, writing documentation and manuals– everything that goes into a real system that can be given to others.

2.1.4 USER FOCUS:

know your users

- who are they?
- probably not like you!
- talk to them
- watch them

use your imagination

2.1.5 SCENARIOS

- stories for design

  Communicate with others – other designers, clients or users. It is easy to misunderstand each other while discussing abstract ideas. Concrete **examples** of use are far easier to share.

  Validate other models A detailed scenario can be 'played' against various more formal representations such as task models or dialog and navigation models

  Express dynamics Individual screen shots and pictures give you a sense of what a system would look like, but not how it behaves.

- linearity

  Time is linear Our lives are linear as we live in time and so we find it easier to understand simple linear narratives. We are natural storytellers and story listeners.

  But no alternatives Real interactions have choices, some made by people, some by systems. A simple scenario does not show these alternative paths. It is easy to miss the unintended things a person may do.

- What will users want to do?
- step-by-step walkthrough
  - what can they see (sketches, screen shots)
  - what do they do (keyboard, mouse etc.)
  - what are they thinking?

use and reuse throughout design

- use scenarios to communicate with others
  - designers, clients, users
- validate other models
  - **'play' it against other models**
- express dynamics
  - screenshots – appearance

Scenarios are a resource that can be used and reused throughout the design process.

2.1.6 NAVIGATION DESIGN

Navigation within the application You need to be able to understand what will happen when a button is pressed, to understand where you are in the interaction.

We are focussing on the computer system itself. You interact at several levels:

Widgets The appropriate choice of widgets and wording in menus and buttons will help you know how to use them for a particular selection or action.

Screens or windows You need to find things on the screen, understand the logical grouping of buttons.

Environment The word processor has to read documents from disk, perhaps some are on remote networks. You swap between applications, perhaps cut and paste. You can see similar levels in other types of application and device, as Table 2.1 shows Level of Interaction

2.1 Levels of Interaction

PC application	Website	Physical device
Widgets	Form elements, tags and links	Buttons, dials, lights, displays
Screen design	Page design	Physical layout
Navigation design	Site structure	Main modes of device
Other apps and operating system	The web, browser, external links	The real world!

There are differences; for **example**, in the web we have less control of how people enter a site and on a physical device we have the same layout of buttons and displays no matter what the internal state.
When considering the structure of an application is to think about actual use:

- Who is going to use the application?
- How do they think about it?
- What will they do with it?

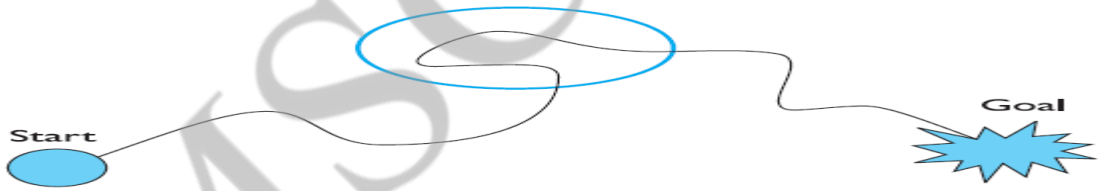
This can then drive the second task – thinking about structure. Individual screens or the layout of devices will have their own structure.

Two main kinds of issue:

- ***local structure***
  looking from one screen or page out
- ***global structure***
  structure of site, movement between screens.

2.1.6.1 LOCAL STRUCTURE

Users have some idea of what they are after and a partial model of the system. If users had perfect knowledge of what they wanted and how the system worked they could simply take the shortest path to what they want, pressing all the right buttons and links. However, in a world of partial knowledge users through the system. The important thing is not so much that they take the most efficient route, but that at each point in the interaction they can make some assessment of whether they are getting closer to their goal.



To do this goal seeking, ***each state of the system or each screen needs to give the user enough knowledge of what to do to get closer to their goal.*** To get you started, here are four things to look for when looking at a single web page, screen or state of a device.

- knowing where you are
- knowing what you can do
- knowing where you are going – or what will happen
- knowing where you've been – or what you've done.

The screen, web page or device displays should make clear *where you are* in terms of the interaction or state of the system.

2.1.6.2 GLOBAL STRUCTURE – HIERARCHICAL ORGANIZATION

One way to organize a system is in some form of hierarchy. This is typically organized along functional boundaries (that is, different kinds of things), but may be organized by roles, user type, or some more esoteric breakdown such as modules in an educational system. ***The hierarchy links screens, pages or states in logical groupings gives a high-level breakdown of some sort of messaging system***. This sort of

hierarchy can be used purely to help during design, but can also be used to structure the actual system. For **example**, this may reflect the menu structure of a PC application or the site structure on the web.

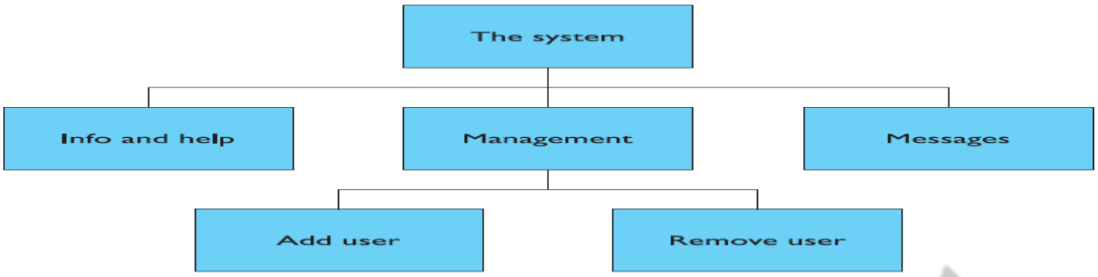


Figure 2.1 Application function hierarchy

2.1.6.3 GLOBAL STRUCTURE – DIALOG

In a pure information system or static website it may be sufficient to have a fully hierarchical structure, perhaps with next/previous links between items in the same group. However, for any system that involves doing things, constantly drilling down from one part of the hierarchy to another is very frustrating. Usually there are ways of getting more quickly from place to place.

As well as these cross-links in hierarchies, when you get down to detailed interactions, such as editing or deleting a record, there is obviously a flow of screens and commands that is not about hierarchy. ***In HCI the word 'dialog' is used to refer to this pattern of interactions between the user and a system.***

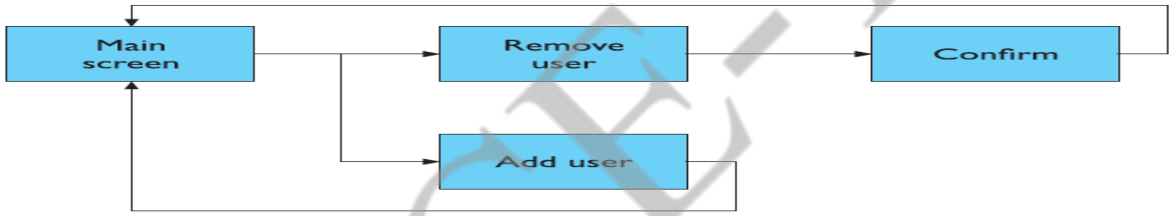


Figure 2.2 Network of Screens and states

Figure 2.2 shows a network diagram illustrating the main screens for adding or deleting a user from the messaging system in Figure 2.1. The arrows show the general flow between the states. We can see that from the main screen we can get to either the 'remove user' screen or the 'add user' screen. This is presumably by selecting buttons or links, but the way these are shown we leave to detailed screen design. We can also see that from the 'add user' screen the system always returns to the main screen, but after the 'remove user' screen there is a further confirmation screen

2.1.6.4 WIDER STILL

Each sits amongst other devices and applications and this in turn has to be reflected within our design has several implications:

Style issues We should normally conform to platform standards, such as positions for menus on a PC application, to ensure consistency between applications. For **example**, on our proposed personal movie player we should make use of standard fast-forward, play and pause icons.

Functional issues On a PC application we need to be able to interact with files, read standard formats and be able to handle cut and paste.

Navigation issues We may need to support linkages between applications, for **example** allowing the embedding of data from one application in another, or, in a mail system, being able to double click an attachment icon and have the right application launched for the attachment.

On the web we have the added difficulty that other sites and applications may include links that bypass our 'home page' and other pages and go direct into the heart of our site or web application. Also, when we link to other sites, we have no control over them or the way their content may change over time.

2.1.7 SCREEN DESIGN AND LAYOUT

A single screen image often has to present information clearly and also act as the locus for interacting with the system.

2.1.7.1 TOOLS FOR LAYOUT

We have a number of visual tools available to help us suggest to the user appropriate ways to read and interact with a screen or device.

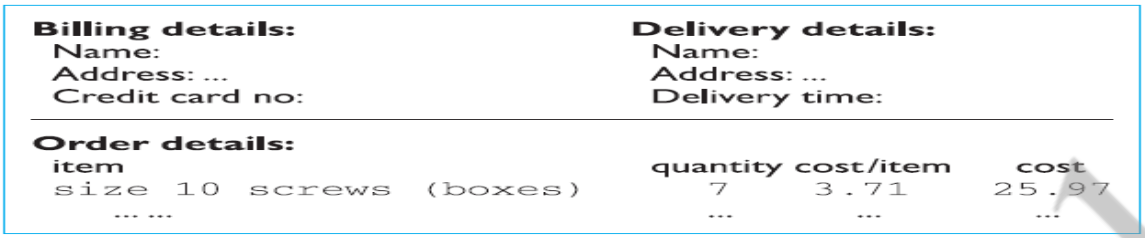


Figure 2.3 Grouping related items in an order screen

Grouping and structure

If things logically belong together, then we should normally physically group them together. This may involve multiple levels of structure. For **example**, in Figure 2.3 we can see a potential design for an ordering screen. Notice how the details for billing and delivery are grouped together spatially; also note how they are separated from the list of items actually ordered by a line as well as spatially. This reflects the following logical structure:

Order:

- Administrative information
  - Billing details
  - Delivery details
- Order information
  - Order line 1
  - Order line 2
  - . . .

Order of groups and items

Figure 2.3 The screen seems to naturally suggest reading or filling in the billing details first, followed by the delivery details, followed by the individual order items. This should normally match the order on screen. For data entry forms or dialog boxes we should also set up the order in which the tab key moves between fields.

Decoration:

Figure 2.3, we can see how the design uses boxes and a separating line to make the grouping clear. Other decorative features like font style, and text or background colours can be used to emphasize groupings.

Alignment:

For users who read text from left to right, lists of text items should normally be aligned to the left. Numbers should normally be aligned to the right (for integers) or at the decimal point. This is because the shape of the column then gives an indication of magnitude – a sort of mini histogram. Items like names are particularly difficult. Figure 2.4. It is clearly hard to look someone up if you only know their surname. To make it easy, such lists should be laid out in columns as in (ii), or have forename and surname reversed as in (iii).

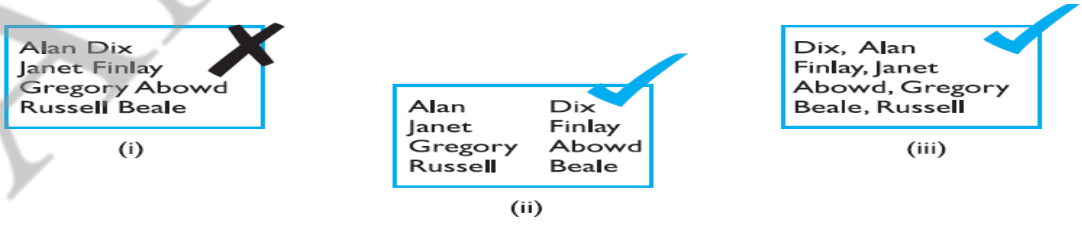


Figure 2.4 Looking up surnames

Multiple column lists require more care. Text columns have to be wide enough for the largest item, which means you can get large gaps between columns.

Figure 2.5 shows an **example** of this (i), and you can see how hard this makes it for your eye to scan across the rows. There are several visual ways to deal with this including: (ii) 'leaders' – lines of dots linking the columns; and (iii) using soft tone gray's or colours behind rows or columns. This is also a time when it may be worth

breaking other alignment rules, perhaps right aligning some text items as in (iv). This last alternative might be a good solution if you were frequently scanning the numbers and only occasionally scanning the names of items, but not if you needed frequently to look up names.

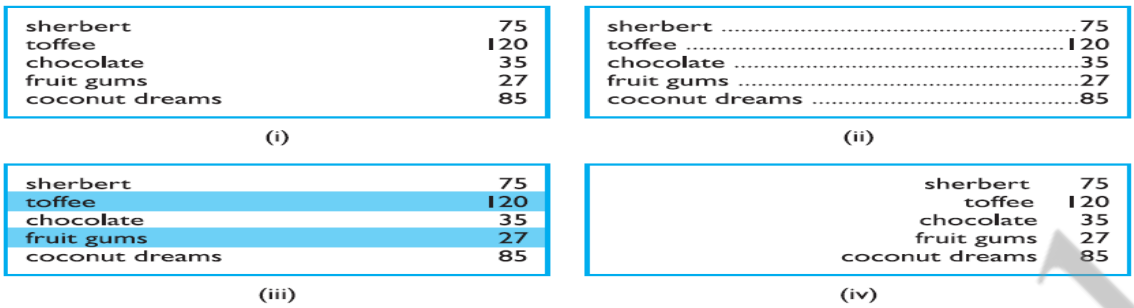


Figure 2.5 Managing Multiple Columns

White space

***In typography the space between the letters is called the counter*.** In painting this is also important and artists may focus as much on the space between the foreground elements such as figures and buildings as on the elements themselves.

Shape of the counter is the most important part of the composition of a painting and in calligraphy and typography the balance of a word is determined by giving an even **weight to the counters. If one ignores the 'content' of a screen and instead concentrates** on the counter –the space between the elements – one can get an overall feel for the layout. If elements that are supposed to be related look separate when you focus on the counter, then something is wrong.

Space can be used in several ways. Some of these are shown in Figure 2.6. The colored areas represent continuous areas of text or graphics.

(i) We can see space used to separate blocks as you often see in gaps between paragraphs or space between sections in a report. Space can also be used to create more complex structures.

(ii) There are clearly four main areas: ABC, D, E and F. Within one of these are three further areas, A, B and C, which themselves are grouped as A on its own, followed by B and C together.

(iii) In Figure 2.6 , we can see space used to highlight. This is a technique used frequently in magazines to highlight a quote or graphic.

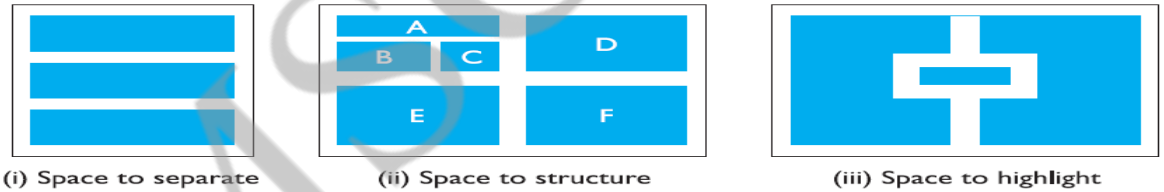


Figure 2.6 using white space in layout

2.1.7.2 USER ACTION AND CONTROL

Entering information

In each case the screen consists not only of information presented to the user, but also of places for the user to enter information or select options. ***Alignment is still important***. It is common to see the text entry boxes aligned in a jagged fashion because the field names are of different length where right-justified text for the field labels may be best or, alternatively, in a graphical interface a smaller font can be used for field labels and the labels placed just above and to the left of the field they refer to.

For both presenting and entering information a clear logical layout is important. The task analysis techniques can help in determining how to group screen items and also the order in which users are likely to want to read them or fill them in. Users are likely to read from left to right and top to bottom means that a screen can be designed so that users encounter items in an appropriate order for the task at hand.

Affordances

Affordances are not intrinsic, but depend on the background and culture of users. Most computer users will click on an icon. This is not because they go around pushing pictures in art galleries, but because they have learned that this is an affordance of such objects in a computer domain.

Similarly, such experienced users may well double click if a single click has no effect, yet novices would not even think of double clicking – after all, double clicking on most real buttons turns them off again.

2.1.7.3 APPROPRIATE APPEARANCE

Presenting information

The way of presenting information on screen depends on the kind of information: text, numbers, maps, tables; on the technology available to present it: character display, line drawing, graphics, virtual reality; and, most important of all, on the purpose for which it is being used.

Consider the window in Figure 2.7. The file listing is alphabetic, which is fine if we want to look up the details of a particular file, but makes it very difficult to find recently updated files. Of course, if the list were ordered by date then it would be difficult to find a particular file. Different purposes require different representations. For more complex numerical data, we may be considering scatter graphs, histograms or 3D surfaces; for hierarchical structures, we may consider outlines or organization diagrams.

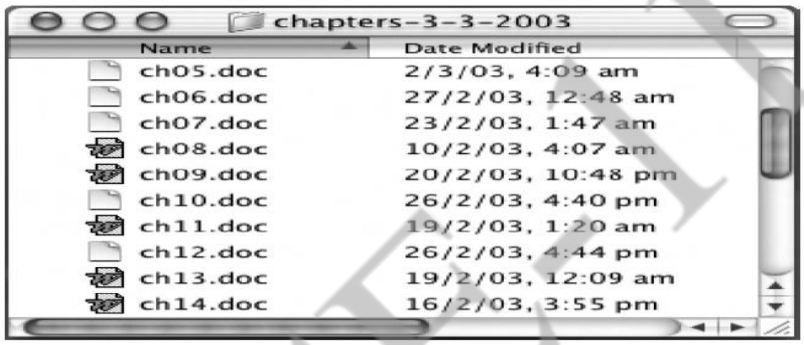


Figure 2.7 Alphabetic file listing. Screen shot reprinted by permission from Apple Computer, Inc.

Aesthetics and utility

An interface should be aesthetically pleasing. Indeed, ***good graphic design and attractive displays can increase users' satisfaction and thus improve productivity.***

For example, an industrial control panel will often be built up of the individual controls of several subsystems, some designed by different teams, some bought in. The resulting inconsistency in appearance may look a mess and suggest tidying up. Certainly some of this inconsistency may cause problems.

The ***conflict between aesthetics and utility can also be seen in many 'well-designed' posters and multimedia systems***. In particular, the backdrop behind text must have low contrast in order to leave the text readable; this is often not the case and graphic designers may include excessively complex and strong backgrounds because they look good. The results are impressive, perhaps even award winning, but completely unusable.

THE COUNTER

Making a mess of it: color and 3D

Many monitors only support a limited range of primary colors .The increasing use of 3D effects in interfaces has posed a whole new set of problems for text and numerical information. For presenting physical information and certain sorts of graphs, text presented in perspective can be very difficult to read and the all too common 3D pie chart is all but useless.

Localization / internationalization

The process of making software suitable for different languages and cultures is called localization or internationalization.

ITERATION AND PROTOTYPING

The result of evaluating the system will usually be a list of faults or problems and this is followed by a redesign exercise, which is then prototyped, evaluated .

Figure 2.8 shows this process. The end point is when there are no more problems that can economically be fixed. So iteration and prototyping are the universally accepted 'best practice' approach for interaction design.

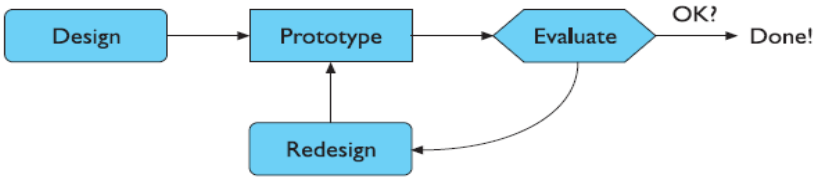


Figure 2.8 Role of prototyping

You cannot iterate the design unless you know what must be done to improve it.

2.2 HCI IN THE SOFTWARE PROCESS:

- The software lifecycle- Software engineering is the discipline for understanding the software design process, or life cycle. Designing for usability occurs at all stages of the life cycle, not as a single isolated activity

The Software Life Cycle

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. Typically, ***the customer and the designer are groups of people and some people can be both customer and designer.*** It is often important to distinguish between the customer who is the client of the designing company and the customer who is the eventual user of the system.

2.2.1 ACTIVITIES IN THE LIFE CYCLE

Requirements specification

Designer and customer try capture what the system is expected to provide can be expressed in natural language or more precise languages, such as a task analysis would provide.

Architectural design

High-level description of how the system will provide the services required factor system into major components of the system and how they are interrelated needs to satisfy both functional and non functional requirements

Detailed design

Refinement of architectural components and interrelations to identify modules to be implemented separately the refinement is governed by the non-functional requirements

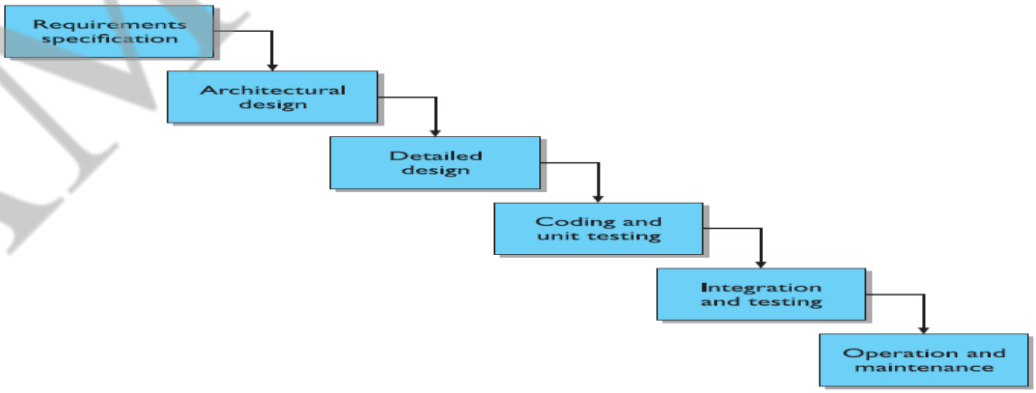


Figure 2.9 The activities in the waterfall model of the software life cycle

Coding and unit testing

The detailed design for a component of the system should be in such a form that it is possible to implement it in some executable programming language. ***After coding, the component can be tested to verify that it performs correctly.***

Integration and testing

Once enough components have been implemented and individually tested, they must be integrated as described in the architectural design. Further testing is done to ensure correct behaviour and acceptable use of any shared resources. It is also possible at this time to perform some ***acceptance testing with the customers to ensure that the system meets their requirements.*** It is only after acceptance of the integrated system that the product is finally released to the customer.

Maintenance

Maintenance involves the correction of errors in the system which is 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, as shown in Figure 2.10.

2.2.2 VALIDATION AND VERIFICATION

Verification of a design will often occur within a single life-cycle activity or between two adjacent activities.

For **example**, in the detailed design of a component

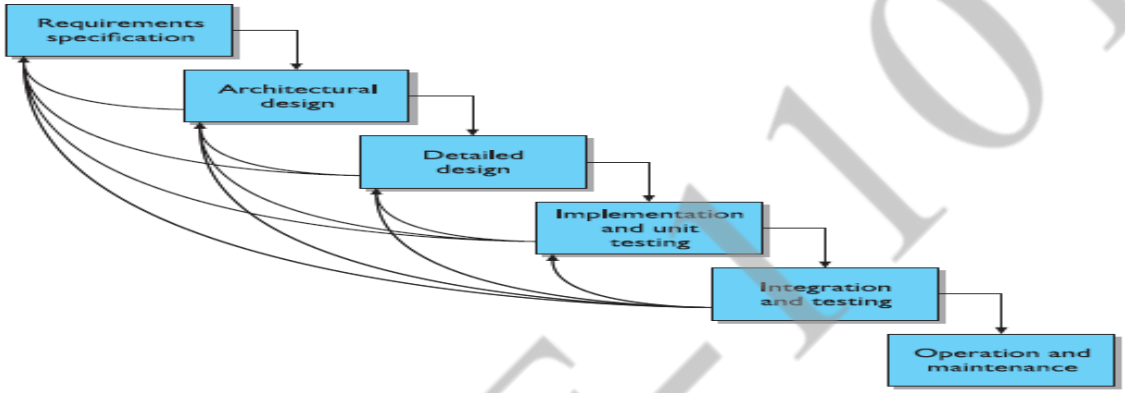


Figure 2.10 Feedback from maintenance activity to other design activities

The formality gap means that validation will always rely to some extent on subjective means of proof. We can increase our confidence in the subjective proof by effective use of real-world experts in performing certain validation chores. These experts will not necessarily have design expertise, so they may not understand the

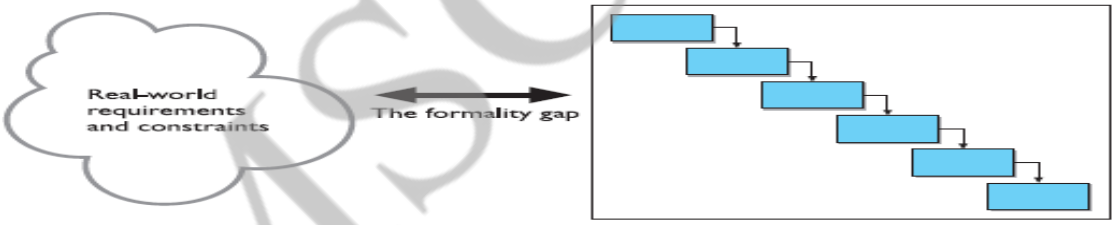


Figure 2.11 The formality gap between the real world and structured design

design notations used. Therefore, it is important that the design notations narrow the formality gap, making clear the claims that the expert can then validate. ***For interactive systems, the expert will have knowledge from a cognitive or psychological domain, so the design specification must be readily interpretable from a psychological perspective in order to validate it against interactive requirements of the system.***

2.2.3 MANAGEMENT AND CONTRACTUAL ISSUES

The technical perspective of the **life cycle is described in *stages*** of activity, whereas the managerial perspective is described in temporally bound *phases*. ***A phase is usually defined in terms of the documentation taken as input to the phase and the documentation delivered as output from the phase.*** So the requirements phase will take any marketing or conceptual development information, identifying potential customers, as input and produce a requirements specification that must be agreed upon between customer and designer.

So, It's necessary to manage software development, but it has negative implications on the design process as well. It is very difficult in the design of an interactive system to determine a priori what requirements to impose on the system to maximize its usability.

2.2.4 INTERACTIVE SYSTEMS AND THE SOFTWARE LIFE CYCLE

The life cycle for development presents the process of design in a somewhat pipeline order. In reality, even for batch-processing systems, **the actual design process is *iterative*, *work in one design activity affecting work in any other*** .We can represent this iterative relationship as in Figure 2.12, but that does not greatly enhance any understanding of the design process for interactive systems.

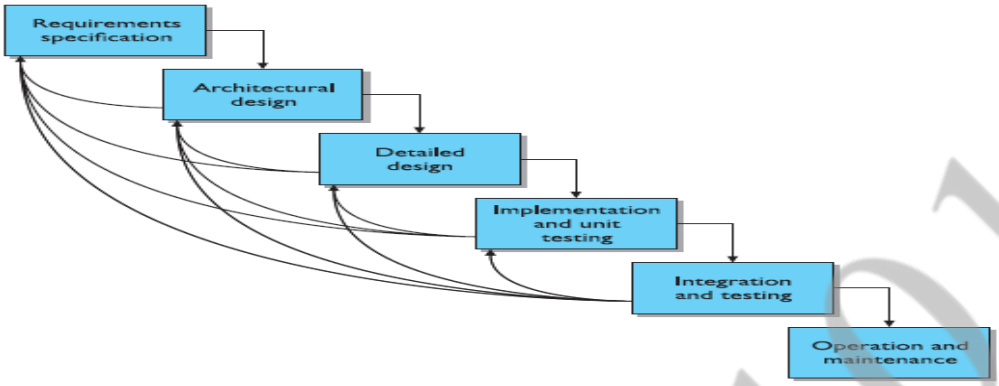


Figure 2.12 Representing iteration in the waterfall model

The traditional software life cycle approach to design that is, ***we can structure our approach to design in order to attain the goal. Designers do not find out all of the requirements for a system before they begin***. Figure 6.4 depicts how discovery in later activities can be reflected in iterations back to earlier stages.

2.2.5 USABILITY ENGINEERING

Usability engineering is the inclusion of a usability specification, forming part of the requirements specification that concentrates on features of the user–system interaction which contribute to the usability of the product. Various attributes of the system are suggested for testing the usability. **For each attribute, six items are defined to form the usability specification of that attribute.** Table 2.1 provides an **example** of a usability specification for the design of a control panel for a video cassette recorder (VCR)

Table 2.1 Sample usability specification for undo with a VCR

Attribute:	Backward recoverability
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program
Now level:	No current product allows such an undo
Worst case:	As many actions as it takes to program in mistake
Planned level:	A maximum of two explicit user actions
Best case:	One explicit cancel action

Table 2.2 Criteria by which measuring method can be determined

Time to complete a task
Per cent of task completed
Per cent of task completed per unit time
Ratio of successes to failures
Time spent in errors
Per cent or number of errors
Per cent or number of competitors better than it
Number of commands used
Frequency of help and documentation use
Per cent of favourable/unfavourable user comments
Number of repetitions of failed commands

Number of runs of successes and of failures
Number of times interface misleads the user
Number of good and bad features recalled by users
Number of available commands not invoked
Number of regressive behaviours
Number of users preferring your system
Number of times users need to work around a problem
Number of times the user is disrupted from a work task
Number of times user loses control of the system
Number of times user expresses frustration or satisfaction

Tables 2.2 and 2.3 provide a list of measurement criteria which can be used to determine the measuring method for a usability attribute and the possible ways to set the worst/best case and planned/ now level targets. Measurements such as those promoted by usability engineering are also called *usability metrics*.

Table 2.3 Possible ways to set measurement levels in a usability

Set levels with respect to information on:
an existing system or previous version
competitive systems
carrying out the task without use of a computer system
an absolute scale
your own prototype
user's own earlier performance
each component of a system separately
a successive split of the difference between best and worst values observed in user tests

Table 2.4 Examples of usability metrics from ISO 9241

Usability objective	Effectiveness measures	Efficiency measures	Satisfaction measures
Suitability for the task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of power features used	Relative efficiency compared with an expert user	Rating scale for satisfaction with power features
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for ease of learning
Error tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

Table 2.4 gives examples of usability metrics categorized by their contribution towards the three categories of usability: effectiveness, efficiency and satisfaction.

2.2.5.1 PROBLEMS WITH USABILITY ENGINEERING

The problem with usability metrics is measurement of very specific user actions in very specific situations. When the designer knows what the actions and situation will be, then she can set goals for measured observations. However, at early stages of design, designers do not have this information.

2.2.6 ITERATIVE DESIGN AND PROTOTYPING

Requirements for an interactive system cannot be completely specified from the beginning of the life cycle. The only way to be sure about some features of the potential design is to build them and test them out on real users. The design can then be modified to correct any false assumptions that were revealed in the testing.

The purpose of design process is to overcome the inherent problems of incomplete requirements specification by cycling through several designs, incrementally improving upon the final product with each pass.

There are three main approaches to prototyping:

Throw-away 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 2.13 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.

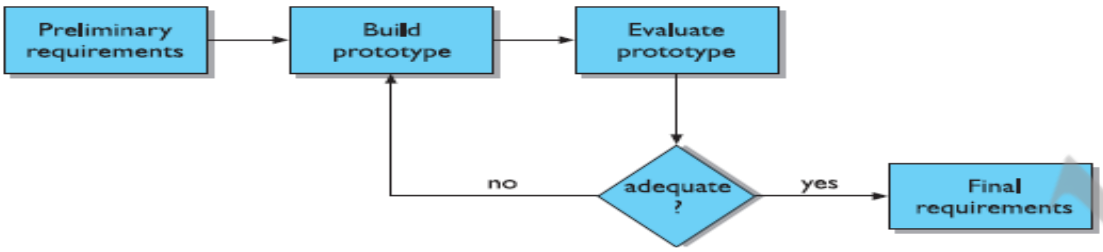


Figure 2.13 Throw-away prototyping within requirements specification

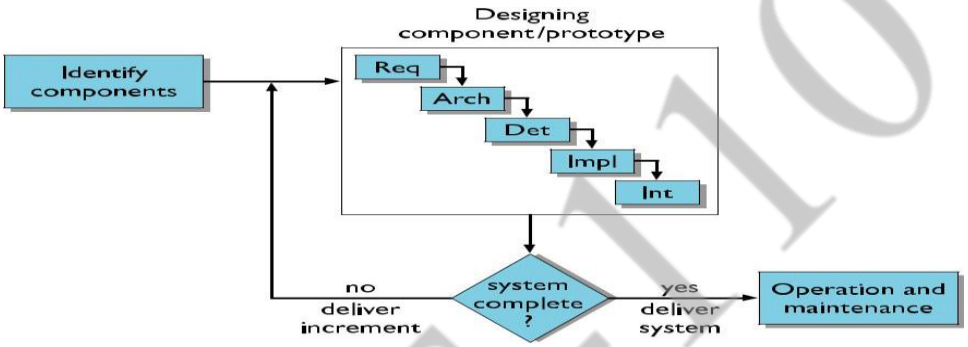


Figure 2.14 Incremental prototyping within the life cycle

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 Figure 2.14.

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 2.15. 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.

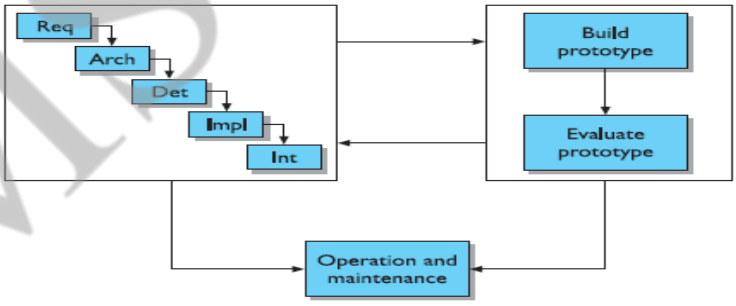


Figure 2.15 Evolutionary prototyping throughout the life cycle

Time Building prototypes take time and, if it is a throw-away prototype, it can be seen as precious time taken away from the real design task. So the value of proto-typing is only appreciated if it is fast, hence the use of the term ***rapid prototyping***.

Planning Most project managers do not have the experience necessary for adequately planning and costing a design process which involves prototyping.

Non-functional features Non-functional features such as safety and reliability, and these are precisely the kinds of features which are sacrificed in developing a prototype.

Contracts The design process is often governed by contractual agreements between customer and designer which are affected by many of these managerial and technical issues.

2.2.6.1 TECHNIQUES FOR PROTOTYPING

Storyboards

Storyboards do not require much in terms of computing power to construct; in fact, they can be mocked up without the aid of any computing resource. The origins of storyboards are in the film industry, where a series of panels roughly depicts snapshots from an intended film sequence in order to get the idea across about the eventual scene. Similarly, for interactive system design, the storyboards provide snapshots of the interface at particular points in the interaction. ***Evaluating customer or user impressions of the storyboards can determine relatively quickly if the design is heading in the right direction.***

Limited functionality simulations

Programming support for simulations means a designer can rapidly build graphical and textual interaction objects and attach some behaviour to those objects, which mimics the system's functionality. Once this simulation is built, it can be evaluated and changed rapidly to reflect the results of the evaluation study with various users.

High-level programming support

HyperTalk is a special-purpose high-level programming language which makes it easy for the designer to program certain features of an interactive system at the expense of other system features like speed of response or space efficiency.

HyperTalk and many similar languages allow the programmer to attach functional behaviour to the specific interactions that the user will be able to do, such as position and click on the mouse over a button on the screen.

2.2.6.2 WARNING ABOUT ITERATIVE DESIGN

The ideal model of iterative design, in which a rapid prototype is designed, evaluated and modified until the *best possible design is achieved in the given project time.*

There are two problems.

First, it is the case that design decisions made at the very beginning of the prototyping process are wrong and, in practice, design inertia can be so great as never to overcome an initial bad decision.

The **second problem** is slightly more subtle, and serious. If, in the process of evaluation, a potential usability problem is diagnosed, it is important to understand the reason for the problem and not just detect the symptom

2.2.7 DESIGN RATIONALE

Design rationale is the information that explains why a computer system is the way it is, including its **structural or architectural description** and its **functional or behavioural description**. Design rationale relates to an activity of both reflection (doing design rationale) and documentation (creating a design rationale) that occurs throughout the entire life cycle.

Benefits of design rationale

- communication throughout life cycle
- reuse of design knowledge across products
- enforces design discipline
- presents arguments for design trade-offs
- organizes potentially large design space

2.2.7.1 PROCESS-ORIENTED DESIGN RATIONALE

A hierarchical structure to a design rationale is created. ***A root issue is identified which represents the main problem or question that the argument is addressing***. **Various *positions* are put as resolutions for the root issue**, and these are depicted as descendants in the IBIS hierarchy directly connected to the root issue. ***Each position is then supported or refuted by arguments, which modify the relationship between issue and position***. The hierarchy grows as secondary issues are raised which modify the root issue in some way. Each of these secondary issues is in turn expanded by positions and arguments, further sub-issues, and so on.

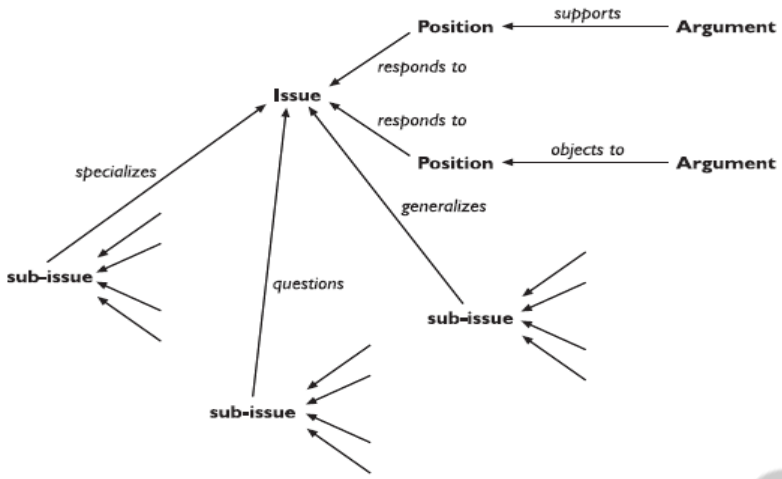


Figure 2.16 The structure of a gIBIS design rationale

Figure 2.16 gives a representation of the gIBIS vocabulary. Issues, positions and arguments are nodes in the graph and the connections between them are labelled to clarify the relationship between adjacent nodes.

2.2.7.2 DESIGN SPACE ANALYSIS

The design space is initially structured by a set of questions representing the major issues of the design. Since **design space analysis is structure oriented.** *Questions, Options and Criteria (QOC) notation, are characterized as design space analysis* (see Figure 2.17).

In Figure 2.17 an option is assessed in terms of a criterion is linked with a solid line, whereas negative links have a dashed line. The most favourable option is boxed in the diagram.

The key to an effective design space analysis using the QOC notation is deciding the right questions to use to structure the space and the correct criteria to judge the options.

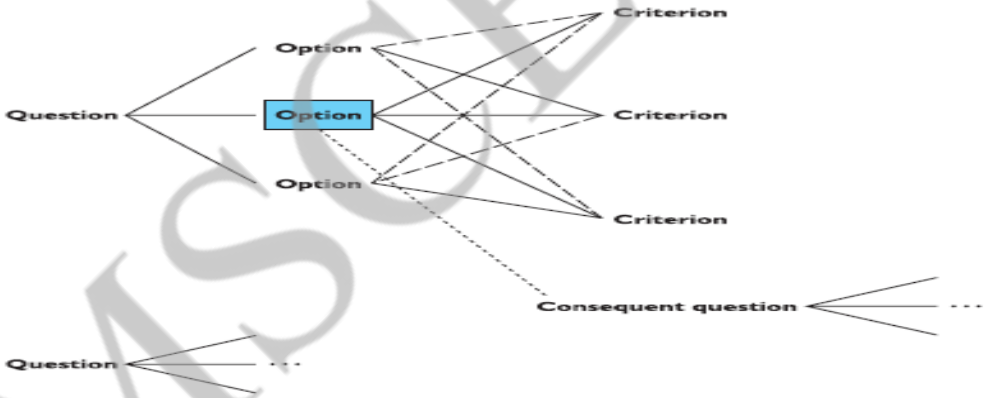


Figure 2.17 the QOC notation

Another structure-oriented technique, called Decision Representation Language (DRL), structures the design space in a similar fashion to QOC, though its language is somewhat larger and it has a formal semantics. The questions, options and criteria in DRL are given the names: decision problem, alternatives and goals. QOC assessments are represented in DRL by a more complex language for relating goals to alternatives.

The ***advantage of the formal semantics of DRL is that the design rationale can be used as a computational mechanism to help manage the large volume of information.***

For **example**, DRL can track the dependencies between different decision problems, so that subsequent changes to the design rationale for one decision problem can be automatically propagated to other dependent problems.

The major ***disadvantage is the increased overhead such an analysis warrants***. More time must be taken away from the design activity to do this separate documentation task. When time is scarce, these kinds of overhead costs are the first to be trimmed.

2.2.7.3 PSYCHOLOGICAL DESIGN RATIONAL

People use computers to accomplish some tasks in their particular work domain. When designing a new interactive system, the designers take into account the tasks that users currently perform and any new ones that they may want to perform. This task identification serves as part of the requirements for the new system, and can be done through empirical observation of how people perform their work currently and presented through informal language or a more formal task analysis language.

When the new system is implemented, or becomes an *artifact*, further observation reveals that in addition to the required tasks it was built to support, it also supports users in tasks that the designer never intended. Once designers understand these new tasks, and the associated problems that arise between them and the previously known tasks, the new task definitions can serve as requirements for future artifacts.

The purpose of psychological design rationale is to support this natural task– artifact cycle of design activity. The main emphasis is not to capture the designer's intention in building the artifact. Rather, psychological design rationale aims to make explicit the consequences of a design for the user, given an understanding of what tasks he intends to perform.

2.3 DESIGN RULES:

principles

- abstract design rules
- low authority
- high generality

standards

- specific design rules
- high authority
- limited application

guidelines

- lower authority
- more general application

2.3.1 PRINCIPLES TO SUPPORT USABILITY

The most abstract design rules are general principles, which can be applied to the design of an interactive system in order to promote its usability.

The principles of usability are 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.

2.3.1.1 LEARNABILITY

Learnability concerns the features of the interactive system that allow novice users to understand how to use it initially and then how to attain a maximal level of performance.

Table 7.1 Summary of principles affecting learnability

Principle	Definition	Related principles
Predictability	Support for the user to determine the effect of future action based on past interaction history	Operation visibility
Synthesizability	Support for the user to assess the effect of past operations on the current state	Immediate/eventual honesty
Familiarity	The extent to which a user's knowledge and experience in other real-world or computer-based domains can be applied when interacting with a new system	Guessability, affordance
Generalizability	Support for the user to extend knowledge of specific interaction within and across applications to other similar situations	–
Consistency	Likeness in input–output behavior arising from similar situations or similar task objectives	–

Predictability

Predictability of an interactive system means that the user's knowledge of the interaction history is sufficient to determine the result of his future interaction with it. There are many degrees to which predictability can be satisfied. The knowledge can be restricted to the presently perceivable information, so that the user need not remember anything other than what is currently observable.

The knowledge requirement can be increased to the limit where the user is actually forced to remember what every previous keystroke was and what every previous screen display contained (and the order of each!) in order to determine the consequences of the next input action.

Synthesizability

Synthesis is the ability of the user to assess the effect of past operations on the current state. When an operation changes some aspect of the internal state, it is important that the change is seen by the user. The principle of ***honesty*** relates to the ability of the user interface to provide an observable and informative account of such change. In the best of circumstances, this notification can come ***immediately,*** requiring no further interaction initiated by the user. At the very least, the notification should appear ***eventually,*** after explicit user directives to make the change observable.

Comparison between command language interfaces and visual desktop interfaces for a file management system.

In a **command language system**, you would have to remember the destination directory and then ask to see the contents of that directory in order to verify that the file has been moved (in fact, you would also have to check that the file is no longer in its original directory to determine that it has been moved and not copied).

In a **visual desktop interface**, a visual representation (or icon) of the file is dragged from its original directory and placed in its destination directory where it remains visible (assuming the destination folder is selected to reveal its contents). In this case, the user need not expend any more effort to assess the result of the move operation. **The visual desktop is immediately honest.**

Familiarity

For a new user, the familiarity of an interactive system measures the correlation between the user's existing knowledge and the knowledge required for effective interaction.

For **example**, when word processors were originally introduced the analogy between the word processor and a typewriter was intended to make the new technology more immediately accessible to those who had little experience with the former but a lot of experience with the latter.

Generalizability

We can apply generalization to situations in which ***the user wants to apply knowledge that helps achieve one particular goal to another situation where the goal is in some way similar.*** Generalizability can be seen as a form of consistency.

Consistency

Consistency is probably the most widely mentioned principle in the literature on user interface design. Consistency is not a single property of an interactive system that is either satisfied or not satisfied. Instead, consistency must be applied relative to something. Thus we have consistency in command naming, or consistency in command/argument invocation.

Another consequence of consistency having to be defined with respect to some other feature of the interaction is that many other principles can be 'reduced' to qualified instances of consistency. Hence, familiarity can be considered as consistency with respect to past real-world experience, and generalizability as consistency with respect to experience with the same system or set of applications on the same platform.

2.3.1.2 FLEXIBILITY

Dialog Initiative

When considering the interaction between user and system as a dialog between partners, it is important to consider which partner has the initiative in the conversation.

The system can initiate all dialogs, in which case the user simply responds to requests for information. We call this type of dialog system pre-emptive.

For **example**, a modal dialog box prohibits the user from interacting with the system in any way that does not direct input to the box.

The user may be entirely free to initiate any action towards the system, in which case the dialog is user pre-emptive.

Table 7.2 Summary of principles affecting flexibility

Principle	Definition	Related principles
Dialog initiative	Allowing the user freedom from artificial constraints on the input dialog imposed by the system	System/user pre-emptiveness
Multi-threading	Ability of the system to support user interaction pertaining to more than one task at a time	Concurrent vs. interleaving, multi-modality
Task migratability	The ability to pass control for the execution of a given task so that it becomes either internalized by the user or the system or shared between them	—
Substitutivity	Allowing equivalent values of input and output to be arbitrarily substituted for each other	Representation multiplicity, equal opportunity
Customizability	Modifiability of the user interface by the user or the system	Adaptivity, adaptability

Multi-threading

Multi-threading of the user–system dialog allows for interaction to support more than one task at a time.

- **Concurrent multi-threading** allows simultaneous communication of information pertaining to separate tasks.
- **Interleaved multi-threading** permits a temporal overlap between separate tasks, but stipulates that at any given instant the dialog is restricted to a single task.

Multi-modality of a dialog is related to multi-threading. Coutaz has characterized **two dimensions of multi-modal systems.**

First, we can consider how the separate modalities (or channels of communication) are combined to form a single input or output expression. Multiple channels may be available, but any one expression may be restricted to just one channel (keyboard or audio).

As an **example,** to open a window the user can choose between a double click on an icon, a keyboard shortcut, or saying 'open window'.

Consider chord sequences of input with a keyboard and mouse (pressing the shift key while a mouse button is pressed, or saying 'drop' as you drag a file over the trash icon). We can also characterize a multi-modality dialog depending on whether it allows concurrent or interleaved use of multiple modes.

A windowing system naturally supports a multi-threaded dialog that is interleaved amongst a number of overlapping tasks. Each window can represent a different task,

For **example** text editing in one window, file management in another, a telephone directory in another and electronic mail in yet another. A multi-modal dialog can allow for concurrent multi-threading.

Task Migratability

Migratability concerns the transfer of control for execution of tasks between system and user. It should be possible for the user or system to pass the control of a task over to the other or promote the task from a completely internalized one to a shared and cooperative venture. Hence, a task that is internal to one can become internal to the other or shared between the two partners.

Spell-checking a paper is a good **example** of the need for task migratability.

Substitutivity

Substitutivity requires that equivalent values can be substituted for each other.

For **example**, in considering the form of an input expression to determine the margin for a letter, you may want to enter the value in either inches or centimeters. You may also want to input the value explicitly (say 1.5 inches) or you may want to enter a calculation which produces the right input value (you know the width of the text is 6.5 inches and the width of the paper is 8.5 inches and you want the left margin to be twice as large as the right margin, so you enter $^2/3\ (8.5 - 6.5)$ inches).

This input substitutivity contributes towards flexibility by allowing the user to choose whichever form best suits the needs of the moment. By avoiding unnecessary calculations in the user's head, substitutivity can minimize user errors and cognitive effort.

We can also consider substitutivity with respect to output, or the system's rendering of state information. ***Representation multiplicity*** **illustrates flexibility for state rendering**.

For example, the temperature of a physical object over a period of time can be presented as a digital thermometer if the actual numerical value is important or as a graph if it is only important to notice trends. *Equal opportunity* blurs the distinction between input and output at the interface. The user has the choice of what is input and what is output; in addition, output can be reused as input.

Customizability

Customizability is the modifiability of the user interface by the user or the system. We distinguish between the user-initiated and system-initiated modification, referring to the former as ***adaptability*** and the latter as ***adaptivity*.**

2.3.1.3 ROBUSTNESS

Table 7.3 Summary of principles affecting robustness

Principle	Definition	Related principles
Observability	Ability of the user to evaluate the internal state of the system from its perceivable representation	Browsability, static/dynamic defaults, reachability, persistence, operation visibility
Recoverability	Ability of the user to take corrective action once an error has been recognized	Reachability, forward/ backward recovery, commensurate effort
Responsiveness	How the user perceives the rate of communication with the system	Stability
Task conformance	The degree to which the system services support all of the tasks the user wishes to perform and in the way that the user understands them	Task completeness, task adequacy

Observability

Observability allows the user to evaluate the internal state of the system by means of its perceivable representation at the interface. It can be discussed through five other principles: browsability, defaults, reachability, persistence and operation visibility.

- ***Browsability*** allows the user to explore the current internal state of the system via the limited view provided at the interface. The availability of *defaults* can assist the user by passive recall (for **example**, a suggested response to a question can be recognized as correct instead of recalled).
- ***Reachability*** refers to the possibility of navigation through the observable system states. There are various levels of reachability that can be given precise mathematical definitions, but the main notion is whether the user can navigate from any given state to any other state.
- ***Persistence*** deals with the duration of the effect of a communication act and the ability of the user to make use of that effect. ***Vocal communication does not persist except in the memory of the receiver. Visual communication can remain as an object which the user can subsequently manipulate long after the act of presentation.***
- ***Recoverability***

  Recoverability is the ability to reach a desired goal after recognition of some error in a previous interaction. **There are two directions in which recovery can occur, forward or backward.**
  - ***Forward error recovery*** involves the acceptance of the current state and negotiation from that state towards the desired state. Forward error recovery may be the only possibility for recovery if the effects of interaction are not revocable
  - ***Backward error recovery*** is an attempt to undo the effects of previous interaction in order to return to a prior state before proceeding. In a text editor, a mistyped keystroke might wipe out a large section of text which you would want to retrieve by an equally simple undo button. Recovery can be initiated by the system or by the user.

Responsiveness

Responsiveness measures the rate of communication between the system and the user. Response time is defined as the duration of time needed by the system to express state changes to the user. Instantaneous means that the user perceives system reactions as immediate.

Absolute response time is response time stability. Response time stability covers the invariance of the duration for identical or similar computational resources. For **example**, pull-down menus are expected to pop up instantaneously as soon as a mouse button is pressed.

Task Conformance

- ***Task completeness*** addresses the coverage issue.

➢ ***Task adequacy*** addresses the user's understanding of the tasks.

Task completeness refers to the level to which the system services can be mapped onto all of the user tasks.

2.3.3 STANDARDS

Standards for interactive system design are usually set by national or international bodies to ensure compliance with a set of design rules by a large community. Standards can apply specifically to either the hardware or the software used to build the interactive system.

Underlying theory

- Hardware standards are based on an understanding of physiology or ergonomics/human factors, the results of which are relatively well known, fixed and readily adaptable to design of the hardware.
- Software standards are based on theories from psychology or cognitive science, which are less well formed, still evolving and not very easy to interpret in the language of software design.
- Hardware standards can directly relate to a hardware specification and still reflect the underlying theory.
- Software standards would have to be more vaguely worded.

Examples of the language of standards for displays:

> 11.3 *Arrangement of displays*
> 11.3.1 Vertical Grouping. The engine display parameters shall be arranged so that the primary or most important display for a particular engine and airplane (thrust, torque, RPM, etc.) be located at the top of the display group if a vertical grouping is provided. The next most important display parameter shall be positioned under the primary display progressing down the panel with the least important at the bottom.

(a) A typical example of a military standard

> 5.1 *Subdivision of the display area*
> In consideration of a simple, fast and accurate visual acquisition, the display area shall be divided into different sub-areas.
> Such a division should be:
> - Input area
> - Output area
> - Area for operational indications (such as status and alarms)

(b) From German standard DIN 66 234 Part 3 (1984), adapted from Smith [324]

> 5.15.3.2.1 *Standardization*
> The content of displays within a system shall be presented in a consistent manner.

(c) From US military standard MIL-STD-1472C, revised (1983), adapted from Smith [324]

Figure 7.1 Sample design standards for displays. Adapted from Smith [324].

Change Hardware is more difficult and expensive to change than software, which is usually designed to be very flexible. Consequently, requirements changes for hardware do not occur as frequently as for software. Since standards are also relatively stable, they are more suitable for hardware than software.

- Set by national or the UK Ministry of defense has published an Interim Defense Standard 00–25 on ***Human Factors for Designers of Equipment***, produced in 12 parts:

- Part 1 Introduction
- Part 2 Body Size
- Part 3 Body Strength and Stamina
- Part 4 Workplace Design
- Part 5 Stresses and Hazards
- Part 6 Vision and Lighting
- Part 7 Visual Displays
- Part 8 Auditory Information
- Part 9 Voice Communication
- Part 10 Controls
- Part 11 Design for Maintainability
- Part 12 Systems

In the beginning of that document, the following definition of usability is given:

- ***Usability*** The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.

- ***Effectiveness*** The accuracy and completeness with which specified users can achieve specified goals in particular environments.
- ***Efficiency*** The resources expended in relation to the accuracy and completeness of goals achieved.
- ***Satisfaction*** The comfort and acceptability of the work system to its users and other people affected by its use.

2.3.4 GUIDELINES

The majority of design rules for interactive systems are suggestive and more general guidelines. The more abstract the guideline, the more it resembles **the principles**. The more specific the guideline, the more suited it is **to detailed design**. The guidelines can also be automated to some extent, providing a direct means for translating detailed design specifications into actual implementation.

Example:

The basic categories of the Smith and Mosier guidelines are:

1. Data Entry
2. Data Display
3. Sequence Control
4. User Guidance
5. Data Transmission
6. Data Protection

Each of these categories is further broken down into more specific subcategories which contain the particular guidelines.

Sample guidelines from Smith and Mosier [325]:

1. **Data Entry**

1.1 *Position Designation*

1.1–1 **Distinctive Cursor**

For position designation on an electronic display, provide a movable cursor with distinctive visual features (shape, blink, etc.).

Exception When position designation involves only selection among displayed alternatives, highlighting selected items might be used instead of a separately displayed cursor.

Comment When choosing a cursor shape, consider the general content of the display. For instance, an underscore cursor would be difficult to see on a display of underscored text, or on a graphical display containing many other lines.

Comment If the cursor is changed to denote different functions (e.g. to signal deletion rather than entry), then each different cursor should be distinguishable from the others.

Comment If multiple cursors are used on the same display (e.g. one for alphanumeric entry and one for line drawing), then each cursor should be distinguishable from the others.

Reference Whitfield, Ball and Bird, 1983

See also 1.1–17 Distinctive multiple cursors
4.0–9 Distinctive cursor

Figure 7.2 Sample guideline from Smith and Mosier [325], courtesy of The MITRE Corporation

Table 2.5 Comparison of dialog styles mentioned in guidelines

Table 7.4 Comparison of dialog styles mentioned in guidelines

Smith and Mosier [325]	Mayhew [230]
Question and answer	Question and answer
Form filling	Fill-in forms
Menu selection	Menus
Function keys	Function keys
Command language	Command language
Query language	—
Natural language	Natural language
Graphic selection	Direct manipulation

2.3.5 GOLDEN RULES AND HEURISTICS

A number of advocates of user-centered design have presented sets of ‘golden rules’ or heuristics. There are many sets of heuristics, but the most well used are Nielsen’s ten heuristics, Shneiderman’s eight golden rules and Norman’s seven principles. Nielsen’s ten heuristics will be discussed in evaluation technique topic.

Schniedermann’s Eight Golden Rules of Interface Design

Shneiderman’s eight golden rules provide a convenient and succinct summary of the key principles of interface design. They are intended to be used during design but can also be applied, like Nielsen’s heuristics, to the evaluation of systems. Notice how they relate to the abstract principles discussed earlier.

- ***Strive for consistency*** in action sequences, layout, terminology, command use and so on.

- ***Enable frequent users to use shortcuts***, such as abbreviations, special key sequences and macros, to perform regular, familiar actions more quickly.
- ***Offer informative feedback*** for every user action, at a level appropriate to the magnitude of the action.
- ***Design dialogs to yield closure*** so that the user knows when they have completed a task.
- ***Offer error prevention and simple error handling*** so that, ideally, users are prevented from making mistakes and, if they do, they are offered clear and informative instructions to enable them to recover.
- ***Permit easy reversal of actions*** in order to relieve anxiety and encourage exploration, since the user knows that he can always return to the previous state.
- ***Support internal locus of control*** so that the user is in control of the system, which responds to his actions.
- ***Reduce short-term memory load*** by keeping displays simple, consolidating multiple page displays and providing time for learning action sequences.

Norman's Seven Principles for Transforming Difficult Tasks into Simple Ones

Norman summarizes user-centered design using the following **seven principles**:

- ***Use both knowledge in the world and knowledge in the head.*** People work better when the knowledge they need to do a task is available externally either explicitly or through the constraints imposed by the environment.

  But experts also need to be able to internalize regular tasks to increase their efficiency. So systems should provide the necessary knowledge within the environment and their operation should be transparent to support the user in building an appropriate mental model of what is going on.
- ***Simplify the structure of tasks.*** Tasks need to be simple in order to avoid complex problem solving and excessive memory load. There are a number of ways to simplify the structure of tasks. One is to provide mental aids to help the user keep track of stages in a more complex task.
- ***Make things visible.*** bridge the gulfs of execution and evaluation. The interface should make clear what the system can do and how this is achieved, and should enable the user to see clearly the effect of their actions on the system.
- ***Get the mappings right.*** User intentions should map clearly onto system controls. User actions should map clearly onto system events. So it should be clear what does what and by how much. Controls, sliders and dials should reflect the task – so a small movement has a small effect and a large movement a large effect.
- ***Exploit the power of constraints*** both natural and artificial. Constraints are things in the world that make it impossible to do anything but the correct action in the correct way. **Example** is a jigsaw puzzle, where the pieces only fit together in one way. Here the physical constraints of the design guide the user to complete the task.
- **Design for error.** To err is human, so anticipate the errors the user could make and design recovery into the system.
- ***When all else fails, standardize.*** If there are no natural mappings then arbitrary mappings should be standardized so that users only have to learn them once. It is this standardization principle that enables drivers to get into a new car and drive it with very little difficulty – key controls are standardized. Occasionally one might switch on the indicator lights instead of the windscreen wipers, but the critical controls (accelerator, brake, clutch, steering) are always the same.

Norman's seven principles provide a useful summary of his user-centered design philosophy.

2.4 EVALUATION TECHNIQUES:

2.4.1 WHAT IS EVALUATION?

Evaluation is assessing the system designs and test systems to ensure that they actually behave as expected and meet user requirements. This is the role of evaluation.

- It should occur throughout the design life cycle and the results should be feedback into modification to the design.
- It can be considered under two broad headings:
  i. Expert analysis
  ii. User Participation

2.4.2 GOALS OF EVALUATION

Evaluation has three main goals:

- ***To assess the extent and accessibility of the system's functionality***
- ***To assess users' experience of the interaction***
- ***To identify any specific problems with the system.***

2.4.3 EVALUATION THROUGH EXPERT ANALYSIS

A number of methods have been proposed to evaluate interactive systems through expert analysis. These depend upon the designer, or a human factors expert, taking the design and assessing the impact that it will have upon a typical user.

The basic intention is to identify any areas that are likely to cause difficulties because they violate known cognitive principles, or ignore accepted empirical results. These methods can be used at any stage in the development process from a design specification, through story boards and prototypes, to full implementations, making them flexible evaluation approaches.

They are also relatively cheap, since they do not assess actual use of the system, only whether or not a system upholds accepted usability principles.

Four approaches to expert analysis are:

1. ***Cognitive Walkthrough***
2. Heuristic Evaluation
3. the use of models
4. use of previous work

2.4.3.1 COGNITIVE WALKTHROUGH

The origin of cognitive walk through is the code walkthrough familiar in software engineering.

The main focus of the cognitive walkthrough is to establish how easy a system is to learn that is learning through exploration. Studies show many users prefer to learn how to use a system by exploring its functionality hands on, and not after sufficient training or examination of user's manual. So the checks that are made during the walkthrough ask questions that address exploratory learning.

For this evaluators go through each step in the task and provide a story about why that step is or is not good for a new user. To do a cognitive walkthrough we need four things:

- A specification or prototype of the system. It doesn't have to be complete, but it should be fairly detailed such as the location and wording for a menu can make a big difference.
- A description of the task the user is to perform on the system. This should be a representation task that most users will want to do.
- A complete, written list of the actions needed to complete the task with the proposed system.
- An indication of who the users are and what kind of experience and knowledge the evaluators can assume about them.

Evaluators tell a believable story about the following four questions for each step in the action sequence.

i. Is the effect of the action the same as the user's goal at that point? **Example**: If the effect of the action is to save a document, is 'saving a document' what the user wants to do?
ii. Will users see that the action is available? **Example**: Will users see the button or menu item that is used to produce the action?
iii. Once users have found the correct action, will they know it is the one they need? **Example**: While the previous question was about the visibility of the action, this one is about whether its meaning and effect is clear.
iv. After the action is taken, will users understand the feedback they get? **Example**: Will the feedback given to user be sufficient confirmation of what has actually happened?

It is important to document the cognitive walkthrough to keep a record of what is good and what needs improvement in the design. For this, standard evaluation forms are used. The cover form and other standard forms will have the above ***evaluation questions and corresponding answers*** as well as date and time of walkthrough, and names of evaluators.

Any negative answer should be documented on a separate usability problem report sheet. This sheet contains version of the system, date, evaluators and a detailed description of the usability problem. It will indicate the severity of the problem to the users. This ***information helps designers to decide priorities for correcting the design, since it is always not possible to fix every problem.***

Example: programming a video recorder by remote control

We can illustrate how the walkthrough method works using a simple example. Imagine we are designing a remote control for a video recorder (VCR) and are interested in the task of programming the VCR to do timed recordings. Our initial design is shown in Figure 9.1. The picture on the left illustrates the handset in normal use, the picture on the right after the timed record button has been pressed. The VCR allows the user to program up to three timed recordings in different 'streams'. The next available stream number is automatically assigned. We want to know whether our design supports the user's task. We begin by identifying a representative task.

Program the video to time-record a program starting at 18.00 and finishing at 19.15 on channel 4 on 24 February 2005.

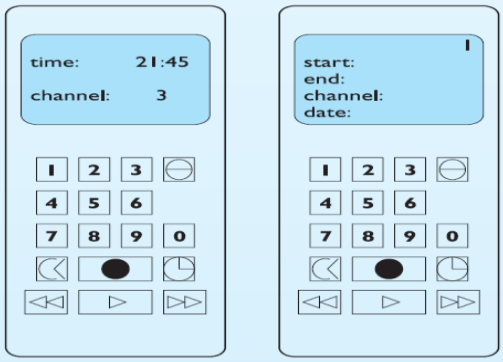


Figure 9.1 An initial remote control design

Having determined our action list we are in a position to proceed with the walkthrough. For each action (1–10) we must answer the four questions and tell a story about the usability of the system. Beginning with UA 1:

UA 1: Press the 'timed record' button
Question 1: Is the effect of the action the same as the user's goal at that point?
The timed record button initiates timer programming. It is reasonable to assume that a user familiar with VCRs would be trying to do this as his first goal.
Question 2: Will users see that the action is available?
The 'timed record' button is visible on the remote control.
Question 3: Once users have found the correct action, will they know it is the one they need?
It is not clear which button is the 'timed record' button. The icon of a clock (fourth button down on the right) is a possible candidate but this could be interpreted as a button to change the time. Other possible candidates might be the fourth button down on the left or the filled circle (associated with record). In fact, the icon of the clock is the correct choice but it is quite possible that the user would fail at this point. This identifies a potential usability problem.
Question 4: After the action is taken, will users understand the feedback they get?
Once the action is taken the display changes to the timed record mode and shows familiar headings (start, end, channel, date). It is reasonable to assume that the user would recognize these as indicating successful completion of the first action.

So we find we have a potential usability problem relating to the icon used on the 'timed record' button. We would now have to establish whether our target user group could correctly distinguish this icon from others on the remote.

The analysis proceeds in this fashion, with a walkthrough form completed for each action. We will leave the rest of the walkthrough for you to complete as an exercise. What other usability problems can you identify with this design?

2.4.3.2 HEURISTIC EVALUATION:

A heuristic is a guideline or general principle or rule of thumb that can guide a design decision or be used to critique a decision that has already been made.

It is useful for evaluating early design but also on prototypes, storyboards and fully functioning systems. It is flexible, relatively cheap approach. So it is called as ***discount usability technique.***

Evaluators assess the severity of each usability problem, based on 4 factors:

- how common is the problem.
- how easy is it for the user to overcome.
- how will it be a one-off problem or a persistent one.
- how seriously will the problem be perceived? These can be combined into a overall rating on a scale of 0-4:
  - 0 = I don't agree that this is a usability problem at all
  - 1 = Cosmetic problem only: need not be fixed unless extra time is available on project
  - 2 = Minor usability problem: fixing this should be given low priority
  - 3 = Major usability problem: important to fix, so should be given high priority
  - 4 = Usability catastrophe imperative to fix this before product can be released

Nielsen's ten heuristics are:

1***. Visibility of system status*** Always keep users informed about what is going on, through appropriate feedback within reasonable time. For **example**, if a system operation will take some time, give an indication of how long and how much is complete.

2. *Match between system and the real* world The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in natural and logical order.

3. ***User control and freedom*** Users often choose system functions by mistake and need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialog. Support undo and redo.

4. ***Consistency and standards*** Users should not have to wonder whether words, situations or actions mean the same thing in different contexts. Follow platform conventions and accepted standards.

5. ***Error prevention*** Make it difficult to make errors. Even better than good error messages is a careful design that prevents a problem from occurring in the first place.

6. ***Recognition rather than recall*** Make objects, actions and options visible. The user should not have to remember information from one part of the dialog to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. ***Flexibility and efficiency of use*** Allow users to tailor frequent actions. Accelerators – unseen by the novice user – may often speed up the interaction for the expert user to such an extent that the system can cater to both inexperienced and experienced users.

8. ***Aesthetic and minimalist design*** Dialog should not contain information that is irrelevant or rarely needed.

9. ***Help users recognize, diagnose and recover from errors*** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. ***Help and documentation*** Few systems can be used with no instructions so it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Once each evaluator has completed their separate assessment, all of the problems are collected and the mean severity ratings calculated. The design team will then determine the ones that are the most important and will receive attention first.

2.4.3.3 MODEL-BASED EVALUATION

Certain cognitive and design models provide a means of combining design specification and evaluation into the same framework.

For **example**, the GOMS (goals, operators, methods and selection) model predicts user performance with a particular interface and can be used to filter particular design options. Similarly, lower-level modeling techniques such as the keystroke-level model pro-vide predictions of the time users will take to perform low-level physical tasks.

Design methodologies such as design rationale have a role to play in evaluation at the design stage. Dialog models can also be used to evaluate dialog sequences for problems, such as unreachable states, circular dialogs and complexity. Models such as state transition networks are useful for evaluating dialog designs prior to implementation.

2.4.3.4 USING PREVIOUS STUDIES IN EVALUATION

A final approach to expert evaluation exploits inheritance, using previous results as evidence to support (or refute) aspects of the design.

2.4.4 EVALUATION THROUGH USER PARTICIPATION

User participation in evaluation tends to occur in the later stages of development when there is at least a working prototype of the system.

2.4.4.1 STYLES OF EVALUATION

There are two distinct evaluation styles: those performed under laboratory conditions and those conducted in the work environment or 'in the field'.

Laboratory Studies

In the first type of evaluation studies, users are taken out of their normal work environment to take part in controlled tests, often in a specialist usability laboratory. ***This approach has a number of benefits and disadvantages.***

A well-equipped usability laboratory may contain sophisticated audio/visual recording and analysis facilities, two-way mirrors, instrumented computers and the like, which cannot be replicated in the work

environment. In addition, the participant operates in an interruption-free environment. However, the lack of context. There are, however, some situations where laboratory observation is the only option.

For **example**, if the system is to be located in a dangerous or remote location, such as a space station. Also some very constrained single-user tasks may be adequately performed in a laboratory. We may want to compare alternative designs within a con-trolled context. For these types of evaluation, laboratory studies are appropriate.

Field Studies

The second type of evaluation takes the designer or evaluator out into the user's work environment in order to observe the system in action. Again this approach has its pros and cons.

High levels of ambient noise, greater levels of movement and constant interruptions, such as phone calls, all make field observation difficult. However, the very 'open' nature of the situation means that you will observe interactions between systems and between individuals that would have been missed in a laboratory study. The context is retained and you are seeing the user in his 'natural environment'. In addition, some activities, such as those taking days or months, are impossible to study in the laboratory

2.4.4.2 EMPIRICAL METHODS: EXPERIMENTAL EVALUATION

One of the most powerful methods of evaluating a design or an aspect of a design is to use a controlled experiment. This provides empirical evidence to support a particular claim or hypothesis. It can be used to study a wide range of different issues at different levels of detail.

Factors of Experimental design

- ***Participants*:** In evaluation experiments, participants should be chosen to match the expected user population as closely as possible. If participants are not actual users, they should be chosen to be of a similar age and level of education as the intended user group.
- ***Variables*:** Experiments manipulate and measure variables under controlled conditions, in order to test the hypothesis. There **are two main types of variable**: those that are 'manipulated' or changed (known as the independent variables) and those that are measured (the dependent variables).
- ***Hypotheses:*** A hypothesis is a prediction of the outcome of an experiment. It is framed in terms of the independent and dependent variables, stating that a variation in the independent variable will cause a difference in the dependent variable. The aim of the experiment is to show that this prediction is correct.
- ***Experimental design*:** In order to produce reliable and generalizable results, an experiment must be carefully designed.
- ***Statistical measures:*** The first two rules of statistical analysis are to *look* at the data and to *save* the data. Our choice of statistical analysis depends on the type of data and the questions we want to answer. It is worth having important results checked by an experienced statistician, but in many situations standard tests can be used.

  Variables can be classified as either ***discrete variables* or *continuous variables*.**

  - A **discrete variable** can only take a finite number of values or *levels.* for **example**, a screen color that can be red, green or blue.
  - A **continuous variable** can take any value (although it may have an upper or lower limit), for **example** a person's height or the time taken to complete a task. A special case of continuous data is when they are *positive*, for **example** a response time cannot be negative

The dependent variable is the measured one and subject to random experimental variation. In the case when this variable is continuous, the random variation may take a special form. If the form of the data follows a known ***distribution*** then special and more powerful statistical tests can be used. Such tests are called ***parametric tests*** and the most common of these are used when the variation follows the ***normal distribution***.

This means that if we plot a histogram of the random errors, they will form the well-known bell-shaped graph (Figure 9.2). Many of these tests are fairly *robust*, that is they give reasonable results even when the data are not precisely normal. This means that you need not worry too much about checking normality during early analysis.

There are ways of checking whether data are really normal. However, as a general rule**, *if data can be seen as the sum or average of many small independent effects they are likely to be normal.***

For **example**, the time taken to complete a ***complex* task** is the sum of the times of all the minor tasks of which it is composed.

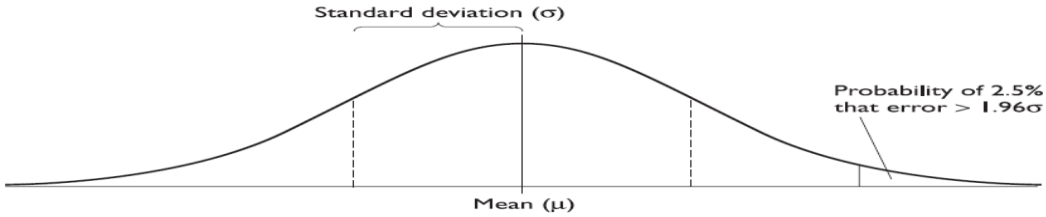


Figure 9.2 Histogram of normally distributed errors

When we cannot assume that data are normally distributed, we must often resort to ***non-parametric* tests.** These are statistical tests that make no assumptions about the particular distribution and are usually based purely on the ranking of the data.

A third sort of test is the contingency table, where we classify data by several discrete attributes and then count the number of data items with each attribute combination.

Table 9.1 lists some of the standard tests categorized by the form of independent and dependent variables (discrete/continuous/normal). Normality is not an issue for the independent variable, but a special case is when it is discrete with only two values, for **example** comparing two systems. We cannot describe all the techniques here; for this you should use a standard statistics text, such as one of those recommended in the reading list. The table is only intended to guide you in your choice of test.

An extensive and accurate analysis is no use if it answers the wrong question.

Table 9.1 Choosing a statistical technique

Independent variable	Dependent variable	
Parametric		
Two valued	Normal	Student's *t* test on difference of means
Discrete	Normal	ANOVA (ANalysis Of VAriance)
Continuous	Normal	Linear (or non-linear) regression factor analysis
Non-parametric		
Two valued	Continuous	Wilcoxon (or Mann–Whitney) rank-sum test
Discrete	Continuous	Rank-sum versions of ANOVA
Continuous	Continuous	Spearman's rank correlation
Contingency tests		
Two valued	Discrete	No special test, see next entry
Discrete	Discrete	Contingency table and chi-squared test
Continuous	Discrete	(Rare) Group independent variable and then as above

Examples of questions one might ask about the data are as follows:

1. Is there a difference?
2. How big is the difference?
3. How accurate is the estimate?

An **example**: Evaluating icon designs

Imagine you are designing a new interface to a document-processing package, which is to use icons for presentation. You are considering two styles of icon design and you wish to know which design will be easier for users to remember. One set of icons uses naturalistic images (based on a paper document metaphor), the other uses abstract images (see Figure 9.3). How might you design an experiment to help you decide which style to use?

The first thing you need to do is form a hypothesis: what do you consider to be the likely outcome? In this case, you might expect the natural icons to be easier to recall since they are more familiar to users. We can therefore form the following hypothesis:

Users will remember the natural icons more easily than the abstract ones.

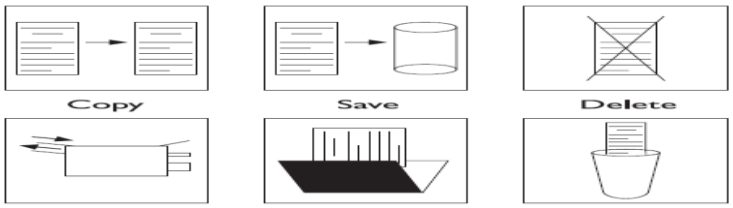


Figure 9.3 Abstract and concrete icons for file operations

The null hypothesis in this case is that there will be no difference between recall of the icon types. This hypothesis clearly identifies the independent variable for our experiment: we are varying the style of icon. The **independent variable has two levels: natural and abstract**. However, when we come to consider the **dependent variable**, things are not so obvious. We have expressed our hypothesis in terms of users being able to remember more easily. How can we measure this? First we need to clarify exactly what we mean by the phrase more easily: are we concerned with the user's performance in terms of accurate recall or in terms of speed.

We need to control the experiment so that any differences we observe are clearly attributable to the **independent variable,** and so that our measurements of the **dependent variables are comparable**. To do this, we provide an interface that is identical in every way except for the icon design, and a selection task that can be repeated for each condition.

The latter could be either a naturalistic task (such as producing a document) or a more artificial task in which the user has to select the appropriate icon to a given prompt.

The **second task has the advantage** that it is more controlled (there is little variation between users as to how they will perform the task) and it can be varied to avoid transfer of learning. Before performing the selection task, the users will be allowed to learn the icons in controlled conditions: for **example**, they may be given a fixed amount of time to learn the icon meanings.

In order to avoid learning effects from icon position, the placing of icons in the block can be randomly varied on each presentation. Each user performs the selection task under each condition. In order to avoid transfer of learning, the users are divided into two groups with each group taking a different starting condition. For each user, we measure the time taken to complete the task and the number of errors made.

Worked exercise *Design an experiment to test whether adding color coding to an interface will improve accuracy. Identify your hypothesis, participant group, dependent and independent variables, experimental design, task and analysis approach.*

Answer The following is only an example of the type of experiment that might be devised.

Participants Taken from user population.

Hypothesis Color coding will make selection more accurate.

IV (Independent Variable) Color coding.

DV (Dependent Variable) Accuracy measured as number of errors.

Design Between-groups to ensure no transfer of learning (or within-groups with appropriate safeguards if participants are scarce).

Task The interfaces are identical in each of the conditions, except that, in the second, color is added to indicate related menu items. Participants are presented with a screen of menu choices (ordered randomly) and verbally told what they have to select. Selection must be done within a strict time limit when the screen clears. Failure to select the correct item is deemed an error. Each presentation places items in new positions. Participants perform in one of the two conditions.

Analysis *t* test.

Studies of group of users

Experiments to evaluate elements of group systems bring additional problems. Given the complexities of human–human communication and group working, it is hardly surprising that experimental studies of groups and of groupware are more difficult than the corresponding single-user experiments.

Example: evaluating a shared application with video connections between the participants and consider some of the problems we will encounter.

- ***The participant groups*** To organize, say, 10 experiments of a single-user system require 10 participants. For an experiment involving groups of three, we will, of course, need 30 participants for the same number of experiments.

  In addition, experiments in group working are often longer than the single-user equivalents as we must allow time for the group to 'settle down' and some rapport to develop. This all means more disruption for participants and possibly more expense payments.

- ***The experimental task*** Choosing a suitable task is also difficult. We may want to test a variety of different task types: creative, structured, information passing, and so on.
- ***Data gathering*** Even in a single-user experiment we may well use several video cameras as well as direct logging of the application. In a group setting this is replicated for each participant. So for a three-person group, we are trying to synchronize the recording of six or more video sources and three keystroke logs. To compound matters, these may be spread over different offices, or even different sites. The technical problems are clearly enormous.
- ***Field studies with groups*** There are, of course, problems with taking groups of users and putting them in an experimental situation. If the groups are randomly mixed, then we are effectively examining the process of group formation, rather than that of a normal working group. Even where a pre-existent group is used, excluding people from their normal working environment can completely alter their working patterns.

2.4.4.3 OBSERVATIONAL TECHNIQUES

The techniques used to evaluate systems by observing user behavior.

i. ***Think aloud and cooperative evaluation*** - Think aloud is a form of observation where the user is asked to talk through what he is doing as he is being observed; for **example**, describing what he believes is happening, why he takes an action, what he is trying to do.

ii. ***Protocol analysis*** – Paper and pencil, Audio recording, Video recording, Computer logging and user notebooks.

iii. ***Automatic protocol analysis tools*** - Analyzing protocols, whether video, audio or system logs, is time consuming and tedious by hand. It is made harder if there is more than one stream of data to synchronize. One solution to this problem is to provide automatic analysis tools to support the task. These offer a means of editing and annotating video, audio and system logs and synchronizing these for detailed analysis. **Example**:

EVA (Experimental Video Annotator) is a system that runs on a multimedia work-station with a direct link to a video recorder. The evaluator can devise a set of buttons indicating different events. These may include timestamps and snapshots, as well as notes of expected events and errors. The buttons are used within a recording session by the evaluator to annotate the video with notes.

During the session the user works at a workstation and is recorded, using video and perhaps audio and system logging as well. The evaluator uses the multimedia workstation running EVA. On the screen are the live video record and a view of the user's screen. The evaluator can use the buttons to tag interesting events as they occur and can record additional notes using a text editor. After the session, the evaluator can ask to review the tagged segments and can then use these and standard video controls to search the information.

iv. ***Post-task walkthroughs*** - Often data obtained via direct observation lack interpretation. A walkthrough attempts to alleviate the problem, by reflecting the participants' actions back to them after the event.

2.4.4.4 QUERY TECHNIQUES - Another set of evaluation techniques relies on asking the user about the interface directly. Query techniques can be useful in eliciting detail of the user's view of a system.

i. **Interviews** - Interviewing users about their experience with an interactive system provides a direct and structured way of gathering information.

Interviews have the advantages that the level of questioning can be varied to suit the context and that the evaluator can probe the user more deeply on interesting issues as they arise. An interview will usually follow a top-down approach, starting with a general question about a task and progressing to more leading questions (often of the form 'why?' or 'what if?') to elaborate aspects of the user's response.

ii**. Questionnaires** - An alternative method of querying the user is to administer a questionnaire. This is clearly less flexible than the interview technique, since questions are fixed in advance, and it is likely that the questions will be less probing.

Styles of Question:

General These are questions that help to establish the background of the user and his place within the user population. They include questions about age, sex, occupation, place of residence, and so on. They may also

include questions on previous experience with computers, which may be phrased as open-ended, multi-choice or scalar questions.

Open-ended These ask the user to provide his own unprompted opinion on a question.

Scalar These ask the user to judge a specific statement on a numeric scale, usually corresponding to a measure of agreement or disagreement with the statement.

For **example**, It is easy to recover from mistakes.

Disagree 1 2 3 4 5 Agree

Multi-choice Here the respondent is offered a choice of explicit responses, and may be asked to select only one of these, or as many as apply. For **example**,

How do you most often get help with the system (tick one)?

Online manual	q
Contextual help system	q
Command prompt	q
Ask a colleague	q

Ranked These place an ordering on items in a list and are useful to indicate a user's preferences.

For **example**, Please rank the usefulness of these methods of issuing a command (1 most useful,

2 next, 0 if not used).

Menu selection	q
Command line	q
Control key accelerator	q

2.4.4.5 EVALUATION THROUGH MONITORING PHYSIOLOGICAL RESPONSES

It is called objective usability testing, ways of monitoring physiological aspects of computer use. Potentially this will allow us not only to see more clearly exactly what users do when they interact with computers, but also to measure how they feel.

The two areas receiving the most attention to date are eye tracking and physiological measurement.

i. *Eye tracking for usability evaluation*

Eye tracking has been possible for many years, but recent improvements in hard-ware and software have made it more viable as an approach to measuring usability.There are many possible measurements related to usability evaluation including:

Number of fixations The more fixations the less efficient the search strategy.

Fixation duration Longer fixations may indicate difficulty with a display.

Scan path indicating areas of interest, search strategy and cognitive load.

Eye tracking for usability is still very new and equipment is prohibitively expensive for everyday use.

ii. *Physiological Measurements*

Emotional response is closely tied to physiological changes. These include changes in heart rate, breathing and skin secretions. Measuring these physiological responses may therefore be useful in determining a user's emotional response to an interface.

Physiological measurement involves attaching various probes and sensors to the user. These measure a number of factors:

- **Heart activity**, indicated by blood pressure, volume and pulse. These may respond to stress or anger.
- ***Activity of the sweat glands***, indicated by skin resistance or galvanic skin response (GSR). These are thought to indicate levels of arousal and mental effort.
- ***Electrical activity in muscle***, measured by the electromyogram (EMG). These appear to reflect involvement in a task.
- ***Electrical activity in the brain***, measured by the electroencephalogram (EEG). These are associated with decision making, attention and motivation.

2.4.5 CHOOSING AN EVALUATION METHOD

2.4.5.1 FACTORS DISTINGUISHING EVALUATION TECHNIQUES

- the stage in the cycle at which the evaluation is carried out (Design Vs Implementation)
- the style of evaluation (Laboratory Vs Field Studies)
- the level of subjectivity or objectivity of the technique (Subjective Vs Objective)

- the type of measures provided (Qualitative Vs Quantitative measures)
- the information provided
- the immediacy of the response
- the level of interference implied (Intrusiveness)
- the resources required.

2.4.5.2 A CLASSIFICATION OF EVALUATION TECHNIQUES

Using the factors discussed in the previous section we can classify the evaluation techniques.

Table 9.4 Classification of analytic evaluation techniques

	Cognitive walkthrough	Heuristic evaluation	Review based	Model based
Stage	Throughout	Throughout	Design	Design
Style	Laboratory	Laboratory	Laboratory	Laboratory
Objective?	No	No	As source	No
Measure	Qualitative	Qualitative	As source	Qualitative
Information	Low level	High level	As source	Low level
Immediacy	N/A	N/A	As source	N/A
Intrusive?	No	No	No	No
Time	Medium	Low	Low–medium	Medium
Equipment	Low	Low	Low	Low
Expertise	High	Medium	Low	High

Table 9.5 Classification of experimental and query evaluation techniques

	Experiment	Interviews	Questionnaire
Stage	Throughout	Throughout	Throughout
Style	Laboratory	Lab/field	Lab/field
Objective?	Yes	No	No
Measure	Quantitative	Qualitative/ quantitative	Qualitative/ quantitative
Information	Low/high level	High level	High level
Immediacy	Yes	No	No
Intrusive?	Yes	No	No
Time	High	Low	Low
Equipment	Medium	Low	Low
Expertise	Medium	Low	Low

Table 9.6 Classification of observational evaluation techniques

	Think aloud[1]	Protocol analysis[2]	Post-task walkthrough
Stage	Implementation	Implementation	Implementation
Style	Lab/field	Lab/field	Lab/field
Objective?	No	No	No
Measure	Qualitative	Qualitative	Qualitative
Information	High/low level	High/low level	High/low level
Immediacy	Yes	Yes	No
Intrusive?	Yes	Yes[3]	No
Time	High	High	Medium
Equipment	Low	High	Low
Expertise	Medium	High	Medium

1 Assuming a simple paper and pencil record
2 Including video, audio and system recording
3 Except system logs

Table 9.7 Classification of monitoring evaluation techniques

	Eye tracking	Physiological measurement
Stage	Implementation	Implementation
Style	Lab	Lab
Objective?	Yes	Yes
Measure	Quantitative	Quantitative
Information	Low level	Low level
Immediacy	Yes	Yes
Intrusive?	No[1]	Yes
Time	Medium/high	Medium/high
Equipment	High	High
Expertise	High	High

1 If the equipment is not head mounted

2.5 UNIVERSAL DESIGN:

Universal design is the process of designing products so that they can be used by as many people as possible in as many situations as possible.

2.5.1 UNIVERSAL DESIGN PRINCIPLES

The principles give us a framework in which to develop universal designs.

- ***Principle one is equitable use:*** the design is useful to people with a range of abilities and appealing to all. No user is excluded or stigmatized.
- ***Principle two is flexibility in use:*** the design allows for a range of ability and preference, through choice of methods of use and adaptively to the user's pace, precision and custom.
- ***Principle three is that the system be simple and intuitive to use***, regardless of the knowledge, experience, language or level of concentration of the user. The design needs to support the user's expectations and accommodate different language and literacy skills. It should not be unnecessarily complex and should be organized to facilitate access to the most important areas.
- ***Principle four is perceptible information***: the design should provide effective communication of information regardless of the environmental conditions or the user's abilities.

  Information should be represented in different forms or modes (e.g. graphic, verbal, text, touch). Presentation should support the range of devices and techniques used to access information by people with different sensory abilities.
- ***Principle five is tolerance for error:*** minimizing the impact and damage caused by mistakes or unintended behavior. Potentially dangerous situations should be removed or made hard to reach. Potential hazards should be shielded by warnings. Systems should fail safe from the user's perspective and users should be supported in tasks that require concentration.
- ***Principle six is low physical effort***: systems should be designed to be comfortable to use, minimizing physical effort and fatigue. The physical design of the system should allow the user to maintain a natural posture with reasonable operating effort. Repetitive or sustained actions should be avoided.
- ***Principle seven requires size and space for approach and use*:** the placement of the system should be such that it can be reached and used by any user regardless of body size, posture or mobility.

  All physical components should be comfortably reachable by seated or standing users. Systems should allow for variation in hand size and provide enough room for assistive devices to be used.

2.5.2 MULTI-MODAL INTERACTION

Providing access to information through more than one mode of interaction is an important principle of universal design. Such design relies on ***multi-modal interaction.*** Since our interaction with the world is improved by multi-sensory input, it makes sense that interactive systems that utilize more than one sensory channel will also provide a richer interactive experience.

In addition, such multi-sensory or multi-modal systems support the principle of redundancy required for universal design, enabling users to access the system using the mode of interaction that is most appropriate to their abilities.

2.5.2.1 SOUND IN THE INTERFACE

Sound is an important contributor to usability. The dual presentation of information through sound and vision supports universal design, by enabling access for users with visual and hearing impairments

respectively. It also enables information to be accessed in poorly lit or noisy environments. Sound can convey transient information and does not take up screen space, making it potentially useful for mobile applications.

1. *Speech in the interface* – Language is rich and complex. Human beings have great and natural mastery of speech.

i. Structure of Speech - If we are fully to appreciate the problems involved with the computer-based recognition and generation of speech. The English language is made up of 40 *phonemes*, which are the atomic elements of speech. Each phoneme represents a distinct sound, there being 24 consonants and 16 vowel sounds. Language is more than simple sounds.

ii. Speech recognition - There have been many attempts at developing speech recognition systems, but, although commercial systems are now commonly and cheaply available, their success is still limited to single-user systems that require considerable training.

Different people to speak differently: accent, idiom, stress, volume, etc. The syntax of semantically similar sentences may vary. Background noises can interfere. **Example**: Phonetic Typewriter – Developed for Finnish (a phonetic language, written as it is said)

iii. Speech synthesis - Most speech synthesizers can deliver a degree of prosody, but in order to decide what intonation to give to a word, the system must have an understanding of the domain.

So an effective automatic reader would also need to be able to understand natural language, which is difficult. However***, for 'canned' messages and responses, the prosody can be hand coded yielding much more acceptable speech.*** Other problems:

Synthesized speech also brings other problems. Being transient, spoken output cannot be reviewed or browsed easily. It is intrusive, requiring either an increase in noise in the office environment or the use of headphones, either of which may be too large a price to pay for the benefits the system may offer.

There are some application areas in which speech synthesis has been successful***. For users who are blind or partially sighted, synthesized speech offers an output medium which they can access.***

Speech synthesis is also useful as a communication tool to assist people with physical disabilities that affect their speech.

iv Uninterpreted Speech - Speech does not have to be recognized by a computer to be useful in the interface. ***Fixed pre-recorded messages can be used to supplement or replace visual information.*** Recordings have natural human prosody and pronunciation, although quality is sometimes low.

Recordings of users' speech can also be very useful, especially in collaborative applications, **for** example many readers will have used voicemail systems. Also, recordings can be attached to other artifacts as *audio annotations* in order to communicate with others or to remind oneself at a later time. **For example**, audio annotations can be attached to Microsoft Word documents.

2. *Non-speech Sound* – Non-speech sound can be used in a number of ways in interactive systems. It is often used to provide transitory information, such as indications of network or system changes, or of errors. It can also be used to provide status information on background processes, since we are able to ignore continuous sounds but still respond to changes in those sounds

i. Auditory icons – Use natural sounds to represent different types of object or action. Natural sounds have associated semantics which can be mapped onto similar meanings in the interaction. **Example:** Throwing something away.

3. Earcons - An alternative to using natural sounds is to devise synthetic sounds. *Earcons* use structured combinations of notes, called *motives*, to represent actions and objects (see Figure 10.2). These vary according to rhythm, pitch, timbre, scale and volume.

There are two types of combination of earcon.

- *Compound earcons* combine different motives to build up a specific action, for **example** combining the motives for 'create' and 'file'.
- *Family earcons* represent compound earcons of similar types. As an **example**, operating system errors and syntax errors would be in the 'error' family.

Earcons provide a structured approach to designing sound for the interface. The most important element in distinguishing different sounds is timbre, the characteristic quality of the sound produced by different instruments and voices. Other factors such as pitch, rhythm and register should be used to supplement timbre in creating distinctive sets of musical earcons.

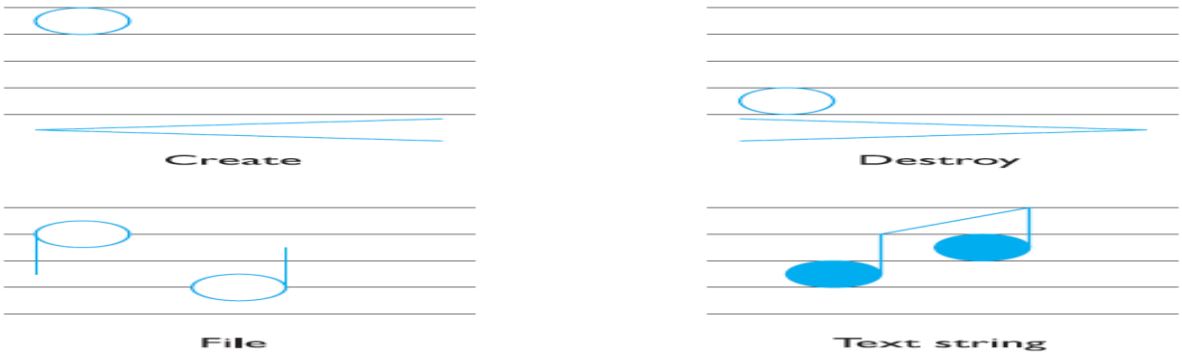


Figure 10.2 Earcons

2.5.2.2 TOUCH IN THE INTERFACE

The use of touch in the interface is known as haptic interaction. Haptics is a generic term relating to touch, but **it can be divided into two areas**:

cutaneous perception is concerned with tactile sensations through the skin.

Kinesthetic is the perception of movement and position. Both are useful in interaction but they require different technologies.

One **example** of a tactile device is an electronic – or soft – braille display. Braille displays are made up of a number of cells (typically between 20 and 80), each containing six or eight electronically controlled pins that move up and down to produce braille representations of characters displayed on the screen. Whereas printed braille normally has six dots per cell, electronic braille typically has eight pins, with the extra two representing additional information about that cell, such as cursor position and character case.

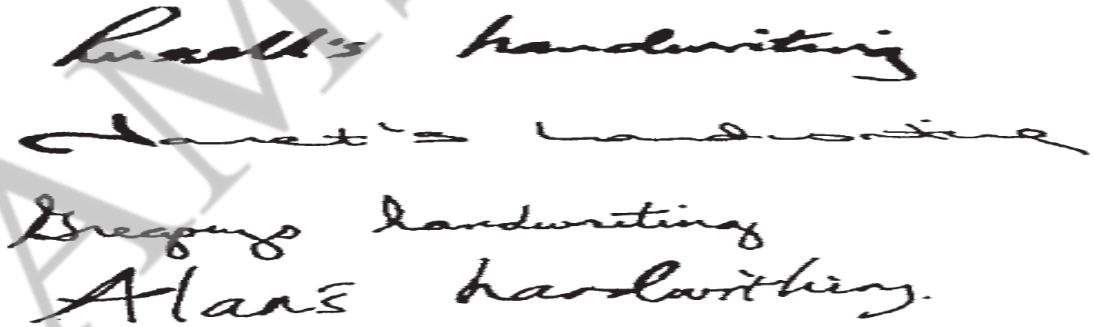
The other main type of haptic device is the force feedback device, which provides kinesthetic information back to the user, allowing him to feel resistance, textures, friction and so on.

2.5.2.3 HANDWRITING RECOGNITION

Like speech, we consider handwriting to be a very natural form of communication. The idea of being able to interpret handwritten input is very appealing, and hand-writing appears to offer both textual and graphical input using the same tools. There are problems associated with the use of handwriting as an input medium.

i Technology for Handwriting Recognition - The major piece of technology used to capture handwriting is the digitizing tablet. . Free-flowing strokes made with a pen are transformed into a series of coordinates, approximately one every 1/50th of a second (or at the sampling rate of the digitizer). Rapid movements produce widely spaced dots, in comparison with slow movements: this introduces immediate errors into the information, since the detail of the stroke between dots is lost, as is the pressure information. ***Digitizing tablets have been refined by incorporating a thin screen on top to display the information, producing electronic paper.***

ii Recognizing handwriting - The variation between the handwriting of individuals is large (see Figure 10.4); moreover, the handwriting of a single person varies from day to day, and evolves over the years.



when letters are individually written, with a small separation, the success of systems becomes more respectable, although they have to be trained to recognize the characteristics of the different users. If tested on an untrained person, success is limited again. Many of the solutions that are being attempted in speech recognition are also being tried in handwriting recognition systems, such as whole-word recognition, the use of context to disambiguate characters, and neural net-works, which learn by example.

2.5.2.4 GESTURE RECOGNITION

Gesture is a component of human–computer interaction that has become the subject of attention in multi-modal systems. Being able to control the computer with certain movements of the hand would be advantageous in many situations where there is no possibility of typing, or when other senses are fully occupied. It could also support communication for people who have hearing loss, if signing could be 'translated' into speech or vice versa. But, like speech, gesture is user dependent, subject to variation and co-articulation.

The technology for capturing gestures is expensive, using either computer vision or a special data glove. ***The data glove provides easier access to highly accurate information, but is a relatively intrusive technology, requiring the user to wear the special Lycra glove***. The interpretation of the sampled data is very difficult, since segmenting the gestures causes problems.

2.5.3 DESIGNING FOR DIVERSITY

Interfaces are usually designed to cater for the 'average' user, but unfortunately this may exclude people who are not 'average'. people are diverse and there are many factors that must be taken into account if we are to come close to universal design.

2.5.3.1 DESIGNING FOR USERS WITH DISABILITY

It is estimated that at least 10% of the population of every country has a disability that will affect interaction with computers. Employers and manufacturers of computing equipment have not only a moral responsibility to provide accessible products, but often also a legal responsibility. In many countries, legislation now demands that the workplace must be designed to be accessible or at least adaptable to all – the design of software and hardware should not unnecessarily restrict the job prospects of people with disabilities.

i Visual impairment **-** The rise in the use of graphical interfaces reduces the possibilities for visually impaired users. In text-based interaction, screen readers using synthesized speech or braille output devices provided complete access to computers: input relied on touch-typing, with these mechanisms providing the output. ***There are two key approaches to extending access: the use of sound and the use of touch*****.** A number of systems use sound to provide access to graphical interfaces for people with visual impairment.

Soundtrack

Soundtrack is an early example of a word processor with an auditory interface, designed for users who are blind or partially sighted [118]. The visual items in the display have been given auditory analogs, made up of tones, with synthesized speech also being used. A two-row grid of four columns is Soundtrack's main screen (see Figure 10.5); each cell makes a different tone when the cursor is in it, and by using these tones the user can navigate around the system. The tones increase in pitch from left to right, while the two rows have different timbres. Clicking on a cell makes it speak its name, giving precise information that can reorient a user who is lost or confused. Double clicking on a cell reveals a submenu of items associated with the main screen item. Items in the submenu also have tones; moving down the menu causes the tone to fall whilst moving up makes it rise. A single click causes the cell to speak its name, as before, whilst double clicking executes the associated action.

Soundtrack allows text entry by speaking the words or characters as they are entered, with the user having control over the degree of feedback provided. It was found that users tended to count the different tones in order to locate their position on the screen, rather than just listen to the tones themselves, although one user with musical training did use the pitch.

Soundtrack provides an auditory solution to representing a visually based word processor, though the results are not extensible to visual interfaces in general. However, it does show that the human auditory system is capable of coping with the demands of highly interactive systems, and that the notion of auditory interfaces is a reasonable one.

File Menu	Edit Menu	Sound Menu	Format Menu
Alert	Dialog	Document1	Document2

Mathematics for the blind

Solve the following equation: $3(x - 2) + 4 = 7 - 2(3 - x)$.

Did you do it in your head or use a piece of paper? When an equation is even slightly complex the instant response of a sighted person is to reach for paper and pencil. The paper acts as an *external memory*, allowing you to record and recall previous steps in a calculation. Blind children learning mathematics have to perform nearly all such calculations in their head, putting them at a severe disadvantage.

Mathtalk is a system developed as part of a European project to create a mathematics workstation for blind people [330]. It uses speech synthesis to speak formulae, and keyboard input to navigate and manipulate them. The first stage, simply speaking a formula out loud, is complex in itself. Given the spoken equation 'three *x* plus four equals seven', how do you know whether this is '$3x + 4 = 7$' or '$3(x + 4) = 7$'? To make it unambiguous one could say the latter as 'three open bracket *x* plus four close bracket equals seven', but this soon becomes very tedious. In fact, when reading mathematics people use several cues in their speech: longer and shorter gaps between terms, and prosody: rising and falling pitch (see Figure 10.6). The Mathtalk system includes a set of rules for generating such patterns suitable for most equations.

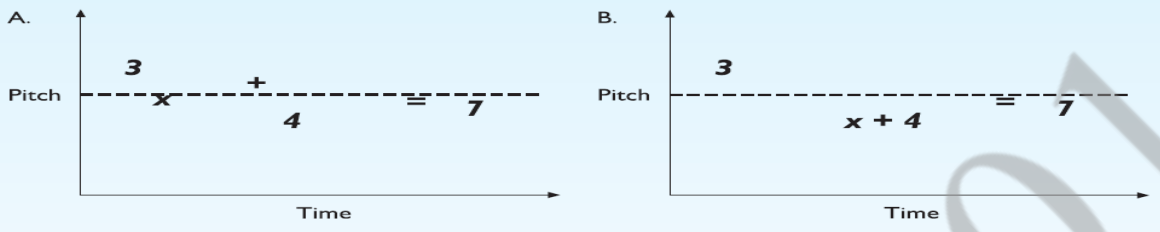


Figure 10.6 Pausing and pitch help distinguish between two expressions

Visual interaction with paper isn't just at the level of reading and writing whole equations. Recall from Chapter 1 that reading usually includes *regressions* where our eyes move backwards as well as forwards through text. Also, when seeing graphical material (remember that mathematics makes heavy use of brackets, symbols, superscripts, etc.), we rely on getting a quick feel for the material at a glance before examining it in detail. Both of these factors are crucial when reading an equation and so Mathtalk supports rapid keyboard-based navigation *within* each equation, and *algebra earcons*, short motives based on the rise and fall of the prosody of an equation.

Notice that Mathtalk uses *keyboard* input combined with speech output. Speech input is slow and error-prone compared with a keyboard. Braille output can also be used for mathematics, but only a small percentage of blind people read braille. Choosing the right input and output devices requires a deep knowledge of the user population and careful analysis of the intended tasks.

A ***limitation of this technology*** at present is that objects must be rendered using specialist software in order for the devices to calculate the appropriate force to apply back to the user.

ii Hearing impairment - Compared with a visual disability where the impact on interacting with a graphical interface is immediately obvious, a hearing impairment may appear to have little impact on the use of an interface. After all, it is the visual not the auditory channel that is predominantly used. To an extent this is true, and computer technology can actually enhance communication opportunities for people with hearing loss. Email and instant messaging are great levellers and can be used equally by hearing and deaf users alike.

Gesture recognition has also been proposed to enable translation of signing to speech or text, again to improve communication particularly with non-signers.

However, the increase in multimedia and the use of sound in interfaces has, ironically, created some access difficulties for people with hearing problems. Many multimedia presentations contain auditory narrative. If this is not supplemented by textual captions, this information is lost to deaf users. Captioning audio content, where there is not already a graphical or textual version, also has the **advantage of making audio files easier and more efficient to index and search, which in turn enhances the experience of all users**

iii Physical impairment - Users with physical disabilities vary in the amount of control and movement that they have over their hands, but many find the precision required in mouse control difficult. Speech input and output is an option for those without speech difficulties.

An alternative is the **eyegaze system**, which tracks eye movements to control the cursor, or a keyboard driver that can be attached to the user's head. If the user is unable to control head movement, gesture and movement tracking can be used to allow the user control.

iv Speech impairment **-** For users with speech and hearing impairments, multimedia systems provide a number of tools for communication, including synthetic speech and text-based communication and conferencing systems. Textual communication is slow, which can lower the effectiveness of the communication. Predictive algorithms have been used to anticipate the words used and fill them in, to reduce the amount of typing required. Conventions can help to provide context, which is lost from face-to-face communication.

For **example** the 'smilie' :-), to indicate a joke.

v Dyslexia **-** Users with cognitive disabilities such as dyslexia can find textual information difficult. In severe cases, speech input and output can alleviate the need to read and write and allow more accurate input and output. In cases where the problem is less severe, spelling correction facilities can help users.

However, these need to be designed carefully: often conventional spelling correction programs are useless for dyslexic users since the programs do not recognize their idiosyncratic word construction methods. As well as simple transpositions of characters, dyslexic users may spell phonetically, and correction programs must be able to deal with these errors.

Colour coding information can help in some cases and provision of graphical information to support textual can make the meaning of text easier to grasp.

vi Autism **-** Autism affects a person's ability to communicate and interact with people around them and to make sense of their environment. This manifests itself in a range of ways but is characterized by the *triad of impairments*:

Social interaction – problems in relating to others in a meaningful way or responding appropriately to social situations.

Communication – problems in understanding verbal and textual language including the use of gestures and expressions.

Imagination – problems with rigidity of thought processes, which may lead to repetitive behavior and inflexibility.

How might universal design of technology assist people with autism? There are two main areas of interest: **communication and education**.

Communication and social interaction are major areas of difficulty for people with autism. Computers, on the other hand, are often motivating, perhaps because they are relatively consistent, predictable and impersonal in their responses. The user is in control.

Computer-mediated communication and virtual environments have been suggested as possible ways of enabling people with autism to communicate more easily with others, by giving the user control over the situation. Some people with autism have difficulties with language and may be helped by graphical representations of information and graphical input to produce text and speech. Again this is supported by providing redundancy in the design.

Computers may also have a role to play in education of children with autism, particularly by enabling them to experience (through virtual environments and games) social situations and learn appropriate responses. This can again provide a secure and consistent environment where the child is in control of his own learning.

Key	*Description*
↑ C (control-C)	Accept the next predicted character
↑ W	Accept the next predicted word
↑ L	Accept the whole predicted line
↑ N	Show the next alternative prediction
↑ P	Show the previous alternative prediction

Reactive keyboard commands

$	mail	↑ N
	cd news	↑ W
	cd news	↑ N
	cd rk/papers/ieee.computer	↑ L
	cd rk/papers/ieee.computer	
$	emacs paper.tex	↑ L
	emacs paper.tex	
$	rm paper.tex.CKP paper.tex.BAK	↑ L
	rm paper.tex.CKP paper.tex.BAK	
$	wc -w paper.tex	↑ L
	wc -w paper.tex	
$	readnews -n comp.sources.unix	↑ N
	mail	↑ W
	mail	↑ N
	mail bdarragh%uncamult.bitnet@ucnet.ucalgary.c	↑ L
	mail bdarragh%uncamult.bitnet@ucnet.ucalgary.c	

User's dialog with the Reactive keyboard.
Only the last line in each group is actually executed.

2.5.3.2 DESIGNING FOR DIFFERENT GROUPS

Older people and children have specific needs when it comes to interactive technology.

i Older people - The requirements of the older population may differ significantly from other population groups, and will vary considerably within the population group. The proportion of disabilities

increases with age: more than half of people over 65 have some kind of disability. Just as in younger people with disabilities, technology can provide support for failing vision, hearing, speech and mobility.

New communication tools, such as email and instant messaging, can provide social interaction in cases where lack of mobility or speech difficulties reduce face-to-face possibilities.

Mobile technologies can be used to provide memory aids where there is age-related memory loss. In spite of the potential benefits of interactive technology to older people, very little attention has been paid to this area until recently.

Researchers are now beginning to address issues such as how technology can best support older people, what the key design issues are, and how older people can be effectively included in the design process, and this area is likely to grow in importance in the future.

ii Children - Like older people, children have distinct needs when it comes to technology, and again, as a population, they are diverse. The requirements of a three year old will be quite different from those of a 12 year old, as will be the methods that can be used to uncover them.

Children are, different from adults, and have their own goals and likes and dislikes. It is therefore important to involve them in the design of interactive systems that are for their use, though this in itself can be challenging as they may not share the designer's vocabulary or be able to verbalize what they think.

Design approaches have therefore been developed specifically to include children actively as members of the design team. As well as their likes and dislikes, children's abilities will also be different from those of adults. Younger children may have difficulty using a keyboard for instance, and may not have well-developed hand–eye coordination. Pen-based interfaces can be a useful alternative input device. Again, universal design principles guide us in designing interfaces that children can use. Interfaces that allow multiple modes of input, including touch or handwriting, may be easier for children than keyboard and mouse. Redundant displays, where information is presented through text, graphics and sound will also enhance their experience.

2.5.3.3 DESIGNING FOR CULTURAL DIFFERENCES

Cultural difference is used synonymously with national differences but this is too simplistic. We can draw out some key factors that we need to consider carefully if we are to practice universal design. These include language, cultural symbols, gestures and use of colour.

Layouts and designs may reflect a language read from left to right and top to bottom, which will be unworkable with languages that do not follow this pattern.

The study of the ***meaning of symbols is known as semantics*** and is a diversion for the student of universal design.

Use of gesture is common in video and animation and care must be taken with differences. Finally, colours are often used in interfaces to reflect 'universal' conventions, such as red for danger and green for go. In fact, red and green mean many different things in different countries. ***As well as danger, red represents life (India), happiness (China) and royalty (France). Green is a symbol of fertility (Egypt) and youth (China) as well as safety (Anglo-American). It is difficult to assume any universal interpretation of colour but the intended significance of particular colours can be supported and clarified through redundancy – providing the same information in another form as well.***

All the Best!

UNIT III MODELS AND THEORIES

Syllabus: Cognitive models –Socio-Organizational issues and stake holder requirements – Communication and collaboration models-Hypertext, Multimedia and WWW.

3.1 COGNITIVE MODELS

- goal and task hierarchies
- linguistic
- physical and device
- architectural

They model aspects of user:

- understanding
- knowledge
- intentions
- processing

Common categorization:

- Competence vs. Performance
  - Computational flavour
  - No clear divide

3.1.1 GOAL AND TASK HIERARCHIES

Many models make use of a model of mental processing in which the user achieves goal by solving sub goals in a divide and-conquer fashion.

Two models are *GOMS* and *CCT*.

Example: sales report

produce report gather data

find book names

do keywords search of names database

… further sub-goals

sift through names and abstracts by hand

… further sub-goals

Produce a report on sales of introductory HCI textbooks. To achieve this goal we divide it into several sub goals, say gathering the data together, producing the tables and histograms, and writing the descriptive material.

Concentrating on the data gathering, we decide to split this into further sub goals:

find the names of all introductory HCI textbooks and then search the book sales database for these books. Similarly, each of the other sub goals is divided up into further sub goals, until some level of detail is found at which we decide to stop.

Goals And Tasks

- goals – intentions
  what you would like to be true
- tasks – actions
  how to achieve it

GOMS– goals are internal

- HTA – actions external
  tasks are abstractions

Issues for Goal Hierarchies:

- ✓ **Granularity**
  Where do we start?
  Where do we stop?
- ✓ **Routine learned behavior, not problem solving**
  The unit task
- ✓ **Conflict**
  More than one way to achieve a goal
- ✓ **Error**

Techniques:

- ✓ Goals, Operators, Methods and Selection (GOMS) – Card, Moran and Newell
- ✓ Cognitive Complexity Theory (CCT) – Kieras & Polson
- ✓ Hierarchical Task Analysis (HTA)

3.1.1.1 GOMS

- **Goals-** what the user wants to achieve
- **Operators-** basic actions user performs
- **Methods-**decomposition of a goal into sub goals/operators
- **Selection-** means of choosing between competing methods

Goals Example:

Goal: CLOSE-WINDOW
[select GOAL: USE-MENU-METHOD
. MOVE-MOUSE-TO-FILE-MENU
. PULL-DOWN-FILE-MENU
. CLICK-OVER-CLOSE-OPTION
GOAL: USE-CTRL-W-METHOD
. PRESS-CONTROL-W-KEYS]

For a particular user:
Rule 1: Select USE-MENU-METHOD unless another rule applies
Rule 2: If the application is GAME, select CTRL-W-METHOD

3.1.1.2 COGNITIVE COMPLEXITY THEORY

Cognitive complexity theory, introduced by Kieras and Polson, begins with the basic premises of goal decomposition from GOMS and enriches the model to provide more predictive power.

CCT has two parallel descriptions:

- **User production rules**
- **Device generalized transition networks**

one of the user's goals and the other of the computer system (called the *device* in CCT).

The description of the user's goals is based on a GOMS-like goal hierarchy, but is expressed primarily using *production rules*.

✓**Production rules are of the form:**
- **if condition then action**

where condition is a statement about the contents of working memory. If the condition is true then the production rule is said to fire. An *action* may consist of one or more elementary actions, which may be either changes to the working memory, or external actions such as keystrokes. The production rule 'program' is written in a LISP-like language.

✓Transition networks covered under dialogue models

Example Ediing with vi

- Production rules are in long-term memory
- Model working memory as attribute-value mapping:

(GOAL perform unit task)
(TEXT task is insert space)
(TEXT task is at 5 23)
(CURSOR 8 7)

- Rules are pattern-matched to working memory,

As an example, we consider an editing task using the UNIX vi text editor. The task is to insert a space where one has been missed out in the text, for instance if we noticed that in the above paragraph we had written 'cognitive complexity theory'. This is a reasonably frequent typing error and so we assume that we have developed good procedures to perform the task.

(SELECT-INSERT-SPACE
IF (AND (TEST-GOAL perform unit task)
(TEST-TEXT task is insert space) (NOT (TEST-GOAL insert space))
(NOT (TEST-NOTE executing insert space))
THEN ((ADD-GOAL insert space)
(ADD-NOTE executing insert space)
(LOOK-TEXT task is at %LINE %COL)))
(INSERT-SPACE-DONE
IF (AND (TEST-GOAL perform unit task)
(TEST-NOTE executing insert space)
(NOT (TEST-GOAL insert space)))
THEN
((DELETE-NOTE executing insert space)

(DELETE-GOAL perform unit task)
(UNBIND %LINE %COL)))
(INSERT-SPACE-1
IF (AND (TEST-GOAL insert space)
(NOT (TEST-GOAL move cursor)) (NOT (TEST-CURSOR %LINE %COL)))
THEN ((ADD-GOAL move cursor to %LINE %COL)))
(INSERT-SPACE-2
IF (AND (TEST-GOAL insert space)
(TEST-CURSOR %LINE %COL))
THEN ((DO-KEYSTROKE 'I')
(DO-KEYSTROKE SPACE)
(DO-KEYSTROKE ESC)
(DELETE-GOAL insert space)))

To see how these rules work, imagine that the user has just seen the typing mistake and thus the contents of working memory (w.m.) are

(GOAL perform unit task)
(TEXT task is insert space)
(TEXT task is at 5, 23)
(CURSOR 8 7)

TEXT refers to the text of the manuscript that is being edited and CURSOR refers to the insertion cursor on the screen. Of course, these items are not actually located in working memory – they are external to the user – but we assume that knowledge from observing them is stored in the user's working memory.

The location (5,23) is the line and column of the typing mistake where the space is required. However, the current cursor position is at line 8 and column 7. This is of course acquired into the user's working memory by looking at the screen. Looking at the four rules above

(SELECT-INSERT-SPACE, INSERT-SPACE-DONE,
INSERT-SPACE-1 and INSERT-SPACE-2), only the first can fire. The condition for SELECT-INSERT-SPACE is:
(AND (TEST-GOAL perform unit task)
true because (GOAL perform unit task) is in w.m.
(TEST-TEXT task is insert space)
true because (TEXT task is insert space) is in w.m.
(NOT (TEST-GOAL insert space))
true because (GOAL insert space) is *not* in w.m.
(NOT (TEST-NOTE executing insert space)))
true because (NOTE executing insert space)
is *not* in w.m.

So, the rule fires and its action is performed. This action has no external effect in terms of keystrokes, but adds extra information to working memory. The (LOOK-TEXT task is at %LINE %COL) looks for a corresponding entry and *binds* LINE and COL to 5 and 23 respectively. These are variables, somewhat as in a normal programming language, which are referred to again in other rules.

The contents of working memory after the firing of rule SELECT-INSERT-SPACE are as follows (note that the order of elements of working memory is arbitrary):

(GOAL perform unit task) (TEXT task is insert space) (TEXT task is at 5 23)
(NOTE executing insert space) (GOAL insert space)
(LINE 5) (COL 23) (CURSOR 8 7)

At this point neither rule SELECT-INSERT-SPACE nor INSERT-SPACE-DONE will fire as the entry (GOAL insert space) will make their conditions false. As LINE is bound to 5 and COL is bound to 23, the condition (TEST-CURSOR %LINE %COL) will be false also, and hence only rule INSERT-SPACE-1 can fire.

After this rule's actions have been performed, the working memory will include the entry (GOAL move cursor to 5 23). The rules for moving the cursor are not included here, but would be quite extensive, moving up/down and right/left depending on the relative positions of the cursor and the target location.

Eventually, assuming the cursor movement is successful, the cursor would be at (5,23) whence rule INSERT-SPACE-2 would be able to fire. This would perform the keystrokes: I, SPACE and ESC, which in vi puts the editor into insert mode, types the space and then leaves insert mode. The action also removes the insert space goal from working memory as this goal has been achieved.

Now the goal has been removed, the second rule INSERT-SPACE-DONE is free to fire, which 'tidies up' working memory. In particular, it 'unbinds' the variables LINE and COL, that is it removes the bindings for them from working memory.

Notice that the rules did not fire in the order they were written. Although they look somewhat like the if–then–else commands one would get in a standard programming language, they behave very differently. The rules are all active and at each moment any rule that has its conditions true may fire. Some rules may never fire; for instance, if the cursor is at the correct position the third rule would not fire. Furthermore, the same rule may fire repeatedly; **for example**, if we were to write out the production rules for moving the cursor, one rule may well be

```
(MOVE-UP
        IF (AND (TEST-GOAL move-up)
                (TEST-CURSOR-BELOW %LINE) )
        THEN (  (DO-KEYSTROKE 'K') ))
```

This rule is to type 'K' (the vi command to move the cursor up one line) while the cursor is below the desired line. It will, of course, be constantly refired until the cursor is at the correct line.

Notice that the keystrokes for actually inserting the space, once you are at the right position, have been ***proceduralized.*** That is, the **user does not go through the subgoals 'enter insert mode', 'type space', 'leave insert mode'**. For a complex insertion, it is quite likely that the user will perform exactly these goals. However, the act of inserting a single space is assumed to be so well rehearsed that it is stored as a single chunk. That is, the rules above represent *expert* knowledge of the vi editor.

The text would read: '**cognitive complexity theory'(CCT)**rules are closely related to GOMS-like goal hierarchies; the rules may be generated from such a hierarchy, or alternatively, we may analyze the production rules to obtain the goal tree:

GOAL: insert space

- GOAL: move cursor – if not at right position
- PRESS-KEY-I

. PRESS-SPACE

- PRESS-ESCAPE

The stacking depth of this goal hierarchy (as described for GOMS) is directly related to the number of (GOAL ...) terms in working memory.

In fact, the ***CCT rules can represent more complex plans than the simple sequential hierarchies of GOMS***. The continuous activity of all production rules makes it possible to represent concurrent plans. **For example**, one could have one set of production rules representing the goal of writing a book, and another set representing the goal of drinking tea. These rules could both be active simultaneously, thus allowing an author to drink tea whilst typing.

Notes on CCT

- Parallel model
- Proceduralisation of actions
- Novice versus expert style rules
- Error behaviour can be represented
- Measures
  - depth of goal structure
  - number of rules
  - comparison with device description

3.1.1.3 PROBLEMS WITH GOAL HIERARCHIES:

The formation of a goal hierarchy is largely a post hoc technique and runs a very real risk of being defined by the computer dialog rather than the user. One way to rectify this is to produce a goal structure based on pre-existing manual procedures and thus obtain a natural hierarchy. GOMS defines its domain to be that of expert use, and thus the goal structures that are important are those which users develop out of their use of the system.

Example: Automated teller machines (ATMs)

GOAL: GET-MONEY
GOAL: USE-ATM
INSERT-CARD

ENTER-PIN
ENTER-AMOUNT
COLLECT-MONEY
outer goal now satisfied goal stack popped >>
COLLECT-CARD – subgoal operators missed

Banks (at least some of them) soon changed the dialog order so that the card was always retrieved before the money was dispensed. A general rule that can be applied to any goal hierarchy from this is that no higher-level goal should be satisfied until all subgoals have been satisfied.

3.1.2 LINGUISTIC NOTATIONS:

- Understanding the user's behaviour and cognitive difficulty based on analysis of language between user and system.
- Similar in emphasis to dialogue models
- Backus–Naur Form (BNF)
- Task–Action Grammar (TAG)

3.1.2.1 BACKUS-NAUR-FORM (BNF)

BNF has been used widely to specify the syntax of computer programming languages, and many system dialogs can be described easily using BNF rules.

For example, imagine a graphics system that has a line-drawing function. To select the function the user must select the 'line' menu option. The line-drawing function allows the user to draw a polyline, that is a sequence of line arcs between points. The user selects the points by clicking the mouse button in the drawing area. The user double clicks to indicate the last point of the polyline.

- ✓ **Basic syntax:**
  - **nonterminal ::= expression**
- ✓ **An expression**
  - **contains terminals and nonterminals**
  - **combined in sequence (+) or as alternatives (|)**

draw-line	**::=**	**select-line + choose-points**
		+ last-point
select-line	**::=**	**position-mouse + CLICK-MOUSE**
choose-points	**::=**	**choose-one**
		| choose-one + choose-points
choose-one	**::=**	**position-mouse + CLICK-MOUSE**
last-point	**::=**	**position-mouse + DOUBLE-CLICK-MOUSE**
position-mouse	**::=**	**empty | MOVE-MOUSE + position-mouse**

The names in the description are of two types:

- ***non-terminals*, shown in lower case,**
- ***terminals*, shown in upper case.**

Terminals represent the lowest level of user behavior, such as pressing a key, clicking a mouse button or moving the mouse. Non-terminals are higher-level abstractions. The non-terminals are defined in terms of other non-terminals and terminals by a definition of the form

Name :: = expression

The '::=' symbol is read as 'is defined as'. Only non-terminals may appear on the left of a definition. The right-hand side is built up using two operators '+' (sequence) and '|' (choice).

For example, the first rule says that the non-terminal draw-line is defined to be select-line followed by choose-points followed by last-point. All of these are non-terminals, that is they do not tell us what the basic user actions are. The second rule says that select-line is defined to be position-mouse (intended to be over the 'line' menu entry) followed by CLICK-MOUSE. This is our first terminal and represents the actual clicking of a mouse.

- ✓ **Terminals**
  - **lowest level of user behaviour**
  - **e.g. CLICK-MOUSE, MOVE-MOUSE**
- ✓ **Nonterminals**
  - **ordering of terminals**
  - **higher level of abstraction**
  - **e.g. select-menu, position-mouse**

- ✓ **For example**, we could have replaced the rules for choose-points and choose-one with the single definition
  choose-points ::= position-mouse + CLICK-MOUSE
  | position-mouse + CLICK-MOUSE + choose-points

Measurements with BNF

- ✓ Number of rules (not so good)
- ✓ Number of + and | operators
- ✓ Complications
  - ➢ same syntax for different semantics
  - ➢ no reflection of user's perception
  - ➢ minimal consistency checking

3.1.2.2 TASK ACTION GRAMMAR:

- ➢ Making consistency more explicit
- ➢ Encoding user's world knowledge
- ➢ Parameterised grammar rules
- ➢ Nonterminals are modified to include additional semantic features

Consistency in tag

- ➢ In BNF, three UNIX commands would be described as:

  To illustrate consistency, we consider the three UNIX commands: cp (for copying files), mv (for moving files) and ln (for linking files).Each of these has two possible forms. They either have two arguments, a source and destination filename, or have any number of source filenames followed by a destination directory:

copy ::=	'cp'	+ filename	+ filename
	| 'cp' + filenames + directory		
move ::=	'mv'	+ filename	+ filename
	| 'mv' + filenames + directory		
link ::=	'ln'	+ filename	+ filename
	| 'ln' +	filenames	+ directory

- ➢ No BNF measure could distinguish between this and a less consistent grammar in which

  link::= ln + filename + filename | ln + directory + filenames
- ➢ consistency of argument order made explicit using a parameter, or semantic feature for file operations
- ➢ Feature Possible values
  Op = copy; move; link
- ➢ Measures based upon BNF could not distinguish between these consistent commands and an inconsistent alternative – say if l took its directory argument first. Task–action grammar was designed to reveal just this sort of consistency. Its description of the UNIX commands would be

file-op[Op] :=	command[Op] + filename + filename	
| command[Op] + filenames + directory		
command[Op=copy]	:=	'cp'
command[Op=move]	:=	'mv'
command[Op=link]	:=	'ln'

Other uses of Tag:

- ➢ User's existing knowledge
- ➢ Congruence between features and commands
- ➢ These are modelled as derived rules

3.1.3 PHYSICAL AND DEVICE MODELS

- The Keystroke Level Model (KLM)
- Buxton's 3-state model
- Based on empirical knowledge of human motor system
- User's task: acquisition then execution.
  these only address execution
- Complementary with goal hierarchies

3.1.3.1 KEYSTROKE LEVEL MODEL (KLM):

Keystroke is aimed at unit tasks within interaction – the execution of simple command sequences, typically taking no more than 20 seconds.

Examples : search and replace feature, or changing the font of a word. It does not extend to complex actions such as producing a diagram. The assumption is that these more complex tasks would be split into subtasks (as in GOMS) before the user attempts to map them into physical actions. The task is split into two phases:

i. **acquisition of the task, when the user builds a mental representation of the task;**

ii. **execution of the task using the system's facilities.**

KLM only gives predictions for the latter stage of activity. **During the acquisition phase, the user will have decided how to accomplish the task using the primitives of the system, and thus, during the execution phase, there is no high-level mental activity** – the user is effectively expert. KLM is related to the GOMS model.

GOMS model decomposes the execution phase into five different physical motor operators, a mental operator and a system response operator:

K Keystroking, actually striking keys, including shifts and other modifier keys.

B Pressing a mouse button.

P Pointing, moving the mouse (or similar device) at a target.

H Homing, switching the hand between mouse and keyboard.

D Drawing lines using the mouse.

M Mentally preparing for a physical action.

- System response which may be ignored if the user does not have to wait for it, as in copy typing.

The execution of a task will involve interleaved occurrences of the various operators. For instance, imagine we are using a mouse-based editor. If we notice a single character error we will point at the error, delete the character and retype it, and then return to our previous typing point. This is decomposed as follows:

1.	Move hand to mouse	**H**[mouse]
2.	Position mouse after bad character	**PB**[LEFT]
3.	Return to keyboard	**H**[keyboard]
4.	Delete character	**MK**[DELETE]
5.	Type correction	**K**[char]
6.	Reposition insertion point	**H**[mouse]**MPB**[LEFT]

Notice that some operators have descriptions added to them, representing which device the hand homes to (**for example**, [mouse]) and what keys are hit (**for example**, LEFT – the left mouse button).

The model predicts the total time taken during the execution phase by adding the component times for each of the above activities. **For example**, if the time taken for one keystroke is t_K, then the total time doing keystrokes is

$T_K = 2t_K$

Table 3.1 Times for various operators in the keystroke-level model

Operator	Remarks	Time (s)
K	Press key	
	good typist (90 wpm)	0.12
	poor typist (40 wpm)	0.28
	non-typist	1.20
B	Mouse button press	
	down or up	0.10
	click	0.20
P	Point with mouse	
	Fitts' law	0.1 $\log_2(D/S$ □ 0.5)
	average movement	1.10
H	Home hands to and from keyboard	0.40
D	Drawing – domain dependent	–
M	Mentally prepare	1.35
R	Response from system – measure	–

wpm = words per minute

screen pointing can be used. Drawing time depends on the number and length of the lines drawn, and is fairly domain specific, but one can easily use empirical data for more general drawing tasks. Finally, homing time and mental preparation time are assumed constant. Typical times are summarized in Table 3.1.

Example for keystroke-level model:

we compare the two methods for iconizing a window.

One used the 'L7' function key, and the other the '**CLOSE**' option from the window's pop-up menu. The latter is obtained by moving to the window's title bar, depressing the left mouse button, dragging the mouse down the pop-up menu to the '**CLOSE**' option, and then releasing the mouse button. We assume that the user's hand is on the mouse to begin with, and hence only the L7-METHOD will require a homing operator. The operators for the two methods are as follows:

L7-METHOD **H**[to keyboard] **MK**[L7 function key]

CLOSE-METHOD **P**[to menu bar] **B**[LEFT down] **MP**[to option] **B**[LEFT up]

The total times are thus

L7-METHOD	=	0.4 + 1.35 + 0.28
	=	2.03 seconds
CLOSE-METHOD	=	1.1 +0.1 + 1.35 + 1.1 + 0.1
	=	3.75 seconds

3.1.3.2 THREE–STATE MODEL

Buxton has developed a simple model of input devices, the ***three-state model***, which captures some of these crucial distinctions. He begins by looking at a mouse. If you move it with no buttons pushed, it normally moves the mouse cursor about. This tracking behavior is termed state 1. Depressing a button over an icon and then moving the mouse will often result in an object being dragged about. This he calls state 2 (see Figure 3.1).

If instead we consider a light pen with a button, it behaves just like a mouse when it is touching the screen. When its button is not depressed, it is in state 1, and when its button is down, state 2. However, the light pen has a third state, when the light pen is not touching the screen. In this state the system cannot track the light pen's position. This is called state 0 (see Figure 3.2).

A touch screen is like the light pen with no button. While the user is not touching the screen, the system cannot track the finger – that is, state 0 again. When the user touches the screen, the system can begin to track – state 1. So a touch screen is a state 0–1 device whereas a mouse is a state 1–2 device. As there is no difference between a state 0–2 and a state 0–1 device, there are only the three possibilities we have seen.

Figure 3.1 Mouse transitions: states 1 and 2

Figure 3.2 Light pen transitions: three states

The only additional complexity is if the device has several buttons, in which case we would have one state for each button: 2_{left}, 2_{middle}, 2_{right}.

Table 3.2 Fitts' law coefficients (after MacKenzie, Sellen and Buxton [221], © 1991 ACM, Inc. Reprinted by permission)

Device	*a* (ms)	*b* (ms/bit)
Pointing (state 1)		
Mouse	−107	223
Trackball	75	300

Dragging (state 2)		
Mouse	135	249
Trackball	−349	688

shown that these differences do exist . Table 3.2 shows the results obtained for a mouse and trackball.

3.1.4 COGNITIVE ARCHITECTURES

All of these cognitive models make assumptions about

- the architecture of the human mind.
  - Long-term/Short-term memory
  - Problem spaces
  - Interacting Cognitive Subsystems
  - ACT

3.1.4.1 THE PROBLEM SPACE MODEL

A problem space consists of a set of states and a set of operations that can be performed on the states. **Behavior in a problem space is a two-step process**.

First, the current operator is chosen based on the current state and then it is applied to the current state to achieve the new state. The problem space must represent rational behavior, and so it must characterize the goal of the agent.

A problem space represents a goal by defining the desired states as a subset of all possible states.

Once the initial state is set, the task within the problem space is to find a sequence of operations that form a path within the state space from the initial state to one of the desired states, whereupon successful termination occurs.

3.1.4.2 INTERACTING COGNITIVE SUBSYSTEMS

Barnard has proposed a very different cognitive architecture, called interacting cognitive subsystems (ICS). ICS provides a model of perception, cognition and action, but unlike other cognitive architectures, it is not intended to produce a description of the user in terms of sequences of actions that he performs. ICS provides a more holistic view of the user as an information-processing machine. The emphasis is on determining how easy particular procedures of action sequences become as they are made more automatic within the user.

ICS is another example of a general cognitive architecture that can be applied to interactive design. One of the features of ICS is its ability to explain how a user proceduralizes action.

Display based Interaction

- Most cognitive models do not deal with user observation and perception
- Some techniques have been extended to handle system output but problems persist (e.g., BNF with sensing terminals, Display-TAG)
- Exploratory interaction versus planning

****

3.2 SOCIO-ORGANIZATIONAL ISSUES AND STAKE HOLDER REQUIREMENTS

3.2.1 INTRODUCTION:

- Organizational issues affect acceptance
  - conflict & power, who benefits, encouraging use
- Stakeholders
  - identify their requirements in organizational context
- Socio-technical models
  - human and technical requirements
- Soft systems methodology
  - broader view of human and organizational issues
- Participatory design
  - includes the user directly in the design process
- Ethnographic methods
  - study users in context, unbiased perspective

3.2.2 ORGANIZATIONAL ISSUES

There are several organizational issues that affect the acceptance of technology by users and that must therefore be considered in system design:

- **systems may not take into account conflict and power relationships**

- **those who benefit may not do the work**
- **not everyone may use systems.**

In addition to generic issues, designers must identify specific stakeholder requirements within their organizational context. Socio-technical models capture both human and technical requirements. Soft systems methodology takes a broader view of human and organizational issues. Participatory design includes the user directly in the design process. Ethnographic methods study users in context, attempting to take an unbiased perspective.

Organizational factors can make or break a system studying the work group is not sufficient

- any system is used within a wider context
- and the crucial people need not be direct users

Before installing a new system must understand:

- who benefits
- who puts in effort
- the balance of power in the organisation … and how it will be affected

3.2.2.1 CONFLICT AND POWER (COOPERATION OR CONFLICT?)

CSCW = computer supported cooperative work

- people and groups have conflicting goals
- systems assuming cooperation will fail!

e.g. computerize stock control stock man looses control of information⇒ subverts the system
identify stakeholders – not just the users

3.2.2.2 CHANGING POWER STRUCTURES (ORGANISATIONAL STRUCTURE)

- Groupware affects organisational structures
- communication structures reflect line management
- email – cross-organisational communication

Disenfranchises lower management
⇒ disaffected staff and 'sabotage'
Technology *can* be used to change management style and power structures

- but need to know that is what we are doing
- and more often an accident !

3.2.2.3 THE INVISIBLE WORKER:

Telecommunications improvements allow:

- neighbourhood workcentres
- home-based tele-working

Many ecological and economic benefits

- reduce car travel
- flexible family commitments

but:

- 'management by presence' doesn't work
- presence increases perceived worth
- problems for promotion

Barriers to tele-working are managerial/social *not* technological

3.2.2.4 WHO BENEFITS? (BENEFITS FOR ALL)

Disproportionate effort
who puts in the effort ≠ who gets the benefit

Example: shared diary:

- effort: secretaries and subordinates, enter data
- benefit: manager easy to arrange meetings
- result: falls into disuse

Solutions:

- coerce use!
- design in symmetry

3.2.2.5 FREE RIDER PROBLEM:

Even where there is no bias toward any particular people, a system may still not function symmetrically, which may be a problem, particularly with shared communication systems. ***One issue is the free rider problem.***

no bias, but still problem
possible to get benefit without doing work
if everyone does it, system falls into disuse
e.g. electronic conferences
– possible to read but never contribute
solutions:

strict protocols (e.g., round robin) increase visibility – rely on social pressure

3.2.2.6 CRITICAL MASS:

Early telephone system:

few subscribers – no one to ring

lots of subscribers – never stops ringing!

Electronic communications similar:
benefit - number of subscribers

early users have negative cost/benefit need critical mass to give net benefits

How to get started?

- look for cliques to form core user base
- design to benefit an initial small user base

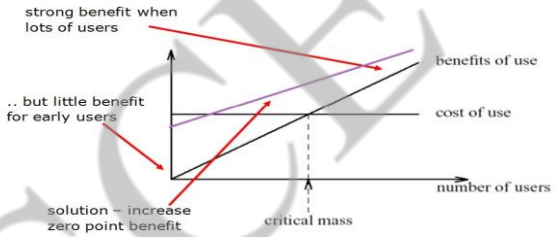


Fig 3.3 Cost/benefit of System use

3.2.2.7 AUTOMATING PROCESSES – WORKFLOW AND BPR

The major task in many organizations is moving pieces of paper around. An order is received by phone and an order form filled in by the sales executive. The order form is passed to accounts who check the credit rating and if all is okay it is passed on to stores who check availability and collect the order together at the picking line. When the order is dispatched, a delivery note is packed with the order and a copy is returned to accounts, who send an invoice to the customer.

Organizations have many such processes, and workflow systems aim to automate much of the process using electronic forms, which are forwarded to the relevant person based on pre-coded rules. Some workflow systems are built using special-purpose groupware, often based on a notation for describing the desired workflow.

A more radical approach to ***organizational processes is found in business process re-engineering (BPR).*** Traditionally, organizations have been structured around functions: sales, accounts, stores, manufacturing. However, the purpose of an organization can be seen in terms of key business processes. The ordering/delivery process described above is a typical and important **example**.

In BPR these processes are recorded and analyzed. Problems in the current process are noted and the whole process may be redesigned in order to make the path of the process more efficient. **For example**, instead of sending an order to the accounts department to approve, a list of customer credit limits could be given to the sales executives. They could then check the credit rating of the customer whilst on the phone and only forward the order to accounts if there are any unusual problems.

Finally, the whole structure of the organization may be modified to reflect and support the key processes more closely. Typically, this involves stripping layers of middle management. BPR as an issue engenders zealots and reactionaries in equal measure.

Evaluating the Benefits:

Assuming we have avoided the pitfalls!

How do we measure our success?

job satisfaction and information flow
– hard to measure

economic benefit
– diffuse throughout organisation

But ..

costs of hardware and software … only too obvious
perhaps we have to rely on hype!

3.2.3 CAPTURING REQUIREMENTS:

- ✓ need to identify requirements within context of use
- ✓ need to take account of
  - ➢ stakeholders
  - ➢ work groups and practices
  - ➢ organisational context
- ✓ many approaches including
  - ➢ socio-technical modelling
  - ➢ soft system modelling
  - ➢ participatory design
  - ➢ contextual inquiry

3.2.3.1 WHO ARE THE STAKE HOLDERS?

- ➢ system will have many stakeholders with potentially conflicting interests
- ➢ A stakeholder, therefore, can be defined as anyone who is affected by the success or failure of the system.

Primary stakeholders are people who actually use the system – the end-users.

Secondary stakeholders are people who do not directly use the system, but receive output from it or provide input to it (**for example**, someone who receives a report produced by the system).

Tertiary stakeholders are people who do not fall into either of the first two categories but who are directly affected by the success or failure of the system (**for example**, a director whose profits increase or decrease depending on the success of the system).

Facilitating stakeholders are people who are involved with the design, development and maintenance of the system.

Example: Classifying stakeholders – an airline booking system

An international airline is considering introducing a new booking system for use by

associated travel agents to sell flights directly to the public. The stakeholders can be classified as follows:

Primary stakeholders: travel agency staff, airline booking staff

Secondary stakeholders: customers, airline management

Tertiary stakeholders: competitors, civil aviation authorities, customers' traveling

companions, airline shareholders

Facilitating stakeholders: design team, IT department staff

- ✓ designers need to meet as many stakeholder needs as possible
  - ➢ usually in conflict so have to prioritise
  - ➢ often priority decreases as move down categories e.g. primary most important

not always e.g. life support machine

3.2.3.2 SOCIO-TECHNICAL MODELS

- ➢ response to *technological determinism*
- ➢ concerned with technical, social, organizational and human aspects of design
- ➢ describes impact of specific technology on organization

The key focus of the socio-technical approach is to describe and document the impact of the introduction of a specific technology into an organization. Methods vary but most attempt to capture certain common elements:

- The problem being addressed: there is a need to understand why the technology is being proposed and

what problem it is intended to solve.

- The stakeholders affected, including primary, secondary, tertiary and facilitating, together with their objectives, goals and tasks.
- The workgroups within the organization, both formal and informal.
- The changes or transformations that will be supported.
- The proposed technology and how it will work within the organization.
- External constraints and influences and performance measures.

Information is gathered using methods such as interviews, observation, focus groups and document analysis.

The methods guide this information-gathering process and help the analyst to make sense of what is discovered. By attempting to understand these issues, socio-technical approaches aim to provide a detailed view of the role technology will play and the requirements of successful deployment.

3.2.3.2.1 CUSTOM METHODOLOGY:

CUSTOM is a socio-technical methodology designed to be practical to use in small organizations.

Six Stage Process - Focus On Stakeholders

- **Describe the organizational context**, including its primary goals, physical characteristics, political and economic background.
- **Identify and describe stakeholders**. All stakeholders are named, categorized (as primary, secondary, tertiary or facilitating) and described with regard to personal issues, their role in the organization and their job.
- **Identify and describe work-groups**. A work-group is any group of people who work together on a task, whether formally constituted or not. Again, work-groups are described in terms of their role within the organization and their characteristics.
- **Identify and describe task–object pairs**. These are the tasks that must be performed, coupled with the objects that are used to perform them or to which they are applied.
- **Identify stakeholder needs**. Stages 2– 4 are described in terms of both the current system and the proposed system. Stakeholder needs are identified by considering the differences between the two.
- **Consolidate and check stakeholder requirements**. Here the stakeholder needs list is checked against the criteria determined at earlier stages.
- **Stages 2 to 4 are described in terms of the current situation** (before the new technology is introduced) and the proposed situation (after deployment). Stakeholders are asked to express their views not only of their current role and position but of their expectations in the light of the changes that will be made. In this way, stakeholder concerns and goals are elaborated. In addition, the impact of the technology on working practices is considered (Stage 3) and the transformations that will be supported by the system specified (Stage 4).
- **The changes from the current position to the proposed position represent the issues that need to be addressed** to ensure successful deployment, and these are made explicit during Stages 5 and 6.

3.2.3.2.2 OPEN SYSTEM TASK ANALYSIS(OSTA)

Eight stage model - focus on task

- ✓ primary task is identified in terms of users' goals
- ✓ Task inputs to the system are identified.
- ✓ The external environment into which the system will be introduced is described, including physical, economic and political aspects.
- ✓ The transformation processes within the system are described in terms of actions performed on or with objects.
- ✓ The social system is analyzed, considering existing work-groups and relationships within and external to the organization.
- ✓ The technical system is described in terms of its configuration and integration with other systems.
- ✓ Performance satisfaction criteria are established, indicating the social and technical requirements of the system.
- ✓ The new technical system is specified.

3.2.3.3 SOFTWARE SYSTEM METHODOLOGY:

✓ There is no assumption of a particular solution: the emphasis is rather on understanding the situation fully. SSM was developed by Checkland

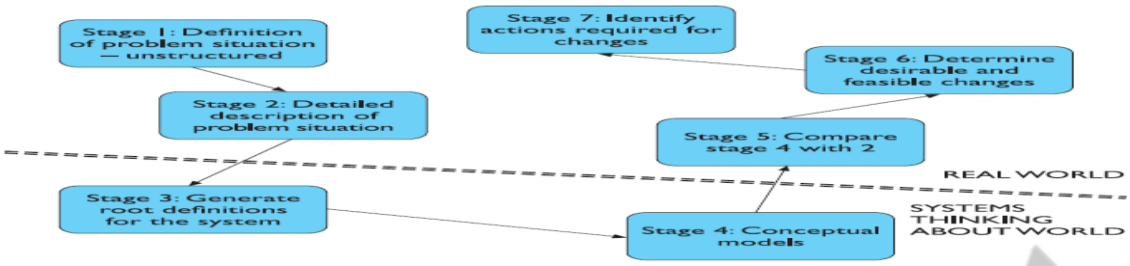


Figure 3.4 The seven stages of soft systems methodology. (Adapted from Checkland)

- ✓ seven stages
- ✓ recognition of problem and initiation of analysis
- ✓ detailed description of problem situation
  - ➢ rich picture
- ✓ generate root definitions of system
  - ➢ CATWOE
- ✓ conceptual model - identifying transformations
- ✓ compare real world to conceptual model
- ✓ identify necessary changes
- ✓ determine actions to effect changes

CATWOE:

- ➢ **Clients:** those who receive output or benefit from the system
- ➢ **Actors:** those who perform activities within the system
- ➢ **Transformations:** the changes that are affected by the system
- ➢ **Weltanschauung:** (from the German) or World View – This is how the system is perceived in a particular root definition
- ➢ **Owner:** those to whom the system belongs, to whom it is answerable and who can authorize changes to it
- ➢ **Environment:** the world in which the system operates and by which it is influenced

Example:

Client: customer

Actor: travel agency staff

Transformation: customer's intention and request to travel transformed into sale of seat on flight and profit for organization

Weltanschauung: profits can be optimized by more efficient sales

Owner: airline management

Environment: Regulations of international civil aviation authorities and national contract leg-islation. Local agency policies worldwide

3.2.3.4 PARTICIPATORY DESIGN:

- ➢ In participatory design:
  workers enter into design context
- ➢ In ethnography (as used for design):
  designer enters into work context
  Both make workers feel valued in design encourage workers to 'own' the products
- ➢ User is an active member of the design team.
  - ➢ Characteristics
    context and work oriented rather than system oriented
    collaborative
    iterative
  - ➢ Methods

brain-storming
storyboarding
workshops
pencil and paper exercises

Participatory design has three specific characteristics.

- It aims to improve the work environment and task by the introduction of the design. This makes design and evaluation context or work oriented rather than system oriented.
- Secondly, it is characterized by collaboration: the user is included in the design team and can contribute to every stage of the design.
- Finally, the approach is iterative: the design is subject to evaluation and revision at each stage.

The participatory design process utilizes a range of methods to help convey information between the user and designer. They include

Brainstorming This involves all participants in the design pooling ideas. This is informal and relatively unstructured although the process tends to involve '***on-the-fly'*** structuring of the ideas as they materialize. All information is recorded without judgment. The session provides a range of ideas from which to work. These can be filtered using other techniques.

Storyboarding Storyboards can be used as a means of describing the user's day-to-day activities as well as the potential designs and the impact they will have.

Workshops These can be used to fill in the missing knowledge of both user and designer and provide a more focussed view of the design. They may involve mutual enquiry in which both parties attempt to understand the context of the design from each other's point of view. The designer questions the user about the work environment in which the design is to be used, and the user can query the designer on the technology and capabilities that may be available. This establishes common ground between the user and designer and sets the foundation for the design that is to be produced. The use of role play can also allow both user and designer to step briefly into one another's shoes.

Pencil and paper exercises These allow designs to be talked through and evaluated with very little commitment in terms of resources. Users can 'walk through' typical tasks using paper mock-ups of the system design. This is intended to show up discrepancies between the user's requirements and the actual design as proposed. Such exercises provide a simple and cheap technique for early assessment of models. PICTIVE is one such approach to paper prototyping, which includes representative stakeholders in a video recorded design session.

Each participant prepares 'homework' focussing on the requirements of the system from their particular perspective, which is then used to introduce and orientate the PICTIVE session. Materials such as sticky notes, highlighters, plastic labels, paper and scissors are used on a shared design surface to produce a low-tech prototype of the proposed system, which is finally tested by the group against the tasks identified.

Such methods are not exclusively used in participatory design, of course, and can be used more widely to promote clearer understanding between designer and stakeholders.

3.2.3.4.1 ETHICS-EFFECTIVE TECHNICAL AND HUMAN IMPLEMENTATION OF COMPUTER BASED SYSTEMS

ETHICS is a method developed by Enid Mumford within the socio-technical tradition, but it is distinct in its view of the role of stakeholders in the process.

- participatory socio-technical approach devised by Mumfor
- system development is about managing change
- non-participants more likely to be dissatisfied
- three levels of participation
  - ✓ consultative, representative, consensus
- design groups including stakeholder representatives make design decisions
- job satisfaction is key to solution

ETHICS methodology, stakeholders are included as participants in the decision-making process. ETHICS considers the process of system development as one of managing change: conflicts will occur and must be negotiated to ensure acceptance and satisfaction with the system. If any party is excluded from the decision-making process then their knowledge and contribution is not utilized and they are more likely to be dissatisfied. However, participation is not always complete. Mumford recognizes three levels of participation:

- **Consultative** – the weakest form of participation where participants are asked for their opinions but are not decision makers.
- **Representative** – a representative of the participant group is involved in the decision-making process.
- **Consensus** – all stakeholders are included in the decision-making process.

The usual practice is that design groups are set up to include representatives from each stakeholder group and these groups make the design decisions, overseen by a steering committee of management and employee representatives.

The design groups then address the following issues and activities:

- ***Make the case for change***. Change for its own sake is inappropriate. If a case can-not be made for changing the current situation then the process ends and the system remains as it is.
- ***Identify system boundaries***. This focusses on the context of the current system and its interactions with other systems, in terms of business, existing techno-logy, and internal and external organizational elements. How will the change impact upon each of these?
- ***Describe the existing system***, including a full analysis of inputs and outputs and the various other activities supported, such as operations, control and coordination.
- ***Define key objectives***, identifying the purpose and function of each area of the organization.
- ***Define key tasks***: what tasks need to be performed to meet these objectives?
- ***Define key information needs*****,** including those identified by analysis of the existing system and those highlighted by definition of key tasks.
- ***Diagnose efficiency needs***, those elements in the system that cause it to under-perform or perform incorrectly. If these are internal they can be redesigned out of the new system; if they are external then the new system must be designed to cope with them.
- ***Diagnose job satisfaction needs***, with a view to increasing job satisfaction where it is low.
- ***Analyze likely future changes***, whether in technology, external constraints (such as legal requirements), economic climate or stakeholder attitudes. This is neces-sary to ensure that the system is flexible enough to cope with change.
- ***Specify and prioritize objectives based on efficiency***, job satisfaction and future needs. All stakeholders should be able to contribute here as it is a critical stage and conflicting priorities need to be negotiated. Objectives are grouped as either primary (must be met) or secondary (desirable to meet).

The final stages of the ETHICS approach focus on the actual design and evaluation of the system.

3.2.3.5 ETHNOGRAPHIC METHODS

Ethnography is based on very detailed recording of the interactions between people and between people and their environment. It has a special focus on social relationships and how they affect the nature of work. The ethnographer does not enter actively into the situation, and does not see things from a particular person's viewpoint. However, ***an aim is to be uncultured, to understand the situation from within its own cultural framework. Culture here means that of the particular work-group or organization, rather than that of society as a whole***. Ethnographers try to take an unbiased and open-ended view of the situation. They report and do not like to speculate, so it is often unclear how well their approach can contribute to the design of new systems.

Ethnography and participatory design

The ethnographic approach differs markedly from the approach of participatory design.

In participatory design the workers come *out* of their work situation, either physically or mentally, and share the design task with the professional designers – effectively the workers become designers. The participatory designer enters into the subjective experience of the workplace.

Ethnographic and other situated approaches take the analyst *into* the workplace, while retaining a level of objectivity. The advantage is that the analyst sees the whole group's perspective, rather than that of involved individuals, but the analyst, however much in tune with the workers, is still 'out there'. On the other hand, involving the workers in the design process in itself increases their motivation and acceptance whether or not the resulting design is 'optimal'.

3.2.3.5.1 CONTEXTUAL INQUIRY:

- Approach developed by Holtzblatt
  - in ethnographic tradition but acknowledges and challenges investigator focus

- The model of contextual inquiry is of the investigator being apprenticed to the user to learn about his work.
- investigation takes place in workplace - detailed interviews, observation, analysis of communications, physical workplace, artefacts
- number of models created:
  - sequence, physical, flow, cultural, artefact
  - models consolidated across users
- output indicates task sequences, artefacts and communication channels needed and physical and cultural constraints

✓ A number of models of the work are developed to capture what is important in the user's work situation:
✓ The sequence model elaborates the steps required to complete a specific task, as well as the triggers that initiate that sequence of steps.
✓ The physical model maps the physical work environment and how it impacts upon work practice, **for example**, an office plan showing where different work activities happen.
✓ The flow model shows the lines of coordination and communication between the user and other participants within and outside the workplace.
✓ The cultural model reflects the influences of work culture and policy and shows the scope of these influences. This may include official or unofficial codes of behavior, common expectations (which may or may not be explicit) and value systems.
✓ The artifact model describes the structure and use of a particular artifact within the work process.

Each of the models above is also consolidated across users to provide a common view of the situation. The result is a representation of the required task sequences, artifacts and communication channels that must be supported in the new system as well as the physical and cultural constraints that must be taken into account.

********** ********** **********

3.3 COMMUNICATION AND COLLABORATION MODELS

3.3.1 CSCW ISSUES AND THEORY:

All computer systems, single-user or multi-user, interact with the work-groups and organizations in which they are used.

✓ We need to understand normal human–human communication:
- face-to-face communication involves eyes, face and body
- conversation can be analyzed to establish its detailed structure.
  - This can then be applied to text-based conversation, which has:

➢ reduced feedback for confirmation
➢ less context to disambiguate utterances
➢ slower pace of interaction but is more easily reviewed.

- Group working is more complex than that of a single person:

✓ it is influenced by the physical environment
✓ experiments are more difficult to control and record
✓ field studies must take into account the social situation.

3.3.2 FACE TO FACE COMMUNICATION:

➢ Most primitive and most subtle form of communication
➢ Often seen as the paradigm for computer mediated communication?

3.3 2.1 TRANSFER EFFECTS AND PERSONAL SPACE

➢ When we come to use computer-mediated forms of communication, we carry forward all our expectations and social norms from face-to-face communication. People are very adaptable and can learn new norms to go with new media (**for example**, the use of 'over' for turn-taking when using a walkie-talkie). However, success with new media is often dependent on whether the participants can use their existing norms. Furthermore, the rules of face-to-face conversation are not conscious, so, when they are broken, we do not always recognize the true problem. We may just have a feeling of unease, or we may feel that our colleague has been rude.

3.3.2.2 EYE CONTACT AND GAZE

➢ to convey interest and establish social presence
➢ Video may spoil direct eye contact.

- but poor quality video better than audio only

3.3.2.3 GESTURES AND BODY LANGUAGE:

- much of our communication is through our bodies
- gesture (and eye gaze) used for deictic reference
- head and shoulders video loses this

So … close focus for eye contact … … or wide focus for body language?

3.3.2.4 BACK CHANNELS, CONFIRMATION AND INTERRUPTION

It is easy to think of conversation as a sequence of utterances: A says something, then B says something, then back to A. This process is called turn-taking and is one of the fundamental structures of conversation. However, each utterance is itself the result of intricate negotiation and interaction. Consider the following transcript:

The nods, grimaces, shrugs of the shoulder and small noises are called back channels. They feed information back from the listener to the speaker at a level below the turn-taking of the conversation. The existence of back channels means that the speaker can afford to be slightly vague, adding details until it is obvious that the listener understands

The back channel responses use a range of sensory channels.

So, as we restrict the forms of communication we lose the back channels. Even video communications tend to use, at most, head and shoulder shots, so we lose some body movement and gestures. On the other hand, a larger view means reduced detail, so we lose information whatever focus we choose.

Audio-only links have to rely on purely verbal back channel responses – the little 'yes'es. Surprisingly, despite the loss of many back channels, people still cope well with these restricted media, and communication is still reasonably effective. However, you may have had the experience, when speaking to someone on the telephone, of suddenly getting the feeling that they have gone away, or the line has gone dead. This is likely to be when you have received insufficient back channel responses.

Transcontinental telephones are especially problematic as they are often only half duplex, that is the sound only goes in one direction at a time. So, while you are speaking, you can hear none of your partner's back channel responses.

Text-based communication, in electronic conferencing, usually has no back channels whatsoever. Any confirmation must be given explicitly in the listener's next utterance. This may confuse an analysis of text-based conversation as the utterances do not correspond simply to utterances in speech.

3.3.2.5 TURN-TAKING

Turn-taking is the process by which the roles of speaker and listener are exchanged. Back channels are often a crucial part of this process.

3.3.3 CONVERSATION

We have looked at the low-level issues of speech and gesture during face-to-face conversation. We now turn to the structure of the conversation itself.

Most analysis of conversation focusses on two-person conversations, but this can range from informal social chat over the telephone to formal courtroom cross-examination. As well as the discipline of ***conversational analysis***, there are other sociological and psychological understandings of conversation.

However, the techniques, as 'bor-rowed' and used to study computer-mediated conversation, would not always find favor with the purist from the discipline from which they originated!

There are three uses for theories of conversation in CSCW.

- First, they can be used to analyze transcripts, for **example** from an electronic conference. This can help us to understand how well the participants are coping with electronic communication.
- Secondly, they can be used as a guide for design decisions – an understanding of normal human–human conversation can help avoid blunders in the design of electronic media.
- Thirdly, and most controversially, they can be used to drive design – structuring the system around the theory.

3.3.3.1 BASIC CONVERSATIONAL STRUCTURE

The most basic conversational structure is turn-taking. On the whole we have an alternating pattern: Alison says something, then Brian, then Alison again. The speech within each turn is called an utterance. There can be exceptions to this turn-taking structure even within two-party conversation.

For example, if there is a gap in the conversation, the same party may pick up the thread, even if she was the last speaker. However, such gaps are normally of short duration, enough to allow turn-claiming if required, but short enough to consider the speech a single utterance.

Often ***we can group the utterances of the conversation into pairs: a question and an answer, a statement and an agreement***. ***The answer or response will normally follow directly after the question or statement and so these are called adjacency pairs.***

We can look at Alison and Brian's conversation above as two adjacency pairs, one after the other. First, Alison asks Brian whether he knows about the film and he responds. Second, she suggests a time to go and he agrees. We can codify this structure as: A-x, B-x, A-y, B-y, where the first letter denotes the speaker (Alison or Brian) and the second letter labels the adjacency pair.

The requirement of adjacency can be broken if the pair is interposed with other pairs for clarification, etc.:

Brian:	Do you want some gateau?
Alison:	Is it very fattening?
Brian:	Yes, very.
Alison:	And lots of chocolate?
Brian:	Masses.
Alison:	I'll have a big slice then.

This conversation can be denoted: B-x, A-y, B-y, A-z, B-z, A-x. Adjacency pair 'x' ('Do you want some gateau?'– 'I'll have a big slice then') is split by two other pairs 'y' and 'z'. One would normally expect the interposed pairs to be relevant to the outer pair, seeking clarification or determining information needed for the response.

Some would say that the adjacency pair is not just a basic structure of conversation but *the* fundamental structure. It is clearly true that we normally respond to the most recent utterance. However, it is less clear whether a simple pairing up of utterances is always possible or useful.

3.3.3.2 CONTEXT

Take a single utterance from a conversation, and it will usually be highly ambiguous if not meaningless: 'the *uh* with the black cat – "The Green whatsit"'. ***Each utterance and each fragment of conversation is heavily dependent on context, which must be used to disambiguate the utterance.***

Two types of context within conversation:

✓ ***external context*** – reference to the environment
e.g., Brian's '*that*' – the thing pointed to

✓ ***internal context*** – reference to previous conversation
e.g., Alison's '*that*' – the last thing spoken of

Referring to Things Deixis:

Often contextual utterances involve indexical: *that, this, he, she, it*
These may be used for internal or external context
Also descriptive phrases may be used:

- ***external:*** '*the corner post is leaning a bit'*
- ***internal:*** '*the post you mentioned'*

A specific form of context dependence is deictic reference. When accompanied by a pointed finger, an expression like 'that post is leaning a bit' is clearly dependent on external context. However, there are very similar uses of internal context:

Brian: (*Points*) That post is leaning a bit.
Alison: That's the one you put in.

Brian's utterance uses external context, whereas Alison's very similar utterance uses internal context. Her 'that' refers to the post Brian was talking about, not the one he is pointing at. To see this, consider the similar fragment:

Brian: The corner post is leaning a bit.
Alison: That's the one you put in.

Real speech, probably more than the written word, is full of ***indexicals,*** words like 'that', 'this', 'he', 'she' and 'it'. Obviously when used in written text, like *this*, words such as *these* make use of purely internal context. In spoken speech any

of the above words can be accompanied by gestures or eyegaze for external context, or simply used, as Alison did, to refer to previous things in the conversation.

Some of the words tend to be more likely to be external ('that', 'this') than others ('he', 'she'), but you can easily think of cases of both forms of use. Furthermore, ***the attachment of pronouns and other indexicals to the things they denote may depend on the semantics of a sentence***: 'Oh no! Eustace has hit Bud. He'll kill him, I know he will.' Does the speaker mean that Eustace will kill Bud, or vice versa? The answer depends on the speaker's knowledge of Eustace and Bud. If Bud is a 22 stone (138 kg) trucker and Eustace has trouble lifting cans of beans then we interpret the sentence one way. If, on the other hand, Eustace has a black belt in karate .

3.3.3.3 TOPICS, FOCUS AND FORMS OF UTTERANCE

Given that conversation is so dependent on context, it is important that the participants have a shared focus. We have addressed this in terms of the external focus – the objects that are visible to the participants – but it is also true of the internal focus of the conversation.

Context resolved relative to current *dialogue focus*

Alison: Oh, look at your roses : : :
Brian: mmm, but I've had trouble with greenfly.
Alison: they're the symbol of the English summer.
Brian: greenfly?
Alison: no roses silly!

Tracing topics is one way to analyse conversation.

- **Alison** begins – *topic* is roses
- Brian shifts topic to greenfly
- **Alison** misses shift in focus … *breakdown*

3.3.3.4 BREAKDOWN AND REPAIR

Breakdown happens at all levels:
topic, indexicals, gesture

Breakdowns are frequent, but

- ✓ redundancy makes detection easy
  (Brian cannot interpret '*they're ... summer'*)
- ✓ people very good at repair
  (Brain and Alison quickly restore shared focus)

3.3.3.5 CONSTRUCTING A SHARED UNDERSTANDING

In a conversation, we know that our partner does not share our knowledge of the world. In addition, we know that our partner will attempt to interpret our utterances. We thus frame our utterances based on this knowledge.

Two guiding principles for our utterances are that they should be relevant and helpful.

- ***To be relevant an utterance should further the current topic***. This is because our partner is expecting an utterance in this context and any sudden shift in our topic focus will make it more difficult for our partner to make sense of the utterance. Such shifts happen in a conversation, but require less ambiguous utterances (as the common ground for that particular utterance is lower).

- ***To be helpful, an utterance should be understandable to the listener and be sufficiently unambiguous given the listener's understanding.*** This requires the speaker to have a model of the listener's understanding and vice versa. So assuming he is being helpful, in saying 'past the pub', Brian implicitly assumes that there is a particular pub, which Alison will recognize as being significant. It is no good the pub being significant to Brian alone; he must know that it will carry its intended significance to Alison.

The ability to build such models is part of our social maturing. One of the key developmental steps for a child is from an egocentric world view, where things are interpreted in relation to the child, to a social one where the child recognizes others' viewpoints. At the age of $2^1/2$, one of the authors' children was interviewed by a linguistics researcher. At one stage the conversation proceeded:

Child: We went to the doctor.
Researcher: Where was the doctor?

Child: Up the steps.

The researcher was clearly (in the context and to an adult) wanting to know whether the doctor was in a hospital or not. The child's answer would have been instantly meaningful to any local parent as the steps to the local doctor were a constant problem for people with prams. However, the child was at that stage unable to phrase the utterance in a way suited to her listener's understanding. At a certain age children assume you know everything they know.

So, we see that conversation is an inherently social activity, based on a constructed shared understanding, and relying on the participants' models of one another. In addition, it depends on continuous interaction to correct misinterpretations and to confirm understanding.

3.3.3.6 SPEECH ACT THEORY:

A specific form of *conversational analysis*
Utterances characterised by what they *do* … … they are *acts*

e.g. '*I'm hungry'*

- ✓ propositional meaning – hunger
- ✓ intended effect – '*get me some food'*

Basic conversational act the illocutionary point:

- ✓ promises, requests, declarations, …

Speech acts need not be spoken

e.g. silence often interpreted as acceptance …

Patterns of acts & Coordinator

- ✓ Generic patterns of acts can be identified
- ✓ Conversation for action (CfA) regarded as central

Basis for groupware tool Coordinator

- ✓ structured email system
- ✓ users must fit within CfA structure
- ✓ not liked by users!

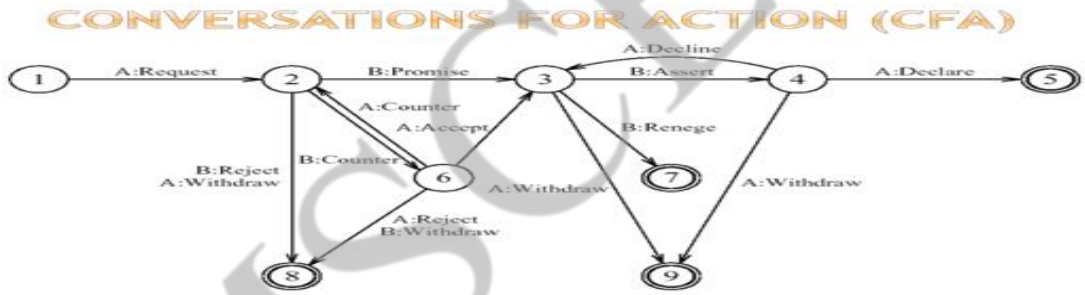


Fig 3.6

CFA in Action:

Simplest route 1–5:

Alison:	have you got the market survey on chocolate mousse?	*request*
Brian:	sure	*promise*
declareBrian:	*there you are*	*assert*
thanks		
Alison:		

More complex routes possible, e.g., 1–2–6–3 …

Alison:	have you got …	*request*
Brian:	I've only got the summary figures	*counter*
Alison:	that'll do	*accept*

Not all speech acts need be spoken! Often a silence or an unspoken action forms a speech act. **For example**, let us imagine that the market survey had not been handy and so Brian answers Alison's request with 'sure, I'll get it later'. Later in the day he finds an electronic copy of the report and then emails it to Alison. His action will be interpreted as *asserting* completion.

If Alison does not respond within a short time, her silence will be read as *declaring* satisfaction and the conversation will be completed.

There are other generic conversation forms as well as CfA. These include:

conversation for clarification usually embedded within a CfA to clarify the required action (different from countering a request);
conversation for possibilities looking toward future actions;
conversation for orientation building up a shared understanding.

In addition, the participants may indulge in meta-conversation, discussing the acts themselves, perhaps questioning the legitimacy of an act: 'I'm hungry' . . . 'well I'm not your skivvy, get your own food'. Also CfA is the most extensive and well developed of the conversational forms. **For example**, the 'creative' conversation for possibilities will have a much less structured form.

The importance of CfA is that actions are central to organizational administration.

3.3.4 TEXT BASED COMMUNICATION:

Most common media for asynchronous groupware exceptions: voice mail, answer-phones. Familiar medium, similar to paper letters but, electronic text may act as speech substitute.

There are four types of textual communication in current groupware:

discrete – directed message as in email. There is no explicit connection between different messages, except in so far as the text of the message refers to a previous one.
linear – participants' messages are added in (usually temporal) order to the end of a single transcript.
non-linear – when messages are linked to one another in a hypertext fashion.
spatial – where messages are arranged on a two-dimensional surface.

In addition, the communication may be connected to other shared computer artefacts,.

3.3.4.1 BACK CHANNELS AND AFFECTIVE STATE(PROBLEMS WITH TEXT):

No facial expression or body language

⇒ weak *back channels*

So, difficult to convey:

affective state – happy, sad, … *illocutionary force* – urgent, important, …

Participants compensate: 'flaming' and smilies

;-) :-(:-) 

In addition to this loss of back channels, the speaker's tone of voice and body language are of course absent. These normally convey the *affective state* of the speaker (happy, sad, angry, humorous) and the *illocutionary force* of the message (an important and urgent demand or a deferential request). Email users have developed explicit tokens of their affective state by the use of 'flaming' and 'smilies', using punctuation and acronyms; for **example**:

:-) – smiling face, happy
:-(– sad face, upset or angry
;-) – winking face, humorous
LOL – laughing out loud.

People tend to use stronger language in email than in face-to-face conversation, **for example** they are more likely to be highly and emotively critical. On the other hand, they are less likely to get emotionally charged themselves. These apparently contradictory findings make sense when you take into account the lack of implicit affective communication. The participants have to put this explicitly into their messages – thus accounting for their stronger language. At the same time, they are emotionally 'distanced' by the text from their conversants and have the conversation spread out over time. In addition, they do not have to express their affective state by *acting* emotionally. Together these factors contribute to a more heated conversation by calmer conversants!

Figure 3.6 Conferencer screen shot showing text transcript and pin-board

3.3.4.2 GROUNDING CONSTRAINTS:

Establishing common ground depends on grounding constraints

contemporality – instant feedthrough simultaneity – speaking together

sequence – utterances ordered

Often weaker in text based communication

e.g., loss of sequence in linear text

Clark and Brennan [71] describe the proper-ties of these channels in terms of *grounding constraints*. These include:
cotemporality – an utterance is heard as soon as it is said (or typed);
simultaneity – the participants can send and receive at the same time;
sequence – the utterances are ordered.

These are all constraints which are weaker in text-based compared with face-to-face interaction.

3.3.4.3 TURN-TAKING

We saw that ***one of the fundamental structures of conversation was turn-taking.*** The last transcript was an **example** of a breakdown in turn-taking. In fact, such breakdowns are quite rare in two-party electronic conversations and are quickly corrected. What is more surprising is that such breakdown so rarely occur during letter writing, which is in some ways similar.

However, when conversing by letter, one has an objective timescale with which to work out whether one's fellow conversant ought to have replied. One therefore does not send a second letter unless the conversant is very remiss in replying to the first missive.

However, in synchronous text-based conversation, the time taken to compose a message (from 30 seconds to several minutes) is far greater than the few seconds which feel 'immediate' on a computer system, but is too short to be able to reason about rationally. The replies always seem a long time coming and hence one is tempted to send a 'follow-on' message.

Despite the occasional breakdown, most observers of two-party text-based inter-action report an overall turn-taking protocol, which exhibits many of the structures of normal conversation including ***adjacency pairs***. However, when we look at three or more participants, turn-taking and adjacency pair structure begin to break down completely.

In a pair of participants, turn-taking is simple; first one person says something, then the other. The only problem is deciding exactly *when* the exchange should happen. With three or more participants, turn-taking is more complex. They must decide *who* should have the next turn. This is resolved by face-to-face groups in a number of ways.

- **First**, the conversation may, for a period, be focused on two of the parties, in which case normal two-party turn-taking holds.
- **Secondly**, the speaker may specifically address another participant as the utterance is finished, either implicitly by body position, or explicitly: 'what do you think Alison?'

- **Finally**, the next speaker may be left open, but the co temporality of the audio channel allows the other participants to negotiate the turn. Basically, whoever speaks first, or most strongly, gets in.

These mechanisms are aided by back channels, as one of the listeners may make it clear that she wants to speak. In this case, either the speaker will explicitly pass the turn (the second option above), or at least the other listeners are expecting her to speak. In addition, the movement between effective two-party conversation (the first option) and open discussion will be mediated by back channel messages from the other participants.

In an unstructured text-based conversation the third option is not available, nor, of course, are the back channels. Paired conversation is quite common and the second option, explicitly naming the next speaker, is possible. However, this naming is not particularly natural unless a direct question is being asked. In both options, the absence of back channels makes it difficult for another listener to interrupt the conversation.

Some systems use more structured mechanisms to get round these problems, perhaps having a ***round-robin protocol*** (each participant 'speaks' in turn) or ***having a queue of turn-requests***. Whether the strictures of such mechanisms are worse than the problems of occasional breakdown depends very much on the context and is a matter of opinion.

Loss of Sequence:

Network delays or coarse granularity ⇒ overlap

1. **Bethan:**how many should be in the group?
2. **Rowena:** maybe this could be one of the 4 strongest reasons
3. **Rowena:** please clarify what you mean
4. **Bethan:**I agree
5. **Rowena:** hang on
6. **Rowena:** Bethan what did you mean?

Message pairs 1&2 and 3&4 composed simultaneously
– lack of *common experience*

Rowena:	2 1 3 4 5 6
Bethan:	1 2 4 3 5 6

N.B. breakdown of turn-taking due to poor back channels

3.3.4.4 CONTEXT AND DEIXIS

Utterances are highly ambiguous and are only meaningful with respect to ***external context***, the state of the world, and ***internal context***, the state of the conversation. ***Both of these are problems in text-based communication.***

The very fact that the participants are not co-present makes it more difficult to use external context to disambiguate utterances. This is why many groupware systems strive so hard to make the participants' views the same; that is, to maintain ***WYSIWIS ('what you see is what I see').***

Whatever the means of direct communication, remote participants have difficulty in using deictic reference. They cannot simply say 'that one', but must usually describe the referrant: 'the big circle in the corner'. If their displays are not WYSIWIS then they must also ensure that their colleague's display includes the object referred to and that the description is unambiguous.

Asynchronous participants have even more problems with deixis as there is no opportunity for their colleagues to clarify a reference (without extremely lengthy exchanges). Furthermore, the objects referred to by a message may have changed by the time someone comes to read it! Similarly, group pointers are not really an option, but one can use methods of linking the conversation to its context, either by embedding it within the objects as annotations or by having hypertext links between the conversation and the object.

The trouble does not end with external context; there are also problems with deictic reference to internal context. In speech, the context is intimately connected to linear sequence and adjacency. As we have seen, even in linear text transcripts, overlap breaks the strict sequentiality of the conversation, and thus causes problems with indexicals and with context in general.

- Alison: Brian's got some lovely roses.
- Brian:I'm afraid they're covered in greenfly.
- Clarise: I've seen them, they're beautiful.

Brian and Clarise both reply to Alison's message at the same time. However, in the transcript, where Clarise says 'they' this appears, at first, to refer to the greenfly. Brian is expecting a consoling reply like 'I've seen them. Have you tried companion planting?' Of course, the breakdown quickly becomes apparent in this case. The problem is not so much that people cannot recover from such breakdowns, as in the extra burden the recovery puts on the participants. If these messages are being sent, say, between continents, network delays and time differences may limit exchanges to once a day. Even one or two messages recovering from breakdown are then a major disaster.

Most email systems and some bulletin boards lack any implied sequentiality and thus any context to the messages. The users (ever inventive) get round this by including copies of previous messages in their replies. This is only partially effective.

Hypertext-based systems avoid the implied sequentiality of a linear transcript. In the above **example**, both Brian and Clarise replied to Alison's message at the same time. In a hypertext these would form parallel conversations. This is shown in Figure 3.7, where in addition Clarise has sent a second message offering advice on Brian's greenfly. The use of 'they' in Clarise's message (3) is now perfectly clear.

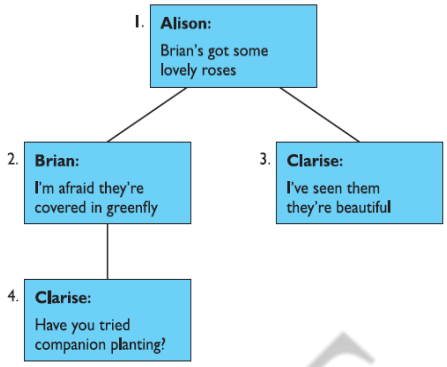


Figure 3.7 Hypertext conversation structure

3.3.4.5 PACE AND GRANULARITY:

Pace of conversation – the rate of turn taking

face-to-face	–	every few seconds
telephone	–	half a minute
email	–	hours or days

face-to-face conversation is highly interactive

- initial utterance is vague
- feedback gives cues for comprehension

lower pace ⇒ less feedback ⇒ less interactive

Coping Strategies:

People are very clever!

they create ***coping strategies*** when things are difficult

Coping strategies for slow communication attempt to increase granularity:

eagerness – looking ahead in the conversation game

Brian: Like a cup of tea? Milk or lemon?

multiplexing – several topics in one utterance

Alison: No thanks. I love your roses.

Conversation Game:

Conversation is like a game

Linear text follows one path through it

Participants choose the path by their utterances

Hypertext can follow several paths at once

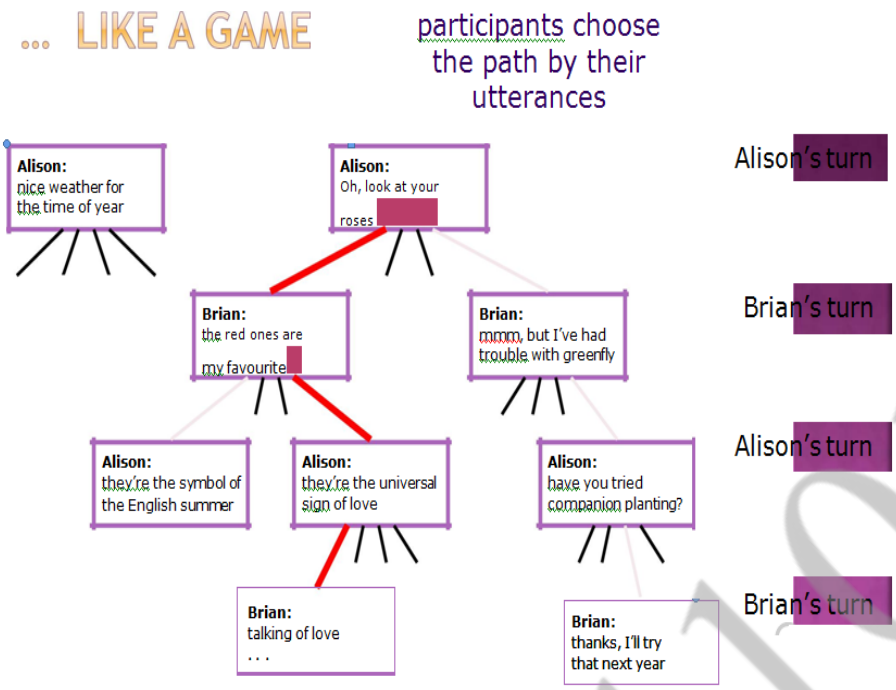


Figure 3.8 The conversation 'game'

3.3.4.6 LINEAR TEXT VS. HYPERTEXT

Considerations of potential overlap suggest that hypertext-based communications may be better suited as a text-based communication medium. Similarly, the problems of pace may be partially solved in a hypertext. Multiplexed messages can be represented as updates to several parts of the hypertext, thus reducing the likelihood of breakdown and lost topics. In addition, if the messages themselves can be mini-hypertexts, then eager messages listing several possible courses of action can be explicitly represented by the message.

Hypertext has its disadvantages. Even static hypertexts, which have been carefully crafted by their authors, can be difficult to navigate. A hypertext that is created 'on the fly' is unlikely to be comprehensible to any but those involved in its creation.

For the asynchronous reader trying to catch up with a conversation, a linear transcript is clearly easier, but it is precisely in more asynchronous settings where overlap in linear text is most likely to cause confusion.

We can see that there is no best solution, with possibly the best course in many situations being linear transcripts arranged by topic, with some automatically generated indication of overlap.

3.3.5 GROUP WORKING

So far we have been principally looking at the properties of direct communication, and largely two-party conversations. Group behavior is more complex still as we have to take into account the dynamic social relationships during group working. We will begin by looking at several factors which affect group working, and then discuss the problems of studying group working. This section deals with groups that are actively working together, rather than the organizational issues considered in the previous chapter, which are primarily concerned with the long-term structures within which people work.

3.3.5.1 GROUP DYNAMICS:

Work groups constantly change:
– in structure – in size
Several groupware systems have explicit rôles
- But rôles depend on context and time
  e.g., M.D. down mine under authority of foreman
- and may not reflect duties
  e.g., subject of biography, author, but now writer

Social structure may change: democratic, autocratic and group may fragment into sub-groups
Groupware systems rarely achieve this flexibility
Groups also change in composition

⇒ new members must be able to `catch up'

3.3.5.2 PHYSICAL LAYOUT(PHYSICAL ENVIRONMENT):

Face-to- face working radically affected by layout of workplace

e.g. meeting rooms:

- recessed terminals reduce visual impact
- inward facing to encourage eye contact
- different power positions

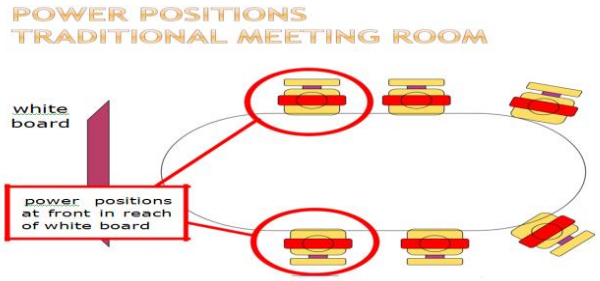


Figure 3.9 Power positions traditional meeting room

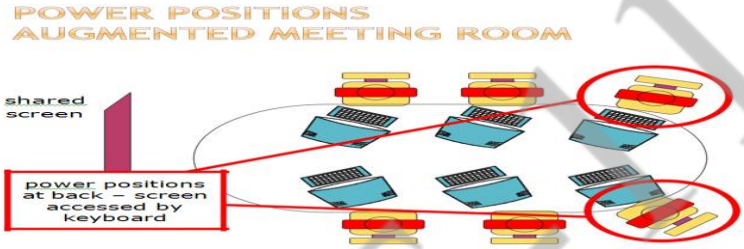


Figure 3.10 Power positions augmented meeting room

3.3.5.3 DISTRIBUTED COGNITION:

Traditional cognitive psychology in *the head*

Distributed cognition suggests look to *the world*

Thinking takes place in interaction

- with other people
- with the physical environment

Implications for group work:

- importance of mediating representations
- group knowledge greater than sum of parts
- design focus on external representation

****

3.4 HYPERTEXT, MULTIMEDIA AND WWW:

Hypertext allows documents to be linked in a non-linear fashion.

- Multimedia incorporates different media: sound, images, video.
- The world wide web is a global hypermedia system.
- Animation and video can show information that is difficult to convey statically.
- Applications of hypermedia include online help, education and e-commerce.
- Design for the world wide web illustrates general hypermedia design, but also has its own special problems.
- Dynamic web content can be used for simple online demonstration or for complete web-based business applications.

Text-imposes strict linear progression on the reader

Hypertext - not just linear

- non-linear structure
  - blocks of text (pages)
  - links between pages create a mesh or network

users follow their own path through information

3.4.1 UNDERSTANDING HYPERTEXT

3.4.1.1 HYPERTEXT DEFINITION – TEXT, HYPERTEXT AND MULTIMEDIA

Text:

All the traditional texts share a common linear nature. This linearity is partly because of the nature of the media used – papyrus scroll, painted frieze or paper book – but perhaps more significantly because we are creatures in linear time. We are natural story-tellers and natural story-hearers.

During some forms of exploratory learning the learners may want to follow their own paths through material: each one delving into details in different parts. Experts in a subject, too, may well want to remind themselves of some particular issue or fact.

Hypertext attempts to get around these limitations of text by structuring it into a mesh rather than a line. This allows a number of different pages to be accessed from the current one, and, if the hypertext is well designed, the user should find it easier to follow his own particular idea through the mesh rather than being forced down one route.

Typically, ***hypertext systems incorporate diagrams, photographs and other media as well as text. Such systems are often known as multimedia or hypermedia systems***, although the three terms are often used interchangeably.

A hypertext system comprises a number of pages and a set of links that are used to connect pages together. The links can join any page to any other page, and there can be more than one link per page. Thus a hypertext document does not simply start a linear progression and follow it to an end, but goes in lots of different directions, some of which terminate, while others link back into different parts of the document .

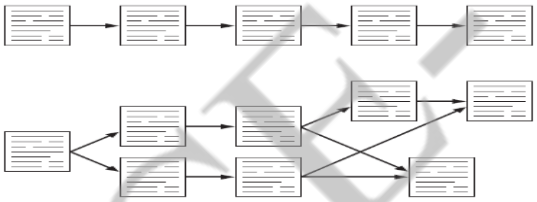


Fig 3.11 Typical Structure of linear text and hypertext

3.4.1.2 RICH CONTENT

3.4.1.2.1 ANIMATION

- adding motion to images
  - for things that change in time
    - digital faces – seconds tick past or warp into the next
    - analogue face – hands sweep around the clock face
    - live displays: e.g. current system load
  - for showing status and progress
    - flashing carat at text entry location
    - busy cursors (hour-glass, clock, spinning disc)
  - for data visualisation
    - abrupt and smooth changes in multi-dimensional data visualised using animated, coloured surfaces
    - complex molecules and their interactions more easily understood when they are rotated and viewed on the screen

3.4.1.2.2 VIDEO AND AUDIO

- now easy to author
  - tools to edit sound & video and burn CDs & DVDs
- easy to embed in web pages
  - standard formats (QuickTime, MP3)
- **still big … but getting manageable**
  - **memory OK … hand held MP3 players, TiVo etc.**

– but download time needs care – tell users how big!

- very linear

hard **to add 'links' often best as small clips or background**

3.4.1.2.3 COMPUTATION, INTELLIGENCE AND INTERACTION

More interactive hypermedia may contain embedded games or applications. **For example**, Figure 3.12 shows a puzzle from the website of one of the authors (Alan), a sort of 2D Rubik's cube that you can play online. Hypermedia running on the user's own computer may interact closely with other applications; **for example**, on a

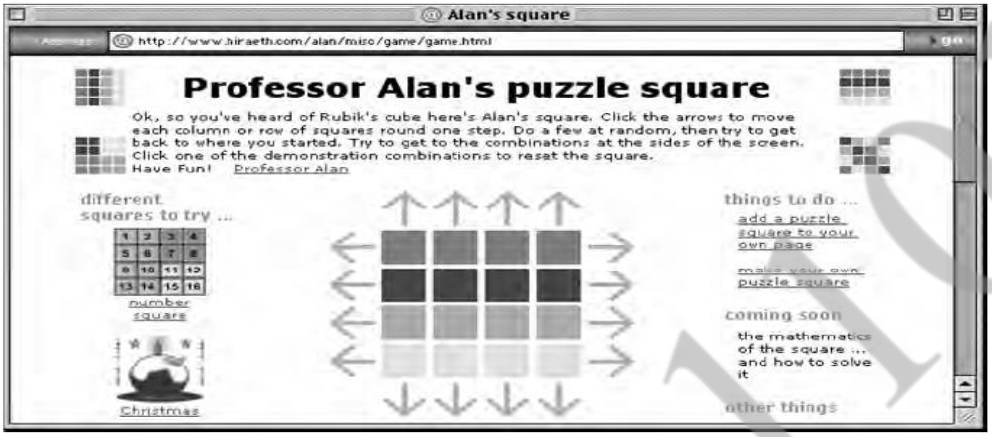


Figure 3.12 Interacting with hypertext – Professor Alan's puzzle square. Screen shot frame reprinted by permission from Microsoft Corporation

Macintosh HyperCard stacks can control applications using AppleEvents, or on a Windows platform hypermedia can include ActiveX components.

*Whilst the 'text' in hypertext suggests passive content under the user's control, some hypermedia may contain more intelligent components or agents actively working to shape the experience for the user. **For example**, some educational hypertexts adapt their content depending on a model of the learner. Similarly e-commerce sites may suggest additional products to buy depending on your previous purchasing and browsing behavior.*

3.4.1.3 DELIVERY TECHNOLOGY

3.4.1.3.1 ON THE COMPUTER

Some hypertexts are downloaded or installed permanently on a computer. However, with media-rich hypertexts containing substantial graphics, video and audio clips it may be impractical to store everything on hard disk. Also, for copy-right protection, some systems will deliberately not allow themselves to be copied from their original distribution media.

Many hypermedia systems are supplied on CD-ROM. This has the advantage of reasonably large capacity (650–700 Mbytes), but access is slower than with installed systems. For highly dynamic material, such as educational media, a special player is installed; alternatively, material such as software documentation may use a standard format such as web pages.

DVD delivered material is interesting as it is not text enriched with video, but instead a movie that has been 'made interactive'.

3.4.1.3.2 ON THE WEB

The ***world wide web*** is the best-known multimedia hypertext system of all. ***The world wide web offers a rich environment for the presentation of information.*** Documents can be constructed that are very different from paper versions; basic text can be augmented through the use of hypertext links to other documents, while graphics can easily be incorporated as pictures, photographs, icons, page dividing bars, or backgrounds***. Pages can also have hypertext links embedded into different regions, which take the user to a different page or graphic if they are clicked on; these are known as active image maps***. These features allow web pages to become interactive, acting as the interface to the information as well as its

holder. Dynamic material in the form of movies and sounds is also available to the designer; all these features push web page design well away from the conventional paper-based kind.

Designing web pages is a developing art, and should be viewed in much the same way as designing any other interactive system. Good pages have been developed with the reader as the focus, and act as effective interaction tools or presentation tools to allow the user to obtain the information he is looking for most effectively.

The web allows the user to browse documents and follow links transparently, with the underlying system taking care of the details of fetching the data from different parts of the world. Theoretically, as far as the user is concerned, any page can be reached as easily as any other; geographical location ceases to become important, whereas linking by content is crucial. The ability for anyone to publish information on the web is one factor in its success as a multimedia system, but the fact that any-one can create a page and, by linking it to others already in existence, immediately integrate their opinions seamlessly into the information space is another.

3.4.1.3.3 ON THE MOVE

Mobile phones, PDAs (personal digital assistants), and notebook computers have all increased the demand to have hypermedia available on the move.

Notebook computers can use just the same mechanisms as desktop computers, using CD-ROM or DVD for standalone material, or connecting to the web through wireless access points or through modems linked to mobile phone networks. However, the fact that the computer is mobile means that location can be used as a key into context-aware hypermedia showing different content depending on location.

PDA access poses different problems. They often have standard web browsers, but of course on a substantially smaller screen. This may mean designing special pages or being especially careful to design ones that resize well. Because PDAs are often not network connected there are also systems to allow access to information when disconnected.

By nature, mobile phones are (nearly) always connected to a network. However, memory and screen size are even more constrained. Some phones allow download-able applets so that small dynamic applications can be used. More web-like content can be accessed via WAP (wireless application protocol), which, like HTTP (hyper-text transfer protocol) for the web, gives access to remote servers. ***WAP content can be produced as static or dynamic content using a mark-up language called WML, which is a simplified version of HTML (hypertext markup language).*** This allows hyperlinks like the web and even simple images, but due to the small screen size most pages consist mainly of small amounts of information or simple lists of links.

3.4.1.4 APPLICATION AREAS

3.4.1.4.1 RAPID PROTOTYPING

HyperCard on Macintosh computers has been very influential as a basis for experimental hypertext systems. ***HyperCard uses the metaphor of a card index***, around which the user can navigate. Each card can hold text, diagrams, photographs, bitmaps and so on, and hot-spots on the cards allow movement between cards. Cards may also contain forward and backward buttons and a home icon, to allow the user to move sequentially and start from scratch respectively. ***HyperCard can be used for a range of applications including information management and teaching.***

However, HyperCard's simple scripting language and easy to produce graphical interfaces meant it was also used extensively as a ***rapid prototyping tool*** for generating interactive systems. In fact, HyperCard stacks for both single-user and networked applications are available from the book website.

For similar reasons, other hypermedia tools such as ***Macromedia Flash and Director are often used to produce dynamic interface mock-ups or even fully functioning systems***. The web, too, is used like this, both to deliver applications and also as a way of mocking up an application interface as a series of storyboard web pages.

3.4.1.4.2 HELP AND DOCUMENTATION

- allows hierarchical contents, keyword search or browsing
- just in time learning
  - what you want when you want it
  - (e.g. technical manual for a photocopier)
    - technical words linked to their definition in a glossary

links between similar photocopiers

3.4.1.4.3 EDUCATION AND E-LEARNING

- animation and graphics allow students to see things happen
- sound adds atmosphere and means diagrams can be looked at while hearing explanation
- non-linear structure allows students to explore at their own pace
- e-learning
  - ✓ letting education out of the classroom!!

  e.g. eClass

3.4.1.4.4 COLLABORATION AND COMMUNITY

Although strictly not hypertext, the web has become a central platform for collaborative applications and community. These use the hypertext structure of the web to structure and access shared resources and message areas. **For example** Yahoo! Groups (groups.yahoo.com) allows mailing lists, shared images (such as family photo albums), web archives of the mailing list and chat, all accessed through a web interface.

Establishing a sense of community can be very important on websites as it is one way to ensure loyalty and get visitors to return. This may involve explicit community features such as chat areas, or may simply be a matter of using a design, language and image that suggests a site which is open and listening to 'readers'.

3.4.1.4.3 E-COMMERCE

For some companies the web is simply another sales opportunity. Many readers will have used online stores such as Amazon or bought from an auction site such as eBay. Hypertext's use of hierarchies, links, images and so on, makes it ideal for displaying certain kinds of product. Actual buying and selling requires not only security at the level of the networks, websites, etc., but also trust. When you walk into a shop you can see the person you are dealing with, and the fact that it is physically there today gives you confidence that it will be there tomorrow if anything goes wrong. How to build and ensure this trust is an active area in HCI research.

3.4.2 FINDING THINGS

3.4.2.1 LOST IN HYPERSPACE

Although the ***non-linear structure of hypertext is very powerful***, it can also be confusing. It is easy to lose track of where you are, a problem that has been called 'lost in hyperspace'.

There are two elements to this feeling of 'lostness'.

The first is cognitive and related to content. In a ***linear text***, when a topic is being described, the writer knows what the reader has already seen. In a ***hypertext***, the reader can browse the text in any order. Each page or node has to be written virtually independently, but, of course, in reality it cannot be written entirely without any assumption of prior knowledge. As the reader encounters fragmentary information, it cannot be properly integrated, leading to confusion about the topic.

The second is related to navigation and structure. Although the hypertext may have a hierarchical or other structure, the user may navigate by hyperlinks that move across this main structure. It is easy to lose track of where you are and where you have been.

The solution to the former issue is to design the information better. The solution to the latter is to give users better ways of understanding where they are and of navigating in the hypertext. To say 'the solution' is disingenuous – there is no simple 'solution'. If we want to provide information that allows complex, unplanned, non-linear access, there will probably always be problems. However, good design can help!

3.4.2.2 DESIGNING STRUCTURE

In a paper format one is stuck with a single structure, which can lead to tensions: for **example**, the fact that in this book structural design is discussed in several places. As another **example**, imagine a car mechanic using a manual. She might want to use the classical breakdown into transmission, fuel system, etc., while fault finding, but if she were dismantling the engine it might be more useful to look at the car components in terms of physical location.

If multiple structures are used, you have to consider what to do about the common material.

For example, if we examine a car hypermedia text under 'engine compartment' and get to the fuel pump, this would also appear in the functional view under 'fuel system'. Such common elements may be replicated. This has the

advantage that the material can be presented in ways that make sense given their context, but it can also lead to inconsistencies.

In all cases it is important that the structure and the naming of parts is meaningful for the user. In a more detailed and theoretical approach to the 'knowing where you are going' principle, Pirolli and others have developed information foraging theory. This uses an analogy with foraging animals searching for patches of food and trying to make decisions about when to move to a different area or stay with the food available, and, if they move, where to go. This is likened to the way an information seeker browses, making decisions about whether to stick with the information available or spend time looking for more, and, if more is needed, deciding where to seek it.

3.4.2.3 MAKING NAVIGATION EASIER

No matter how well designed the site structure is, there will still be problems: because the user does not understand the structure; or because the user has individual needs that the designer has not foreseen; or because even a good structure is not perfect. However, there are various things that can make it easier for users.

One solution is to provide a map of the hypertext document, identifying the current position of the reader within it. Links to home or end points can then be identified and the user is less likely to get lost. This may be a separate part of the hypertext; **for example**, some websites have a site map link leading to a special page, and many help systems have a table of contents view. Alternatively, the site map can be woven into the layout of the document; f**or example**, some sites have an outline-style sidebar listing the main sections and drilling down to the current location. This acts both as an indication of where you are in the site and as a constant reminder of the overall site structure.

Another type of hypertext takes the form of 'levels of access' to a document. Different levels of access privilege 'see' different amounts of information.

A document structured in this way may provide ***one level of access that gives only a brief overview of the topic***. ***The next level of access presents a fuller description of the system***, while the ***next level may also include information regarding the precise meaning of technical terms used in the system***. The ***final level of access may add historical information*** and such like. The user can choose at which level he wants to read the document, cutting out irrelevant information while obtaining all the necessary details. Such a document tends to be linear in nature, which makes navigating and printing it easier, but removes the user's choice in structuring his progress through it.

Once information has been retrieved, a paper version is often needed. Printing a document requires the pages to be in a particular order, but hypertext does not support the concept of one single order. This is against the ethos of hypertext, which intends the user to structure the information in the way that suits him best. It can therefore be difficult to get a hard copy of the information that is required.

Although there is no simple way to linearize a hypertext, one can at least make it possible to print individual linear parts, whether single pages or groups of linearly linked pages. In general, you should not rely on the print facility of a browser as this is printing a page designed for on-screen viewing. You may notice websites offering printer friendly pages. These may be in a different format such as PDF, or may simply be web pages without side bar navigation aids, etc.

3.4.2.4 HISTORY, BOOKMARKS AND EXTERNAL LINKS

Hypertext viewers and web browsers usually have some sort of history mechanism to allow you to see where you have been, and a more stack-based system using the 'back' button that allows you to backtrack through previously visited pages. ***The back button may be used where a user has followed a hyperlink and then decided it was to the wrong place, or alternatively, when browsing back and forth from a central page that contains lots of links. The latter is called hub and spoke browsing***. In fact, in studies of web browsing the back button accounted for 30% of all navigation actions. Other studies have shown frequent revisiting of pages during a single browsing session.

Although the back button is used extensively, it is used relatively little to go back more than one step. For error correction this makes sense, but for general revisiting one might think that moving back several steps would be common. Possibly, one reason for this is confusion about the meaning of the back button; indeed a formal comparison of back and history mechanisms in four different hypertext and web browsers found that the operation of back and history were subtly different in each.

For longer-term revisiting, browsers typically support some form of bookmarking of favorite pages. Both this and, ***on the web, external links from other people's sites mean that users may enter your hypertext at locations other than the top level or home page. On the web this is called deep linking***. Many websites rely on the user remembering where they have come from to make sense of a page. If a page does not adequately show where it fits, then a user coming to it from outside may have no idea what site it is from, or why they are reading the material. Furthermore, ***if the original site depended on the***

user pressing 'back' to return to higher levels of the site hierarchy it may be impossible for a visitor to find the rest of your site at all!

All pages should therefore make clear where they belong and have links into the full site structure.

Framed websites are particularly difficult. The material you want to bookmark or link to may be one of the frame content pages, but the URL you can see or bookmark is that of the overall frameset. This may either discourage linking to the site or, if circumvented, it may mean linking directly to the content of one of the frames, which is then very likely to lack sufficient context, being designed to be seen within the frameset.

Search engines, too, may generate links to individual frames in a frame-set. Many web style guides heavily discourage the use of frames for this reason. If the site is designed using a development tool that supports page templates, or is being dynamically generated, there is rarely any need for frames as most of the effects can be obtained using other page formatting.

Very occasionally you may want to discourage deep linking; **for example**, if the framed page is more of an interactive application or you know the inner structure is unlikely to stay constant. In such cases you can include a small piece of script in the inner framed pages that makes them redirect to the outer frame if they are ever opened 'bare' in a window. This means that if a site or a search engine does link into the inner frames, following the link takes the user to the full, framed site.

3.4.2.5 INDICES, DIRECTORIES AND SEARCH

A hierarchical table of contents structure, many help systems, hypertexts, and for that matter paper books, have some kind of index. Note that an ***index is not a complete list of all words in a document***. If this were the case then the index for this book would be as big as the rest of the book! The words in an index are chosen because they are significant key phrases or words with a domain meaning, and not every occurrence of a word is indexed, only those deemed in some way important. ***The main difference between an electronic index and a paper one is that with the paper index you have to physically look up the page after finding the word in the index, whereas in an electronic index the links are 'live' so you can simply click to the content.***

On the web an index would be very big (!); however, directory services such as Yahoo! (www.yahoo.com) or the Open Directory Project (ODP) (www.dmoz.org) can be seen as a form of index. The main difference is that while an index is simply an alphabetical list of keywords, web directories give a hierarchical categorization to sites. The categorization is done either by self-submission, or, in the case of quality directories such as ODP or Yahoo!, by experts in the relevant field.

For exhaustive searching by keywords, some kind of automated search is required. ***In the case of a standalone hypertext, the viewer application may do this either by using a pre-computed electronic index of all word occurrences used by the hypertext, or by scanning it on demand.*** The latter will take longer for each search, but may be more effective if the hypertext is not too big or the material is rapidly changing. Where the hypertext is generated from a database, the search may be performed on the underlying data rather than the generated pages.

In the case of the web the content is dynamically changing, but it would be impossible to scan the whole web every time you wanted to find anything! Search engines such as Google or AltaVista use web crawling. Starting from an initial collection of pages they look for all links from these pages. These links are followed and the new pages reached are scanned, and so on. As pages are visited, an index is built of which words occur in which pages.

The search engines do not keep a copy of every page visited, but may just keep the title and the first hundred or so words on each page. Even the index is vast and so the most common words are usually not indexed; these are called stop words. When you do a search, the search engine uses the index and the summary information to construct the results page with links to the actual pages. Because it is using the page summaries and not looking at the pages themselves, it is possible that a page may have been removed or changed since the index was constructed.

The web is enormous and so the number of pages containing a given word is enormous. Web search engines allow you to search for several words at once or for exact phrases, or, with Boolean searches, to specify using logical and/or options what is required: **for example**, 'engine AND NOT car'.

Even when looking for multiple words or Boolean queries, the number of results may be in the tens or hundreds of thousands. So search engines need some way to rank pages. Some use simple, content-based measures such as the number of times the requested words occur, whether they occur in the title or body, whether they occur near the beginning or end of the page. Some search engines keep track of how many times users click through for specific pages and so can build up a model of popularity. In addition, some search engines sell the right to be top, based on keywords, or have a special advertisers or sponsors links section.

Some specialist searches, **for example** for video, books, etc., and even ***some more general-purpose search engines, allow you to rate pages you have visited***. Ratings are then used to rank the more popular pages for future visitors. These are called **recommender systems**. As well as explicit recommendations, e-commerce sites often track your browsing

on the site and use this to build a profile of which users are similar to you. The books or goods they purchased may then be suggested to you.

When designing web pages, it is possible to make them more 'search engine friendly' by adding 'META' tags in the head section of the web page, in particular keywords and description, as well as a relevant 'TITLE' tag. In the early days, people tried to fool search engines by including invisible lines with lots of popular keywords at the top of their document.

Search engines have trouble scanning sites with many generated pages, especially if they are accessed through a search box only, **for example** a dictionary or thesaurus. ***There are many large data sets available on the web, often public domain or freely accessible, which contain high-quality information – often better quality than any old web page – but are not easy to find unless you know the site***. ***This has been called the hidden web*** and some estimates say that it is an order of magnitude larger than the visible web. Currently, there are no accepted ways to link such material into broader web searches although some products.

3.4.3 WEB TECHNOLOGY AND ISSUES

3.4.3.1 BASICS

The web consists of a set of protocols built on top of the internet that, in theory, allow multimedia documents to be created and read from any connected computer in the world. The web supports hypertext, graphics, sound and movies, and, to structure and describe the information, uses a language called *HTML* (hypertext markup language) or in some cases, XML (extensible markup language). ***HTML is a markup language that allows hypertext links, images, sounds and movies to be embedded into text***, and it provides some facilities for describing how these components are laid out. **HTML documents are interpreted by a viewer, known as a browser**; there are many browsers, and each can interpret HTML in subtly different ways, or support different levels of functionality, which means that a ***web page viewed through one browser can look very different from the same page viewed through another. The web requires no particular multimedia capabilities from the machines that run the browsers***; **for example**, if sound is unavailable on a particular machine, then obviously no sound is heard but the browser still displays the text happily.

The web owes its success to many factors, including the robustness and (relative) ease of use offered by popular browsers from the very first graphical browser *Mosaic*, and continued in commercial browsers such as *Netscape Navigator*, *Microsoft Internet Explorer* and *Opera*. These offer a graphical interface to the document, controlled by the mouse.

Hypertext links are shown by highlighting the text that acts as the link in an alternative color, and are activated by clicking on the link. A further color is used to indicate a link that has already been visited. Hypertext links can also be embedded into regions within an image.

Although the browser contains most of the functionality required to view a web document, supporting text and graphics in an integrated package, special file formats and media, including some movie formats, may require additional plug-ins or helper applications.

As well as ***static web content such as text and images, many pages are dynamic***. **for example**, they may be generated from data held in databases, respond to individual information entered into forms, or include dynamic elements such as Java applets.

3.4.3.2 WEB SERVERS AND WEB CLIENTS

- the web is distributed in
  - different machines far across the world
  - pages stored on servers
  - browsers (the clients) ask for pages sent to and fro across the internet

Whereas a conventional PC program runs and is displayed on one computer, the ***web is distributed***. Different parts of it run on different computers, often in different countries of the world. They are linked, of course, by the internet, an enormous global computer network

The pages are stored on web servers that may be on a company's own premises or in special data centers. Because they are networked, the webmaster for a site can upload pages to the server from wherever she is.

For example, the web pages for www.hcibook.com are stored in a data center several thousand miles from where any of the authors live!

Your machine, the PC running the web browser, is called a *client* because it wants the pages from the servers. When you click on a link your web browser works out the full URL of the page it needs: say

'http://www.hcibook.com/e3/authors.html'. It splits this into parts. The first part is the protocol 'http' which says how it talks to the server (other alternatives include 'ftp'). The second part 'www.hcibook.com' is the host name, that is the name of the web server containing the requested page. The last part '/e3/authors.html' gives the particular file on the site. The browser then establishes a connection to the required web server (in this case 'www.hcibook.com'), and sends a message, formatted using the HTTP protocol, to the web server, which then finds the requested html file (or image, or other file type) and returns it to the browser, which then displays it to you.

If the page contains images the same process is repeated for each image, and if the page is a framed one for each frame within the page.

3.4.3.3 NETWORK ISSUES

- QoS (quality of service)
  - **bandwidth**
    - ← how much information per second**(Network capacity is called *bandwidth)***
  - **latency**
    - ← how long it takes (delay)

There is also the time it takes for a message to get across the network from your machine to the web server and back. This delay is called *latency*. Latency is caused by several factors – the finite speed of electrical or optical signals (no faster than the speed of light), and delays

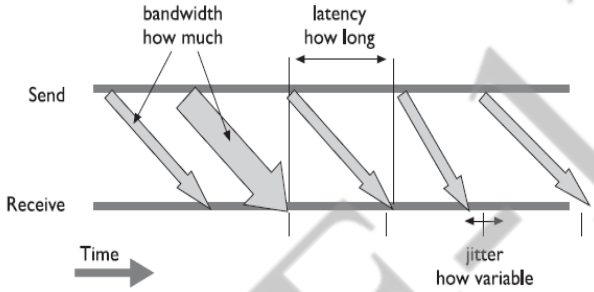


Figure 3.13 Bandwidth, latency and jitter

at routers along the way that take messages from one computer network and pass them on. This latency may not always be the same, varying with the exact route through the network traveled by a message, the current load on the different routers, etc. Variability in the latency is called *jitter* (see Figure 21.7).

- jitter
  - ← how consistent is the delay**(Variability in the latency is called *jitter)***
- reliability
  - ← some messages are lost
- connection set-up

Delivering WAP content – balancing usability and feedback

Because of the tiny screens on phones it is difficult to scroll through a long WAP page. However, if every link involved going back to the WAP server the feedback would be very slow. For this reason WML divides WAP content into *stacks* of *notes*. For the user browsing the content, the note is the parallel of the normal web page. A link may be to a note in the same stack or one in a different stack; the user is largely unaware of which it is. However, when you request a note in a stack the whole stack is downloaded to the WAP browser on your phone. This means that links to notes in the same stack have much faster feedback. By carefully arranging content within stacks and notes, the overall user experience can be improved.

3.4.4 STATIC WEB CONTENT

3.4.4.1 THE MESSAGE AND THE MEDIUM

Excellent page design can make useless material look attractive, but it still remains useless material. On the other hand, poor design can mean that excellent material is never seen by potential readers, as they have become bored, or intolerant of the medium, or confused, or for a host of other reasons have aborted their attempts to download and view the information. Pages do have to look immediately interesting and attractive if people are to spend time, effort and, because of the communication costs, money, in viewing them; the user-centered nature of the medium makes this imperative. This is in marked contrast to television or cinema or other dynamic media, which are not under any direct user control, where

information is presented to a passive audience. With web documents, people have actually to want to see the information, and make an effort to retrieve it, which clearly must have an influence on design.

Whatever is being presented, underlying all the comments made on good and bad design, the fundamental message is that, for the user group or groups who are targeted, the content should be worth reading.

When it is likely that a user will require a paper copy of the information made available over the web, ideally they should be able to download it in one go as single complete file, with the same information content but possibly a different layout. Paper does not have the same inbuilt hyper textual and active capabilities as the web page, and will be accessed in a predominantly linear fashion.

3.4.4.2 TEXT

Because web pages are displayed on many different machines, there are only a small set of fonts that can be guaranteed to be available: a standard font and a type-writer font (e.g. courier) with bold and italic versions in different sizes. However, it is possible to specify preferred fonts and many of these such as *Arial*, Verdana or Comic Sans are available on most web platforms. The difficult thing is to balance fine tuning the appearance of the text on one platform with making it readable on all.

The various structured styles such as headings allow the web designer to create material that will lay out passably on all platforms. But these offer a fairly coarse level of control. The size and boldness of the heading should be chosen carefully. **for example**, huge dark fonts on a page can look loud and brash.

There is an increasing desire to have fine control. Cascading style sheets (CSS) allow you to specify fonts, line spacing, size, etc., in a similar way to styles in a word processor or DTP package. However, care must be taken. **For example**, many pages specify fixed point sizes that may not display well on different platforms and can cause problems for people with visual impairments.

The use of color is of great importance for web pages, but it is often abused. First, it should be remembered that a significant proportion of the potential viewers of the page will have problems with color, either because they are using older machines with a limited color palette, or because they have some form of color blindness.

Users also bring a deep-rooted emotional interpretation to colors; as we have seen, in some cultures, red is associated with danger and anger, whilst green is regarded as go, or safe. Blue can be a cool color, orange a warm one, and so on.

Links usually change color once they have been accessed, providing cues to the user about what material they have already explored. This means that two distinct but still suitable colors need to be associated with each link, so that the system is acceptable whether or not the links have been activated.

Note, too, that consistent use of color can help the user understand the role of particular elements more intuitively, whereas color used for no clear purpose is often distracting.

One common mistake is to put colored text onto a similar colored background so that it becomes nearly invisible. One of the authors had a student who designed a beautifully laid out page of text, and decided to add a background to the page just before demonstrating it to the rest of the group. It was only at the demonstration that he realized that the cool black background he had added made the black text impossible to see!

There are only a limited number of text-placing options: ***text can be left or right justified, or centered. There are a few predefined formatting styles such as ordered***

and unordered lists that have additional structure, in the form of indentation from the left margin, with numbering in the case of ordered lists. Vertical positioning is even more limited, but tables and frames allow a greater degree of horizontal and vertical placement. More precise ***positioning still can be obtained using 'dynamic HTML' (DHTML), which allows parts of an HTML document (called layers or 'div' sections) to be positioned*** as if they were separate mini-pages within the browser window. The word 'dynamic' is used because these can then be controlled using JavaScript to produce various animated effects .

Remember that monitors are different sizes and that some people use full-screen windows and others smaller ones. To prevent very long lines, many designers lay out pages within tables that put maximum widths (in pixels) for columns based on typical minimum expected monitor sizes (perhaps 800 × 600 or even 640 × 480). If fixed layouts or large graphics are used then they may either display strangely on smaller windows or force the user to scroll horizontally, which many users find confusing.

The lack of explicit textual positioning makes it very difficult to produce complex mathematical equations, and the font set available is not rich enough to provide a suitable approximation. Developments in the specification are addressing this, though the intrinsic complexity of typesetting mathematics suggests that it may be a while before a simple, usable solution is found that is acceptable to readers, page designers, and implementers of web browsers alike.

- ✓ text style
  - generic styles universal: serif, sans, fixed, bold, *italic*
  - specific fonts too, but vary between platforms

- cascading style sheets (CSS) for fine control
  - ✓ **but beware older browsers and fixed font sizes**
- **colour … often abused!**
- positioning
- easy .. left, right justified or centred
- precise position**ing with DHTML**
  - ✓**but beware platforms and screen size**

3.4.4.3 GRAPHICS

Obtaining graphics

There are a number of sites on the web that contain archives of graphical images, icons, backgrounds and so on. There is also paint and image manipulation packages available on almost all computer systems, and scanners and digital cameras, where available, enable the input of photographs and diagrams.

Using graphics

While graphics and icons tend to play a significant role in web page design, their use should be carefully thought out. Graphical images take longer to load than text, and this may become a problem. Text uses 8 bits to represent a character: some rough calculations show that approximately 2000 characters represent about a screenful of information, and so 16,000 bits (2 K) are required. For graphics, one pixel may use 8 bits to represent its color: a page-sized image will be at least 600 by 400 pixels, which will take 1,920,000 bits (240 K), or 120 times as long to load. Put another way, while a picture may tell a thousand words, it takes approximately 50 times as long to appear! Users become bored with operations that take a long time to complete, and are unlikely to wait for ages while a page appears.

Complex backgrounds are the worst offenders in this area; they offer little in the way of added value to the information presented on the page, and cause great frustration for the poor reader. They tend to be designed and tested only on local machines, with high-bandwidth connections between them, which means that the time factor is negligible for the designer/user. However, this disregards the fact that many people accessing the page will be using congested, slow networks, with a transfer rate sometimes down to a few kilobits per second, rather than fast megabit links. Fussy backgrounds also have the unfortunate ability to obscure text, making it very difficult or impossible to read.

Different browsers support different types of functionality, with more recent versions having features that try to alleviate the usability problems introduced by the delay involved in downloading graphics. Most browsers support caching, in which graphics are downloaded once and temporarily stored on the user's local machine. If the same image is reused, it is fetched from the local store far more rapidly than if it was retrieved from the remote site. This clearly has implications for page design: if graphics are to be used, then their reuse wherever possible speeds up the whole process of drawing the page.

Complex graphics can sometimes be broken down into a set of items, many of which can be reused and assembled in different ways to add visual impact to the page without causing large delays. Most browsers also offer the option of turning off automatic image loading, so that only the text is downloaded. If a page then appears to be of interest, the graphics can be explicitly requested. It is sometimes possible to set out a page so that it still looks attractive even without the graphics, which is necessary if the user has turned off image loading. There are other browsers that are purely text based and do not support graphics of any sort, and for these HTML offers an additional image attribute that allows a textual description of the image to be used as an alternative. The need to support these different user preferences and browser capabilities provides a great challenge in designing pages that are acceptable to all.

Some browsers have additional features related to image handling as a technological response to the problem of page usability. If the designer specifies the size of the image in advance, the browser can lay out the text on the page first, leaving spaces for the images. This allows the user to continue to read the page contents whilst the images are being downloaded into their respective slots. This capability improves the usability of the page, and so should be supported by the page designer whenever possible, by incorporating the necessary information into the image reference.

Both GIF (graphics interchange format) and JPEG (Joint Photographic Experts Group), the most widely used web graphic image formats, can be saved in forms that allow them to be progressively transmitted. This means that images appear as a whole, but very blurred, version that becomes gradually sharper, rather than appearing in perfect resolution a line at a time. An overall impression of the page and the graphic information appearing is thus given to the user, who is then better informed about whether or not to continue the download.

The JPEG format is optimized for photographic images and makes use of their properties to offer a higher compression ratio and hence faster loading. However**,its compression is *lossy***, that is the image reproduced is slightly different from the

original, losing certain kinds of visually indistinguishable colors and losing high-frequency change. The latter is because photographs tend to have slowly varying changes with few sharp edges. If sharp-edged images such as diagrams or text labels are stored as JPEGs, small *artifacts* are produced such as ripples appearing around letters and lines. In contrast GIF uses a *lossless* compression so that the image appears exactly as it started. Although GIFs can be used for photographic images, the compression is very poor.

The GIF format also allows *animated GIFs*. These are a sort of mini-slide show or movie where several images are stored in the same file and play one after another. These can be used to produce simple and effective animations, but when overused can lead to very 'noisy' pages.

Active image maps are pictures with defined 'hot' areas, which, when clicked, execute a particular script, usually calling another web page or graphic. The careful use of such maps can transform an interface, but there is an overhead to pay in loading the map and calling and running the script, and this should be considered carefully by a page designer.

Another characteristic of image maps is that there is rarely any indication of which areas are active and which is not, and it can be difficult for users to ensure that they have visited all the active regions. For accessing spatially organized information, image maps are very suitable .

Icons

on the web just small images

- for bullets, decoration
- or to link to other pages
- lots available!
- **design … just like any interface**
- need to be understood
- designed as **collection to fit …**
- under construction

Graphics and color

Using many different colors within graphics may well result in the browsers for older machines running out of entries in the color map, with unpredictable consequences. This is often problematical as the browser may be running in tandem with other color applications, and only has a restricted range of colors to begin with.

For many consumer markets, **for example** in the UK and the US, this is unlikely to be a problem as home machines are often relatively recent. However, many businesses continue to use older PCs so long as they 'do the job' and PDAs may not have a full color palette. Furthermore, in economically deprived areas, where there is computer access it may well be through older or second-hand machines.

If universal access is required it is therefore still wise, where possible, to restrict images to a limited number of colors, taken from the standard 216 color web palette, and to reduce complex color images to simpler approximations. Reducing the number of colors used also allows the depth of the images to be reduced; a change from a default of 8 bits to, say, 4 bits will produce a twofold speedup in image loading. The earlier comments on the use of color obviously apply as much to graphics as they do to text.

One further point should be made about graphics: computer screens are typically limited to a resolution of around 72 dpi (dots per inch), and so either high-resolution images will have to be displayed much larger than actual size, or the increased resolution will be forfeited.

3.4.4.4 MOVIES AND SOUND

Movies and sound are both available to users of the web, and hence to page designers. One problem associated with them is actually obtaining appropriate sound and video clips, as they usually require some sort of multimedia capability on behalf of the host machine in order to be able to digitize sound and capture and digitize video. Video suffers from the same problems as graphics, magnified by an order of magnitude or two; it can take extremely large amounts of time for a video segment to download. Video is also not well integrated into the web, requiring the creation of a process to run it that is separate from the page from whence it came. Not all receiving machines have the capability to play video, or sound, and so it is unwise for a designer to rely on these dynamic media to convey information without replicating it elsewhere.

The use of sound and video moves page design further away from the typesetter and toward the sound engineer and cinematographer; the integration of these cinematic media with the enhanced textual capabilities offered by the web is a new domain, in which the techniques that work and those that fail have not yet been fully explored, let alone understood.

The need to download movies and sound (see Figure 3.14) puts sharp limits on the length of clip that can be shown. Streaming media over the internet, such as RealVideo, RealAudio and CuSeeMe, allow potentially unlimited sources. As well as longer prepared clips, these techniques allow live transmission (e.g. live radio broadcasts over

RealAudio) and long recorded sequences for asynchronous communication. An excellent use of the latter is the eClass project, which links recordings of audio and video during a lecture with pen strokes on an electronic whiteboard, so that students can replay the part of a lecture associated with any slide or annotation.

Acceptable streaming video and audio is achieved by a combination of high compression and large client-end buffers. The former leads to loss of quality including blurring and ghosting after rapid changes in screen content. The latter leads to delays, often of several seconds, which makes it impossible to support video conferencing. The challenges of achieving high quality transmissions (e.g. for video on demand) and low latency (e.g. for video conferencing) are active research topics in multimedia technology.

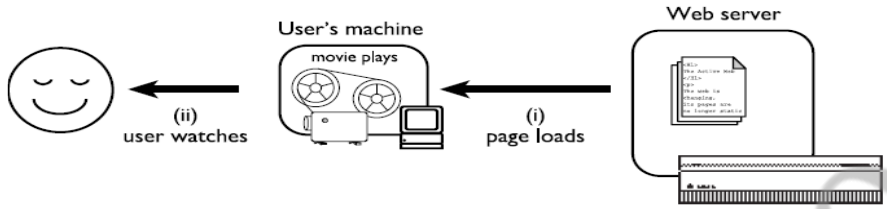


Figure 3.14 Animated GIF or movie needs to download completely

Buffering effectively solves this by trading off quality against delay, which is okay for fixed content, or low pace change (as in eClass), but is problematic when we require a *high pace* of interaction (as in video conferencing). CuSeeMe uses little buffering and hence is more likely to suffer break-up of video and audio.

3.4.5 DYNAMIC WEB CONTENT

3.4.5.1 THE ACTIVE WEB

In the early days, the web was simply a collection of (largely text) pages linked together. The material was static or slowly changing and much of it authored and updated by hand. Some pages were generated on the fly, in particular the gateways into ftp servers and to gophers, which were so important in adding 'free' content to the web. However, even here the user's model was still of a static repository of information. Web surfers may not have always known where they were, but they had a pretty good idea of what they were seeing and that if they came back it would be the same.

3.4.5.2 WHAT HAPPENS WHERE

When considering dynamic material on the web we need to take the external, user's viewpoint and ask *what* is changing: media, presentation or actual data; **by *whom*: by the computer automatically, by the author, by the end-user or by another user;** and ***how often***, the ***pace*** of change: seconds, days or months? From a technical standpoint, we also need to know ***where* 'computation' is happening**: in the user's web-browsing client, in the server, in some other machine or in the human system surrounding it? The 'what happens where' question is the heart of architectural design. It has a major impact on the pace of interaction, both ***feedback*, how fast users see the effects of their own actions**, and ***feed through*, how fast they see the effects of others' actions**. Also, where the computation happens influences where data has to be moved to with corresponding effects on download times and on the security of the data.

The user view

One set of issues is based on what the end-user sees, the end-user here being the web viewer.

What changes? This may be a media stream (video, audio or animation) which is changing simply because it is the fundamental nature of the medium. It may be the presentation or view the user has of the underlying content; **for example**, sorting by different categories or choosing text-only views for blind users.

A special form of presentation change is when only a selection of the full data set is shown, and that selection changes. The deepest form of change is when the actual content changes.

By whom? Who effects the changes? In the case of a media stream or animation, the changes are largely automatic – made by the computer. The other principal sources of change are the site author and the user. However, in complex sites users may see each other's changes – feed through.

How often? Finally, what is the pace of change? Months, days, or while you watch?

Technology and security

The fundamental question here is where 'computation' is happening. If pages are changing, there must be some form of computation' of those changes. Where does it happen?

Client One answer is in the user's web-browsing client enabled by Java applets, various plug-ins such as Flash, scripting using JavaScript or VBScript with dynamic HTML layers, CSS and DOM (Domain Object Model).

Server A second possibility is at the web server using CGI scripts (written in Perl, C, UNIX shell, Java or whatever you like!), Java Servlets, Active Server Pages or one of the other server-specific scripting languages such as PHP. In addition, client-side Java applets are only allowed to connect to networked resources on the same machine as they came from. This means that databases accessed from client-side JDBC (Java database connectivity) must run on the web server (see below).

Another machine Although the pages are *delivered* from the web server, they may be *constructed* elsewhere. For hand-produced pages, this will usually be on the page author's desktop PC. For generated pages, this may be a PC or a central database server.

People The process of production and update may even involve people!

It is easy to roll out maxims such as 'users first', but, in reality, the choice between these options is not solely a matter of matching the end-user requirements. The best choice also depends on the expertise of the web developer and external limitations. If the server runs on a UNIX machine, you can't expect to use Microsoft Active Server Pages.

If you are designing for an intranet you may even get to influence the choice of client software and so make it easier to use more complex client-end solutions.

3.4.5.3 FIXED CONTENT – LOCAL INTERACTION AND CHANGING VIEWS

Probably the most hyped aspect of the web in recent years has been Java. In fact, Java can be used to write server-end software and platform independent standalone pro-grams (not to mention the embedded systems for which it was originally designed!), but the aspect that most people think of is Java applets.

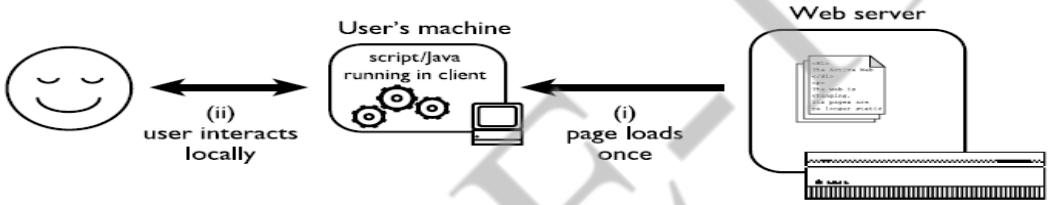


Figure 3.15 Java applet or JavaScript running locally

Applets are just one of the techniques that can be added to give client-end inter-action (and about the least well integrated into the rest of the page). The most common alternatives are JavaScript, Flash and if you are prepared to limit yourself to Windows platforms, ActiveX plug-ins. These techniques share the characteristic that they are downloaded to the user's own machine (see Figure 3.15) and thereafter all interaction happens on the PC, not across the network (with caveats – see below).

The simplest use of this is to add interaction widgets such as roll-over buttons (usually using JavaScript). More complex pages may add the equivalent of an

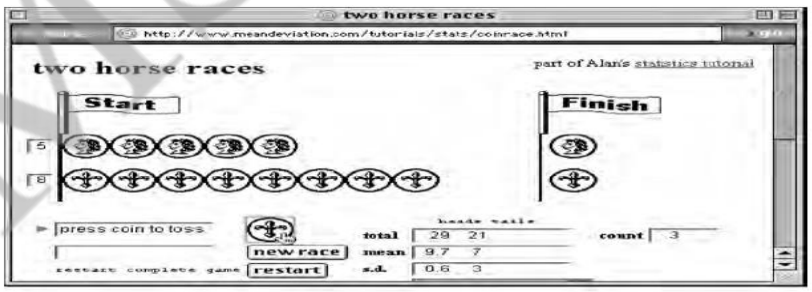


Figure 3.16 Simulated coin tossing using JavaScript. Screen shot frame reprinted by permission from Microsoft Corporation

interactive application on the page. **For example**s, see Alan's pages on coin tossing experiments (Figure 3.16), which use JavaScript to emulate real and biased coins, and dancing histograms (Figure 3.17), which use a Java applet. See Sun and JavaSoft's own sites for many more examples. The addition of DHTML gives even more opportunities for dynamic pages where parts of the page can move, change size, or change content all without any interaction with the web server.

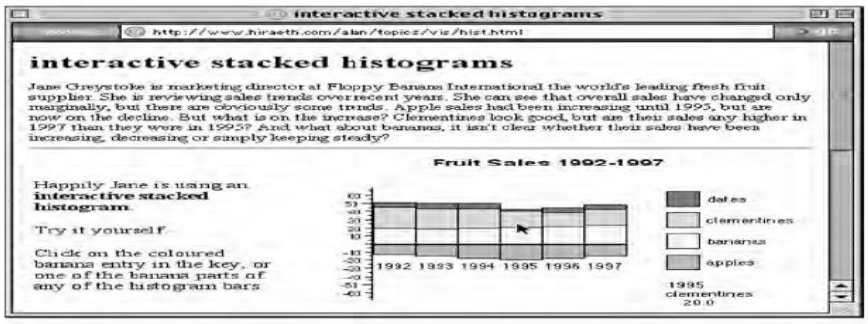


Figure 3.17 Dancing histograms using Java applet. Screen shot frame reprinted by permission from Microsoft Corporation

3.4.5.4 SEARCH

All common web search pages work by submitting forms to the server where CGI programs perform the searches and return results. An additional reason for this approach is that most browsers support forms, but some still do not support Java or scripting in a consistent manner. The web search engine for this book works in this way.

The user's keywords are submitted to the server using an HTML form, they are compared against pre-prepared indexes at the server and all matching paragraphs in the book are returned (Figure 3.18).

The variable factor is the user's input. The computation needs both the data supplied by the web author and the user's input. The result must end up on the user's screen. Either the data must come to the user's machine or the user's input must go to the server.

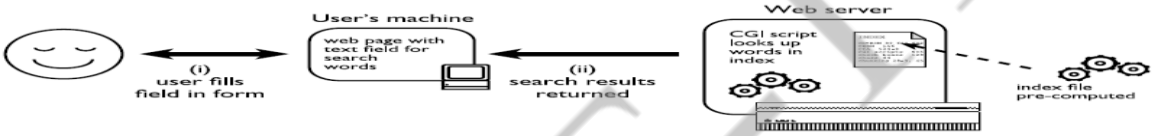


Figure 3.18 HCI book search

3.4.5.5 AUTOMATIC GENERATION

Automatic generation includes vendor-specific products such as Oracle Web Server and Domino (for publishing Lotus Notes), and also more general techniques such as using SQL (structured query language) or JDBC to access databases from CGI scripts or even from running Java applets.

Database-generated websites have many advantages.

- They make use of existing data sources.
- They guarantee consistency of different views of the data within the site and between the site and the corporate data.
- They allow easy generation of tables of contents, indices, and inter-page links.
- They separate content and layout.

The most dynamic way to get database content on the web is by accessing a database directly from a running applet.

The interface can then have any look-and-feel that can be programmed in Java and can allow very rapid interaction with the user. The Java applet can establish an internet connection back to the web server to access data files using HTTP (as if they were web pages), it can connect to a bespoke server (e.g. for chat type applications) or it can use standard data-base access methods. The latter would normally use JDBC, the Java database access package. Using JDBC the applet can issue complex SQL queries back to the database meaning that some of the most complex work happens there (Figure 3.19).

In all cases, the Java security model built into most web browsers means that the applet can only connect back to the machine from which it came. This means that ***the database server must run on the same machine as the web server***.

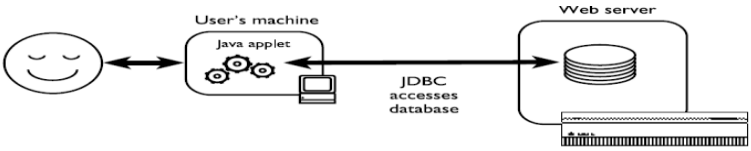


Figure 3.19 Java applet accesses database using JDBC

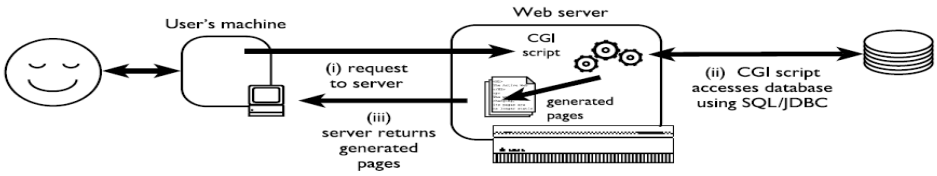


Figure 3.20 CGI script accesses database

The most insecure part of any system is usually the web server, both because it is easy to leave loop holes in the many file access permissions and also because it often sits outside the most secure part of a corporate firewall.

The more common solution is where the user uses a web forms interface (or special URL) and then a CGI script runs at the server end accessing the database (Figure 3.20). The CGI script generates a web page, which is then returned to the user. Some of the vendor-specific solutions use essentially this approach but bypass the web-server/CGI step by having their own special web server which accesses the database directly using their own scripting language or templates.

The user interface of such systems is limited to standard HTML features. This is a limitation, but is at least consistent and means that it will work with virtually any browser. Java applets can offer more rapid surface interaction, but both have to wait for the actual data to move between server and client. Of course, the pages generated by a CGI script can themselves contain JavaScript or Java applets for local inter-action, so the difference between the two solutions is not so radical as first appears.

From a security angle, the database accessed from the CGI script can run on a separate machine (using standard database remote access methods or even a Java/JDBC CGI program), thus making the system more secure. However, the database cannot be entirely secure – if the web-server machine is compromised the CGI scripts can be altered to corrupt or modify the database. The special vendor-specific web servers are probably more secure as they don't require a standard web server to be running.

3.4.5.6 BATCH GENERATION

A low-tech but very secure solution is to generate pages offline from a database and then upload them to the web server (Figure 3.21).

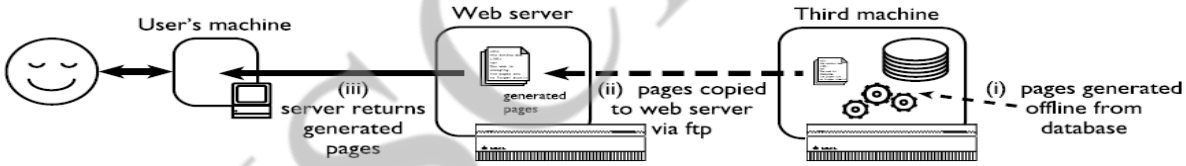


Figure 3.21 Batch pre-generation of web pages

This is certainly a simple solution as it separates out the task of page generation from that of page delivery. ***Pages can be generated directly using many standard database packages such as Access or HyperCard.*** Alternatively, **standalone programs in languages such as Visual Basic, Java or C can access a database and output HTML pages.** These programs can run on a central computer, or on your own PC. ***The generating program simply produces a set of HTML pages on your own disk that can be checked locally and then copied onto the web server using ftp or shared network disks.***

The snippet of Visual Basic in Figure 2.22 is a trivial but fully functioning HTML generator.

```
Set db = openDatabase("C:\test.mdb"); sql = "select Name, Address from Personnel;"
Set query = db.OpenRecordset(sql) Open "out.html" For Output As #1
Print #1, "<h1>Address List</h1>" query.MoveFirst
While Not query.EOF
Print #1, "<p>" & query("Name") & " "; query("Address")
query.MoveNext Wend
Close #1 query.Close
```

Some web scripting languages can be used in this mode too. **For example** PHP allows you to send the page being generated into a buffer, which can then be saved to a file. This can be run on a separate machine, or on the web server itself.

The offline generation of web pages means that there is no need for an online connection between the web server and the database. The web server can be configured without CGI scripting enabled at all, which considerably increases its security.

```
Set db = openDatabase("C:\test.mdb");
sql = "select Name, Address from
Personnel;"
Set query = db.OpenRecordset(sql)
Open "out.html" For Output As #1

Print #1, "<h1>Address List</h1>"
query.MoveFirst
While Not query.EOF
  Print #1, "<p>" & query("Name") & " ";
query("Address")
  query.MoveNext
Wend

Close #1
query.Close
```

Figure 3.22 Visual Basic code to generate a web page

3.4.5.7 DYNAMIC CONTENT

The most 'active' web pages are those where the content of the pages reacts to and is updateable by the web user.

If pages are generated from database content using either the Java-applet/JDBC method or the CGI method, the same mechanisms can as easily be used to update as to access the database. The feedback of changes to the user is effectively instantaneous

For example, you check for seat availability on the theatre web page, select a seat, enter your credit card details and not only is the seat booked, but you can see it change from free to booked on the web page.

Many business applications operate an *n*-tier web architecture. This involves multiple layers of software where the outer layers are concerned more with the user interface and the inner layers more with business functionality. Figure 3.23 shows this using several web standards.

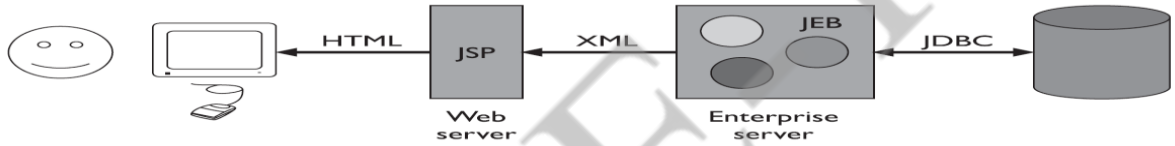


Figure 3.23 n-tier architecture

The user interacts through a web browser with a web server. ***The pages are generated using Java Servlet Pages (JSP). To generate the page the servlets connect to Java Enterprise Beans (JEB) on an enterprise server.*** These are components that encapsulate 'business logic'. **For example**, in a banking system this could include rules on whether a particular transaction is allowed. These Java Enterprise Beans draw their data from the corporate database using JDBC connections.

All the Best!

4. MOBILE HCI

Syllabus: Mobile Ecosystem: Platforms, Application frameworks - Types of Mobile Applications: Widgets, Applications, Games - Mobile Information Architecture, Mobile 2.0, Mobile Design: Elements of Mobile Design, Tools.

Topics

4.1 Mobile Ecosystem:
- Platforms
- Application frameworks

4.2 Types of Mobile Applications:
- Widgets
- Applications
- Games

4.3 Mobile Information Architecture

4.4 Mobile 2.0

4.5 Mobile Design:
- Elements of Mobile Design
- Tools

4.1 Mobile Ecosystem:

Mobile is an entirely unique ecosystem and like the Internet, it is made up of many different parts that must all work seamlessly together.

If the Internet is a cloud, then the mobile ecosystem would be the atmosphere,made up of many clouds.

The internet is just one of these clouds, though a very large one.

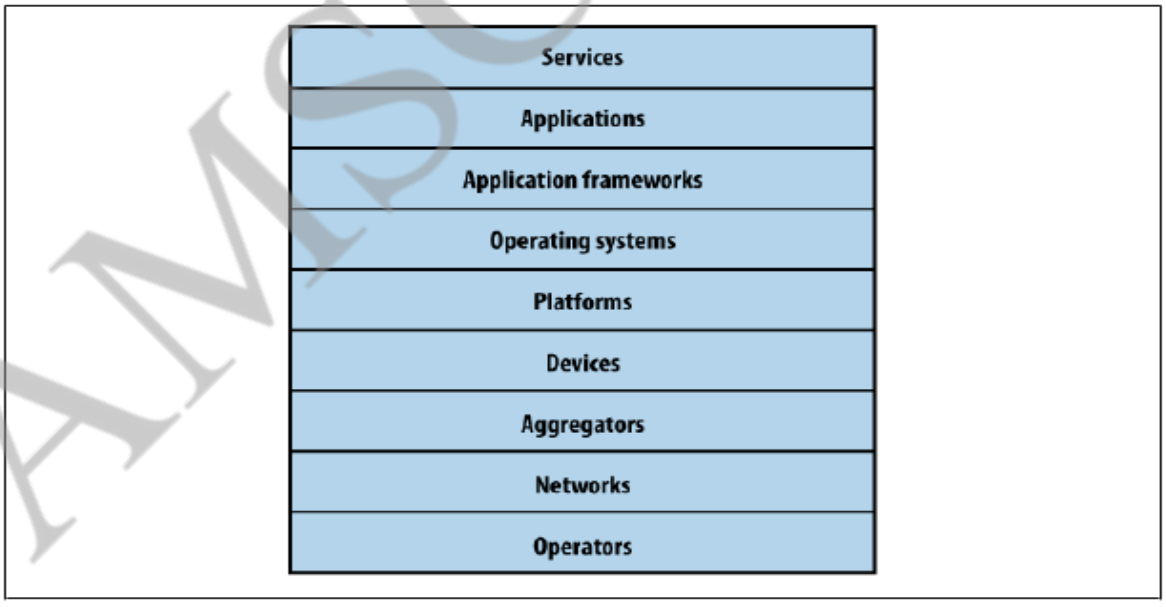


Fig 4.1 Layers of Mobile Ecosystem

Each layer is reliant on the others to create a seamless, end-to-end experience. Although not every piece of the puzzle is included in every mobile product and service, for the majority of the time, they seem to add

complexity to our work, regardless of whetherwe expressly put them there. The following sections expand on each of these layers and the roles they play in themobile ecosystem.

4.1.1 Platforms

A mobile platform's primary duty is to provide access to the devices. To run software and services on each of these devices, we need a *platform*, or a core programminglanguage in which all of the software is written. Like all software platforms, these aresplit into three categories: **licensed, proprietary, and open source**.

4.1.1.1 Licensed

Licensed platforms are sold to device makers for nonexclusive distribution on devices.
The goal is to create a common platform of development Application Programming Interfaces (APIs) that work similarly across multiple devices with the least possible effort required to adapt for device differences, although this is hardly reality.
Following are the **licensed platforms**:

i. ***Java Micro Edition (Java ME)***

Formerly known as J2ME, Java ME is by far the most predominant software platform of any kind in the mobile ecosystem. It is a licensed subset of the Java platform and provides a collection of Java APIs for the development of software for resourceconstrained devices such as phones.

ii. ***Binary Runtime Environment for Wireless (BREW)***

BREW is a licensed platform created by Qualcomm for mobile devices, mostly for the U.S. market. It is an interface-independent platform that runs a variety of application frameworks, such as C/C++, Java, and Flash Lite.

iii. ***Windows Mobile***

Windows Mobile is a licensable and compact version of the Windows operating system, combined with a suite of basic applications for mobile devices that is based on the Microsoft Win32 API.

iv. ***LiMo***

LiMo is a Linux-based mobile platform created by the LiMo Foundation. Although Linux is open source, LiMo is a licensed mobile platform used for mobile devices. LiMo includes SDKs for creating Java, native, or mobile web applications using the WebKit browser framework.

4.1.1.2 Proprietary

Proprietary platforms are designed and developed by device makers for use on theirdevices. They are not available for use by competing device makers. **These include**:

i. ***Palm***

Palm uses three different proprietary platforms. Their first and most recognizable is the Palm OS platform based on the C/C++ programming language; this was initially developed for their Palm Pilot line, but is now used in low-end smartphones such as the Centro line. As Palm moved into higher-end smartphones, they started using the Windows Mobile-based platform for devices like the Treo line. The most recent platform is called webOS, is based on the WebKit browser framework, and is used in the Pre line.

ii. ***BlackBerry***

Research in Motion maintains their own proprietary Java-based platform, used exclusively by their BlackBerry devices.

iii. ***iPhone***

Apple uses a proprietary version of Mac OS X as a platform for their iPhone andiPod touch line of devices, which is based on Unix.

4.1.1.3 Open Source

Open source platforms are mobile platforms that are freely available for users to download, alter, and edit.

They are newer and slightly controversial, but increasingly gaining traction with device makers and developers.

Example: Android. It is developed by the Open Handset Alliance, which is spearheaded by Google. The Alliance seeks to develop an open source mobile platform based on the Java programming language.

4.1.2 Application Frameworks

Application frameworks are used to create applications, such as a game, a web browser, a camera, or media player. Although the frameworks are well standardized, the devices are not.
The largest challenge of deploying applications is knowing the specific device attributes and capabilities.

Example: For creating an application using the Java ME application framework, we need to know what version of Java ME the device supports, the screen dimensions, the processor power, the graphics capabilities, the number of buttons it has, and how the buttons are oriented.
Multiply that by just a few additional handsets and we have hundreds of variables to consider when building an application.
Multiply it by the most popular handsets in a single market and we can easily have a thousand variables, quickly doomingour application's design or development.

Although mobile applications can typically provide an excellent user experience, it almost always comes at a fantastic development cost, making it nearly impossible to create a scalable product that could potentially create a positive return on investment.

A common alternative these days is creating applications for only one platform, such as the iPhone or Android. By minimizing the number of platforms the developer has to support and utilizing modern application frameworks, the time and cost of creation go down significantly.
The mobile web browser is an application that renders content that is device-, platform-, and operating-system-independent.

The web browser knows its limitations, enabling content to scale gracefully across multiple screen sizes. However, like all applications, mobile web browsers suffer from many of the same device fragmentation problems.
When a device is sold to an operator, it is provisioned (customized) to their requirements. This means the operators will often put customized applications on each of the devices sold. With the example of the RAZR, every operator had it and every operator put a different web browser on it. To make matters worse, the RAZRs, like most phones, are not field-refreshable, meaning that you can't update the software, upgrade the applications, or eliminate bugs.

For example, if a device manufacturer makes a device called the MDv1, they must strike a deal with an operator if they want to preload an operator store application, a different web browser, and bowling game. The device is sold as the MDv1.1. The operator sells the devices, or worse, gives them away for free. A couple hundred thousand of them go out into the marketplace before a glitch in the hardware is detected, such as dropped calls. Because the device cannot be upgraded by cable or over the air, the operator stops selling the MDv1.1, but seeing that they have a hit, they quickly replace it with the MDv1.1.1. The whole process is repeated as it is provisioned to each operator. Suddenly, there is an MDv1.2, an MDv1.3, an MDv1.4, and so on. Then we have the next generations—the MDv1.2.1, the MDv1.3.1, the MDv1.4.1, and so on, spreading like a virus. This is essentially what causes device fragmentation, making application development a costly and timely endeavor.

4.2 Types of Mobile Applications:

4.2.1 Mobile Web Widgets

Largely in response to the poor experience provided by the mobile web over the years, there has been a growing movement to establish mobile widget frameworks and platforms.

For years the mobile web user experience was severely underutilized and failed to gain traction in the market, so several operators, device makers, and publishers began creating widget platforms to counter the mobile web's weaknesses.

According to ***Webster's Dictionary*** **"**Mobile Widget is a component of a user interface that operates in a particular way". ***Wikipedia*** defines a web widget this way: "A portable chunk of code that can be installed and executed within any separate HTML based web page by an end user without requiring additional compilation".

Between these two definitions is a better answer: **"A mobile web widget is a standalone chunk of HTML-based code that is executed by the end user in a particular way".**

Basically, mobile web widgets are small web applications that can't run by themselves; they need to be executed on top of something else. Opera Widgets, Nokia Web RunTime (WRT), Yahoo! Blueprint, and Adobe Flash Lite are all examples of widget platforms that work on a number of mobile handsets. Using a basic knowledge of HTML, we can create compelling user experiences that tap into device features and, in many cases, can run while the device is offline.

Widgets, however, are not to be confused with the utility application context, a user experience designed around short, task-based operations.

Pros and Cons of Mobile Web Widgets

Pros

The pros of mobile web widgets are:

- They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
- They can be simple to deploy across multiple handsets.
- They offer an improved user experience and a richer design, tapping into device features and offline use.

Cons

The cons of mobile web widgets are:

- They typically require a compatible widget platform to be installed on the device.
- They cannot run in any mobile web browser.
- They require learning additional proprietary, non-web-standard techniques.

4.2.2 Mobile Web Applications

Mobile web applications are mobile applications that do not need to be installed or compiled on the target device. Using XHTML, CSS, and JavaScript, they are able toprovide an application-like experience to the end user while running in any mobile web browser. Web applications allow users to interact with content in real time, where a click or touch performs an action within the current view.

The history of how mobile web applications came to be so commonplace is interesting. Shortly after the explosion of Web 2.0, web applications like Facebook, Flickr, and Google Reader hit desktop browsers, and there was discussion of how to bring those same web applications to mobile devices. The Web 2.0 movement brought user-centered design principles to the desktop web, and those same principles were sorely needed in the mobile web space as well.

The challenge, as always, was device fragmentation. The mobile browsers were years behind the desktop browsers, making it nearly impossible for a mobile device to render a comparable experience. While XHTML support had become fairly commonplace across devices, the rendering of CSS2 was wildly inconsistent, and support for Java-

Script, necessary or simple DHTML, and Ajax was completely nonexistent.

To make matters worse, the perceived market demand for mobile web applications was not seen as a priority with many operators and device makers. It was the classic chicken or-the-egg scenario. What had to come first, market demand to drive browser innovation or optimized content to drive the market?

With the introduction of the first iPhone, we saw a change across the board.

Using WebKit, the iPhone could render web applications not optimized for mobile devices as perfectly usable, including DHTML- and Ajax-powered content. Developers quickly got on board, creating mobile web applications optimized mostly for the iPhone. The combination of a high-profile device with an incredibly powerful mobile web browser and a quickly increasing catalog of nicely optimized experiences created the perfect storm the community had been waiting for.

Pros and Cons of Mobile Web Applications

Pros

The pros of mobile web applications are:

• They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
• They are simple to deploy across multiple handsets.
• They offer a better user experience and a rich design, tapping into device features and offline use.
• Content is accessible on any mobile web browser.

Cons

The cons of mobile web applications are:

• The optimal experience might not be available on all handsets.
• They can be challenging (but not impossible) to support across multiple devices.
• They don't always support native application features, like offline mode, locationlookup, filesystem access, camera, and so on.

Native Applications

The next mobile application medium is the oldest and the most common; it is referredto as native applications. These applications actually should be called "platform applications," as they have to be developed and compiled for each mobile platform.

These native or platform applications are built specifically for devices that run the platform in question. The most common of all platforms is Java ME (formerly J2ME). A device written as a Java ME MIDlet should work on the vast majority of feature phones sold around the world. The reality is that even an application written as a Java ME MIDlet still requires some adaptation and testing for each device it is deployed on.

In the smartphone space, the platform SDKs get much more specific. Although many smartphones are also powered by Java, an operating system layer and APIs added to allow developers to more easily offload complex tasks to the API instead of writing methods from scratch. In addition to Java, other smartphone programming languages include versions of C, C++, and Objective-C.

Creating a platform application means deciding which devices to target, having a means of testing and certification, and a method to distribute the application to users. Thevast majority of platform applications are certified, sold, and distributed either through an operator portal or an app store. It is possible to create a Java ME MIDlet application and publish it for free on the Web, but it is rarely done.

Pros and Cons of Native Applications

Pros

The pros of native applications include:

• They offer a best-in-class user experience, offering a rich design and tapping into device features and offline use.
• They are relatively simple to develop for a single platform.
• You can charge for applications.

Cons

The cons of native applications include:

• They cannot be easily ported to other mobile platforms.
• Developing, testing, and supporting multiple device platforms is incredibly costly.
• They require certification and distribution from a third party that you have no control over.
• They require you to share revenue with the one or more third parties.

4.2.3 Games

The final mobile medium is games, the most popular of all media available to mobile devices. Technically games are really just native applications that use the similar platform.

They are different from native applications for two reasons: they cannot be easily duplicated with web technologies, and porting them to multiple mobile platforms is a bit easier than typical platform-based applications. Adobe's Flash and the SVG (scalable vector graphics) standard are the only way for the mobile games on the Web now, and will likely be how it is done on mobile devices in the future, the primary obstacle being the performance of the device in dealing with vector graphics.

The game mechanics are the only thing that needs to adapt to the various platforms. Like in console gaming, there are a great number of mobile game portingshops that can quickly take a game written in one language and port it to another.

Mobile games stand apart from all other application genres—their capability to be unique and difficult to duplicate in anotherapplication type, though the game itself is relatively easy to port.

Pros and Cons of game applications

Pros

• They provide a simple and easy way to create an immersive experience.
• They can be ported to multiple devices relatively easily.

Cons

The cons of game applications are:

• They can be costly to develop as an original game title.
• They cannot easily be ported to the mobile web.

4.3 Mobile Information Architecture:

What Is Information Architecture?

• The structural design of shared information environments.
• The combination of organizations, labeling, search, and navigation systems within websites and intranets.
• The art and science of shaping information products and experiences to support usability and findability.
• An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape

Information architecture, describe several **unique disciplines**, including the following:

i. ***Information architecture***

The organization of data within an informational space. In other words, how the user will get to information or perform tasks within a website or application.

ii. ***Interaction design***

The design of how the user can participate with the information present, either in a direct or indirect way, meaning how the user will interact with the website of application to create a more meaningful experience and accomplish the goals.

iii. ***Information design***

The visual layout of information or how the user will assess meaning and direction given the information presented to him.

iv. ***Navigation design***

The words used to describe information spaces; the labels or triggers used to tell the users what something is and to establish the expectation of what they will find.

v. ***Interface design***

The design of the visual paradigms used to create action or understanding.

Information architecture composes the core of the user experience. The role of information architecture is played by a variety of people, from product managers to designers and even developers. Words like intuitive, simple, findable, usable, or the executive favorite—easy to-use—all describe the role that information architects play in creating digital experiences.

The visual design of product, what frameworks we use, and how it is developed are integral to the success of any product, but the information architecture stands apart as being the most crucial element of the product. It isthe first line of scrimmage—the user's first impression of product. Even if we have the best design, the best code, and the best backend service, if the user cannot figure out how to use it, she will fail—and so will the product.
When thinking about mobile information architecture, we want to keep it as simple as possible.

Support the defined goals. If something doesn't support the defined goals, lose it. Go back to user goals and needs, and identify the tasks that map to them. Find those needs and fill them.

Ask yourself: what need does my application fill? What are people trying to do here?

What is their primary goal? Once we understand that, it is a simple process of reverse engineering the path from where they want to be to where they are starting. Cut out everything else—the site or application doesn't need it. For example, to get some news and information on a mobile device, we need to first ask what the goal is. What is the need we are trying to fill? Then we need to apply context. Where are the users? What are they doing? Are they waiting for the bus? Do they have only a minute to spare? Or, do they have five minutes to spare? With these answers, we get our information architecture.

Keep all labels short and descriptive, and never try to be clever with the words we use to evoke action. The worst sin is to introduce branding or marketing into information architecture; this will just serve to confuse and distract users. Executives love to use the words they use internally to external communications on websites and applications, but these words have no meaning outside of company walls.

Based on what we know from web design, we should use simple, direct terms for navigating around pages rather than overly clever terms. That latter typically result in confused visitors who struggle to find the content they are looking for. When that happens, they will go elsewhere to look for the information they want. So, if we apply these same mistakes to a constrained device like mobile, then we end up adding confusion to the user experience at a higher magnitude than the Web.

1. **Site Maps**

The first deliverable we use to define mobile information architecture is the site map.

Site maps are a classic information architecture deliverable. They visually represent the relationship of content to other content and provide a map for how the user will travel through the informational space, as shown in Figure 4.2.Mobile site maps aren't that dissimilar from site maps we might use on the Web. But there are a few tips specific to mobile that we want to consider.

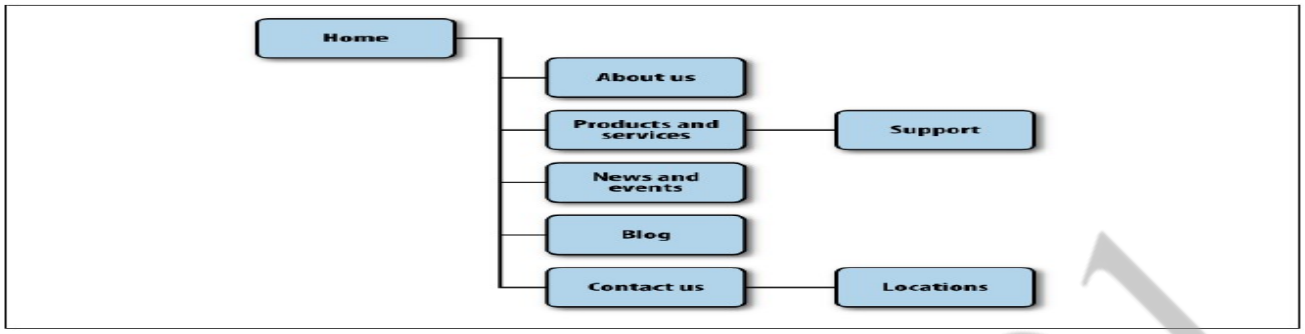


Fig 4.2 An Example of site map

In Figure 4.3, we can see a poorly designed mobile information architecture that too closely mimics its desktop cousin; it was not designed with the mobile user in mind.

But in mobile, we cannot make this assumption. In the mobile context, tasks are short and users have limited time to perform them. And with mobile websites, we can't assume that the users have access to a reliable broadband connection that allows them to quickly go back to the previous page. The users pay cash for viewing the wrong page by mistake, they pay to again download the page they started from: we can't assume that pages will be cached properly.

Therefore, the advice is to limit users' options.

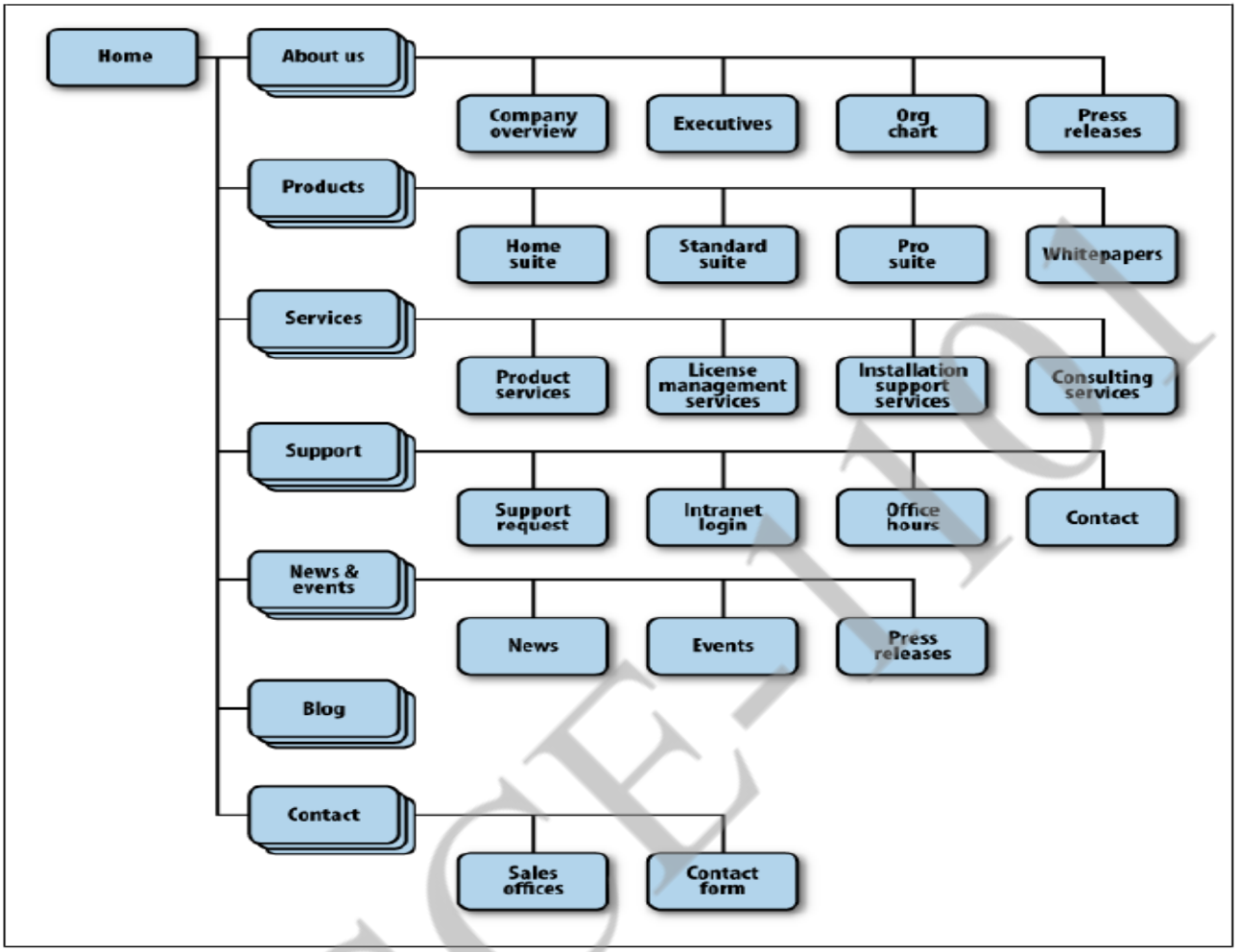


Fig 4.3 An example of a bad Mobile Information Architecture that was designed with desktop users in mind rather than mobile users

After the users have selected a path, it isn't always clear whether they are getting to where they need to be. Information-heavy sites and applications often employ nested or drill-down architectures, forcing the user to select category after category to get to their target. To reduce risking the user's time and money, we want to make sure we present enough information for the user to wade through information architecture successfully. On the Web, we take these risks very lightly, but with mobile, we must give our users a helping hand. We do this by teasing content within each category— that is, providing at least one content item per category.

The challenge with ringtone sites is you have a lot of items, grouped by artist, album, genre, and so on. The user starts with a goal like "I want a new ringtone" and finds an item that suits his taste within a catalog of tens of thousands of items. In order to make sense of a vast inventory of content, we have to group, subgroup, and sometimes even subgroup again, creating a drill-down path for the user to browse.

Though on paper this might seem like a decent solution, once you populate an application with content, the dreaded "Page 1 of 157" appears.

Users would flip through a few pages of content, then give up or go back and visit another area. We could see a direct relationship to the number of pages viewed to sales—essentially, more pages loaded meant fewer sales. Then we realized we could take the most popular item based on sales and place it as the first item in any list, which is teasing the content.

In Figure 4.4, we can see in a constrained screen that teasing the first few items of the page provides the user with a much more intuitive interface, immediately indicating what type of content the user can expect.
We immediately saw that users were finding content more quickly, driving up our sales.

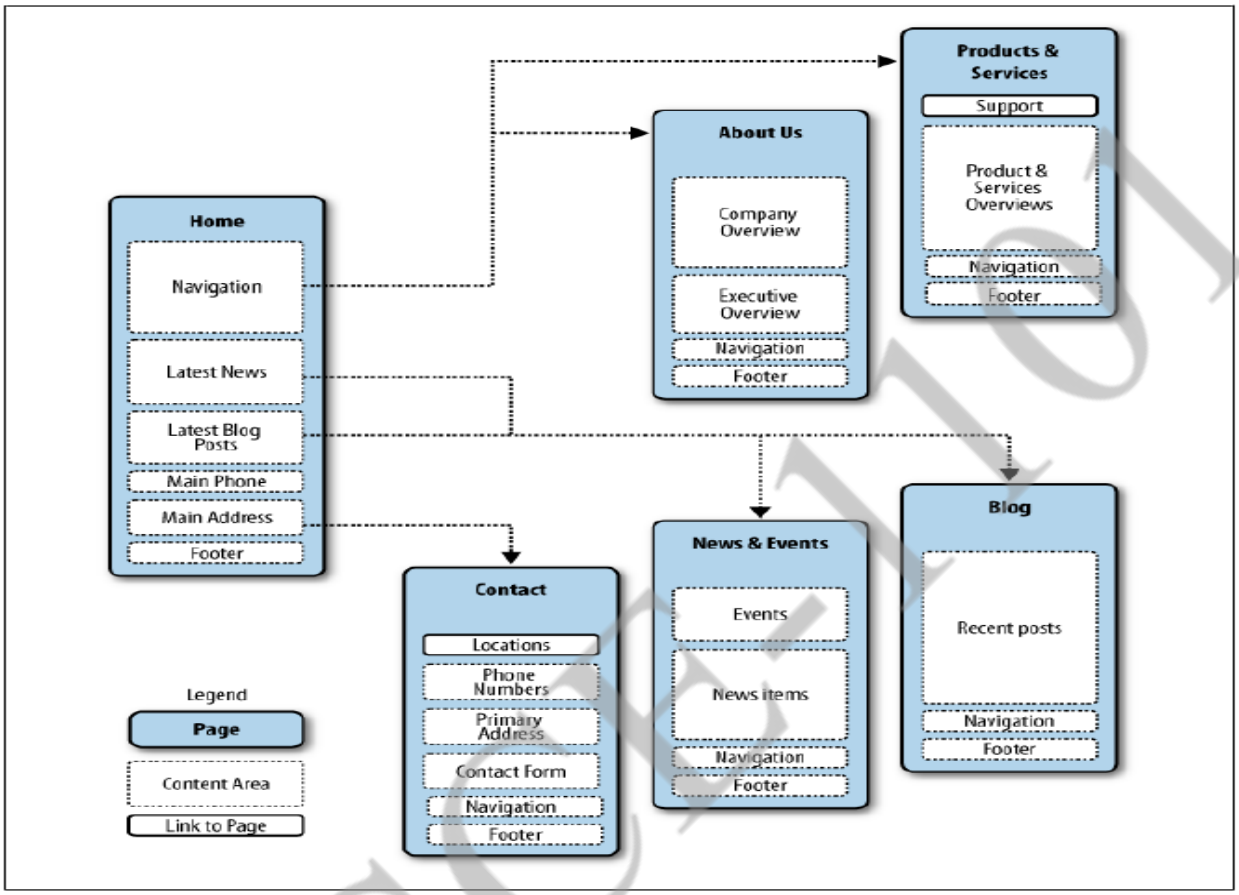


Fig 4.4 Teasing content to confirm the user's expectations of the content within

2. **Clickstreams**

Clickstream is a term used for showing the behavior on websites, displaying the orderin which users travel through a site's information architecture, usually based on data gathered from server logs. Clickstreams are usually historical, used to see the flaws in information architecture, typically using heat-mapping or simple percentages to show where users are going. They are a useful tool for re architecting large websites.

However, information architecture in mobile is more like software than it is the Web, meaning that we create clickstreams in the beginning, not the end. This maps the idealpath the user will take to perform common tasks. Being able to visually lay out the path users will take gives a holistic or bird's-eye view of mobile information architecture, just as a road map does. When we can see all the paths next to each other and take a step back, we start to see shortcuts and how we can get users to their goal faster or easier, as shown in Figure 4.5.

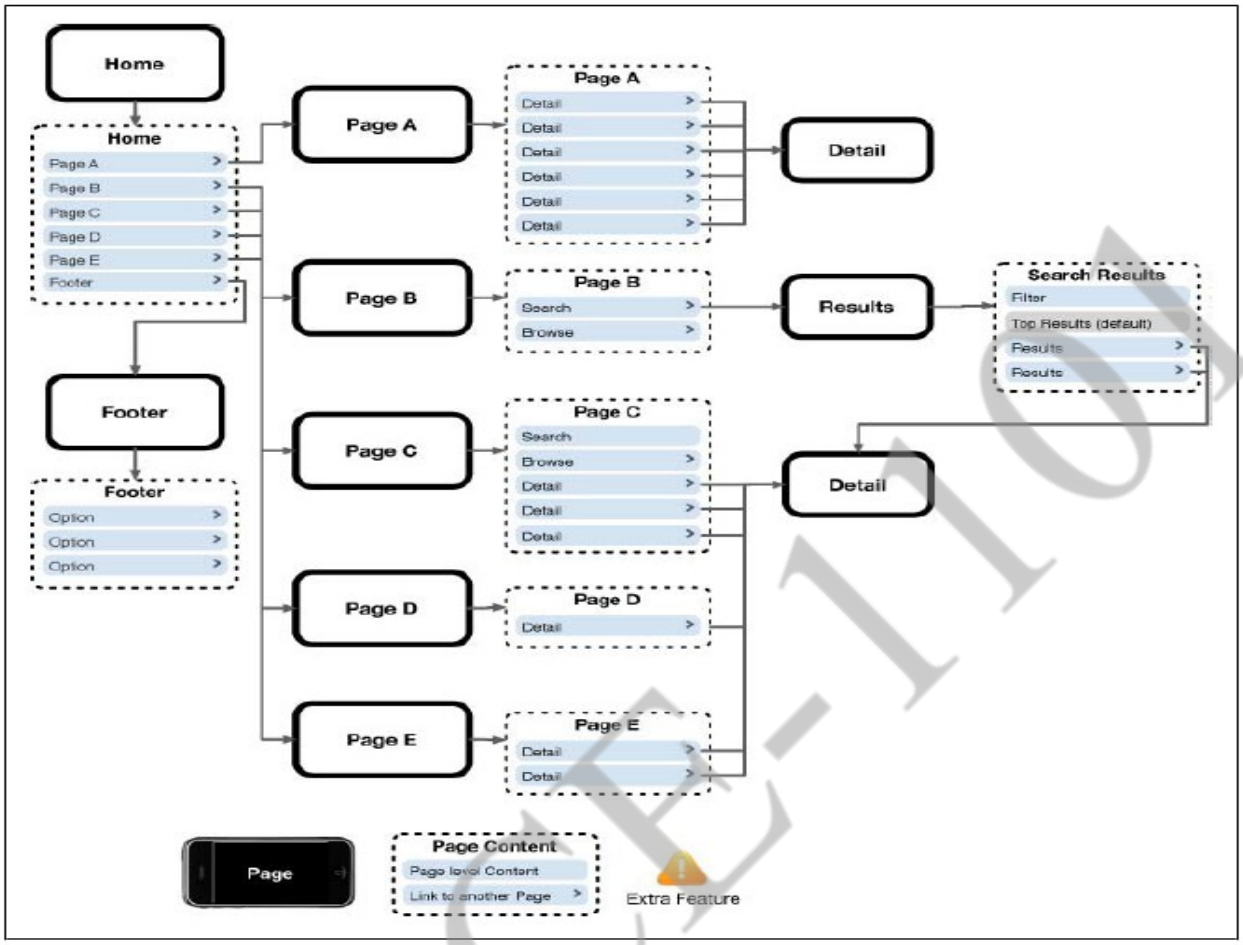


Fig 4.5An example clickstream for an iPhone web application

The esoteric diagram shown in Figure 4.6, which is made up of boxes and diamonds that look more like circuit board diagrams than an information architecture.

A good architect's job is to create a map of user goals, not map out every technical contingency or edge case. Too often, process flows go down a slippery slope of adding every project requirement, bogging down the user experience with unnecessary distractions, rather than focusing on streamlining the experience. In mobile, our job is to keep it as simple as possible. We need to have an unwavering focus on defining an excellent user experience first and foremost.

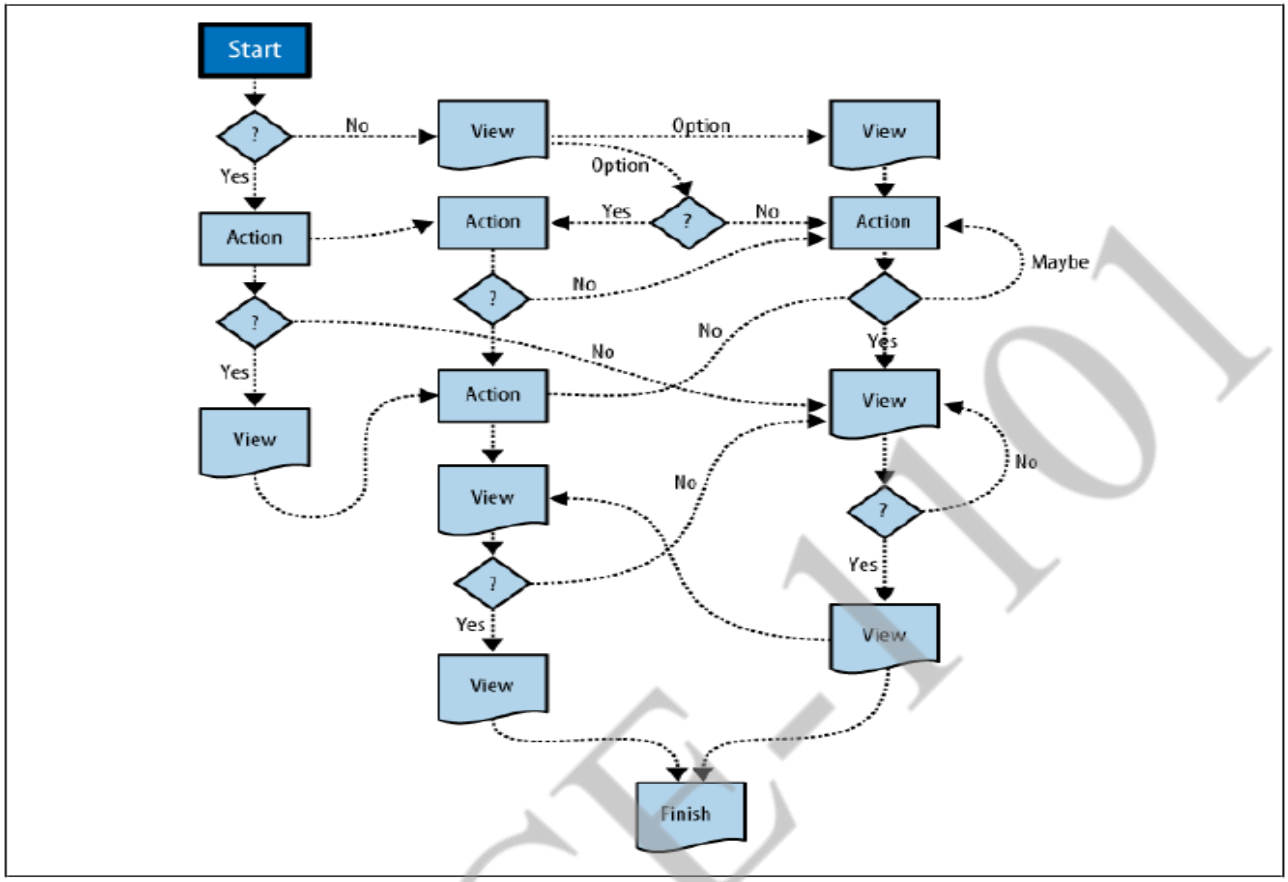


Fig 4.6 An example process flow diagram

3. **Wireframes**

The next information architecture tool is wireframes. *Wireframes* are a way to lay out information on the page, also referred to as *information design*. Site maps show how our content is organized in our informational space; wireframes show how the user will directly interact with it. Wireframes are like the peanut butter to the site map jelly in our information architecture sandwich. Wireframes serve to make information space tangible and useful.

But the purpose of wireframes is not just to provide a visual for our site map; they alsoserve to separate layout from visual design, defining how the user will interact with the experience. How do we lay out our navigation? What visual or interaction metaphors will we use to evoke action? What are the best ways to communicate and show information in the assumed context of the user? These questions and many more are answered with wireframes.

The challenge is that a diagram on a piece of paper doesn't go a long way toward describing how the interactions will work. Most common are "in-place" interactions or areas where the user can interact with an element without leaving the page. This can be done withAjax or a little show/hide JavaScript. These interactions can include large amounts of annotation, describing each content area in as much length as we can fit in the margins of the page, as shown in Figure 4.7.

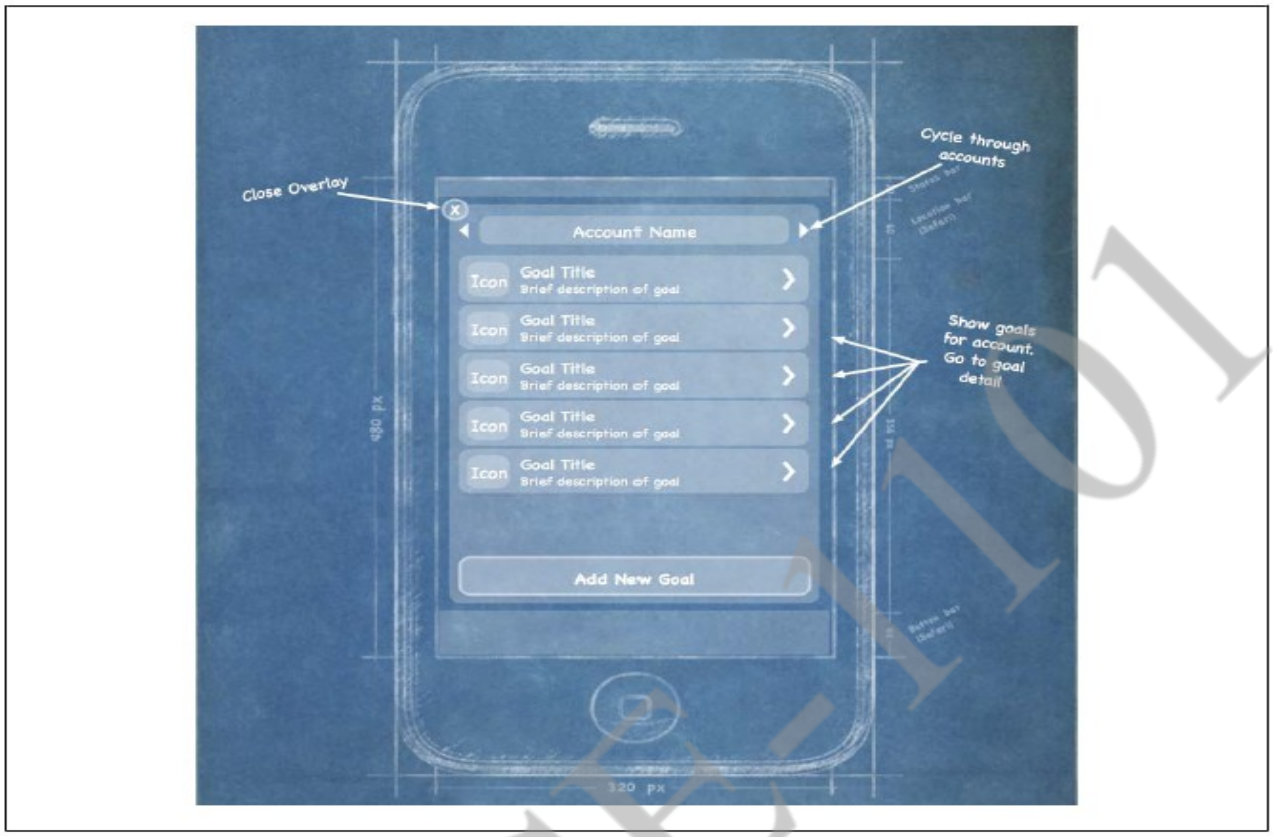


Fig 4.7 Using annotations to indicate the desired interactions of the site or application

Often we get an idea that we think is brilliant, but once we say it out loud, it just sounds absurd. In mobile, it is this kind of feedback, using wireframes as the key deliverable that turns good ideas into excellent mobile products.

4. **Prototyping**

Wireframes lack the capability to communicate more complex, often in-place, interactions of mobile experiences. This is where prototypes come in.

Prototypes might sound like a scary (or costly) step in the process. Some view them as redundant or too time-consuming, preferring to jump in and start coding things. But as with wireframes, each product we've built out some sort of prototype has saved both time and money. The following sections discuss some ways to do some simple and fast mobile prototyping.

5. **Paper prototypes**

The most basic level we have is paper prototyping: taking our printed-out wireframes or even drawings of our interface and putting them in front of people.

Create a basic script of tasks (hopefully based on either context or user input) and ask users to perform them, pointing to what they would do. We act as the device, changing the screens for them. For paper prototypes, try using small blank note cards, and for lower-end devices, use business card-sized paper. The size matters and we'll learn as much from how the user manages working with small media as we will what information is actually on it.

6. **Context prototype**

The next step is creating a context prototype. Take a higher-end device that enables to load full-screen images on it. Take wireframes or sketches and load them onto the device, sized to fill the device screen. Leave the office.

Go for a walk down to nearest café. Or get on a bus or a train. As you are traveling about, pull out your device and start looking at the interface in the various contexts you find yourself currently in.

Pay particular attention to what you are thinking and your physical behavior while you are using your interface and then write it down. If you are brave and don't have strict nondisclosure issues, ask the people around you to use it, too. Try to keep an eye on a clock to determine how long the average session is.

7. **HTML prototypes**

The third step is creating a lightweight, semifunctional static prototype using XHTML,CSS, and JavaScript, if available. This is a prototype that we can actually load onto a device and produce the nearest experience to the final product, but with static dummycontent and data. It takes a little extra time, but it is worth the effort.

With a static XHTML prototype, you use all the device metaphors of navigation, you see how much content will really be displayed on screen (it is always less than you expect), and you have to deal with slow load times and network latency. In short, you will feel the same pains your user will go through.

Whatever route you wish to take, building a mobile prototype takes you one very big leap forward to creating a better mobile experience. Different Information Architecture for Different Devices

Depending on which devices you need to support, mobile wireframes can range from the very basic to the complex. On the higher-end devices with larger screens, we might be inclined to add more interactions, buttons, and other clutter to the screen, but this would be a mistake. Just because the user might have a more advanced phone, that doesn't mean that he is giving you license to pack his screen with as much informationas you can muster.

The motivations, goals, and how users will interact with a mobile experience are the same at the low end as they are on a high-end device. For the latter, you just have better tools to express the content. You can learn a lot from designing for the lower end first.

The greatest challenge in creating valuable experiences is knowing when to lose what you don't need. You don't have a choice on lower-end devices—it must be simple.

When designing for both, it is best to try and to keep your information architecture as close to each other as possible without sacrificing the user experience. They say that simple design is the hardest design, and this principle certainly is true when designing information architecture for mobile devices.

The Design Myth

A little secret about interactive design is that people don't respond to the visual aestheticas much. What colors you use, whether you use square or rounded corners, or, gradients or flat backgrounds, helps build first impressions, but it doesn't do too much to improve the user's experience. Users appreciategood design, but they are quickly indifferent about the visual aesthetic and move almostimmediately to the layout (information design), what things are called (taxonomy), thefindability of content, and how intuitive it is to perform tasks. These are all facets ofinformation architecture.

Just look at one of the top-selling iPhone Twitter applications, Tweetie. Many consider Tweetie to be a "well-designed" application, but because it is built from the same API as all other iPhone applications, at first glance there is little that is actually visually distinctive between this and other applications. What makes this application "well designed" is how the content is applied to the context of the user—in other words, the mobile information architecture. The point is great information design is often mistaken for great visual design.

Most non–information architects almost always do information architecture in someform or another; often, they don't even know they are doing it. They might do a few wireframes, or maybe a site map. Sometimes designers will jump in and incorporate information architecture deliverables directly into their designs. By not focusing on the information architecture exclusively from the start, we risk confusing our disciplines, our deliverables, and ultimately our direction. The more time we spend focusing on just information architecture, the faster and less costly the project will be.

4.4 Mobile 2.0:

What Is Mobile 2.0?

Mobile 2.0, refers to a perceived next generation of **mobile** internet services that leverage the social web, or what some call Web **2.0**. The social web includes social networking sites and wikis that emphasise collaboration and sharing amongst users.

Mobile 2.0: The Convergence of the Web and Mobile

It is obvious that in the minds of many, Mobile 2.0 is the Web. At this point, the mobile web has always been viewed as a second-class citizen within the mobile ecosystem, for many reasons, as discussed later.
Mobile is already a medium, but the consensus is that by leveraging the power of theWeb, integrating web services into the mobile medium is the future of mobile development.

When the iPhone exploded onto the scene, it increased the usage of the mobile web by its users to levels never seen before. The spur of new mobile web apps created just for the iPhone doubled the number of mobile websites available in under a year.

If Web 2.0 taught us that the Web is the platform, then Mobile 2.0 tells us that mobile will be the primary context in which we leverage the Web in the future.

The Mobile Web Browser As the Next Killer App

If the future of mobile is the Web, then it only makes sense that the mobile web browser is the next killer app of mobile. This is something we saw confirmed with WebKitin the iPhone and later in Android.
However, of particular concern is how device fragmentation factors into mobile browsers.

For example, how can we expect developers to support more than 30 different mobile browsers? A fellow panelist from the Mozilla Minimo project offered a potential solution in consolidation—that we will see only a few mobile browsers in the future; specifically, browsers built on Mozilla, Opera, Internet Explorer, and WebKit technologies.

The line between smartphone and feature phone seems to be going away, so this prediction is fairly accurate.

But the single biggest challenge in mobile remains device fragmentation. The mobile browser enables us to penetrate the problem by not having our content locked so specifically to the device abilities, screen size, and form factor, but device fragmentation still causes old, outdated browsers to remain in the market long after they should be put out to pasture.

What appears to be solving browser fragmentation is actually the iPhone. The MobileSafari browser included with the iPhone provided such an excellent web experience on a mobile device that it drove use of the mobile web to huge levels, which means big profits for the operators. This also means that the mobile web is no longer a second class citizen. In the post-iPhone market, all new devices are judged by the quality of their mobile web browser. Operators know it and therefore are demanding better browsers from device makers and browser makers.

Mobile Web Applications Are the Future

Creating mobile web applications instead of mobile software applications has remained an area of significant motivation and interest. The mobile community is looking at theWeb 2.0 revolution for inspiration, being able to create products and get them to market quickly and at little cost. They see the success of small iterative development cycles and want to apply this to mobile development, something that is not that feasible in the traditional mobile ecosystem.

Developers have been keen for years to shift away from the costly mobile applications that are difficult to publish through the mobile service provider, require massive testing cycles and costly porting to multiple devices, and can easily miss the mark with users after loads of money have been dumped into them.The iPhone App Store and the other mobile device marketplaces have made it far easier to publish and sell, but developers still have to face difficult approval processes, dealing with operator and device maker terms and porting challenges.

Mobile software has two fundamental problems that mobile **web applications solve**.

The **first** is forcing users through a single marketplace. We know from years of this model that an app sold through a marketplace can earn huge profits if promoted correctly.

Being *promoted correctly* is the key phrase. The companies that know how to work the system are the ones that get the big prizes, making it increasingly hard for the small developer to see any kind of success. But the mobile web provides any size of developer with the ability to promote and distribute their app on their terms, building a relationship directly with their customers and not by proxy.

The **second** problem is the ability to update your application. It is certainly possible on modern marketplaces like the App Store, but we are still years from that being the norm.

Mobile web apps enable you to make sure that you never ship a broken app, or if yourapp breaks in the future due to a new device, to be able to fix it the same day the device hits the street. This flexibility isn't possible in the downloaded app market.

JavaScript Is the Next Frontier

If we are going to provide mobile web applications, we have to have a mobile web browser that supports Ajax, or, XMLHttpRequest. It makes a lot of those cool interactions in web browser work via the capability to load content asynchronously into browser view.

But it isn't just Ajax; it is JavaScript, a web technology that has largely been ignored with most mobile web browsers. Ajax is great, but just being able to do a little show/ hide or change a style after click or touch it goes a long way toward improving the user experience.

This is probably where mobile web browsers fall behind desktop browsers the most.

Because they both support XHTML and CSS relatively well, JavaScript has been a nogoin mobile for years. In order for mobile web apps to rival native applications, wehave to support some JavaScript.

Modern mobile browsers have made much progress over the last few years, but there is still plenty of work to be done. For example, accessing the device capabilities like the phone book or filesystem with JavaScript doesn't work in a consistent way. These problems still need to be solved in order to truly reap the benefits of the Web.

Rich interactions kill battery life

JavaScript and Ajax have been ignored because using an Ajax-based web application on phone can drain battery at a rate of four to five times normal power consumption.The two most prevalent reasons are:

• JavaScript consumes more processor power and therefore more battery life.

• Ajax apps fetch more data from the network, meaning more use of the radio and more battery life.

Unless we are in the habit of carrying around a bunch of extra batteries, expect to charge phone every hour or two as a penalty for using the modern mobile web.

Apple and the open source WebKit browser have made huge strides by releasing aJavaScript engine that is incredibly efficient on mobile devices, though the other big mobile browser technologies aren't far behind. This problem is going away quickly as the mobile browsers get better, batteries improve efficiency, and devices get more powerful.

The Mobile User Experience Is Awful

Traditionally, the user experience available in the mobile web has been like using a website from 1995: mostly text-based, difficult to use, and ugly as sin. This isn't to say that the user experience of mobile applications has been much better, but it used to be that if wanted a good experience, build a native app.

Mobile user experience was largely ignored for close to a decade. People in mobile treat the user experience like a chicken-and-egg scenario: bad input/output of the user experience prevents adoption, but designing a shiny user experience with bells and whistles will bring them in droves.

Device APIs usually force to use their models of user experience, meaning that wehave to work in the constraints of the API. This is good for creating consistent experiences for that platform, but these experiences don't

translate to others. For example, we cannot take an iPhone app design and put it on an Android device. The user experience for these devices is similar but still remains different.

The beauty of the Web, literally, is that we can design whatever experience we want, for better or worse. We are in control of the presentation and can establish own visual metaphors. The problem has been that traditionally complex (which often equates to good) user experiences haven't been possible on mobile devices. Modern mobile web browsers, as they come closer to their desktop counterparts, remove this distinction, giving us the same canvas on mobile devices that we have for the desktop.

This means that creating mobile experiences just got a whole lot easier. It also means we can have a consistent user experience across multiple mediums.

Mobile Widgets Are the Next Big Thing

The challenges with the mobile web is to create a series of "small webs" targeted at a specific user or task. Though I couldn't figure out the problem being solved with these widgets,

Carrier Is the New "C" Word

It is clear that one of the key drivers of Mobile 2.0 and the focus on the mobile web is to find a way to build a business that doesn't rely on carrier control.

Mobile Needs to Check Its Ego

The mobile community and the web communityhave treated each other almost like rivals. It is the mobile camp that needs to check their egos at the door and get into the game, before they learn that all the rules have changed.

On the mobile side, we have some incredibly intelligent people who have been innovating amazing products under insane constraints for years. On the web side, we have creative amateurs who have helped build a community and ecosystem out of passion and an openness to share information.

The web guys want to get into the game and move the medium forward, partly out of desire open up a new market for themselves, but mostly out of passion for all things interactive. But, to the mobile community, they are seen as a threat to expertise. On the other hand, to the web community, the mobile guys come off as overly protective, territorial, selfish, and often snobbish or egotistical, effectively saying, "Go away."

Unless the mobile community comes together with the web community by sharing information, experience, and guidance, one day they will find that their experience has become obsolete. In return, the web guys will share their enthusiasm, willingness to learn, and passion that many in mobile development have forgot.

It's that one principle of Web 2.0 that the mobile community has left out: harnessing collective intelligence. The Web and the mobile community are reaching a point where the two worlds can no longer afford not to be working together, sharing what they know and harnessing the collective intelligence of both media.

We Are Creators, Not Consumers

The final principle of Mobile 2.0 is recognizing that we are in a new age of consumerism.

Yesterday's consumer does not look anything like today's consumer. The people of today's market don't view themselves as consumers, but rather as creators.

The web is about content. Sure, there are programming languages, APIs, and other technical underpinnings, but what we do when we open a web browser? We read.

Our primary task online is to read, to gain information. During the early days of the Web, it took tools and know-how in order to publish to the Web. But early in the Web2.0 evolution, we saw a rise in tools that allowed us to publish to the Web easily, giving individuals a voice online, with a massive audience.

This democratization of the Web took many forms that some call "social media," like blogging, social networks, media sharing, microblogging, and lifestreams. Although social media may have many facets, they all share the same goal: to empower normal, everyday people to become creators and publishers of content. It started with the written word, then music, then photos, and more recently video was added. Entire markets have been

created to provide today's consumer with gadgets, software, and web services to record and publish content so that we can share it with our friends and loved ones.

At the center of this revolution in publishing is the mobile device. As networked portable devices become more powerful, allowing us to capture, record, and share content in the moment, we are able to add a new kind of context to content—the likes of which we haven't seen since satellite television. Now you can share any moment with any group of people in real time. Think about how powerful a concept that is! It could change entire cultures.

Tony Fish, coauthor of *Mobile Web 2.0* (futuretext), says:

When everyone has the tools to create content, in addition to zero-cost publishing, we do not consume content, we create it.

The early "Web 1.0" days clearly changed how business is done, because businesses are the primary consumer of desktop computers. It probably is no coincidence that Web 2.0 occurred around the same time that laptop computers became affordable for the average person, making the Web a more personal medium.

With Mobile 2.0, the personal relevance of the content matches how personal the device is and how personally it applies to our everyday situations or our context.

4.5 Mobile Design:

4.5.1 The Elements of Mobile Design

The good design requires **three abilities**: the first is a natural gift for being able to see visually how something should look that produces a desired emotion with the target audience. The second is the ability to manifest that vision into something for others to see, use, or participate in. The third is knowing how to utilize the medium to achieve our design goals.

To think like Mobile designerinvolves knowing the **six elements of mobile design** that we need to consider, starting with the context and layering in visual elements or laying out content to achieve the design goal. Then, we need to understand how to use the specific tools to create mobile design, and finally, need to understand the specific design considerations of the mobile medium.

1. **Context**

Context is core to the mobile experience. As the designer, it isthe job to make sure that the user can figure out how to address context using app. Some context based questions:

• Who are the users? What do you know about them? What type of behavior can you assume or predict about the users?

• What is happening? What are the circumstances in which the users will best absorb the content you intend to present?

• When will they interact? Are they at home and have large amounts of time? Are they at work where they have short periods of time? Will they have idle periods of time while waiting for a train, for example?

• Where are the users? Are they in a public space or a private space? Are they inside or outside? Is it day or is it night?

• Why will they use your app? What value will they gain from your content or services in their present situation?

• How are they using their mobile device? Is it held in their hand or in their pocket?

How are they holding it? Open or closed? Portrait or landscape?

The answers to these questions will greatly affect the course of design. Treat these questions as a checklist to the design from start to finish. They can provide not only great inspiration for design challenges, but justification for design decisions later.

2. **Message**

Another design element is message, or what we are trying to say about site or application visually. One might also call it the "branding,".The message is the overall mental impression we create explicitly through visual design. Branding shouldn't be confused with messaging. Branding is the impression company name and logo gives—essentially, reputation. Branding serves to reinforce the message with authority, not deliver it. In mobile, the opportunities for branding are limited, but the need for messaging is great. With such limited real estate, the users don't care about brand, but they will care about the messaging, asking themselves questions like, "What can this do for me?" or "Why is this important to me?"

The approach to the design will define that message and create expectations. A sparse, minimalist design with lots of whitespace will tell the user to expect a focus on content.

A "heavy" design with use of dark colors and lots of graphics will tell the user to expect something more immersive.

3. **Look and Feel**

The concept of "look and feel" is an odd one, being subjective and hard to define.

Typically, look and feel is used to describe appearance, as in "I want a clean look and feel" or "I want a usable look and feel." The problem is: as a mobile designer, what does it mean? And how is that different than messaging? Look and feel in a literal sense, as something real and tactile that the users can "look" at, then "feel"—something they can touch or interact with. Look and feel is used to evoke action—how the user will use an interface. Messaging is holistic, as the expectation the users will have about how you will address their context. It is easy to confuse the two, because "feel" can be interpreted to mean our emotional reaction to design and the role of messaging.

Establishing a look and feel usually comes from wherever design inspiration comes from. However, personal inspiration can be a hard thing to justify. Therefore we have "design patterns," or documented solutions to design problems, sometimes referred to as style guides. On large mobile projects or in companies with multiple designers, a style guide or pattern library is crucial, maintaining consistency in the look and feel and reducing the need for each design decision to be justified.

Although a lot of elements go into making Apple's App Store successful, the most important design element is how it looks and feels. Apple includes a robust user interface tool that enables developers to use prebuilt components, supported with detailedHuman Interface Guidelines (or HIG) of how to use them, similar to a pattern library.

This means that a developer can just sit down and create an iPhone application that looks like it came from Apple in a matter of minutes. During the App Store submission process, Apple then ensures that the developer uses these tools correctly according to the HIG.

The look and feel can either be consistent with the stock user interface elements thatApple provides; they can be customized, often retaining the "spirit" of Apple's original design; or an entirely new look and feel can be defined—this approach is often used for immersive experiences.

The stock user experience that Apple provides is a great example of how look and feel works to supporting messaging. For the end user, the design sends a clear message: by using the same visual interface metaphors that Apple uses throughout the iPhone, we can expect the action, or how this control will behave, but we can also expect the same level of quality. This invokes the message of trust and quality in the application and in the platform as a whole. Apple isn't the first to use this shared look and feel model in mobile—in fact, it is incredibly common with most smartphone platforms—but they are surely making it incredibly successful, with a massive catalog of apps and the sales to support it.

Mobile designers haveto be creative and remember the context. Like in the early days of the Web, people tend to be skeptical about mobile experiences. The modal context of the user—in this case, what device he is using —should be considered during the design, as it will help to establish the user's expectations of the experience.

4. **Layout**

Layout is an important design element, because it is how the user will visually processthe page, but the structural and visual components of layout often get merged together, creating confusion and making design more difficult to produce.

The first time layout should rear its head is during information architecture.

Why define the layout before the mobile design? Design is just too subjective of anissue. If we are creating a design for anyone but our self, chances are good that there will be multiple loosely-based-on-experience opinions that will be offered and debated.

There is no right answer—only opinions and gut instincts. Plus, in corporate environments we have internal politics we have to consider, where the design opinions of theCEO or Chief Marketing Officer (CMO) might influence a design direction more than, say, the Creative Director or Design Director.

By defining design elements like layout prior to actually applying the look and feel, wecan separate the discussion. The majority of comments that reviewers would make were about the layout. They focus on the headers, the navigation, the footer, or how content blocks are laid out, and so on. But their feedback will get muddied with the "look and feel, the colors, and other design elements."

Reviewers do make remarks like "I like the navigation list, but can you make it look more raised?" Most designers don't hear that; they hear "The navigation isn't right, do it again." But, with this kind of feedback, there are two important pieces of information about different types of design. First, there is confirmation that the navigation and layout are correct. Second, there is a question about the "look and feel." Because designers hear "Do it again," they typically redo the layout, even though it was actually fine.

Creating mobile designs in an environment with multiple reviewers is all about gettingthe right feedback at the right time. Our job is to create a manifestation of a shared vision. Layout is one of the elements we can present early on and discuss independently.

People confuse the quality and fidelity of deliverables as design. By keeping it basic, we don't risk having reviewers confuse professionalism with design.

Different layouts for different devices

The second part of layout design is how to visually represent content. In mobile design, the primary content element we deal with is the navigation. Whether we are designing a site or app, you need to provide users with methods of performing tasks, navigating to other pages, or reading and interacting with content. This can vary, depending on the devices we support.

There are **two distinct types** of navigation layouts for mobile devices: **touch and scroll**.

With touch, we literally point to where we want to go; therefore, navigation can beanywhere on the screen. But we tend to see most of the primary actions or navigation areas living at the bottom of the screen and secondary actions living at the top of thescreen, with the area in between serving as the content area, like what is shown inFigure 4.8.

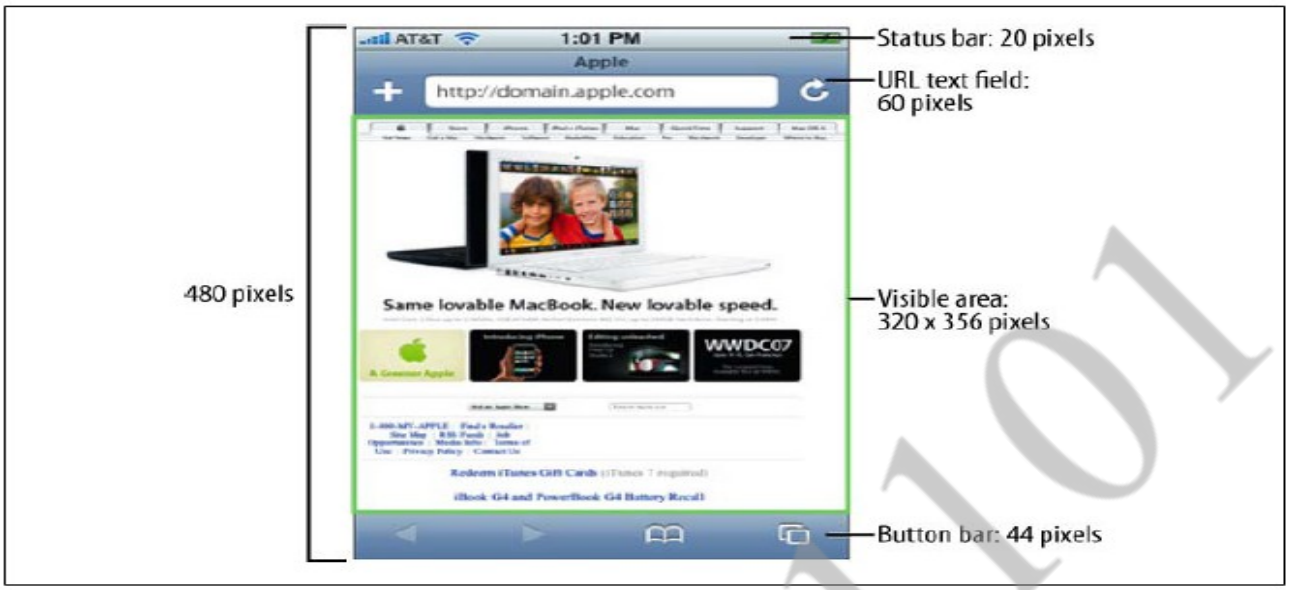


Fig 4.8 iPhone HIG, showing the layout dimensions of Safari on the iPhone

This is the opposite of the scroll navigation type, where the device's D-pad is used togo left, right, up, or down. When designing for this type of device, the primary and often the secondary actions should live at the top of the screen. This is so the user doesn't have to press down dozens of times to get to the important stuff. In Figure 4.9, we can actually see by the bold outline that the first item selected on the screen is the link around the logo.

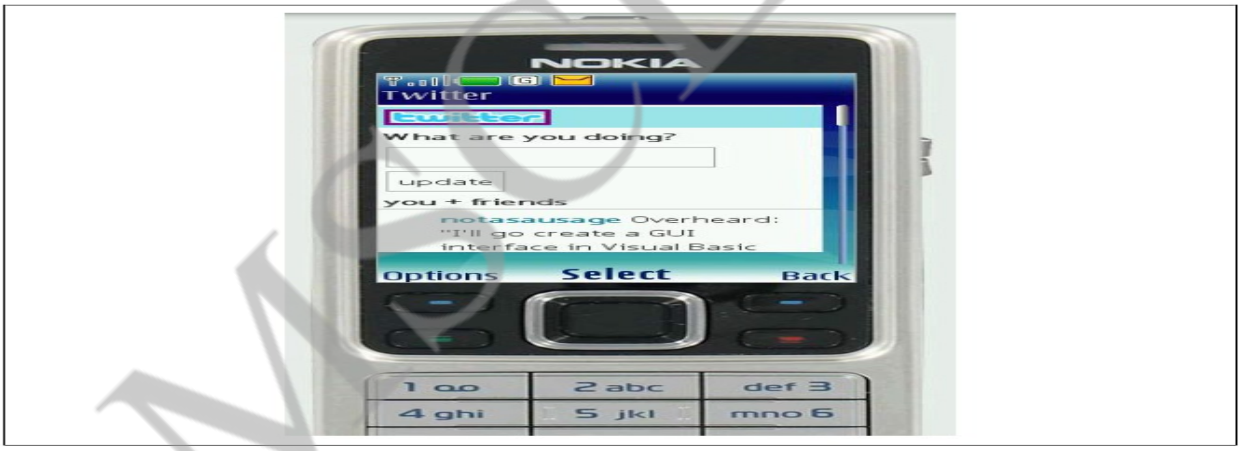


Fig 4.9 Example layout of a scroll based application, where the user had to press the D-pad past each link to scroll the page

When dealing with scroll navigation, we also have to make the choice of whether to display navigation horizontally or vertically. Visually, horizontally makes a bit more sense, but when we consider that it forces the user to awkwardly move left and right, it can quickly become a bit cumbersome for the user to deal with. There is no right or wrong way to do it, but try and keep it as simple as possible.

Fixed versus fluid

Another layout consideration is how design will scale as the device orientation changes, for example if the device is rotated from portrait mode to landscape and vice versa. This is typically described as either being fixed (a

set number of pixels wide), or fluid (having the ability to scale to the full width of the screen regardless of the device orientation).

Orientation switching has become commonplace in mobile devices, and designshould always provide the user with a means to scale the interface to take full advantage of screen real estate.

5. Color

The fifth design element, color, is hard to talk about in a black-and-white book. Maybe it is fitting, because it wasn't that long ago those mobile screens were available only in black and white (well, technically, it was black on a green screen). These days, we havenearly the entire spectrum of colors to choose from for mobile designs.

The most common obstacle we encounter when dealing with color is mobile screens,which come in a number of different color or bit depths, meaning the number of bits (binary digits) used to represent the color of a single pixel in a bitmapped image. When complex designs are displayed on different mobile devices, the limited color depth on one device can cause banding, or unwanted posterization in the image.

Different devices have different color depths. In Table 4.1, we can see the supported colors and a few example devices.

Bit depth	Supported colors	Description	Example devices
12-bit	4,096 colors	Used with older phones; dithering artifacts in photos can easily be seen.	Nokia 6800
16-bit	65,536 colors	Also known as HighColor; very common in today's mobile devices. Can cause some banding and dithering artifacts in some designs.	HTC G1, BlackBerry Bold 9000, Nokia 6620
Bit depth	**Supported colors**	**Description**	**Example devices**
18-bit	262,144 colors	Used in mobile devices to offer Truecolor (see following entry) levels through dithering. Limited banding may be seen.	Samsung Alias, Sony Ericsson TM506
24-bit	16.7 million colors	Also known as Truecolor; supports millions of colors and produces little banding.	iPhone, Palm Prē, Nokia N97

Table 4.1 Supported Colors and Example Devices

The psychology of color

People respond to different colors differently. It is fairly well known that different colors produce different emotions in people, but surprisingly few talk about it outside of art school. Thinking about the emotions that colors evoke in people is an important aspect of mobile design, which is such a personal medium that tends to be used in personalways. Using the right colors can be useful for delivering the right message and setting expectations.

One of the examples used earlier was the ESPN mobile site, which uses a bold red header to create a stark and prominent tone to the design. But what does that say about
ESPN? What does it tell the user about the experience?

For the purposes of reference, Table 4.2provides some of the characteristics of various colors that naturally evoke certain emotions in people.

Color	Represents
White	Light, reverence, purity, truth, snow, peace, innocence, cleanliness, simplicity, security, humility, sterility, winter, coldness, surrender, fearfulness, lack of imagination, air, death (in Eastern cultures), life, marriage (in Western cultures), hope, bland
Black	Absence, modernity, power, sophistication, formality, elegance, wealth, mystery, style, evil, death (in Western cultures), fear, seriousness, conventionality, rebellion, anarchism, unity, sorrow, professionalism
Gray	Elegance, humility, respect, reverence, stability, subtlety, wisdom, old age, pessimism, boredom, decay, decrepitude, dullness, pollution, urban sprawl, strong emotions, balance, neutrality, mourning, formality
Yellow	Sunlight, joy, happiness, earth, optimism, intelligence, idealism, wealth (gold), summer, hope, air, liberalism, cowardice, illness (quarantine), fear, hazards, dishonesty, avarice, weakness, greed, decay or aging, femininity, gladness, sociability, friendship
Green	Intelligence, nature, spring, fertility, youth, environment, wealth, money (U.S.), good luck, vigor, generosity, go, grass, aggression, coldness, jealousy, disgrace (China), illness, greed, drug culture, corruption (North Africa), life eternal, air, earth (classical element), sincerity, renewal, natural abundance, growth
Blue	Seas, men, productiveness, interiors, skies, peace, unity, harmony, tranquility, calmness, trust, coolness, confidence, conservatism, water, ice, loyalty, dependability, cleanliness, technology, winter, depression, coldness, idealism, air, wisdom, royalty, nobility, Earth (planet), strength, steadfastness, light, friendliness, peace, truthfulness, love, liberalism (U.S. politics), and conservatism (UK, Canadian, and European politics)
Violet	Nobility, envy, sensuality, spirituality, creativity, wealth, royalty, ceremony, mystery, wisdom, enlightenment, arrogance, flamboyance, gaudiness, mourning, exaggeration, profanity, bisexuality, confusion, pride
Red	Passion, strength, energy, fire, sex, love, romance, excitement, speed, heat, arrogance, ambition, leadership, masculinity, power, danger, gaudiness, blood, war, anger, revolution, radicalism, aggression, respect, martyrs, conservatism (U.S. politics), Liberalism (Canadian politics), wealth (China), and marriage (India)
Orange	Energy, enthusiasm, balance, happiness, heat, fire, flamboyance, playfulness, aggression, arrogance, gaudiness, overemotion, warning, danger, autumn, desire
Pink	Spring, gratitude, appreciation, admiration, sympathy, socialism, femininity, health, love, romance, marriage, joy, flirtatiousness, innocence and child-like qualities
Brown	Calm, boldness, depth, nature, richness, rustic things, stability, tradition, anachronism, boorishness, dirt, dullness, heaviness, poverty, roughness, earth

Table 4.2 Color Characteristics

Note what some of the different colors can mean in different cultures. In some cases,the color we use can have opposing meanings in different cultures. This is something to consider when thinking of deploying mobile experience to countries with the highest number of mobile devices, such as China or India.

Color palettes

Defining color palettes can be useful for maintaining a consistent use of color in mobile design. Color palettes typically consist of a predefined number of colors to use throughout the design. Selecting what colors to use varies from designer to designer, each having different techniques and strategies for deciding on the colors. **Three** basic ways to define a color palette:

i. ***Sequential***

In this case, there are primary, secondary, and tertiary colors. Often the primary color is reserved as the "brand" color or the color that most closely resembles the brand's meaning. The secondary and tertiary colors are often complementary colors that can be selected using a color wheel.

ii. ***Adaptive***

An adaptive palette is one in which we leverage the most common colors present in a supporting graphic or image. When creating a design that is meant to look native on the device, an adaptive palette can be usedto make sure that colors are consistent with the target mobile platform.

iii. ***Inspired***

This is a design that is created from the great pieces of design we might see onlineor offline, in which a picture of the design might inspire us. This could be anything from an old poster in an alley, a business card, or some packaging. Like with the adaptive palette, we actually extract the colors from the source image, though we should never ever use the source material in a design.

Typography

The sixth element of mobile design is typography, which in the past would bring tomind the famous statement by ***Henry Ford:***

Any customer can have a car painted any color that he wants so long as it is black.

Traditionally in mobile design, you had only one typeface that we could use, and that was the device font. The only control over the presentation was thesize.

As devices improved, so did their fonts. Higher-resolution screens allowed for a morerobust catalog of fonts than just the device font. First, let's understand how mobile screens work.

Subpixels and pixel density

There seem to be two basic approaches to how type is rendered on mobile screens: using subpixel-based screens or having a greater pixel density or pixels per inch (PPI).

A subpixel is the division of each pixel into a red, green, and blue (or RGB) unit at a microscopic level, enabling a greater level of antialiasing for each font character or glyph. The addition of these RGB subpixels enables the eye to see greater variations of gray, creating sharper antialiasing and crisp text.

In Figure 4.10, you can see three examples of text rendering. The first line shows a simple black and white example, the second shows text with grayscale antialiasing, and the third line shows how text on a subpixel display would render.

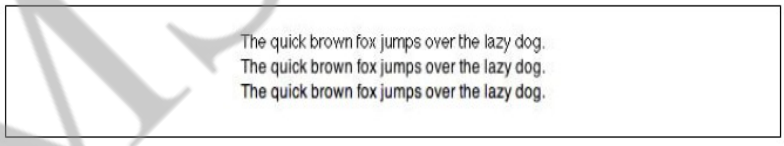


Fig 4.10 Different ways text can render on mobile screens

The Microsoft Windows Mobile platform uses the subpixel technique with its Clear-Type technology, as shown in Figure 4.11.

Fig 4.11 Microsoft Clear-Type using sub pixels to sharp text

The second approach is to use a great pixel density, or pixels per inch. We often refer to screens by either their actual physical dimensions or their pixel dimensions, or resolution. The pixel density is determined by dividing the width of the display area in pixels by the width of the display area in inches. So the pixel density for15.4-inchlaptop would be 110 PPI. In comparison, a 1080p HD television has a PPI of 52.

As this applies to mobile devices, the higher the density of pixels, the sharper the screenappears to the naked eye. This guideline especially applies to type, meaning that as text is antialiased on a screen with a high density of tiny pixels, the glyph appears sharper to the eye. Some mobile screens have both a high PPI and subpixel technology, though these are unnecessary together.

Table 4.3 provides the dimensions and PPI for a few mobile devices.

Mobile device	Diagonal	Pixels	PPI
Nokia N95	2.6"	240×320	153
Apple iPhone 3G	3.5"	320×480	163
Amazon Kindle	6.0"	600×800	167
HTC Dream	3.2"	320×480	181
Sony Ericsson W880i	1.8"	240×320	222
Nokia N80	2.1"	352×416	256

Table 4.3 Dimensions and PPI for some Mobile Devices

Type options

Fortunately, today's mobile devices have a few more options than a single typeface, butthe options are still fairly limited. Coming from web design, where we have a dozen or so type options, the limited choices available in mobile design won't come as a big surprise. Essentially, we have a few variations of serif, sans-serif, and monospace fonts, and depending on the platform, maybe a few custom fonts (Figure).

Fig 4.12 Options in Typography increase as the device become more sophisticated

Therefore, when creating mobile designs for either web or native experiences, stick with either the default device font, or web-safe fonts—the basic serif variants like Times New Roman and Georgia or sans-serif typefaces like Helvetica, Arial, orVerdana.

Font replacement

The ability to use typefaces that are not already loaded on the device varies from model to model and chosen platform. Some device APIs will allow to load a typeface into native application. Some mobile web browsers support various forms of fontreplacement; the two most common are sIFR and Cufon. sIFR uses Flash to replaceHTML text with a Flash representation of the text, but the device ofcourse has to support Flash. Cufon uses JavaScript and the canvas element draws the glyphs in the browser, but the device of course needs to support both JavaScript and the canvas element.

In addition, the @font-face CSS rule allows for a typeface file to be referenced and loaded into the browser, but a license for web use is usually not granted by type foundries.

Readability

The most important role of typography in mobile design is to provide the user with excellent readability, or the ability to clearly follow lines of text with the eye and not lose one's place or become disoriented, as shown in Figure 4.13. This can be done byfollowing these six simple rules:

Fig 4.13 Classics, an iPhone application designed with readability and typography in mind

i. ***Use a high-contrast typeface***

Remember that mobile devices are usually used outside. Having a high-contrast typeface with regard to the background will increase visibility and readability.

ii. ***Use the right typeface***

The type of typeface you use tells the user what to expect. For example, a sans-serif font is common in navigation or compact areas, whereas serif typefaces come in handy for lengthy or dense content areas.

iii. ***Provide decent leading (rhymes with "heading") or line spacing***

Mobile screens are often held 10–12" away from the eye, which can make tracking each line difficult. Increase the leading to avoid having the users lose their place.

iv. ***Leave space on the right and left of each line; don't crowd the screen***

Most mobile frameworks give you full access to the screen, meaning that you normally need to provide some spacing between the right and left side of the screen's edge and text—not much, typically about three to four character widths.

v. ***Generously utilize headings***

Break the content up in the screen, using text-based headings to indicate to the user what is to come. Using different typefaces, color, and emphasis in headings can also help create a readable page.

vi. ***Use short paragraphs***

Like on the Web, keep paragraphs short, using no more than two to three sentences per paragraph.

6. **Graphics**

The final design element is graphics, or the images that are used to establish or aid avisual experience. Graphics can be used to supplement the look and feel, or as content displayed in line with the text.

For example, in Figure 4.14, we can see Ribot's Little Spender application for the iPhone and the S60 platform. The use of graphical icons in the iPhone experience helps to establish a visual language for the user to interact with to quickly categorize entries.

On the S60 application, the wallet photo in the upper-right corner helps communicate the message of the application to the user.

Fig 4.14 Ribot's Little Splendor application uses graphics to define the experience

Iconography

The most common form of graphics used in mobile design is icons. Iconography is useful to communicate ideas and actions to users in a constrained visual space. The challenge is making sure that the meaning of the icon is clear to the user. For example, looking at Figure 4.15, we can see some helpful icons that clearly communicate an idea and some perplexing icons that leave us scratching our head.

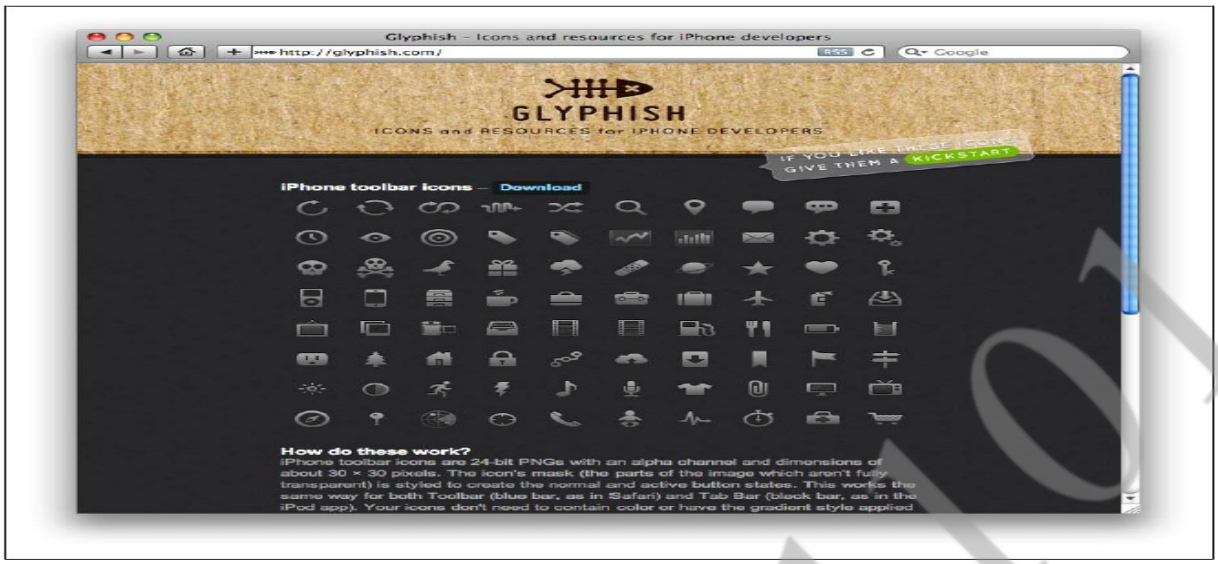


Fig 4.15 Glyphish provides free iPhone icons

Photos and images

Photos and images are used to add meaning to content, often by showing a visual display of a concept, or to add meaning to a design. Using photos and images isn't as common in mobile design as we might think. Because images have a defined height and width, they need to be scaled to the appropriate device size, either by the server, **using a content adaptation model**, or **using the resizing properties of the device**. In the latter approach, this can have a cost in performance. Loading larger images takes longer and therefore costs the user more.

Using graphics to add meaning to a design can be a useful visual, but we can encounter issues regarding how that image will display in a flexible UI—for example, when the device orientation is changed. In Figure 4.16, we can see how the pig graphic is designedto be positioned to the right regardless of the device orientation.

Fig 4.16 Using Graphics in Multiple Design Orientations

4.5.2 Mobile Design Tools

Mobile design requires understanding the design elements and specific tools. The closest thing to a common design tool is Adobe Photoshop, though each framework has a different method of implementing the design into the application.

Some frameworks provide a complete interface toolkit, allowing designers or developers to simply piece together the interface, while others leave it to the designer to define from scratch.

In Table 4.4, we can see each of the design tools and what interface toolkits are available for it.

Mobile framework	Design tool	Interface toolkits
Java ME	Photoshop, NetBeans	JavaFX, Capuchin
BREW	Photoshop, Flash	BREW UI Toolkit, uiOne, Flash
Flash Lite	Flash	Flash Lite
iPhone	Photoshop, Interface Builder	iPhone SDK
Android	Photoshop, XML-based themes	Android SDK
Palm webOS	Photoshop, HTML, CSS, and JavaScript	Mojo SDK
Mobile web	Photoshop, HTML, CSS, and JavaScript	W3C Mobile Web Best Practices
Mobile widgets	Photoshop, HTML, CSS, and JavaScript	Opera Widget SDK, Nokia Web Runtime
Mobile web apps	Photoshop, HTML, CSS, and JavaScript	iUI, jQTouch, W3C Mobile Web App Best Practices

Table 4.4 Design Tools and Interface Toolkits

All the Best!

UNIT 5 – WEB INTERFACE DESIGN

Syllabus: Designing Web Interfaces – Drag & Drop, Direct Selection, Contextual Tools, Overlays, Inlays and Virtual Pages, Process Flow. Case Studies.

Topics:

5.1Drag and Drop
5.2 Direct Selection
5.3 Contextual Tools
5.4 Overlays
5.5 Inlays
5.6 Virtual Pages
5.7 Process Flow
5.8 Case Studies

5.1 DRAG AND DROP

One of the great innovations that the Macintosh brought to the world in 1984 was Drag and Drop. Influenced by the graphical user interface work on Xerox PARC's Star Information System and subsequent lessons learned from the Apple Lisa, the Macintosh team invented drag and drop as an easy way to move, copy, and delete files on the user's desktop.

In 2000, a small startup, HalfBrain, launched a web-based presentation application, BrainMatter. It was written entirely in DHTML and used drag and drop as an integral part of its interface.

Interesting Moments

There are a number of **individual states at which interaction is possible**. We call these **microstates interesting moments**as follows:

- How will users know what is draggable?
- What does it mean to drag and drop an object?
- Where can we drop an object, and where is it not valid to drop an object?
- What visual affordance will be used to indicate draggability?
- During drag, how will valid and invalid drop targets be signified?
- Do we drag the actual object?
- Or do we drag just a ghost of the object?
- Or is it a thumbnail representation that gets dragged?
- What visual feedback should be used during the drag and drop interaction?

What makes it challenging is that there are a lot of events during drag and drop that canbe used as opportunities for feedback to the user. Additionally, there are a number of elementson the page that can participate as actors in this feedback loop.

The Events

There are at least 15 events available for cueing the user during a drag and drop interaction:

i. **Page Load**

Before any interaction occurs, we can pre-signify the availability of drag and drop.For example, we could display a tip on the page to indicate draggability.

ii. **Mouse Hover**

The mouse pointer hovers over an object that is draggable.

iii. **Mouse Down**

The user holds down the mouse button on the draggable object.

iv. **Drag Initiated**

After the mouse drag starts (usually some threshold—3 pixels).

v. **Drag Leaves Original Location**

After the drag object is pulled from its location or object that contains it.

vi. **Drag Re-Enters Original Location**

When the object re-enters the original location.

vii. **Drag Enters Valid Target**

Dragging over a valid drop target.

viii. **Drag Exits Valid Target**

Dragging back out of a valid drop target.

ix. **Drag Enters Specific Invalid Target**

Dragging over an invalid drop target.

x. Drag Is Over No Specific Target

Dragging over neither a valid or invalid target. Do we treat all areas outside of validtargets as invalid?

xi. Drag Hovers Over Valid Target

User pauses over the valid target without dropping the object. This is usually when aspring loaded drop target can open up. For example, drag over a folder and pause, thefolder opens revealing a new area to drag into.

xii. Drag Hovers Over Invalid Target

User pauses over an invalid target without dropping the object.

xiii. Drop Accepted

Drop occurs over a valid target and drop has been accepted.

xiv. Drop Rejected

Drop occurs over an invalid target and drop has been rejected. Do we zoom back the dropped object?

xv. Drop on Parent Container

Is the place where the object was dragged from special? Usually this is not the case,but it may carry special meaning in some contexts.

The Actors

During each event we can visually manipulate a number of actors. The page elementsavailable include:

i. **Page (e.g., static messaging on the page)**
ii. **Cursor**
iii. **Tool Tip**
iv. **Drag Object (or some portion of the drag object, e.g., title area of a module)**
v. **Drag Object's Parent Container**
vi. **Drop Target**

Interesting Moments Grid

That's 15 events times 6 actors. That means there are 90 possible interesting moments—eachrequiring a decision involving an almost unlimited number of style and timing choices. We can pull all this together into a simple interesting moment's grid for Drag and Drop.

	Page Generation	Mouse Hover	Drag Initiated	Drag over Valid	Drag over Invalid	Drag over Original	Drop Accepted	Drop Rejected	Drop on Original
Page Content	Hint	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Cursor	Normal	Move Cursor	Move Cursor	Move Cursor	Move Cursor	Move Cursor	Normal	Normal	Normal
Drag Object	Normal	Normal	Reduced Opacity & Tracking	Reduced Opacity & Tracking	Reduced Opacity & Tracking + Invalid Badge	Reduced Opacity & Tracking	2. Modules animates into the area just below insertion bar 3. Module comes to rest in new area 4. Modules slide up in a self-healing transition to close hole	Normal Opacity + Zoom Back to Original	Normal Opacity + Zoom Back to Original
Orig Location	Normal	Normal	Hole Opens	Hole Remains	Hole Remains	Hole Remains	Hole Remains	Hole refilled with drag object	Hole refilled with drag object
Drop Target	Normal	Normal	Normal	Insertion Bar	N/A	N/A	1. Insertion Bar Removed	N/A	N/A

Table 5.1 A simplified interesting moments grid for the original My Yahoo! drag and drop design;it provided a way to capture the complexities of drag and drop into a single page

The grid is a handy tool for planning out interesting moments during a drag and drop interaction. It serves as a checklist to make sure there are no "holes" in the interaction. Just place the actors along the lefthand side and the moments along the top. In the grid intersections, place the desired behaviors.

Purpose of Drag and Drop

Drag and drop can be a powerful idiom if used correctly. Specifically it is useful for:

i. **Drag and Drop Module**

Rearranging modules on a page.

ii. **Drag and Drop List**

Rearranging lists.

iii. **Drag and Drop Object**

Changing relationships between objects.

iv. **Drag and Drop Action**

Invoking actions on a dropped object.

v. **Drag and Drop Collection**

Maintaining collections through drag and drop.

5.1.2 Drag and Drop Module

One of the most useful purposes of drag and drop is to allow the user to directly placeobjects where she wants them on the page. A typical pattern is Drag and Drop Moduleson a page. Netvibes provides a good example of this interaction pattern (Figure 5.1).

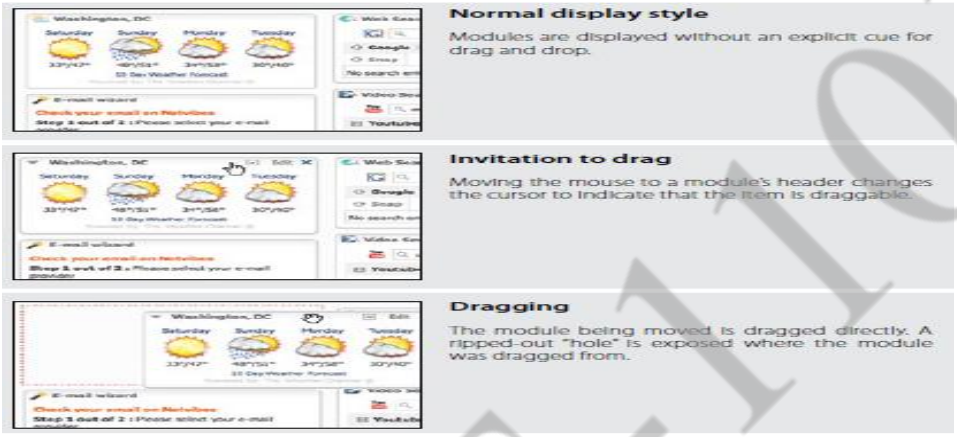


Fig 5.1 Netvibes allows modules to be arranged directly via drag and drop; the hole cues what will happen when a module is dropped

Considerations

Netvibes allows its modules to be rearranged with drag and drop. A number of interestingmoments decide the specific interaction style for this site. Fig 5.2 shows the interestingmoments grid for Netvibes.

	Mouse Hover	Mouse Down	Drag Initiated	Drag Hovers over Valid Target*	Drop Accepted
Cursor	Change to a hand with finger pointing.*	Change to a hand/move cursor.	No change.*		Cursor returns to normal style.
Dragged Module			Module is dragged with full opacity.		Dragged version is removed.
Dragged Modules Original Location			Hole is shown as a red dashed outline.		Hole is removed.
Drop Target				Hole (red dashed outline) is moved to the new drop spot. Other modules shift to close prior hole.	Module is placed in the new location.
Notes	* *A better approach might be to signal draggability with the hand/move cursor.*		* *On drag initiated, it would be better to switch to a hand that looks like a grab.*	* *Triggers when the dragged module's title bar has moved past the midpoint of the dragged over module's header.*	

Fig 5.2 Interesting moments grid for Netvibes: there are 20 possible moments of interaction; Netvibes specifically handles 9 of these moments

While dragging, it is important to make it clear what will happen when the user drops thedragged object. There are two common approaches to targeting a drop:

i. **Placeholder target**

ii. **Insertion target**

Placeholder target

Netvibes uses a placeholder (hole with dashed outline) as the drop target. The idea (in Fig5.3) is to always position a hole in the spot where the drop would occur.

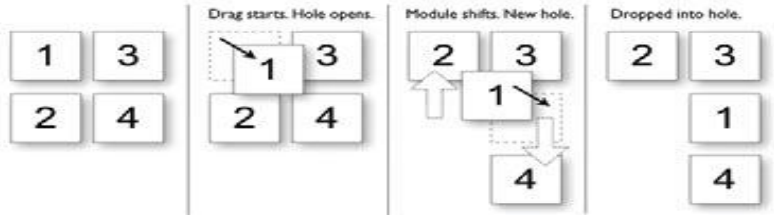


Fig 5.3 A placeholder target always shows where the dragged module will end after the drop; module 1 is being dragged from the upper right to the position between modules 3 and 4

When module 1 starts dragging, it gets "ripped" out of the spot. In its place is theplaceholder target (dashed outline). As 1 gets dragged to the spot between 3 and 4, theplaceholder target jumps to fill in this spot as 4 moves out of the way.

The hole serves as a placeholder and always marks the spot that the dragged module willland when dropped. It also previews what the page will look like (in relation to the othermodules) if the drop occurs there. For module drag and drop, the other modules onlyslide up or down within a vertical column to make room for the dragged module.

One complaint with using placeholder targets is that the page content jumps around a lot during the drag. This makes the interaction noisier and can make it harder to understandwhat is actually happening. This issue is compounded when modules look similar. Theuser starts dragging the modules around and quickly gets confused about what just gotmoved. One way to resolve this is to provide a quick animated transition as the modulesmove. It is important, however, that any animated transitions not get in the way of thenormal interaction.

There is a point in Figure 5.3 where the placeholder shifts to a new location. The position of the mouse, the boundary of the dragged object, and the boundary of the dragged-over object can all be used to choose the module'snew location.

Boundary-based placement.

Since most sites that use placeholder targeting drag the modulein its original size, targeting is determined by the boundaries of the dragged object andthe boundaries of the dragged-over object. The mouse position is usually ignored becausemodules are only draggable in the title (a small region). **Both Netvibes and iGoogletake the boundary-based approach.** But, interestingly, they calculate the position of theirplaceholders differently.

In Netvibes, the placeholder changes position only after the dragged module's title bar hasmoved beyond the dragged-over module's title bar. In practice, this means if we are movinga small module to be positioned above a large module, we have to move it to the very top ofthe large module.

In contrast, moving the small module below the large module actually requires less drag distance since we only have to get the title bar of the small module below the title bar of the large module.

This approach to boundary-based drop targeting is non-symmetrical in the drag distancewhen dragging modules up versus dragging modules down.

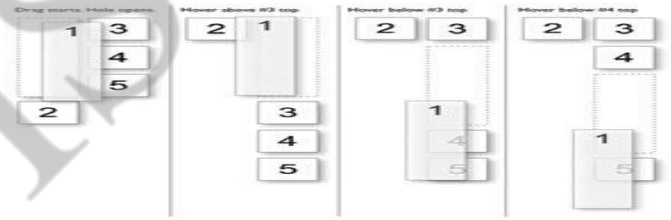


Fig 5.4 The Netvibes approach requires the dragged object's title to be placed above or below a module before the placement position changes; this results in inconsistent drag distances

A more desirable approach is that taken by iGoogle. Instead of basing the drag on the titlebar, iGoogle calculates the placeholder targeting on the dragged-over object's midpoint.

In Fig 5.5, the stock market module is very large (the module just above the moonphase module).

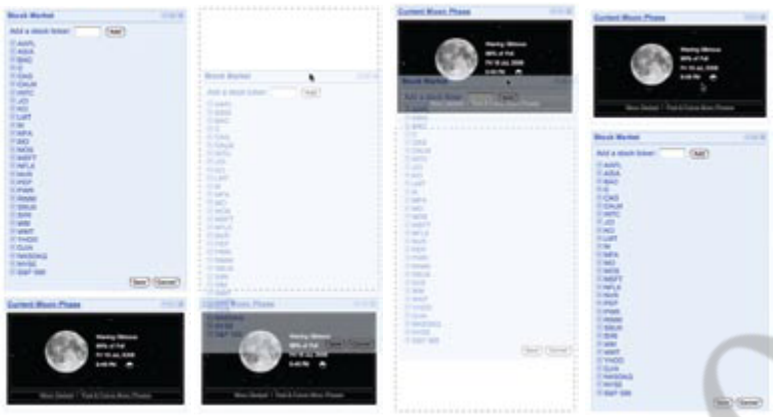


Fig 5.5 When dragging a module downward, iGoogle moves the placeholder when the bottom of the dragged module crosses the midpoint of the object being dragged over; the distance to accomplish a move is less than in the Netvibes approach

With the Netvibes approach, we would have to drag the stock module's title below themoon phase module's title. iGoogle instead moves the placeholder when the bottom of thedragged module (stock module) crosses the midpoint of the dragged over module (moonphase module).

What happens when we head the other way? When we drag the stock module up to place it above the moon phase module, iGoogle moves the placeholder when the top of the stock module crosses the midpoint of the moon phase module (Fig5.6).

Fig 5.6 When dragging a module upward, iGoogle moves the placeholder when the top of the dragged module crosses the midpoint of the object being dragged over; dragging modules up or down requires the same effort, unlike in the Netvibes example

As Figure 5.7 illustrates, module 1 is dragged from the first column to the second column, the placeholder moves above module 3. As module 1 is dragged downward, the placeholder moves below 3 and 4 as the bottom of module 1 crosses their midpoints.

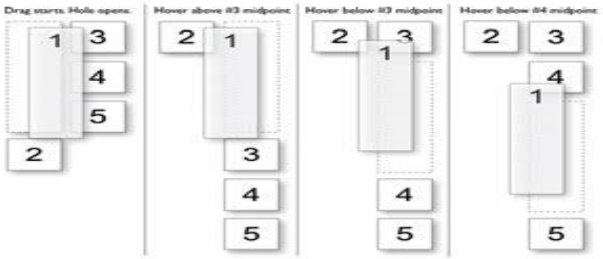


Fig 5.7To create the best drag experience, use the original midpoint location of the module beingdragged over to determine where to drop the dragged module: module 1 is being dragged into theposition just below module 4

The net result is that the iGoogle approach feels more responsive and requires less mouse movement to position modules. Figure 5.8 shows the interesting moments grid for the iGoogle drag and drop interaction.

	Mouse Hover	Mouse Down	Drag Initiated	Drag Hovers over Valid Target*	Drop Accepted
Cursor	Change to a hand pointer.	Change to normal style.*			
Dragged Module			Slightly transparent.		Dragged module removed.
Dragged Modules Original Location			Hole is shown as a gray, thick, dashed outline.		Hole is removed.
Drop Target				Hole (gray, thick, dashed outline) is moved to the new drop spot. Other modules shift to close prior hole.	Module is placed in the new location.
Notes		** A better approach is to switch to a hand that looks like it grabbed the module.*	** Drag initiates instantly on mouse down.*	** Triggers when the mid-point of the dragged object enters a valid drop target.*	

Fig 5.8 Interesting moments grid for iGoogle: as in the Netvibes grid, there are 20 possible moments of interaction; iGoogle specifically handles 8 of these moments

Insertion target

Placeholder positioning is a common approach, but it is not the only way to indicate drop targeting. An alternate approach is to keep the page as stable as possible and only move around an insertion target (usually an insertion bar). A previous version of My Yahoo! used the insertion bar approach as the dragged module was moved around

While the module is dragged, the page remains stable. No modules move around. Instead an insertion bar marks where the module will be placed when dropped. This technique is illustrated in Figure 5.9. When module 1 is dragged to the position between 3 and 4, an insertion bar is placed there. This indicates that if 1 is dropped, then 4 will slide down to open up the drop spot.

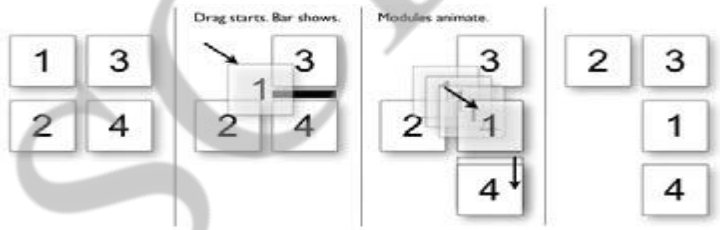


Fig 5.9 Using an insertion bar keeps the page stable during dragging a nd makes it clear how things get rearranged when the module is dropped

Unlike with the placeholder target, the dragged module 1 is usually represented with a slightly transparent version of the module (also known as ghosting). In the most current version, fullsize module dragging has been replaced with a thumbnail representation. This is somewhat unfortunate since the small gray outline is not very visible.

As we can see in Fig 5.10, the My Yahoo! page makes different decisions about how drag and drop modules are implemented as compared to Netvibes and iGoogle.

	Mouse Hover	Mouse Down*	Drag Initiated	Drag Hovers over Valid Target	Drag Hovers over Invalid Target	Drop Accepted	Drop Rejected	Drop On Parent Container
Cursor	Change to a hand pointer.					Change back to normal style cursor.	Change back to normal style cursor.	Change back to normal style cursor.
Dragged Module			Thumbnail represents dragged module. (small, gray outline.)			Thumbnail removed.	Thumbnail removed by zooming back to original location.	Thumbnail removed by zooming back to original location.
Dragged Module's Original Location			Original module shown dimmed at original location.			Modules get rearranged.	Original module is brightened back to original opacity.	
Drop Target (Insertion Bar)			No insertion bar shown until valid drop target available.	Insertion bar shown where module can be dropped.	Insertion bar removed.	Module is placed in the new location, modules re-arrange.	No insertion bar was visible.	No insertion bar was visible.
Notes		* If mouse is held down for more than one second, Drag is initiated.	* Drag is also initiated if the mouse is moved more than 3 pixels after the mouse down.	Triggers when the mid-point of the dragged object enters a valid drop target.				

Fig 5.10 My Yahoo! uses 15 of the possible 32 moments to interact with the user during drag and drop; the biggest difference between My Yahoo!, Netvibes, and iGoogle is the insertion bar placement—another subtle difference is how drag gets initiated

Drag distance

Dragging the thumbnail around does have other issues. Since the object being dragged is small, it does not intersect a large area. It requires moving the small thumbnail directly to the place it will be dropped. With iGoogle, the complete module is dragged. Since the module will always be larger than the thumbnail, it intersects a drop target with much less movement. The result is a shorter drag distance to accomplish a move.

Tip- Drag and Drop takes additional mouse dexterity. If possible, shorten the necessary drag distance to target a drop.

Drag rendering

My Yahoo! uses a small gray rectangle to represent a module. Netvibes represents the dragged module in full size as opaque, while iGoogle uses partial transparency (Fig 5.11). The transparency (ghosting) effect communicates that the object being dragged is actually a representation of the dragged object. It also keeps more of the page visible, thus giving a clearer picture of the final result of a drop.

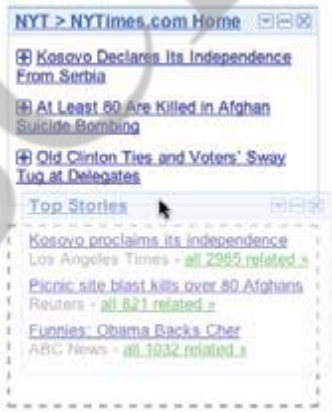


Fig 5.11 On iGoogle the dragged module Top Stories is given transparency to make it easier to see the page and to indicate that we are in a placement mode

Ghosting the module also indicates that the module is in a special mode. It signals that the module has not been positioned; instead, it is in a transitional state.

Tip - For Drag and Drop Modules, use the module's midpoint to control the drop targeting.

Of the various approaches for Drag and Drop Modules, iGoogle combines the best approaches into a single interface:

i. **Placeholder targeting**

Most explicit way to preview the effect.

ii. **Midpoint boundary**

Requires the least drag effort to move modules around.

iii. **Full-size module dragging**

Coupled with placeholder targeting and midpoint boundary detection, it means drag distances to complete a move are shorter.

iv. Ghost rendering

Emphasizes the page rather than the dragged object. Keeps the preview clear.

Best Practices for Drag and Drop Module

Here are some best practices to keep in mind:

- Use the placeholder approach when showing a clear preview during drag is important.
- Use the insertion bar approach when you want to avoid page jitter.
- Use the midpoint of the dragged object to determine drag position.
- Use a slightly transparent version of the object being dragged (ghost) instead of an opaque version.
- If you drag thumbnail representations, use the insertion bar targeting approach.

5.1.3 Drag and Drop List

Rearranging lists is very similar to rearranging modules on the page but with the added constraint of being in a single dimension (up/down or left/right). The Drag and Drop List pattern defines interactions for rearranging items in a list. 37 Signal's Backpackit allows to-do items to be rearranged with Drag and Drop List (Figure 5.12).

Fig 5.12 Backpackit allows to-do lists be arranged directly via drag and drop

Considerations

Backpackit takes a real-time approach to dragging items. Since the list is constrained, this is a natural approach to moving objects around in a list. We immediately see the result of the drag.

Placeholder target

This is essentially the same placeholder target approach we discussed earlier for dragging and dropping modules. The difference is that when moving an item in a list, we are constrained to a single dimension. Less feedback is needed. Instead of a "ripped-out" area (represented earlier with a dotted rectangle), a simple hole can be exposed where the object will be placed when dropped.

A good example from the desktop world is Apple's iPhoto. In a slideshow, we can easily rearrange the order of photos with drag and drop. Dragging the photo left or right causes the other photos to shuffle open a drop spot.

The difference between iPhoto and Backpackit is that instead of using the dragged photo's boundary as the trigger for crossing a threshold, iPhoto uses the mouse cursor position. In the top row of Figure 5.13, the user clicked on the right side of the photo. When the cursor crosses into the left edge of the next photo, a new space is opened. In the bottom row, the user clicked on the top left side of the photo. Notice in both cases it is the mouse position that determines when a dragged photo has moved into the space of another photo, not the dragged photo's boundary.

Figure 5.13 iPhoto uses cursor position: when the cursor crosses a threshold (the edge of the next photo), a new position is opened up

Insertion target

Just as with Drag and Drop Modules, placeholder targeting is not the only game in town. We can also use an insertion bar within a list to indicate where a dropped item will land. Netflix uses an insertion target when movies are dragged to a new location in a user's movie queue (Figure 5.14).

Fig 5.14 A Netflix queue can be rearranged via drag and drop

The upside to this approach is that the list doesn't have to shuffle around during drag. The resulting experience is smoother than the Backpackit approach. The downside is that it is not as obvious where the movie is being positioned. The insertion bar appears under the ghosted item. The addition of the brackets on the left and right of the insertion bar is an attempt to make the targeting clearer

Non–drag and drop alternative

Besides drag and drop, the Netflix queue actually supports two other ways to move objects around:

- Edit the row number and then press the "Update DVD Queue" button.
- Click the "Move to Top" icon to pop a movie to the top.

Modifying the row number is straightforward. It's a way to rearrange items without drag and drop. The "Move to Top" button is a little more direct and fairly straightforward (if the user really understands that this icon means "move to top"). Drag

and drop is the least discoverable of the three, but it is the most direct, visual way to rearrange the list. Since rearranging the queue is central to the Netflix customer's satisfaction, it is appropriate to allow multiple ways to do so.

Hinting at drag and drop

When the user clicks the "Move to Top" button, Netflix animates the movie as it moves up. But first, the movie is jerked downward slightly and then spring-loaded to the top.

Click "Move to Top":

Clicking the "Move to Top"button starts the movie movingto the top.

Spring loaded:

The movie does not immediatelystart moving up. Instead,it drops down and tothe right slightly. This gives the feeling that the movie isbeing launched to the top.

Animated move to top:

The movie then animates veryquickly to show it is movingto the top.

The combination of the downward jerk and then the quick animation to the top gives a subtle clue that the object is draggable. This is also an interesting moment to advertise drag and drop. After the move to top completes, a simple tip could appear to invite users to drag and drop. The tip should probably be shown only once, or there should be a way to turn it off. Providing an invitation within a familiar idiom is a good way to lead users to the new idiom.

Tip: If drag and drop is a secondary way to perform a task, use the completion of the familiartask as an opportunity invite the user to drag and drop the next time.

Drag lens

Drag and drop works well when a list is short or the items are all visible on the page. But when the list is long, drag and drop becomes painful. Providing alternative ways to rearrange is one way to get around this issue. Another is to provide a drag lens while dragging.

A drag lens provides a view into a different part of the list that can serve as a shortcut target. It could be a fixed area that is always visible, or it could be a miniature view of the list that provides more rows for targeting. The lens will be made visible only during dragging. A good example of this is dragging the insertion bar while editing text on the iPhone (Figure 5.15).

Figure 5.15 The iPhone provides a drag magnifier lens that makes it easier to position the cursor

Best Practices for Drag and Drop List

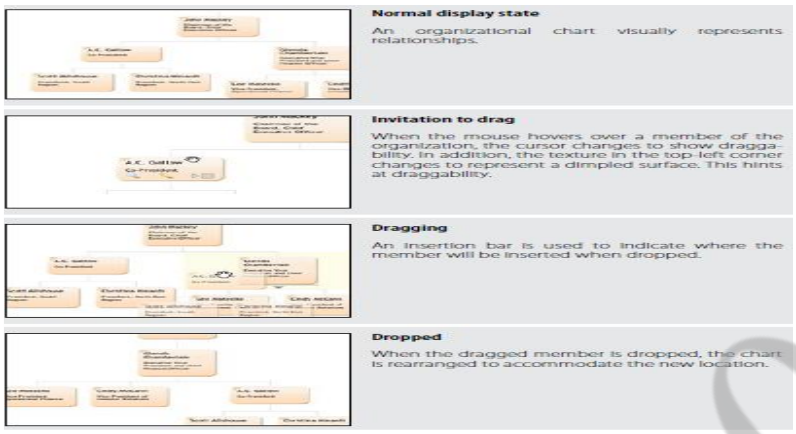
Here are some best practices to keep in mind:

- If possible, drag the items in a list in real time using the placeholder target approach.
- Use the mouse position for drag target positioning.
- If the goal is speed of dragging or if dragged items are large, consider using the insertion target approach, as rendering an insertion bar is inexpensive compared to dynamically rearranging the list.
- Since drag and drop in lists is not easily discoverable, consider providing an alternate way to rearrange the list.
- When the user rearranges the list with an alternate method, use that moment for a one-time advertisement for drag and drop.

Drag and Drop Object

Another common use for drag and drop is to change relationships between objects. This is appropriate when the relationships can be represented visually. Drag and drop as a means of visually manipulating relationships is a powerful tool.

Cogmap is a wiki for organizational charts. Drag and Drop Object is used to rearrange members of the organization.



Considerations

When object relationships can be clearly represented visually, drag and drop is a natural choice to make these types of changes. Cogmap uses the target insertion approach. This allows the dragging to be nondistracting, since the chart does not have to be disturbed during targeting.

Drag feedback: Highlighting

Bubbl.us, an online mind-mapping tool, simply highlights the node that will be the new parent (Figure 5.16).

Fig 5.16 Bubbl.us provides a visual indication of which node the dropped node will attach itself to

In both cases, immediate preview is avoided since it is difficult to render the relationships in real time without becoming unnecessarily distracting.

Looking outside the world of the Web, the desktop application Mind Manager also uses highlighting to indicate the parent in which insertion will occur. In addition, it provides insertion targeting to give a preview of where the employee will be positioned once dropped (Figure 5.17).

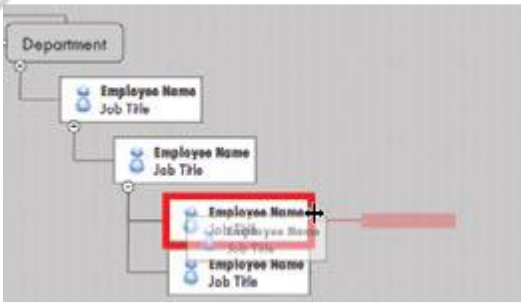


Fig 5.17 Mind Manager is a desktop tool that uses a combination of insertion targeting plus a clear preview of the drop

Drag feedback:Dragged object versus drop target

As we mentioned at the beginning of this chapter, one of the first serious uses for drag and drop was in the Oddpost web mail application. Oddpost was eventually acquired by Yahoo! and is now the Yahoo! Mail application. Yahoo! Mail uses drag and drop objects for organizing email messages into folders.

Drag initiated:

When a message drag is initiated,a snippet of the messageis shown, along withan icon denoting whether adrop can be made.

Valid drop target:

When the dragged messagemay be dropped, theicon portion of the draggedobject changes from ared invalid sign to a greencheckmark.

Instead of signaling that a drop is valid or invalid by changing the visual appearance of the area dragged over, Yahoo! Mail shows validity through the dragged object.

When a drop will be invalid:

• The dragged object's icon becomes a red invalid sign.

• If over an invalid folder, the folder is highlighted as well.

When a drop will be valid:

• The dragged object's icon changes to a green checkmark.

• The drop target highlights.

Another approach is to signal both validity and location in the drop target itself. In this case we would highlight the valid drop target when it is dragged over and not highlight the drop target if it is invalid. In Yahoo! Mail's interaction, the signaling of validity and where it can be dropped are kept separate. This allows a drag to indicate that a target is a drop target, just not valid for the current object being dragged.

One odd situation occurs when we first start dragging a message and then later drag it back into the inbox area. At first it shows the inbox as an invalid drop area. Then it shows it as a valid drop area. Here the interface needs to display the same indicator in both cases.

Drag feedback: Drag positioning

Another slightly troublesome approach is positioning the dragged object some distance away from the mouse. The reason the object is positioned in this manner is to avoid obscuring dragged-over folders. While this may alleviate that problem, it introduces a second problem: when we initiate the drag, the dragged message jumps into the offset position. Instead of conveying that the first message in the list is being dragged, it feels like the second message in the list is being dragged.

Drag feedback: Drag start

In Yahoo! Mail, message dragging is initiated when the mouse is dragged about four or five pixels.

A good rule of thumb on drag initiation comes from the Apple Human Interface Guidelines: *Your application should provide drag feedback as soon as the user drags an item at least three pixels. If a user holds the mouse button down on an object or selected text, it should become draggable immediately and stay draggable as long as the mouse remains down.*

It might seem like a small nit, but there is quite a difference between starting a drag after three pixels of movement versus four or five pixels. The larger value makes the object feel hard to pull out of its slot to start dragging. On the flip side, starting a drag with too small a value can cause drag to initiate accidentally, usually resulting in the interface feeling too finicky.

Tip - Start a drag when the object is dragged three pixels or the mouse is held down for halfa second.

Best Practices for Drag and Drop Object

Here are some best practices to keep in mind:

- If objects are represented in a complex visual relationship, use insertion targeting to indicate drop location (minimizes disturbing the page during drag).
- For parent/child relationships, highlight the parent as well to indicate drop location.
- If possible, reveal drag affordances on mouse hover to indicate draggability.
- Initiate drag when the mouse is dragged three pixels or if the mouse is held down for at least half a second.
- Position dragged objects directly in sync with the cursor. Offsetting will make the drag feel disjointed.
- When hovering over a draggable object, change the cursor to indicate draggability.

Drag and Drop Action

Drag and drop is also useful for invoking an action or actions on a dropped object. The Drag and Drop Action is a common pattern. Its most familiar example is dropping an item in the trash to perform the delete action. Normally uploading files to a web application includes pressing the upload button and browsing for a photo. This process is repeated for each photo.

When Yahoo! Photos was relaunched in 2006, it included a drag and drop upload feature. It allowed the user to drag photos directly into the upload page. The drop signified the upload action (Figure 2-30).

Normal display state:

"Add Photos" allows either browsing for photos or simply dragging anddropping them into the target zonebelow.

Invitation to drag:

The invitation is clear. By using thedrop target area as an advertisementfor the drag feature, the process is discoverable (as well as natural).

Dropped:

Photos dropped are collected into anupload area. Pressing "Start Upload" starts the uploading process.

Completed:

All items are marked complete when finished.

Considerations

This is not a trivial implementation. But it does clearly illustrate the benefit of drag and drop for operating on a set of files. The traditional model requires each photo to be selected individually for upload. Drag and drop frees us to use whatever browsing method is available on our system and then drop those photos for upload.

Anti-pattern: Artificial Visual Construct

Unfortunately, drag and drop can sometimes drive the design of an interface instead of being an extension of a natural interface. These interactions are almost always doomed, as they are the tail wagging the proverbial dog. Rating movies, books, and music is a common feature found on many sites. But what happens if we try to use drag and drop to rate movies?

While this certainly would work, it is wrong for several reasons:

Non-obvious:

Requires some additional instructions to "Drag the DVDs into the boxes below" in order for the user to know how to rate the movies.

Too much effort:

Requires too much user effort for a simple task. The user needs to employ mouse gymnastics to simply rate a movie. Drag and drop involves these discrete steps: target, then drag, then target, and then drop. The user has to carefully pick the movie, drag it to the right bucket, and release.

Too much space:

Requires a lot of visual space on the page to support the idiom. Is it worth this amount of screen real estate?

Direct rating systems (thumbs up/down, star ratings, etc.) are a much simpler way to rate a movie than using an Artificial Visual Construct. A set of stars is an intuitive, compact, and simple way to rate a movie.

This method still falls way short since the amount of space needed for this far outweighs simpler approaches such as providing an action button for the selected objects.

Tip - Drag and drop should never be forced. Don't create an artificial visual construct to support it.

Natural Visual Construct

Another example of Drag and Drop Action is demonstrated in Google Maps. A route is visually represented on the map with a dark purple line. Dragging an arbitrary route point to a new location changes the route in real time (Figure 5.18).

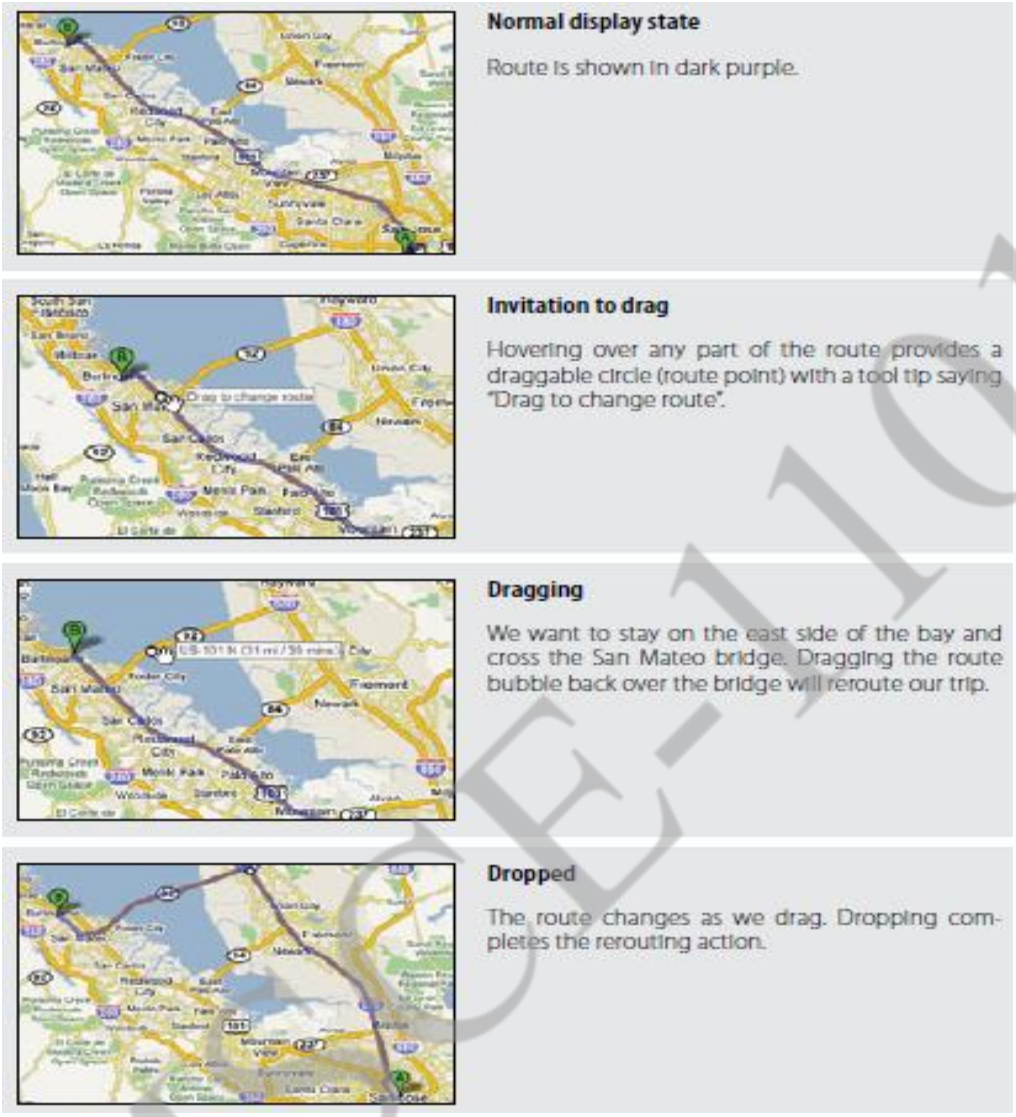


Fig 5.18 Rerouting in Google Maps is as simple as drag and drop

This is the opposite of the Artificial Visual Construct anti-pattern. The route is a Natural Visual Construct. Since anywhere along the route is draggable, there are a lot of opportunities to discover the rerouting bubble. When the route is being dragged, Google dynamically updates it. The constant feedback forms the basis of a Live Preview.

Best Practices for Drag and Drop Action

Here are some best practices to keep in mind:

- Use **Drag and Drop Actions** sparingly in web interfaces, as they are not as discoverable or expected.
- Provide alternate ways to accomplish the action. Use the **Drag and Drop Action** as a shortcut mechanism.
- Don't use drag and drop for setting simple attributes. Instead use a more direct approach to setting attributes on the object.
- Don't construct an artificial visual representation for the sole purpose of implementing drag and drop. Drag and drop should follow the natural representation of the objects in the interface.
- Provide clear invitations on hover to indicate the associated action.

Drag and Drop Collection

A variation on dragging objects is collecting objects for purchase, bookmarking, or saving into a temporary area. This type of interaction is called Drag and Drop Collection. Drag and drop is a nice way to grab items of interest and save them to a list. The Laszlo shopping cart example illustrates this nicely (Figure 5.19).

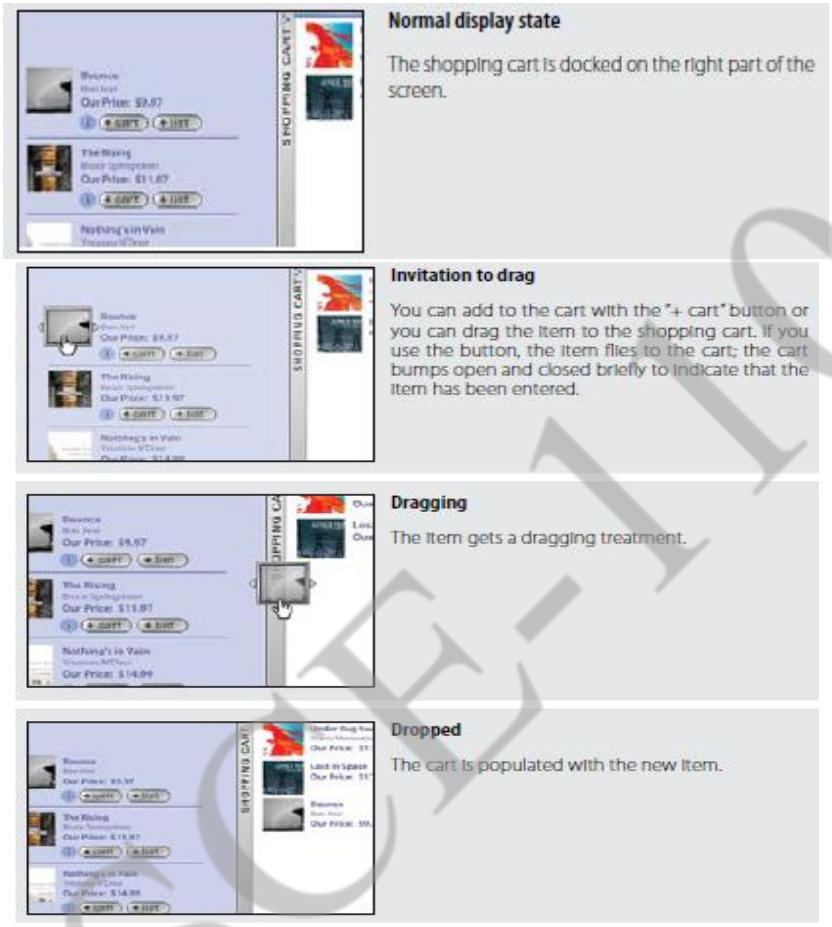


Fig 5.19 This Laszlo shopping cart demo uses both drag and drop and a button action to add items to its shopping cart

Considerations

There are a few issues to consider in this example.

Discoverability

Drag and drop is a natural way collect items for purchase. It mimics the shopping experience in the real world. Grab an item. Drop it in our basket. This is fast and convenient once we know about the feature. However, as a general rule, we should never rely solely on drag and drop for remembering items.

Parallel, more explicit ways to do the same action should be provided. In this example, Laszlo provides an alternative to dragging items in the cart. Notice the "+ cart" button in Figure 5.19. Clicking this button adds the item to the shopping cart.

Teachable moment

When providing alternates to drag and drop, it is a good idea to hint that dragging is an option. In the Laszlo example, clicking the "+ cart" button causes the shopping cart tray to bump slightly open and then closed again. This points to the physicality of the cart. Using another interaction as a teachable moment to guide the user to richer interactions is a good way to solve discoverability issues.

The Challenges of Drag and Drop

As we can see from the discussion in this chapter, Drag and Drop is complex. There are four broad areas where Drag and Drop may be employed: **Drag and Drop Module, Drag and Drop List, Drag and Drop Object, and Drag and Drop**

Action. And in each area, there are a large number of interesting moments that may be handled in numerous ways. Being consistent in visual and interaction styles across all of these moments for all of these types of interactions is a challenge in itself. And keeping the user informed throughout the process with just the right amount of hints requires design finesse.

****

5.2 DIRECT SELECTION

When the Macintosh was introduced, it ushered into the popular mainstream the ability to directly select objects and apply actions to them. Folders and files became first-class citizens. Instead of a command line to delete a file, we simply dragged a file to the trash.

Treating elements in the interface as directly selectable is a clear application of the Make It Direct principle. On the desktop, the most common approach is to initiate a selection by directly clicking on the object itself. We call this selection pattern Object Selection.

Types of selection patterns:

Toggle Selection

Checkbox or control-based selection.

Collected Selection

Selection that spans multiple pages.

Object Selection

Direct object selection.

Hybrid Selection

Combination of Toggle Selection and Object Selection.

5.2.1 Toggle Selection

The most common form of selection on the Web is Toggle Selection. Checkboxes and toggle buttons are the familiar interface for selecting elements on most web pages. An example is Yahoo! Mail Classic.

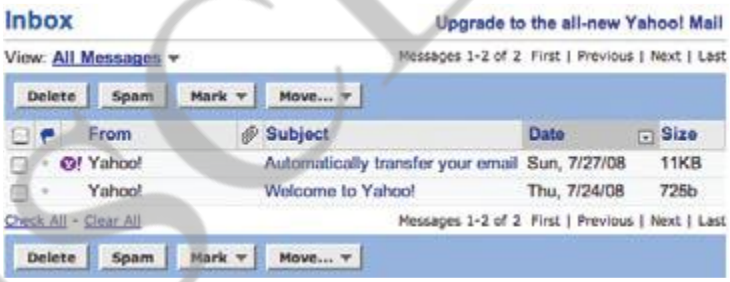


Fig 5.20 In Yahoo Mail Classic a mail message can be selected by clicking on the corresponding row's checkbox

The way to select an individual mail message is through the row's checkbox. Clicking onthe row itself does not select the message. We call this pattern of selection Toggle Selectionsince toggle-style controls are typically used for selecting items.

Tip -Toggle Selection is the easiest way to allow discontinuous selection.

Once items have been check-selected, actions can be performed on them. Usually these actions are performed on the selection by clicking on a separate button (e.g., the Delete button). Gmail is a good example of actions in concert with Toggle Selection.

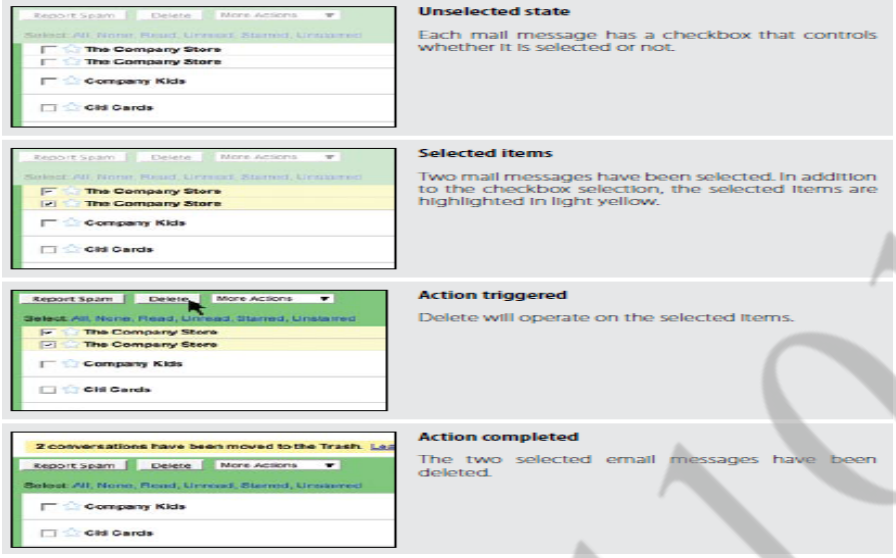


Fig 5.21 Gmail uses checkbox selection to operate on messages

Action completed

The two selected email messages have beendeleted.

Considerations

Toggle Selection with checkboxes has some nice attributes:

• Clear targeting, with no ambiguity about how to select the item or deselect it.

• Straightforward discontinuous selection, and no need to know about Shift or Controlkey ways to extend a selection. Just click the checkboxes in any order, either in a continuous or discontinuous manner.

• Clear indication of what has been selected.

Scrolling versus paging

The previous examples were with paged lists. Yahoo! Mail uses a scrolled list to show all of its mail messages. While not all messages are visible at a time, the user knows that scrolling through the list retains the currently selected items. Since the user understands that all the messages not visible are still on the same continuous pane, there is no confusion about what an action will operate on—it will affect all selected items in the list. Sometimes the need for clarity of selection will drive the choice between scrolling and paging.

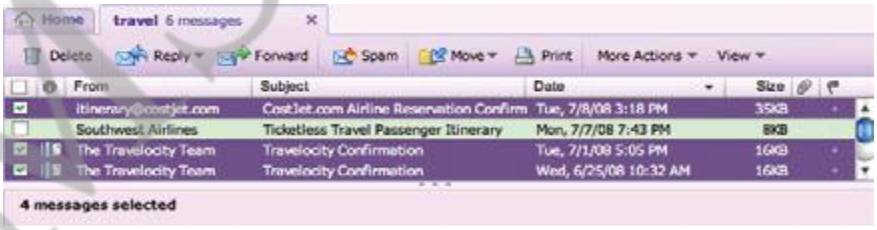


Fig 5.22 Yahoo! Mail uses a scrolled list for its messages; selection includes what is in the visible part of the list as well as what is scrolled out of view

Making selection explicit

With Yahoo! Bookmarks we can manage bookmarks by selecting bookmarked pages and then acting on them. The selection model is visually explicit (Figure 3-6).

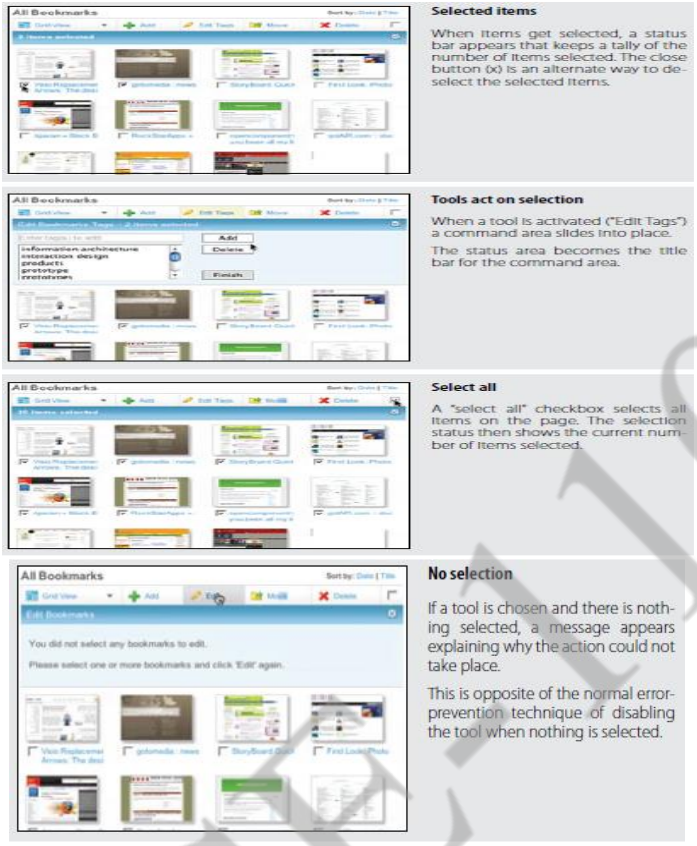


Fig 5.23 Yahoo! Bookmarks explicitly displays the state of the selection

The advantage of this method is that it is always clear how many items have been selected. Visualizing the underlying selection model is generally a good approach. This direct approach to selection and acting on bookmarks creates a straightforward interface.

One interesting question: what happens when nothing is selected? One approach is to disable any actions that require at least one selected item. Yahoo! Bookmarks takes a different approach. Since buttons on the Web do not follow a standard convention, we often can't rely on a color change to let us know something is not clickable. Yahoo! Bookmarks chose to make selection very explicit and make it clear when a command is invalid because nothing is selected ("No selection" in Figure 5.23). This is not normally the optimal way to handle errors. Generally, the earlier we can prevent errors, the better the user experience.

Netflix disables the "Update DVD Queue" button when nothing is selected and enables it when a movie gets selected (Figure 5.24).

Fig 5.24 When nothing is selected, Netflix disables the "Update DVD Queue" button to prevent errors early

Best Practices for Toggle Selection

Here are some best practices to keep in mind:

- Use **Toggle Selection** for selecting elements in a row.
- Use **Toggle Selection** to make it easy to select discontinuous elements.
- In a list, highlight the row in addition to the checkbox to make the selection explicit.
- When moving from page to page, actions should only operate on the items selected on that page.
- If offering a "select all" option, consider providing a way to select all elements across all pages.
- Provide clear feedback for the number of elements selected.
- If possible, disable unavailable actions when nothing is selected. If you keep the action enabled, you will need additional interface elements to signal that it can't be completed.

5.2.2 Collected Selection

Toggle Selection is great for showing a list of items on a single page. But what happens if we want to collect selected items across multiple pages? Collected Selection is a pattern for keeping track of selection as it spans multiple pages. In Gmail, we can select items as we move from page to page. The selections are remembered for each page. If we select two items on page one, then move to page two and select three items, there are only three items selected. This is because actions only operate on a single page. This makes sense, as users do not normally expect selected items to be remembered across different pages.

Considerations

Gmail does provide a way to select all items across different pages. When selecting all items on an individual page (with the "All" link), a prompt appears inviting the user to "Select all 2785 conversations in Spam". Clicking that will select all items across all pages (Figure 5.25). The "Delete Forever" action will operate on all 2785 conversations, not just the 25 selected on the page.

Fig 5.25 Gmail provides a way to select all items across all pages, allowing the user to delete all items in a folder without having to delete all items on each page individually

Keeping the selection visible

The real challenge for multi-page selection is finding a way to show selections gathered across multiple pages. We need a way to collect and show the selection as it is being created. Here is one way that Collected Selection comes into play. LinkedIn uses Collected Selection to add potential contacts to an invite list (Figure 5.26).

Fig 5.26 LinkedIn provides a holding place for saving selections across multiple pages

The list of potential invitees is shown in a paginated list on the lefthand side. Clicking the checkbox adds them to the invite list. The invite list becomes the place where selected contacts across multiple pages are remembered.

Any name in the invite list can be removed by clicking the "X" button beside it. Once the complete list of invitees is selected, clicking the "Invite selected contacts" sends each selected contact a LinkedIn invitation. **Collected Selection and actions**

When Yahoo! Photos was working its way through an early design of its Photo Gallery, the plan was to show all photos in a single, continuous scrolling page. In a long virtual list, the selection model is simple. Photos are shown in a single page and selection is easily understood in the context of this single page.

However, due to performance issues, the design was changed. Instead of a virtual page, photos had to be chunked into pages. In order to support Collected Selection, Yahoo! Photos introduced the concept of a "tray" into the interface (Figure 5.27). On any page, photos can be dragged into the tray. The tray keeps its contents as the user moves from page to page. So, adding a photo from page one and three more from page four would yield four items in the tray. As a nice touch, the tray would make itself visible (by sliding into view) even when the user was scrolled down below the fold.

Fig 5.27 Yahoo! Photos used a "tray" to implement a form of Collected Selections; the confusing aspect was which actions in the menu operated on the tray versus the photos selected on the page

There was a problem with the design, however. In the menu system it was hard to discern whether the user meant to operate on the selection (photos on the page could be selected through an Object Selection model) or on the collected items in the tray. To resolve this ambiguity, the drop-down menus contained two identical sets of commands. The first group of commands in the menu operated on the collected items in the tray. The second set of commands operated on the selected objects. Needless to say, this was confusing since it required the user to be fully aware of these two selection models when initiating a command.

One way to remove this ambiguity would have been to have a single set of commands that operated on either the tray or the photos—depending on which had the focus. This would require a way to select the tray and a way to deselect it (by clicking outside the tray). A possible approach would be to slightly dim the photo gallery when the tray is selected (causing it to clearly have the focus), and do the opposite when the tray is not the focus.

Best Practices for Collected Selection

Here are some best practices to keep in mind:

- If you allow selection across page boundaries, accumulate the selected items (from each page) into a separate area. This makes the selection explicit even when moving from page to page.
- Use **Collected Selection** to blend **Toggle Selection** and **Object Selection** in the same interface.
- Watch out for ambiguity between items selected with **Collected Selection** and any items or objects that can be normally selected on the page.

3.2.3 Object Selection

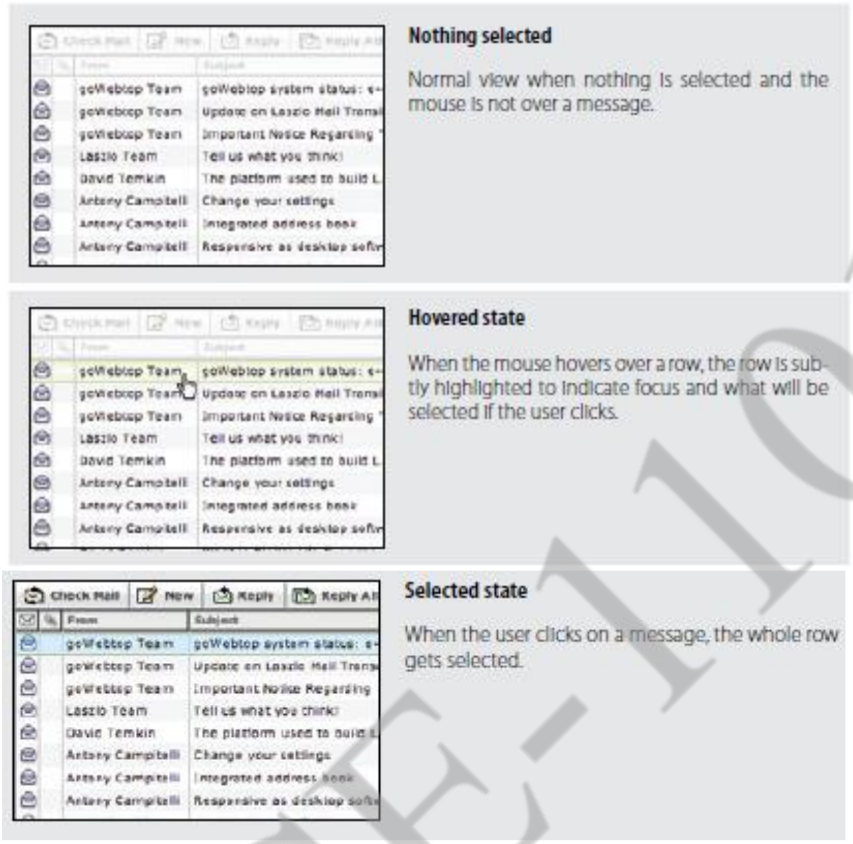


Fig 5.28 Laszlo Web Top Mail uses highlighting to indicate row selection

As we mentioned earlier, Toggle Selection is the most common type of selection on the Web. The other type of selection, Object Selection, is when selection is made directly on objects within the interface.

Sometimes using a checkbox does not fit in with the style of interaction desired. Laszlo's WebTop mail allows the user to select messages by clicking anywhere in the row. The result is that the whole row gets highlighted to indicate selection (Figure 5.28).

Considerations

Desktop applications tend to use Object Selection. It is also natural that web-based mail applications that mimic desktop interactions employ this same style of selection. Instead of showing a control (like a checkbox), the object itself can be selected and acted on directly.

Object Selection can be extended by holding down the Shift key while clicking on a different item. The Command key (Macintosh) or Control key (Windows) can be used to individually add items in a discontinuous manner. The downside to this approach is that it is not obvious to use the modifier keys for extending the selection. Toggle Selection's use of toggle buttons makes the selection extension model completely obvious.

Flickr is a simple example of the keyboard being used to extend the selection in a web application. In the Organizr tool, multiple photos can be selected by using modifier keys to extend the selection (Figure 5.29).

Fig 5.29 Flickr allows for discontinuous selection by using the Command/Control key to extend

Desktop-style selection

For now Object Selection is not as common on the Web. Given that most sites have been content-oriented, there have been few objects to select. Also, with the Web's simple event model, Object Selection was not easy to implement. In typical web pages, keyboard events have rarely made sense since they are also shared with the browser. However, all of this is changing as the capabilities of web technologies continue to improve.

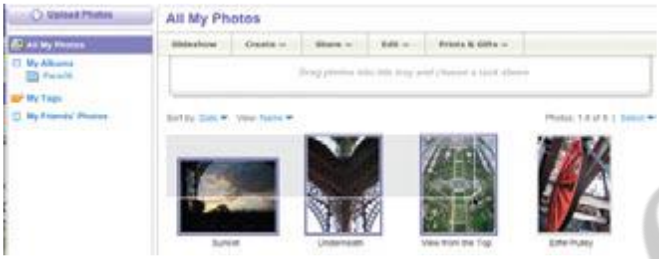


Fig 5.30 Yahoo! Photos 3.0 created a rich drag selection mechanism for selecting photos

Most desktop Object Selection interactions include ways to use the mouse to drag-select objects. Yahoo! Photos introduced this same type of object selection to its photo gallery (Figure 5.30). Individually clicking on a photo selects it. Using the Shift key and clicking also extends the selection. In addition, using the Control key and clicking discontinuously selects photos. And like most desktop applications, we can drag a selection box around a group of items to add them to the selected set (in this case, photos).

Best Practices for Object Selection

Here are some best practices to keep in mind:

- Use **Object Selection** when selectable elements can be dragged.
- Use **Object Selection** when the application will simulate desktop style interactions.
- Allow standard modifier key extensions (Shift to extend selection; Ctrl for discontinuous selection).
- If possible, degrade **Object Selection** to **Toggle Selection** when browser capabilities are limited.

3.2.4 Hybrid Selection

Mixing Toggle Selection and Object Selection in the same interface can lead to a confusing interface. Referring back to Yahoo! Bookmarks, we'll see an odd situation arise during drag and drop (Figure 5.31).

Fig 5.31 In Yahoo! Bookmarks, one item is selected, but two items can be dragged by dragging on the unselected item

Considerations

There are a few important issues to consider when using Hybrid Selection.

Confusing two models

In the left panel of Figure 3-14, one bookmark element is selected (notice the checkbox Toggle Selection). The second bookmark element ("Dr. Dobb's") is unselected (the checkbox is clear). In the right panel of Figure 3-14, clicking and dragging on the unselected bookmark element initiates a drag. The drag includes both the selected element and the unselected element. Since only one is shown as selected, this creates a confusing situation.

This occurs because three things are happening in the same space:

• Toggle Selection is used for selecting bookmarks for editing, deleting, etc.

• Object Selection is used for initiating a drag drop.

• Mouse click is used to open the bookmark on a separate page.

The problem is that more than one interaction idiom is applied to the same place on the same page. In this case, if we happen to try to drag, but instead click, we will be taken to a new page. And if we drag an unselected item, we now have two items selected for drag but only one shown as selected for other operations (Figure 3-14, right). This is definitely confusing. Simply selecting the item (automatically checking the box) when the drag starts would keep the selection model consistent in the interface. However, it might lead the user to expect a single click to also do the same (which it cannot since it opens the bookmark).

So, mixing the two selection models together can be problematic. However, there is a way to integrate the Toggle Selection and Object Selection and have them coexist peacefully as well as create an improved user experience.

Blending two models

Yahoo! Mail originally started with the Toggle Selection model (Figure 5.32). When the new Yahoo! Mail Beta was released, it used Object Selection exclusively (Figure 5.33). But since there are advantages to both approaches, the most recent version of Yahoo! Mail incorporates both approaches in a Hybrid Selection (Figure 3-17).

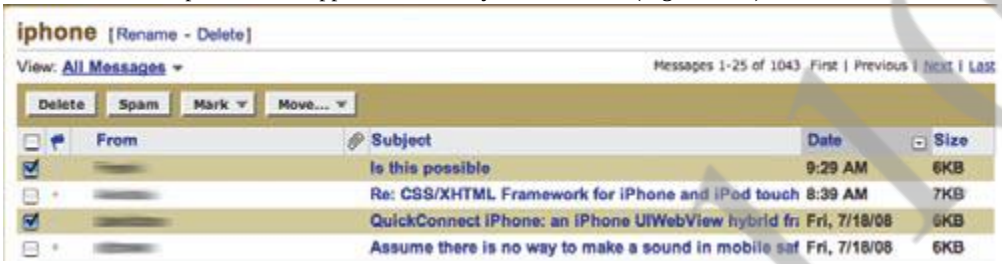


Fig 5.32 Yahoo! Mail Classic uses Toggle Selection; it also highlights selected rows, but rows can only be selected by clicking the message's checkbox

Fig 5.33 Yahoo! Mail Beta launched with Object Selection; no checkboxes were provided, and discontiguous selection could only be done by using keyboard modifiers

Fig 5.34 Yahoo! Mail now uses a hybrid approach; it incorporates both the Toggle Selection and the Object Selection patterns; Toggle Selection selects the message without loading the message in the viewing pane

Hybrid Selection brings with it the best of both worlds. We can use the checkbox selection model as well as normal row selection. We get the benefit of explicit selection and simplified multiple selections that Toggle Selection brings. And we get the benefit of interacting with the message itself and direct object highlighting.

Tip - Combining Toggle Selection and Object Selection is a nice way to bridge a common web idiom with a common desktop idiom.

There is an additional meaning applied to Toggle Selection versus Object Selection. Clicking on a row with the checkbox has the benefit of selecting the message without loading its contents in the message pane (think spam!). Clicking on a message it will load the contents in the message pane.

Best Practices for Hybrid Selection

Here are some best practices to keep in mind:

- Use checkbox selection to select an object without opening it.
- Use object selection to select and open an object.

****

5.3 CONTEXTUAL TOOLS

5.3.1 Interaction in Context

Desktop applications separate functionality from data. Menu bars, toolbars, and palettes form islands of application functionality. Either the user chooses a tool to use on the data or makes a selection and then applies the tool.

Early websites were just the opposite. They were completely content-oriented. Rich tool sets were not needed for simply viewing and linking to content pages. Even in e-commerce sites like Amazon or eBay, the most functionality needed was the hyperlink and "Submit" button.

Touch-based interfaces were the stuff of research labs and, more recently, interesting YouTube videos. But now they're as close as our phones. Most notably, the Apple iPhone brought touch to the masses (Figure 5.35).

Gesture-based interfaces seemed even further out. Yet these became reality with the Nintendo Wii.

With both gesture and touch-based interfaces, interaction happens directly with the content.

Fig 5.35 The Apple iPhone introduced touch-based interfaces to the consumer market

5.3.2 Fitts's Law

Fitts's Law is an ergonomic principle that ties the size of a target and its contextual proximity to ease of use. Bruce Tognazzini restates it simply as:

The time to acquire a target is a function of the distance to and size of the target.

In other words, if a tool is close at hand and large enough to target, then we can improve the user's interaction. Putting tools in context makes for lightweight interaction.

5.3.3 Contextual Tools

Contextual Tools are the Web's version of the desktop's right-click menus. Instead of having to right-click to reveal a menu, we can reveal tools in context with the content. We can do this in a number of ways:

i. **Always-Visible Tools**
   Tools Place Contextual Tools directly in the content.
ii. **Hover-Reveal Tools**
   Show Contextual Tools on mouse hover.
iii. **Toggle-Reveal Tools**
   A master switch to toggle on/off Contextual Tools for the page.
iv. **Multi-Level Tools**
   Progressively reveal actions based on user interaction.
v. **Secondary Menus**

Show a secondary menu (usually by right-clicking on an object).

5.3.4 Always-Visible Tools

The simplest version of Contextual Tools is to use Always-Visible Tools. Digg is an example of making Contextual Tools always visible (Figure 5.36).

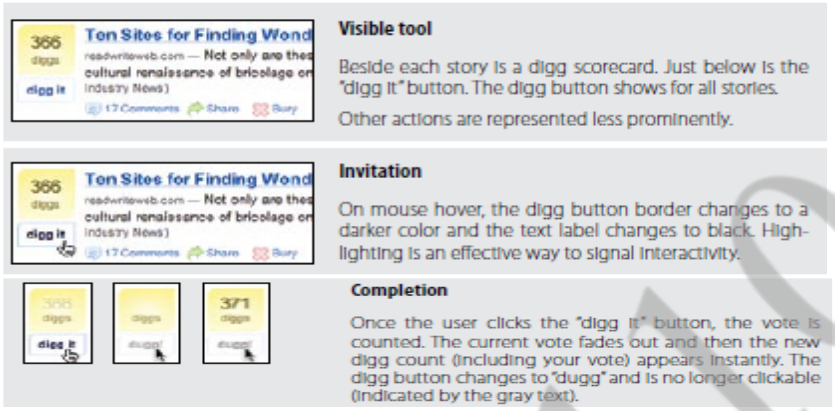


Fig 5.36 Digg's "dig it" button is a simple Contextual Tool that is always visible

Considerations

The "digg it" button and Digg scorecard provide Always-Visible Tools next to each story.

Clear call to action

Why not hide the tools and only reveal them when the mouse is over the story? Since digging stories is central to the business of Digg, always showing the tool provides a clear call to action. There are other actions associated with news stories (comments, share, bury, etc.) but they are represented less prominently. In the case of Digg, the designers chose to show these at all times. An alternate approach would be to hide them and show them on mouse hover (we will discuss this approach in the next section). It turns out that voting and rating systems are the most common places to make tools always visible. Netflix was the earliest to use a one-click rating system

Relative importance

The "digg it" action is represented as a button and placed prominently in the context of the story. The "bury it" action is represented as a hyperlink along with other "minor" actions just below the story. The contrast of a button and a hyperlink as well as its placement gives a strong indication as to the relative importance of each action.

Discoverability

Discoverability is a primary reason to choose Always-Visible Tools. On the flip side, it can lead to more visual clutter. In the case of Digg and Netflix, there is a good deal of visual space given to each item (story, movie). But what happens when the items we want to act on are in a list or table?

Generally Contextual Tools in a list work well when the number of actions is kept to a minimum. Gmail provides a single Always-Visible Tool in its list of messages—the star rating—for flagging emails (Figure 5.37).

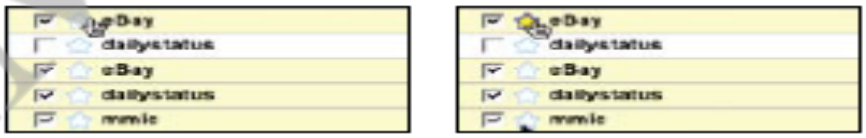


Fig 5.37 Google Mail uses contextual tools to flag favorites

Simply clicking the star flags the message as important. The unstarred state is rendered in a visually light manner, which minimizes the visual noise in the list.

Best Practices for Always-Visible Tools

Here are some best practices to keep in mind:

- Make your **Contextual Tools** always visible if it is important to make a prominent call to action.
- Keep visual clutter to a minimum.
- Keep the number of visual items to a minimum.

5.3.5 Hover-Reveal Tools

Instead of making Contextual Tools always visible, we can show them on demand. One way to do this is to reveal the tools when the user pauses the mouse over an object. The Hover-Reveal Tools pattern is most clearly illustrated by 37 Signal's Backpackit (Figure 5.38). To-do items may be deleted or edited directly in the interface. The tools to accomplish this are revealed on mouse hover.

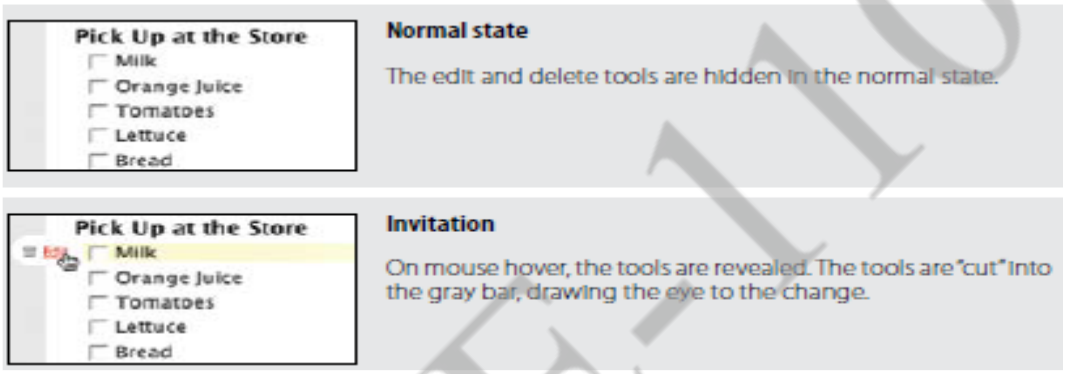


Fig 5.38 Backpackit reveals its additional tools on mouse hover

Considerations

The gray bar on the left is a nice visual reinforcement for the interaction. By allowing the tools to "cut" into the sidebar, the designers draw our eye to the available tools. The light yellow background draws attention to the to-do item being acted on. These two simple treatments make it clear which line has the focus and that additional tools have been revealed.

Visual noise

Showing the items on hover decreases the visual noise in the interface. Imagine if instead the delete and edit actions were always shown for all to-do items. Yahoo! Buzz reveals its tools on mouse hover for both its Top Searches (Figure 5.39) and Top Stories (Figure 5.40).

Fig 5.39 Yahoo! Buzz reveals additional tools for the top searches when the user hovers over each item

Fig 5.40 Yahoo! Buzz high lights the row and brings in additional tools

For Top Searches, it is important to keep the top-ten list as simple as possible. Showing tools would compete with the list itself. Since the actions "Search Results" and "Top Articles" (Figure 4-10, right) are less important, they are revealed on hover. The actions may be important, but making the content clear and readable is a higher priority.

Similarly, for Top Stories, Yahoo! Buzz shows only "Share", "Post", and "Buzz Down" tools on hover. "Buzz Up" is shown at all times, but gets extra visual treatment on mouse hover (Figure 4-11, right). "Buzz Up" is important enough to show at all times, but can be toned down when not the focus.

Discoverability

A serious design consideration for Hover-Reveal Tools is just how discoverable the additional functionality will be. While the Contextual Tools are revealed on hover, the checkbox is always visible for each to-do item. To check off an item, users have to move the mouse over it. When they do, they will discover the additional functionality.

Contextual Tools in an overlay

Sometimes there are several actions available for a focused object. Instead of placing tools beside the object being acted on, the revealed tools can be placed in an overlay. However, there can be issues with showing contextual tools in an overlay:

1. Providing an overlay feels heavier. An overlay creates a slight contextual switch for the user's attention. 2. The overlay will usually cover other information—information that often provides context for the tools being offered.
3. Most implementations shift the content slightly between the normal view and the overlay view, causing the users to take a moment to adjust to the change.
4. The overlay may get in the way of navigation. Because an overlay hides at least part of the next item, it becomes harder to move the mouse through the content without stepping into a "landmine."

5.3.6 Toggle-Reveal Tools

A variation on the two previous approaches is to not show any Contextual Tools until a special mode is set on the page. A good example of Toggle-Reveal Tools is in Basecamp's category editing (Figure 5.41).

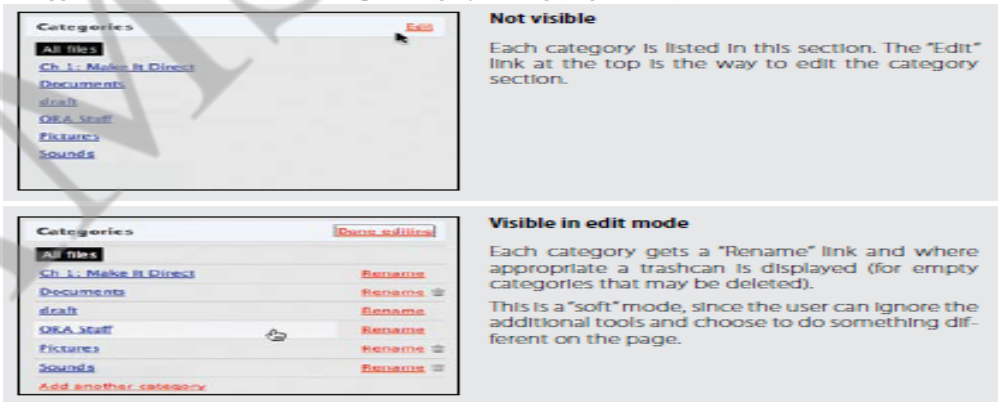


Fig 5.41 Basecamp reveals category-editing tools only when the edit mode is turned on for the area

Considerations

Here are a few considerations to keep in mind when using Toggle-Reveal Tools.

Soft mode

Generally, it is a good thing to avoid specific modes in an interface. However, if a mode is soft it is usually acceptable. By "soft" we mean the user is not trapped in the mode. With Basecamp, the user can choose to ignore the tools turned on. It just adds visual noise and does not restrict the user from doing other actions. This is a nice way to keep the interaction lightweight.

It is common, however, to want to click through and see the contents of a category (the category is always hyperlinked). Hence, make it readable and easily navigable in the normal case—but still give the user a way to manage the items in context. Google Reader could potentially be improved in this manner. In the current interface, clicking "Manage Subscriptions" takes the user to another page to edit subscriptions. One possible change is the addition of an "edit" button that toggles in a set of context tools for each subscription (Figure 5.42). This would allow the user to rename and unsubscribe without leaving the context of the reading pane.

Fig 5.42 Adding an "edit" link to Google Reader's feed list and toggling in common actions could potentially make it easier to manage subscriptions

5.3.7 Multi-Level Tools

Contextual Tools can be revealed progressively with Multi-Level Tools. Songza provides a set of tools that get revealed after a user clicks on a song. Additional tools are revealed when hovering over the newly visible tools.

Considerations

Songza reveals the four options "play", "rate", "share", and "add to playlist" after the user clicks on a song title. Hovering over "share" or "rate" reveals a secondary set of menu items (Figure 5.43, center).

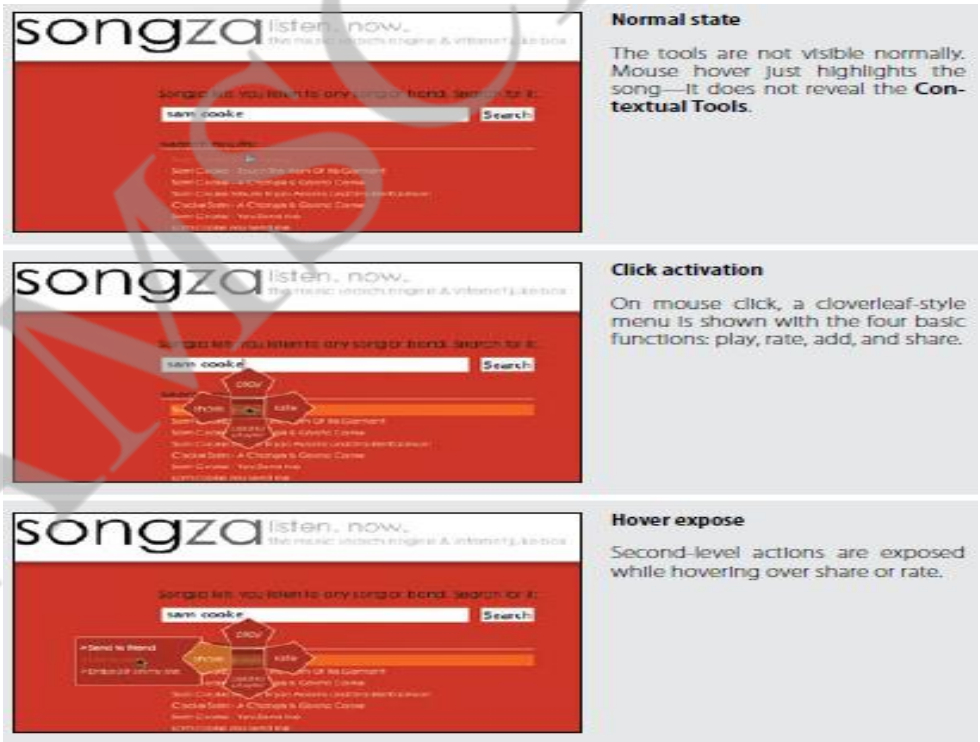


Fig 5.43 Songza uses a multi-level contextual tool menu

Radial menus

Radial menus such as in Songza have been shown to have some advantages over more traditional menus. First, experienced users can rely on muscle memory rather than having to look directly at the menu items. Second, the proximity and targeting size make the menu easy to navigate since the revealed menu items are all equally close at hand (recall Fitts's Law). The one potential downside to this approach is that rating a song requires several steps: an initial click on the song, moving the mouse over the "rate" menu item, then clicking either the thumbs up or thumbs down option. If rating songs was an important activity, the extra effort might prevent some users from doing so. An alternate approach would be to replace "rate" directly with the thumbs up and the thumbs down options.

Activation

Another interesting decision Songza made was to not activate the radial menu on hover. Instead, the user must click on a song to reveal the menu. Activating on click makes the user intent more explicit. Making activation more explicit avoids the issues described earlier in the Hover and Cover anti-pattern. The user has chosen to interact with the song. Conversely, with a mouse hover, it's never quite clear if the user meant to activate the menu or just happened to pause over a song title.

Default action

However, this does mean there is no way to start a song playing with just a simple click. Playing a song requires moving to the top leaf. One possible solution would be to place the "play" option in the middle of the menu (at the stem) instead of in one of the leaves. Clicking once would activate the menu. Clicking a second time (without moving the mouse) would start playing the song. This interaction is very similar to one commonly used in desktop applications: allowing a double-click to activate the first item (default action) in a right-click menu.

Contextual toolbar

Picnik is an online photo-editing tool that integrates with services like Flickr. In all, there are six sets of tools, each with a wide range of palette choices. Picnik uses Multiple-Level Tools to expose additional functionality. By wrapping the photo with tools in context and progressively revealing the levels of each tool, Picnik makes editing straightforward (Figure 5.44).

Fig 5.44 Picnik wraps layers of Contextual Tools around the image being edited

Muttons

Another variation on Multi-Level Tools is the "mutton" (menu + button = mutton). Muttons are useful when there are multiple actions and we want one of the actions to be the default. Yahoo! Mail uses a mutton for its "Reply" button (Figure 5.45).

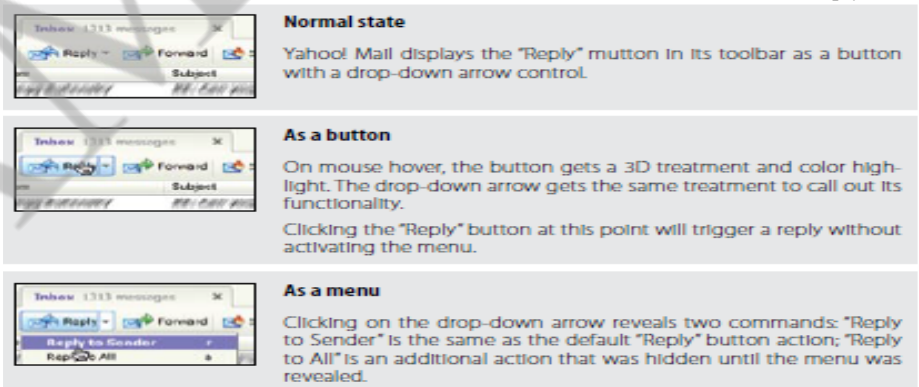


Fig 5.45 Yahoo! Mail's "Reply" button looks like a drop-down when hovered over; clicking "Reply" relies to sender, and clicking the drop-down offers the default action as well as "Reply to All"

Clicking “Reply” performs the individual reply. To reply to all, the menu has to be activated by clicking on the drop-down arrow to show the menu.

Muttons are used to:

- Provide a default button action (“Reply to Sender”)
- Provide a clue that there are additional actions
- Provide additional actions in the drop-down

If muttons are not implemented correctly, they can be problematic for those using accessibility technologies. Because an earlier version of Yahoo! Mail did not make the mutton keyboard accessible, Yahoo!’s accessibility guru, Victor Tsaran, was convinced that there was no “Reply to All” command in the Yahoo! Mail interface. **Anti-**

Secondary Menu

Desktop applications have provided **Contextual** Tools for a long time in the form of **Secondary Menus**. These menus have been rare on the Web. Google Maps uses a secondary menu that is activated by a right-click on a route. It shows additional route commands (Figure 5.46).

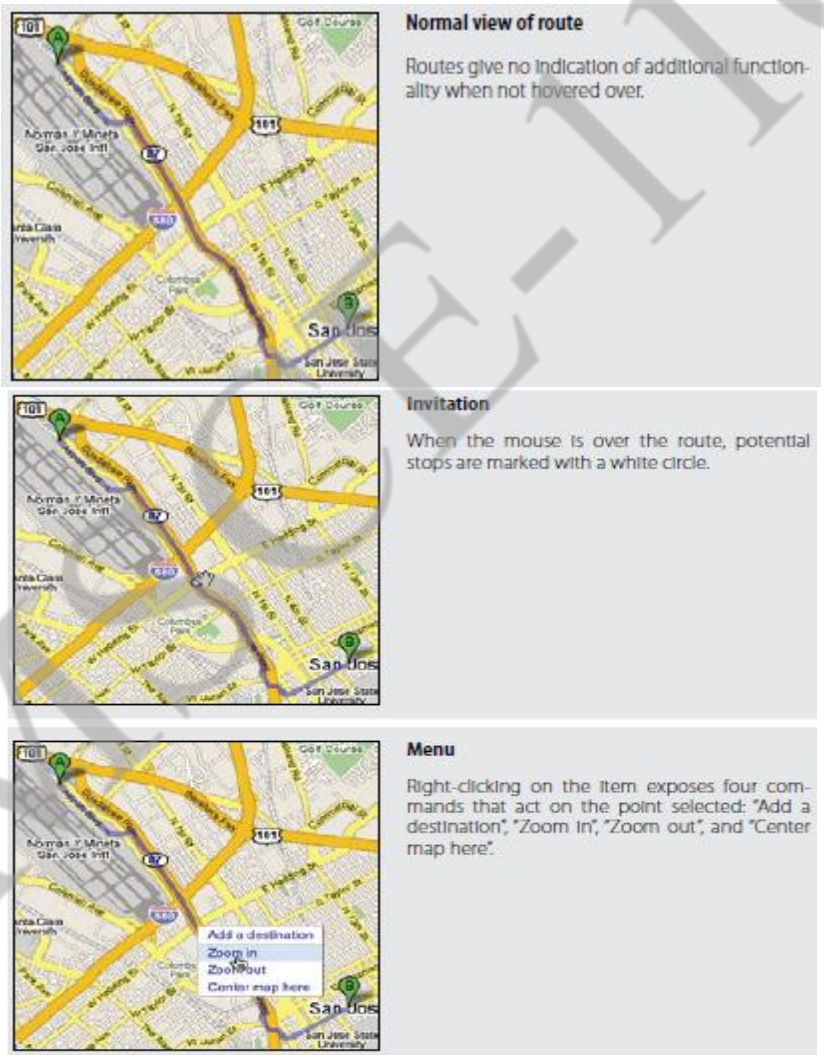


Fig 5.46 Google Maps uses a right-click menu to add new route stops or to adjust the map around the current point on the route

Considerations

Secondary Menus have not been common in web applications.

Conflict with browser menu

One problem is the browser inserts its own right-click menu. Replacing the menu in normal content areas can confuse users, as they may not know if the standard browser menu or the application-specific menu will be shown. It will depend on whether it is clear that an object exists in the interface (as in the route line above), and if the menu is styled differently enough to disambiguate the menus.

Discoverability

As a general rule, never put anything in the Secondary Menu that can't be accomplished elsewhere. Secondary Menus are generally less discoverable. More advanced items or shortcuts, however, can be placed in the Secondary Menu as an alternate way to accomplish the same task.

Accessibility

Right-click is not the only way to activate a Secondary Menu. We can activate the menu by holding down the mouse for about one second. This provides a more accessible approach to popping up a Secondary Menu. This technique is used in the Macintosh Dock. Clicking and holding down on an application in the dock will reveal the Secondary Menu without requiring a right-click activation.

Acting on multiple objects Keep in mind that all of the other Contextual Tools presented in this chapter have a limitation on the number of items they can operate on. Always-Visible Tools, Hover-Reveal Tools, Toggle-Reveal Tools, and Multi-Level Tools all operate on a single item at a time (even Toggle-Reveal Tools just shows a tool per item). Secondary Menus are different. They can be combined with a selection model (as described in Chapter 3) to perform actions on selected set of items.

Four ways to keep the user on the page:

Overlays

Instead of going to a new page, a mini-page can be displayed in a lightweight layerover the page.

Inlays

Instead of going to a new page, information or actions can be inlaid within the page.

Virtual Pages

By revealing dynamic content and using animation, we can extend the virtual spaceof the page.

Process Flow

Instead of moving from page to page, sometimes we can create a flow within a pageitself.

****

5.4 OVERLAYS

Overlays are really just lightweight pop ups. We use the term *lightweight* to make a cleardistinction between it and the normal idea of a *browser pop up*. Browser pop ups are createdas a new browser window (Figure 5.47).

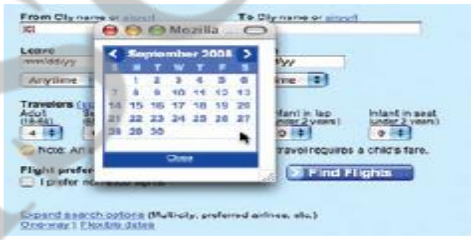


Fig 5.47 If Orbitz used a browser pop-up window for its calendar chooser, this is how it might look

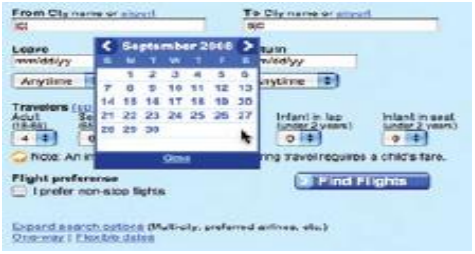


Fig 5.48 Orbitz uses a lightweight DHTML overlay for its calendar chooser, since it does not require the overhead of a separate browser window, it can pop up quickly and is better integrated into the page visually

Lightweight overlays are shown within thebrowser page as an overlay (Figure 5.48).

Older style browser pop ups are undesirablebecause:
Browser pop ups display a new browser window.
As a result these windows often taketime and a sizeable chunk of system resources to create.Browser pop ups often display browser interface controls (e.g., a URL bar). Due tosecurity concerns, in Internet Explorer 7 the URL bar is a permanent fixture on anybrowser pop-up window.
By using either Flash or Ajax-style techniques (Dynamic HTML), a web applicationcan present a pop up in a lightweight overlay within the page itself. This has distinctadvantages:

- Lightweight overlays are just a lightweight in-page object. They are inexpensive tocreate and fast to display.
- The interface for lightweight overlays is controlled by the • web application and notthe browser.
- There is complete control over the visual style for the overlay. This allows the overlayto be more visually integrated into the application's interface.
- Lightweight overlays can be used for asking questions, obtaining input, introducing features,indicating progress, giving instructions, or revealing information. They can be activateddirectly by user events (e.g., clicking on an action, hovering over objects) or beprovided by the web application at various stages in the completion of an action.

We will look at three specific types of overlays: Dialog Overlays, Detail Overlays, andInput Overlays.

Dialog Overlay

Dialog Overlays replace the old style browser pop ups. Netflix provides a clear exampleof a very simple Dialog Overlay. In the "previously viewed movies for sale" section, a usercan click on a "Buy" button to purchase a DVD. Since the customer purchasing the DVDis a member of Netflix, all the pertinent shipping and purchasing information is alreadyon record. The complete checkout experience can be provided in a single overlay (Figure
5.49).

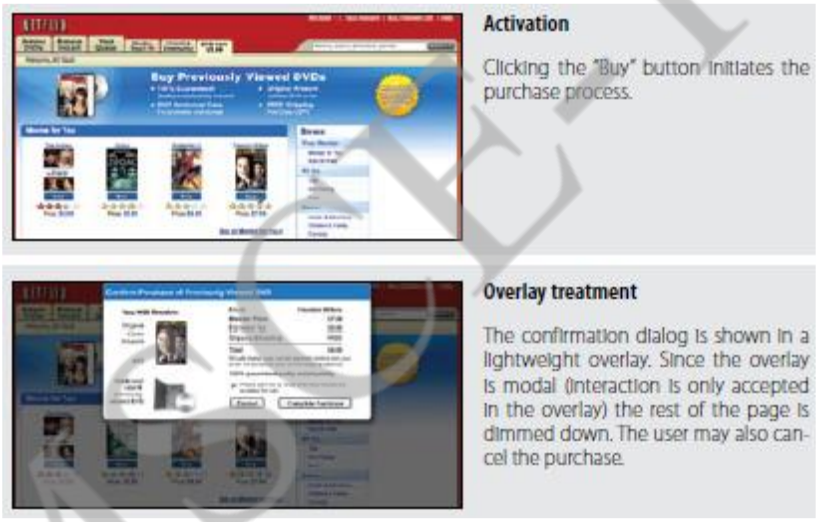


Fig 5.49 Netflix uses a lightweight pop up to confirm a previously viewed DVD purchase; in addition, it uses the Lightbox Effect to indicate modality

Considerations

Because the overlay is a lightweight pop up, the confirmation can be displayed more rapidlyand the application has complete control over its look and placement.

Lightbox Effect

One technique employed here is the use of a Lightbox Effect. In photography a lightboxprovides a backlit area to view slides. On the Web, this technique has come to mean bringingsomething into view by making it brighter than the background. In practice, this isdone by dimming down the background.

The Lightbox Effect is useful when the Dialog Overlay contains important informationthat the user should not ignore. Both the Netflix Purchase dialog and the Flickr Rotatedialog are good candidates for the Lightbox Effect. If the overlay contains optional information,then the Lightbox Effect is overkill and should not be used.

Modality

Overlays can be modal or non-modal. A modal overlay requires the user to interact withit before she can return to the application.

The Lightbox Effect emphasizes that we are in a separate mode. As a consequence, it isnot needed for most non-modal overlays. As an example, refer back to theOrbitz calendar pop up. Since the overlay is really more like an in-page widget, it wouldnot be appropriate to make the chooser feel heavier by using a Lightbox Effect.

Staying in the flow

Overlays are a good way to avoid sending a user to a new page. This allows the user tostay within the context of the original page. However, since overlays are quick to displayand inexpensive to produce, sometimes they can be tempting to use too freely, and in theprocess, may actually break the user's flow.

Detail Overlay

The second type of overlay is somewhat new to web applications. The Detail Overlay allowsan overlay to present additional information when the user clicks or hovers over alink or section of content. Toolkits now make it easier to create overlays across differentbrowsers and to request additional information from the server without refreshing thepage.

Taking another example from Netflix, information about a specific movie is displayed asthe user hovers over the movie's box shot (Figure 5.50).

Fig 5.50 Netflix shows "back of the box" information in an overlay as the user hovers over a movie's box shot

Activation

The overlay is displayed when the mouse hovers a box shot. There is about a halfsecond delay after the user pauses over a movie. The delay on activation prevents users from accidentally activating the overlay as they move the cursor around the screen. Once the user moves the cursor outside the box shot, the overlay is removed immediately. Removing it quickly gives the user a fast way to dismiss it without having to look for a Close box.

Anti-pattern: Mouse Traps

It is important to avoid activating the Detail Overlay too easily. We have seen usabilitystudies that removed the delay in activation, and users reported that the interface was "toonoisy" and "felt like a trap". We label this anti-pattern the Mouse Trap.

The reasoning for this is not clear, but Amazon uses the Mouse Trap anti-pattern in one of its "associate widgets". Original Motion Picture Soundtrack"activates an overlay providing information on the soundtrack and a purchase option.

Input Overlay

Input Overlay is a lightweight overlay that brings additional input information for eachfield tabbed into. American Express uses this technique in its registration for premiumcards such as its gold card (Figure 5.51).

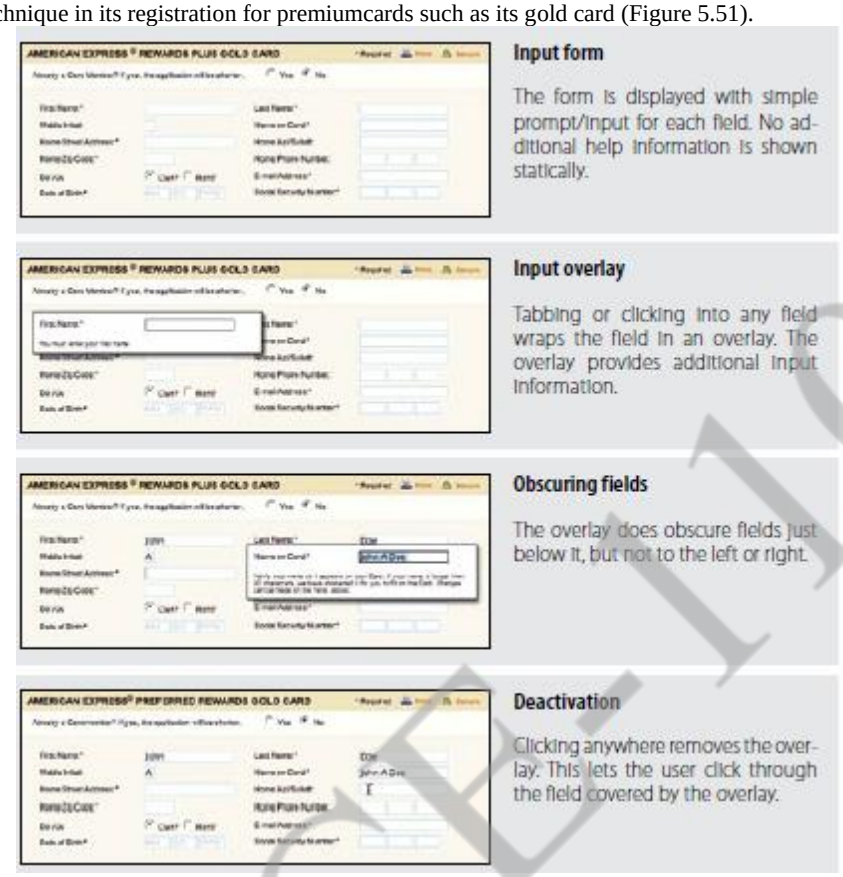


Fig 5.51 American Express provides Input Overlays to guide the user through the signup process

Considerations

There are a few things to keep in mind when using Input Overlays.

Clear focus

As the user tabs or clicks from field to field, the field gets wrapped in an overlay. The overlaycontains additional input help information. This allows the normal display of the formto be displayed in a visually simple manner (just prompts and inputs). The overlay createsfocus on the given input field. Instead of seeing an ocean of inputs, the user is focused onjust entering one field.

Display versus editing

When the Input Overlay is shown, the prompt is displayed in exactly thesame manner as when the overlay doesn't show. This is critical, as it makes the overlay feeleven more lightweight. If the overlay prompt were bold, for example, the change wouldbe slightly distracting and take the focus away from input. The only difference betweenthe non-overlay field and the overlay version is a slightly thicker input field border. Thisdraws the eye to the task at hand—input.

- *Field traversal*
- *Tab navigation*
- *One-click deactivation*

****

5.5 INLAYS

Not every bit of additional control, information, or dialog with the user needs to be anoverlay. Another approach is to inlay the information directly within the page itself. Todistinguish from the pop-up overlay, we call these in-page panels Inlays.

Dialog Inlay

A simple technique is to expand a part of the page, revealing a dialog area within the page.

The BBC recently began experimenting with using a Dialog Inlay as a way to reveal customizationcontrols for its home page (Figure 5.52).

Fig 5.52 The BBC home page put its customization tools in an inlay that slides out when activated

Considerations

Of course an overlay could have been used instead. However, the problem with overlaysis that no matter where they get placed, they will end up hiding information. Inlays getaround this problem by inserting themselves directly into the context of the page.

In context

This Dialog Inlay is similar to a drawer opening with a tray of tools. Instead of being takento a separate page to customize the home page appearance, the user can make changesand view the effects directly. The advantage is the ability to tweak the page while viewingthe actual page.

List Inlay

Lists are a great place to use Inlays. Instead of requiring the user to navigate to a new pagefor an item's detail or popping up the information in an Overlay, the information can beshown with a List Inlay in context. The List Inlay works as an effective way to hide detailuntil needed—while at the same time preserving space on the page for high-level overviewinformation.

Google Reader provides an expanded view and a list view for unread blog articles. In thelist view, an individual article can be expanded in place as a List Inlay (Figure 5.53).

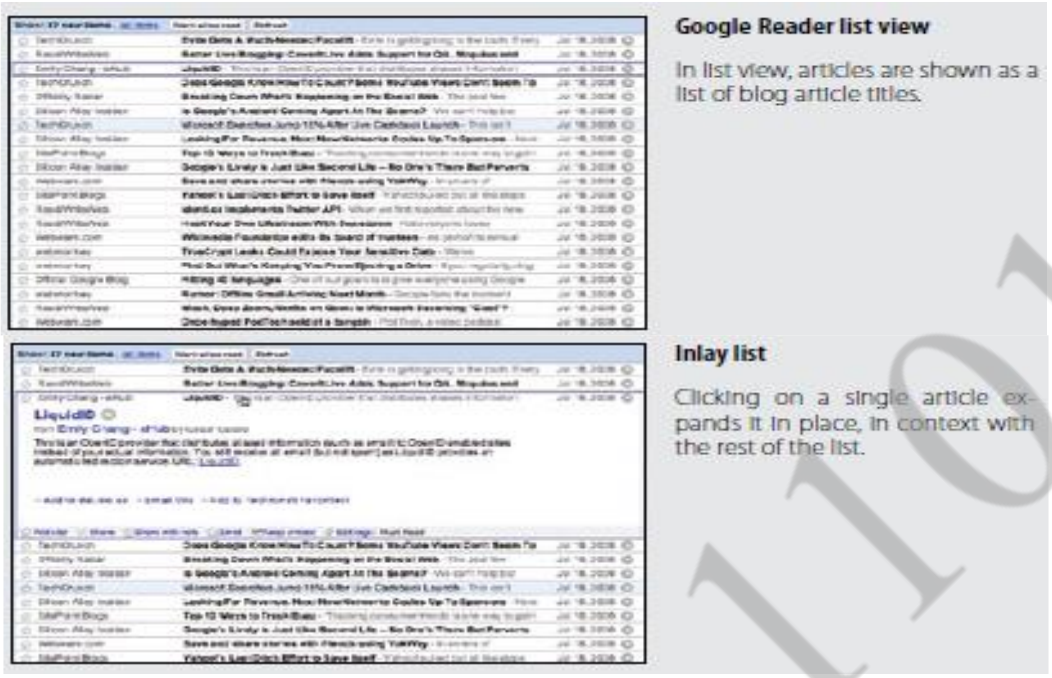


Fig 5.53 In list view, Google Reader shows all articles as a collapsed list – except for the one that is currently selected

Detail Inlay

A common idiom is to provide additional detail about items shown on a page. We saw thiswith the example of the Netflix movie detail pop up in Chapter 5 (Figure 5-8). Hoveringover a movie revealed a Detail Overlay calling out the back-of-the-box information.

Details can be shown inline as well. Roost allows house photos to be viewed in-context fora real estate listing with a Detail Inlay.

Combining inlays and overlays

Roost's solution was to combine several patterns. First, it uses the Hover Reveal, a ContextualTools pattern, to reveal a set of tools when the user hovers over a listing. Second, ituses the Detail Inlay pattern to show a carousel of photos when the user clicks on the "Viewphotos" link. And finally, it uses a Detail Overlay to blow up a thumbnail when clicked on.

****

5.6 VIRTUAL PAGES

Patterns that support virtual pages include:

- Virtual Scrolling
- Inline Paging
- Scrolled Paging
- Panning
- Zoomable User Interface

Virtual Scrolling

The traditional Web is defined by the "page." In practically every implementation of websites(for about the first 10 years of the Web's existence) pagination was the key way to getto additional content. Of course, websites could preload data and allow the user to scrollthrough it. However, this process led to long delays in loading the page. So most sites keptit simple: go fetch 10 items and display them as a page and let the user request the nextpage of content. Each fetch resulted in a page refresh.

The classic example of this is Google Search. Each page shows 10 results. Moving throughthe content uses the now-famous Google pagination control.Another approach is to remove the artificial page boundaries created by paginating thedata with Virtual Scrolling.

Loading status

There are a few downsides to the Yahoo! Mail version of Virtual Scrolling. First, if theloading is slow, it spoils the illusion that the data is continuous. Second, since the scrollbardoes not give any indication of where users are located in the data, they have to guess howfar down to scroll. A remedy would be to apply a constantly updating status while the useris scrolling.

Progressive loading

Microsoft has applied Virtual Scrolling to its image search. However, it implements it ina different manner than Yahoo! Mail. Instead of all content being virtually loaded (andthe scrollbar reflecting this), the scrollbar reflects what has been loaded.

Inline Paging

By only switching the content in and leaving the rest of the page stable,we can create an Inline Paging experience. This is what Amazon's Endless.com site doeswith its search results.

Natural chunking

Inline Paging can also be useful when reading news content online. The *InternationalHerald Tribune* applied this as a way to page through an article while keeping the surroundingcontext visible at all times.

Back button

The biggest issue with Inline Paging is whether the back button works correctly. Onecriticism of Endless.com is that if the user pages through search results and then hits theback button, it jumps to the page just before the search. This unexpected result could befixed by making the back button respect the virtual pages as well. This is the way Gmail handles the back button. Clicking back moves we through the virtual pages.

Interactive content loading

The iPhone employs inline paging when displaying search results in the iTunes store.

Scrolled Paging: Carousel

Besides Virtual Scrolling and Virtual Paging, there is another option. We can combineboth scrolling and paging into Scrolled Paging. Paging is performed as normal. But insteadthe content is "scrolled" into view.

The Carousel pattern takes this approach. A Carousel provides a way to page-in moredata by scrolling it into view. On one hand it is a variation on the Virtual Scrolling pattern.In other ways it is like Virtual Paging since most carousels have paging controls. Theadditional effect is to animate the scrolled content into view.

Time-based

Carousels work well for time-based content. Flickr employs a Carousel to let users navigateback and forth through their photo collection.

Animation direction

Inexplicably, AMCtheatres.com animates its Carousel the opposite way. This leads to aconfusing experience, and it's harder to know which control to click.

Virtual Panning

A great place for Virtual Panning is on a map. Google Maps allows us to pan in anydirection by clicking the mouse down and dragging the map around (Figure 5.54).

Fig 5.54 Google Maps creates a virtual canvas; one tool that helps with that illusion is the ability to pan from area to area

Zoomable User Interface

A Zoomable User Interface (ZUI) is another way to create a virtual canvas. Unlike panningor flicking through a flat, two-dimensional space, a ZUI allows the user to also zoomin to elements on the page. This freedom of motion in both 2D and 3D supports the conceptof an infinite interface.

Practically speaking, ZUIs have rarely been available in everyday software applications,much less on the Web. But with more advanced features added to Flash and the adventof Silverlight, this type of interface is starting to emerge and may be commonplace in thenot-too-distant future.

Paging Versus Scrolling

Leading web designers and companies have taken different approaches to solving thesame problems. Yahoo! Mail chose Virtual Scrolling. Gmail chose Inline Paging.

How do we choose between paging and scrolling? While there are no hard and fast rules,here are some things to consider when making the decision:

- When the data feels "more owned" by the user—in other words, the data is not transientbut something users want to interact with in various ways. If they want to sort

it, filter it, and so on, consider Virtual Scrolling (as in Yahoo! Mail).

- When the data is more transient (as in search results) and will get less and less relevantthe further users go in the data, Inline Paging works well (as with the iPhone).
- For transient data, if we don't care about jumping around in the data to specific sections,consider using Virtual Scrolling (as in Live Image Search).
- If we are concerned about scalability and performance, paging is usually the bestchoice. Originally Microsoft's Live Web Search also provided a scrollbar. However,the scrollbar increased server-load considerably since users are more likely to scrollthan page.
- If the content is really continuous, scrolling is more natural than paging.
- If we get our revenue by page impressions, scrolling may not be an option for our business model.
- If paging causes actions for the content to become cumbersome, move to a scrollingmodel. This is an issue in Gmail. The user can only operate on the current page.

Changing items across page boundaries is unexpected. Changing items in a continuousscrolled list is intuitive.

****

5.7 PROCESS FLOW

Process Flow Google Blogger

The popular site Google Blogger generally makes it easy to create and publish blogs. Onething it does not make easy, though, is deleting comments that others may leave on our blog. This is especially difficult when we are the victim of hundreds of spam commentsleft by nefarious companies hoping to increase their search ranking.

1. Scroll to find the offending comment.
2. Click the trash icon to delete the comment.
3. After page refreshes, click the "Remove Forever" checkbox.
4. Click the "Delete Comment" button.
5. After the page refreshes, click the link to return to my blog article.
6. Repeat steps 1–5 for each article with spam comments.

The Magic Principle

Alan Cooper discusses a wonderful technique for getting away from a technology-drivenapproach and discovering the underlying mental model of the user. He calls it the "magicprinciple."* Ask the question, "What if when trying to complete a task the user could invokesome magic?" For example, let's look at the problem of taking and sharing photos.

The process for this task breaks down like this:

• Take pictures with a digital camera.
• Sometime later, upload the photos to a photo site like Flickr. This involves:
— Finding the cable.
— Starting iTunes.
— Importing all photos.
—Using a second program, such as Flickr Uploadr, to upload the photos to Flickr.
— Copying the link for a Flickr set (which involves first locating the page for theuploaded set).

- Send the link in email to appropriate friends.

If some magic were invoked, here is how it might happen:

• The camera would be event-aware. It would know that is your daughter's eighth birthday.
• When finished taking pictures of the event, the camera would upload the pictures toFlickr.
• Flickr would notify family and friends that the pictures of the birthday party areavailable.

Thinking along these lines gets some of the artifacts out of the way. Of course the magiccould be taken to the extreme: just eliminate the camera altogether! But by leaving someelements in the equation, the potentially unnecessary technology pieces can be exposed.How about the cable? What if the camera could talk magically to the computer?

Process Flow patterns:

- Interactive Single-Page Process
- Inline Assistant Process
- Configurator Process
- Overlay Process
- Static Single-Page Process

Interactive Single-Page Process

Consumer products come in a variety of shapes, sizes, textures, colors, etc. Online shopperswill not only have to decide that they want shoes, but do they want blue suede shoes?And what size and width do they want them in? In the end the selection is constrainedby the available inventory. As the user makes decisions, the set of choices gets more andmore limited.

This type of product selection is typically handled with a multi-page workflow. On onepage, the user selects a shirt and its color and size. After submitting the choice, a new pageis displayed. Only when the user arrives at this second page does he find out that the "truenavy" shirt is not available in the medium size.

Benefits

Adobe calls out the Broadmoor one-page reservation interface in its Adobe Showcase.

Itstates the benefits of this method:

- Reduces entire reservation process to a single screen.
- Reduces the number of screens in the online reservation process from five to one.

Other online reservation applications average 5 to 10 screens.

- Seventy-five percent of users choose OneScreen in favor of the HTML version.
- Allows users to vary purchase parameters at will and immediately view results.
- Reduces the time it takes to make a reservation from at least three minutes to lessthan one.

Additionally, Adobe notes that conversion rates (users who make it through the reservationprocess) are much higher with the Interactive Single-Page Process.

Inline Assistant Process

The Gap employed an Inline AssistantProcess pattern for its shopping cart when it re-launched its site a few years back. Blending quick and easy with the additional step

Dialog Overlay Process

The Netflix approach just described uses a Dialog Overlay Process to encapsulatea multi-step flow inside a Dialog Overlay. Discover.com recently expanded its account section with a more detailed profile. The profile captures things like our payment date, mobile fraud alerts, paperless statements, and general contact information. The overlay pops up when we first enter our account.

Configurator Process

Sometimes a Process Flow is meant to invoke delight. In these cases, it is the engagementfactor that becomes most important. This is true with various Configurator Process interfaces on the Web. We can see this especially at play with car configurators. Porsche provides a configurator that allows users to build their own Porsche.

Static Single-Page Process

Just put the complete flow on one page in a Static Single-Page Process. The user sees allthe tasks needed to complete the full process. This can be both good and bad. Seeing just one step to complete the process can encourage users to finish the task. But if the single step seems too long or too confusing, the user will most likely bail out of the process early.

In other words, if placing all the tasks on a single page is enough to cause the user to bail out, it is not a good idea. In the case of the Apple store, each item is optionally set, and it's just a single click to include or exclude an item from the purchase.

****

Case studies for Web interface Design

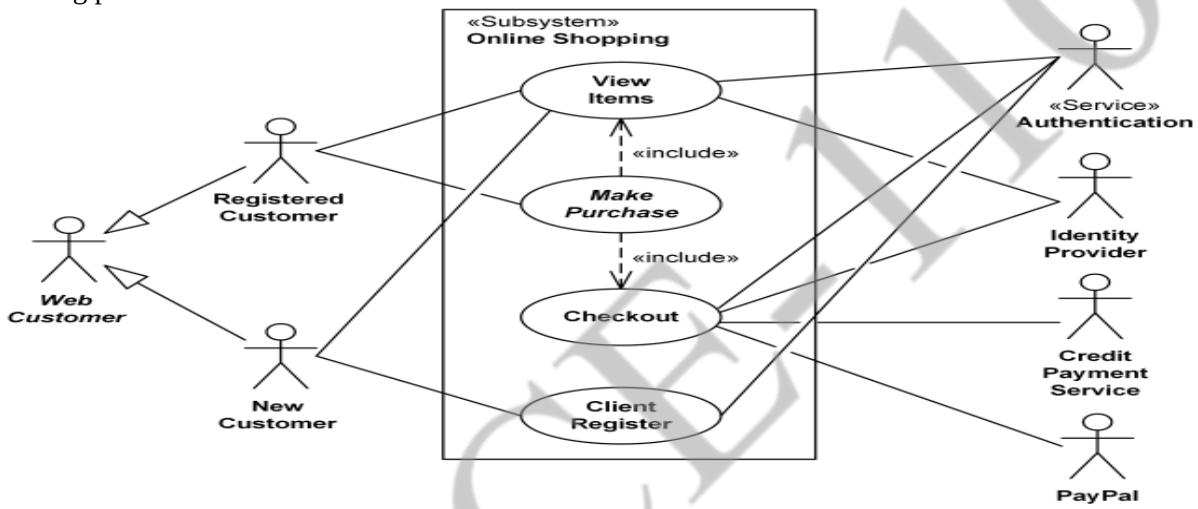
Online Shopping

UML Use Case Diagram Example

Web Customer **actor** uses some web site to make purchases online. Top level **use cases** are **View Items**, **Make Purchase** and **Client Register**.

View Items use case could be used by customer as top level use case if customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customer to register on the web site, for example to get some coupons or be invited to private sales.

Note, that **Checkout** use case is **included use case** not available by itself - checkout is part of making purchase.



Online shopping UML use case diagram example - top level use cases.

View Items use case is **extended** by several optional use cases - customer may search for items, browse catalog, view items recommended for him/her, add items to shopping cart or wish list. All these use cases are extending use cases because they provide some optional functions allowing customer to find item.

Customer Authentication use case is **included** in **View Recommended Items** and **Add to Wish List** because both require the customer to be authenticated. At the same time, item could be added to the shopping cart without user authentication.

Online shopping UML use case diagram example - view items use case.

Checkout use case includes several required uses cases. Web customer should be authenticated. It could be done through user login page, user authentication cookie ("Remember me") or Single Sign-On (SSO). Web site authentication service is used in all these use cases, while SSO also requires participation of external identity provider.

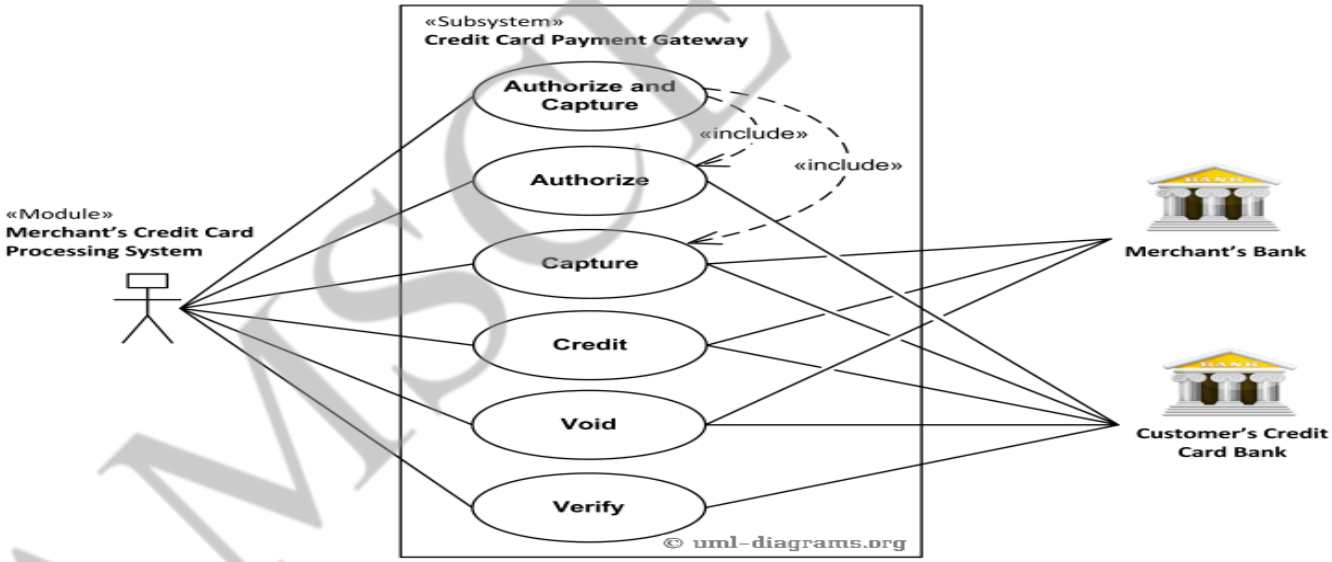
Checkout use case also includes **Payment** use case which could be done either by using credit card and external credit payment service or with PayPal.

This UML use case diagram example shows some use cases for a system which processes credit cards. **Credit Card Processing System** (aka Credit Card Payment Gateway) is a **subject**, i.e. system under design or consideration. Primary **actor** for the system is a **Merchant's Credit Card Processing System**. The merchant submits some credit card transaction request to the credit card payment gateway on behalf of a customer. Bank which issued customer's credit card is actor which could approve or reject the transaction. If transaction is approved, funds will be transferred to merchant's bank account.

Authorize and Capture use case is the most common type of credit card transaction. The requested amount of money should be first authorized by **Customer's Credit Card Bank**, and if approved, is further submitted for settlement. During the settlement funds approved for the credit card transaction are deposited into the **Merchant's Bank** account.

In some cases, only **authorization** is requested and the transaction will not be sent for settlement. In this case, usually if no further action is taken within some number of days, the authorization expires. Merchants can submit this request if they want to verify the availability of funds on the customer's credit card, if item is not currently in stock, or if merchant wants to review orders before shipping.

Capture (request to capture funds that were previously authorized) use case describes several scenarios when merchant needs to complete some previously authorized transaction - either submitted through the payment gateway or requested without using the system, e.g. using voice authorization.

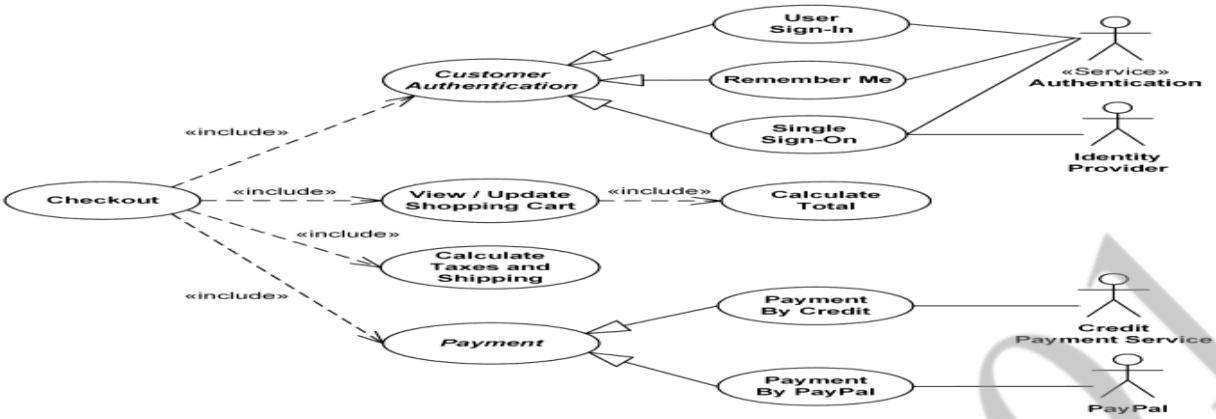


UML use case diagram example for a credit cards processing system.

Credit use case describes situations when customer should receive a refund for a transaction that was either successfully processed and settled through the system or for some transaction that was not originally submitted through the payment gateway.

Void use case describes cases when it is needed to cancel one or several related transactions that were not yet settled. If possible, the transactions will not be sent for settlement. If the Void transaction fails, the original transaction is likely already settled.

Verify use case describes zero or small amount verification transactions which could also include verification of some client's data such as address.

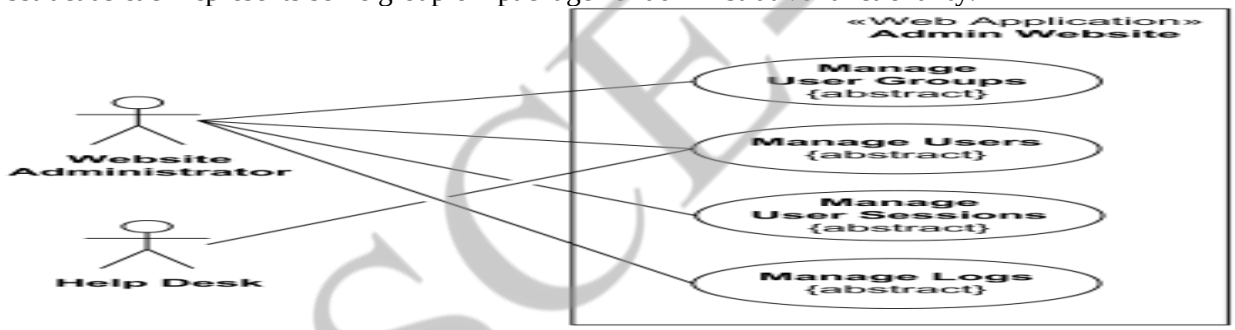


Online shopping UML use case diagram example - checkout, authentication and payment use cases.

xcept for administrators, some part of the administrative interfaces should be also available to the Help desk staff, as they need to be able to assist customers having issues while using the customer oriented website.

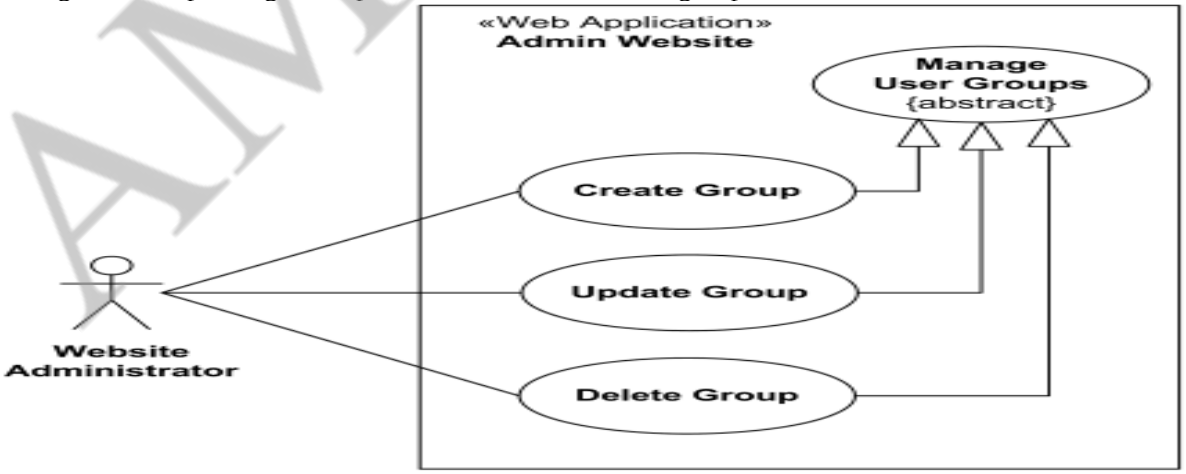
Top level use case diagram below shows some administrative functions that administration website could provide.

Two **actors** using administrative interfaces are **Website Administrator** and **Help Desk**. Help Desk uses a subset of functions available to the Website Administrator. All top level **use cases** shown are abstract as each represents some group or "package" of administrative functionality.



Top level use case diagram for the administration website.

Manage User Groups abstract use case is specialized by **Create Group**, **Update Group**, and **Delete Group** use cases. The idea is that website administrator could create different user groups, for example, having different privileges or options, and later some user groups could be modified or even deleted.

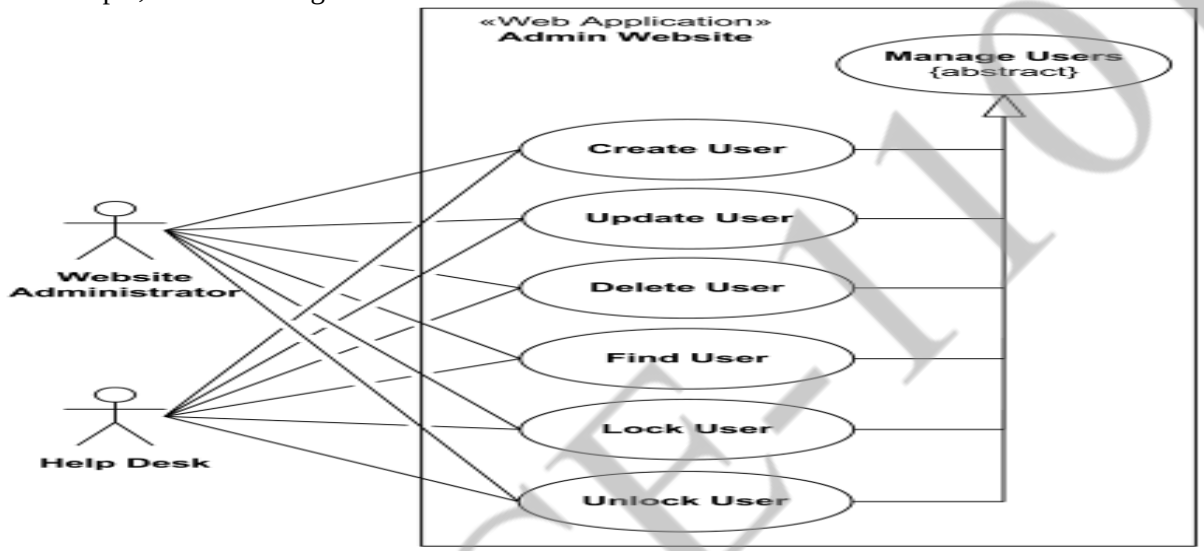


User group management use case diagram for the administration website.

User management use cases are available both to the **Website Administrator** and to the **Help Desk**. There is standard user CRUD (Create, Retrieve/Find, Update, Delete) functionality set.

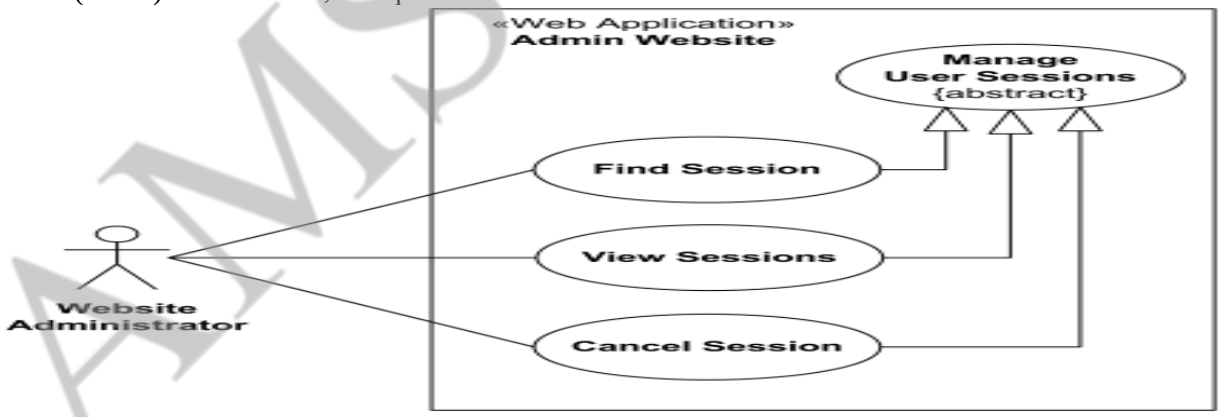
Two other use cases, **Lock User** and **Unlock User** are specific to website security. For example, if during some predefined period of time there were several unsuccessful login attempts using wrong user password, user account should be locked for some predefined time to prevent possible brute force password guessing attack.

This locking and unlocking is usually done automatically by intrusion detection or website authentication subsystem, but this functionality needs to be available in the manual mode too, just in case. For example, some user might call and ask to lock his or her account.



User management use case diagram for the administration website.

User session is created either for each new incoming request that is not yet part of a session, or/and after user was authenticated. Website administrator should have ability to see how many sessions were created, including some statistics about sessions, to find some specific session and see status of that session, and to cancel (delete) some session, if required.

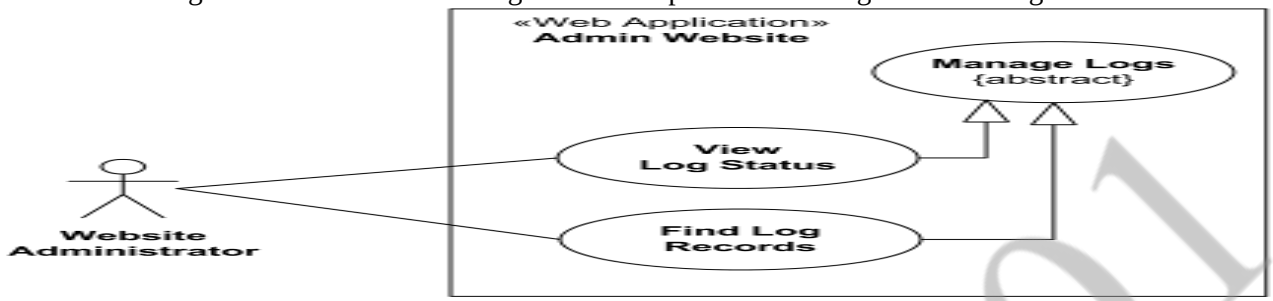


User sessions management use case diagram for the administration website.

List of administrative functions included in the **log management** depend on the security requirements supported and implemented by the website.

It is a standard security requirement (e.g., see OWASP Guide 2.0) for the logs that new records can be only appended while older log records should not be rewritten or deleted. It could be implemented e.g. by writing logs to a write once / read many (WORM) device such as a CD-R.

Website administrator should be able to see status of logs. The status could include verification that logging is still functional (there is enough space on disk and/or connection to database is not stale), and that older log files are on schedule being moved to a permanent storage for archiving.



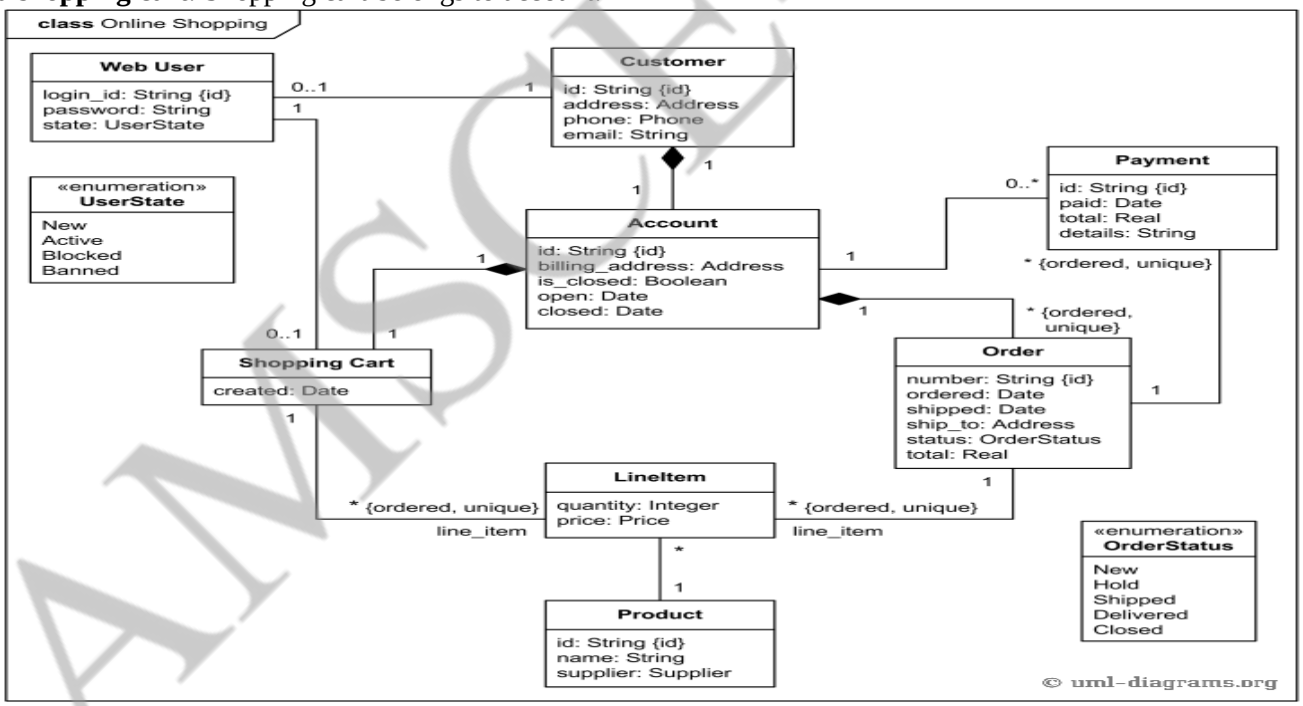
Logs management use case diagram for the administration website.

It is also common requirement to allow website administrator to find and see some log records related to a specific user or an exceptional situation.

UML Class Diagram Example

Each customer has unique id and is linked to exactly one **account**. Account owns shopping cart and orders. Customer could register as a web user to be able to buy items online. Customer is not required to be a web user because purchases could also be made by phone or by ordering from catalogues. Web user has login name which also serves as unique id.

Web user could be in several states - new, active, temporary blocked, or banned, and be linked to a **shopping cart**. Shopping cart belongs to account.



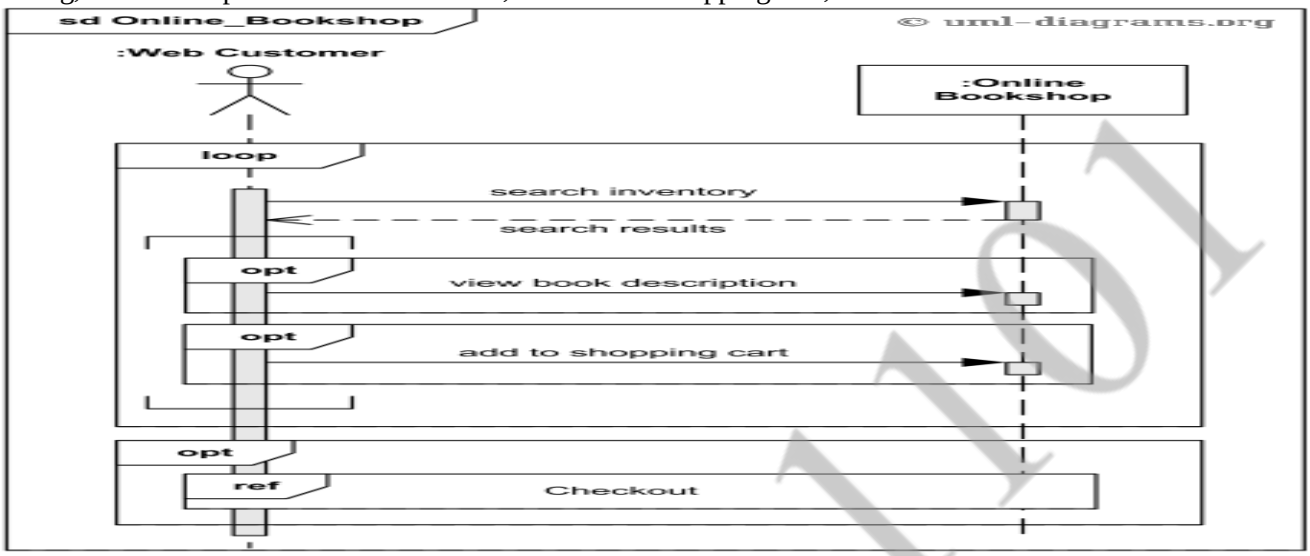
Online shopping domain UML class diagram example.

Account owns customer orders. Customer may have no orders. Customer orders are sorted and unique. Each order could refer to several **payments**, possibly none. Every payment has unique id and is related to exactly one account.

Each order has current order status. Both order and shopping cart have **line items** linked to a specific product. Each line item is related to exactly one product. A product could be associated to many line items or no item at all.

UML Sequence Diagram Example

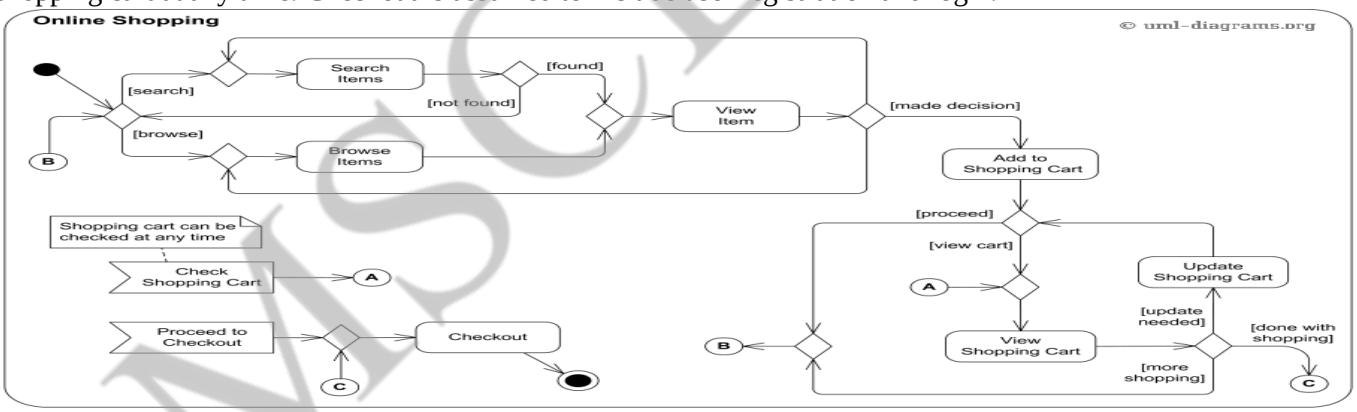
An example of high level **sequence diagram** for **online bookshop**. Online customer can search book catalog, view description of a selected book, add book to shopping cart, do checkout.



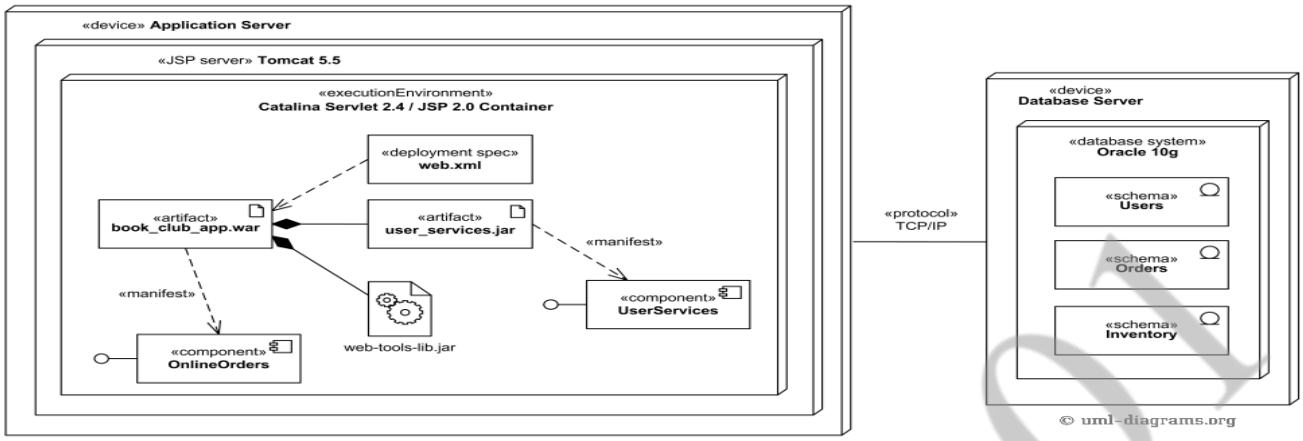
An example of UML sequence diagram for online bookshop.

UML Activity Diagram Example

An example of **activity diagram** for **online shopping**. Online customer can browse or search items, view specific item, add it to shopping cart, view and update shopping cart, checkout. User can view shopping cart at any time. Checkout is assumed to include user registration and login.



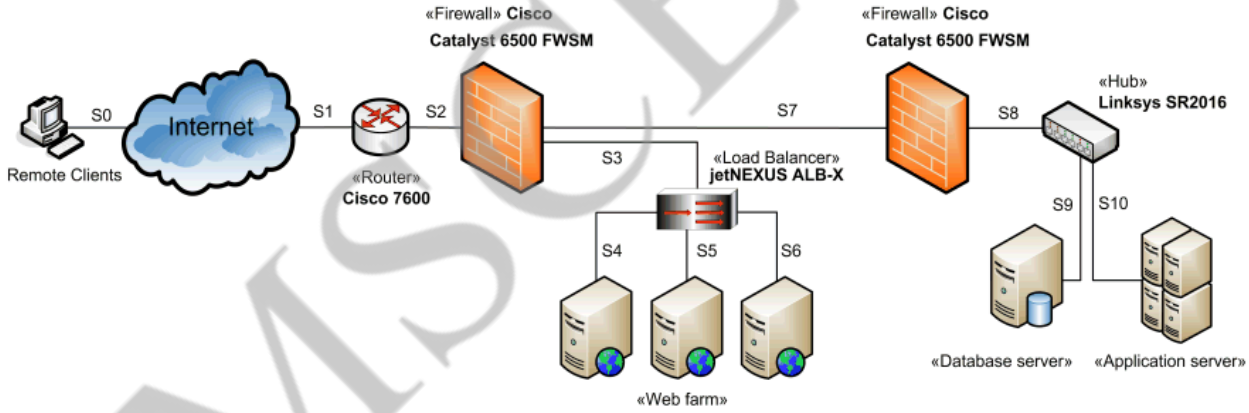
An example of UML activity diagram for online shopping.



Web Application

Network Diagram Example

UML provides no special kind of diagram to describe logical or physical **network architecture** of the designed or existing system. **Deployment diagrams**could be used for this purpose with elements limited mostly to **devices** with neither artifacts nor actual deployments shown.
The example of the network diagram below shows network architecture with configuration usually called "two firewall demilitarized zone". Demilitarized zone (DMZ) is a host or network segment located in a "neutral zone" between the Internet and an organization's intranet (private network). It prevents outside users from gaining direct access to an organization's internal network while not exposing a web, email or DNS server directly to the Internet.



An example of networking diagram for web application with two firewall DMZ configuration.

A two firewall DMZ configuration with complex security rules provides better protection over a router firewall DMZ configuration and is often able to analyze incoming and outgoing HTTP traffic and protect against application layer attacks aimed at the web servers.

Load balanced web servers shown in the DMZ communicate to the application and database servers located in the private network (intranet).