ESTD. 2001

PRATHYUSHA ENGINEERING COLLEGE

Poonamallee – Tiruvallur Road, Chennai – 602025.

CS 6008

Human Computer Interaction

(Anna University - Regulation)

Ms. N.Sripriya

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.

INTRODUCTION

. In 1983, Card, Moran and Newell described the Model Human Processor, which is a simplified view of the human processing involved in interacting with computer systems. The model comprises three subsystems: the perceptual system, handling sensory stimulus from the outside world, the motor system, which controls actions, and the cognitive system, which provides the processing needed to connect the two. Each of these subsystems has its own processor and memory, although obviously the complexity of these varies depending on the complexity of the tasks the subsystem has to perform. The model also includes a number of principles of operation which dictate the behavior of the systems under certain conditions.

INPUT–OUTPUT CHANNELS

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 becomes the computer's input and vice versa. Consequently the use of the terms input and output may lead to confusion so we shall blur the distinction somewhat and concentrate on the channels involved. This blurring is appropriate since, although 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 eyegaze system. Input in the human occurs mainly through the senses and output through the motor control of the effectors. There are five major senses: sight, hearing, touch, taste and smell. Of these, the first three are the most important to HCI. Taste and smell do not currently play a significant role in HCI, and it is not clear whether they could be exploited at all in general computer systems, although they could have a role to play in more specialized systems (smells to give warning of malfunction, for example) or in augmented reality systems. However, vision, hearing and touch are central. 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.

1.2.1 Vision

Human vision is a highly complex activity with a 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 the outside world, and the processing and interpretation of that stimulus. On the one hand the physical properties of the eye and the visual system mean that there are certain things that cannot be seen by the human; on the other the interpretative capabilities of visual processing allow images to be constructed from incomplete information. We need to understand both stages as both influence what can and cannot be perceived visually by a human being, which in turn directly affects the way that we design computer systems.

The human eye

Vision begins with light. The eye is a mechanism for receiving light and transforming it into electrical energy. Light is reflected from objects in the world and their image is focussed upside down on the back of the eye. The receptors in the eye transform it into electrical signals which are passed to the brain. The eye has a number of important components (see Figure 1.1) which we will look at in more detail. The cornea and lens at the front of the 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 are highly sensitive to light and therefore allow us to see under a low level of illumination. However, they are unable to resolve fine detail and are subject to light saturation. This 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. The cones do not operate either as they are suppressed by the rods. We are therefore temporarily unable to see at all. There are approximately 120 million rods per eye which are mainly situated towards the edges of the retina. Rods therefore dominate peripheral vision. 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, a small area of the retina on which images are fixated.

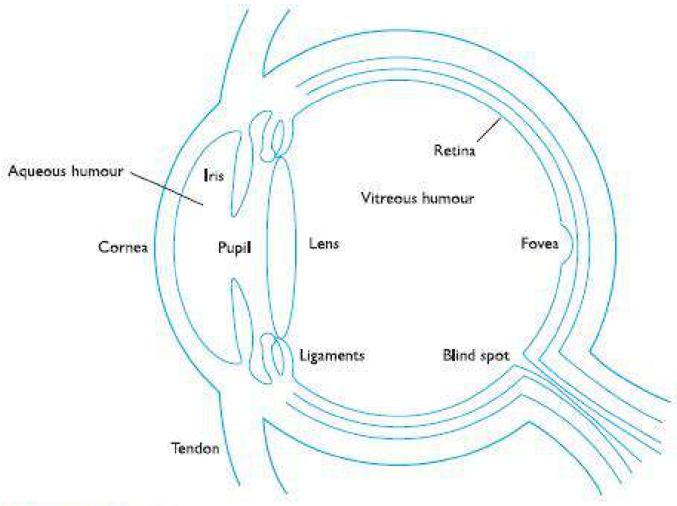


Figure 1.1 The human eye

Visual perception

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. On the hillside is a farm house withoutbuildings and farm vehicles. Someone is on the track, walking toward the summit. Below in the valley is a small market town. Even in describing such a scene the notions of size and distance predominate. Our visual system is easily able to interpret the images which it receives to take account of these things. We can identify similar objects regardless of the fact that they appear to us to be of vastly different sizes. In fact, we can use this information to judge distances. 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 noted in the previous section, reflected light from the object forms an upside-down image on the retina. The size of that image is specified as a *visual angle*. Figure 1.2 illustrates how the visual angle is calculated.

If we 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. Similarly, if two objects of the same size are placed at different distances from the eye, the

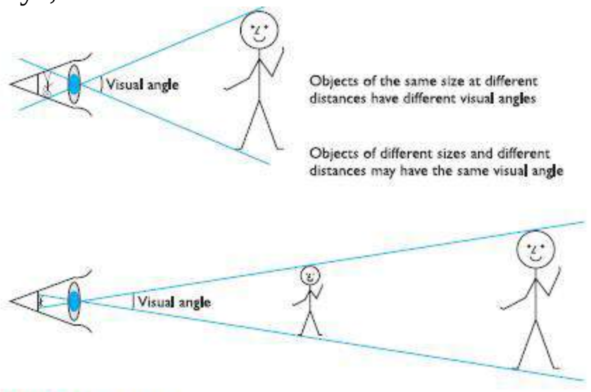

Visual angle
Objects of the same size at different distances have different visual angles
Objects of different sizes and different distances may have the same visual angle
Visual angle

Figure 1.2 Visual angle

furthest one will have the smaller 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. So how does an object's visual angle affect our perception of its size? First, if the visual angle of an object is too small we will be unable to perceive it at all. *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. Assuming that we can perceive the object, does its visual angle affect our

perception of its size? Given that the visual angle of an object is reduced as it gets further away, we might expect that we would perceive the object as smaller. In fact, our perception of an object's size remains constant even if its visual angle changes.
So a person's height is perceived as constant even if they move further from you. This is the *law of size constancy*, and it indicates that our perception of size relies on factors other than the visual angle.

The capabilities and limitations of visual processing

In considering the way in which we perceive images we have already encountered some of the capabilities and limitations of the human visual processing system. However, we have concentrated largely on low-level perception. Visual processinginvolves the transformation and interpretation of a complete image, from the light that is thrown onto the retina. As we have already noted, our expectations affect theway an image is perceived. 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 retinawhich 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. This ability to interpret and exploit our expectations can be used to resolve ambiguity. For example, consider the image shown in Figure 1.3. What do you perceive?Now consider Figure 1.4 and Figure 1.5. The context in which the object appears

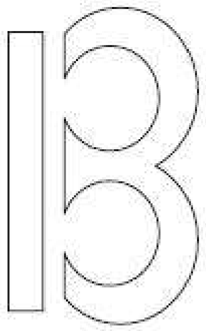


Figure 1.3 An ambiguous shape?

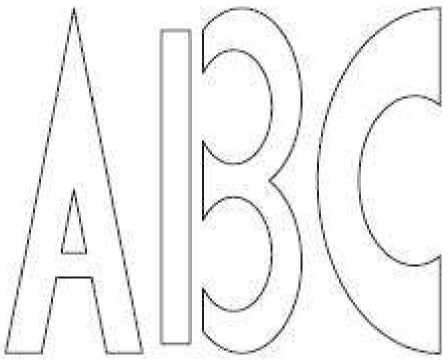


Figure 1.4 ABC

Reading
So far we have concentrated on the perception of images in general. However,the perception and processing of text is a special case that is important to interfacedesign, which invariably requires some textual display. We will therefore endthis section by looking at *reading*. There are several stages in the reading process.First, the visual pattern of the word on the page is perceived. It is then decodedwith reference to an internal representation of language. The final stages of languageprocessing include syntactic and semantic analysis and operate on phrases orsentences.We are most concerned with the first two stages of this process and how theyinfluence interface design. During reading, the eye makes jerky movements called*saccades* followed by fixations. Perception occurs during the fixation periods, whichaccount for approximately 94% of the time elapsed. The eye moves backwards overthe text as well as forwards, in what are known as *regressions*. If the text is complexthere will be more regressions.Adults read approximately 250 words a minute. It is unlikely that words arescanned serially, character by character, since experiments have shown that words canbe recognized as quickly as single characters. Instead, familiar words are recognizedusing word shape. This means that removing the word shape clues (for example, bycapitalizing words) is detrimental to reading speed and accuracy.

1.2.2 Hearing

The sense of hearing is often considered secondary to sight, but we tend to under estimate the amount of information that we receive through our ears. Close your eyes for a moment and listen. What sounds can you hear? Where are they coming from? What is making them? As I sit at my desk I can hear cars passing on the road outside, machinery working on a site nearby, the drone of a plane overhead and bird song. But I can also tell *where* the sounds are coming from, and estimate how far away they are. So from the sounds I hear I can tell that a car is passing on a particular road near my house, and which direction it is traveling in. I know that building work is in progress in a particular location, and that a certain type of bird is perched in the tree in my garden.

The human ear

Just as vision begins with light, hearing begins with vibrations in the air or *sound waves*. The ear receives these vibrations and transmits them, through various stages, to the auditory nerves. The ear comprises three sections, commonly known as the *outer ear*, *middle ear* and *inner ear*.
The outer ear is the visible part of the ear. It has two parts: the *pinna*, which is the structure that is attached to the sides of the head, and the *auditory canal*, along which sound waves are passed to the middle ear. The outer ear serves two purposes. First, it protects the sensitive middle ear from damage. The auditory canal contains wax which prevents dust, dirt and over-inquisitive insects reaching the middle ear. It also maintains the middle ear at a constant temperature. Secondly, the pinna and auditory canal serve to amplify some sounds. The middle ear is a small cavity connected to the outer ear by the *tympanic membrane*, or ear drum, and to the inner ear by the *cochlea*. Within the cavity are the *ossicles*, the smallest bones in the body. Sound waves pass along the auditory canal and vibrate the ear drum which in turn vibrates the ossicles, which transmit the vibrations to the cochlea, and so into the inner ear. This 'relay' is required because, unlike the air-filled outer and middle ears, the inner ear is filled with a denser cochlean liquid. If passed directly from the air to the liquid, the transmission of the sound waves would be poor. By transmitting them via the ossicles the sound waves are concentrated and amplified.

Processing sound

As we have seen, sound is changes or vibrations in air pressure. It has a number of characteristics which we can differentiate. *Pitch* is the frequency of the sound. A low frequency produces a low pitch, a high frequency, a high pitch. *Loudness* is proportional to the amplitude of the sound; the frequency remains constant. *Timbre* relates to the type of the sound: sounds may have the same pitch and loudness but be made by different instruments and so vary in timbre. We can also identify a sound's location, since the two ears receive slightly different sounds, owing to the time difference between the sound reaching the two ears and the reduction in intensity caused by the sound waves reflecting from the head. The human ear can hear frequencies from about 20 Hz to 15 kHz. It can distinguish frequency changes of less than 1.5 Hz at low frequencies but is less accurate at high frequencies. Different frequencies trigger activity in neurons in different parts of the auditory system, and cause different rates of firing of nerve impulses. The auditory system performs some filtering of the sounds received, allowing us to ignore background noise and concentrate on important information. We are selective in our hearing, as illustrated by the *cocktail party effect*, where we can pick out our name spoken across a crowded noisy room. However, if sounds are too loud, or frequencies too similar, we are unable to differentiate sound.

1.2.3 Touch

The third and last of the senses that we will consider is touch or *haptic perception*. Although this sense is often viewed as less important than sight or hearing, imagine life without it. 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. Watching such users can be an informative and amusing experience! Touch is therefore an important means of feedback, and this is no less so in using computer systems. Feeling buttons depress is an important part of the task of pressing the button. Also, we should be aware that, although for the average person, haptic perception is a secondary source of information, for those whose other senses are impaired, it may be vitally important. For such users, interfaces such as braille may be the primary source of information in the interaction. We should not therefore un derestimate the importance of touch.

1.2.4 Movement

Before leaving this section on the human's input–output channels, we need to consider motor control and how the way we move affects our interaction with computers. 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. However, a combined signal will result in the quickest response. Factors such as skill or practice can reduce reaction time, and fatigue can increase it. A second measure of motor skill is accuracy. One question that we should ask is whether speed of reaction results in reduced accuracy. This is dependent on the task and the user. In some cases, requiring increased reaction time reduces accuracy. This is the premise behind many arcade and video games where less skilled users fail at levels of play that require faster responses. However, for skilled operators this is not necessarily the case. Studies of keyboard operators have shown that, although the faster operators were up to twice as fast as the others, the slower ones made 10 times the errors. 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* [135]. There are many variations 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. This affects the type of target we design. Since users will find it more difficult to manipulate small objects, targets should generally be as large as possible and the distance to be moved as small as possible. This has led to suggestions that piechart- shaped menus are preferable to lists since all options are equidistant. However, the trade-off is increased use of screen estate, so the choice may not be so simple. If lists are used, the most frequently used options can be placed closest to the user's start point (for example, at the top of the menu).

HUMAN MEMORY

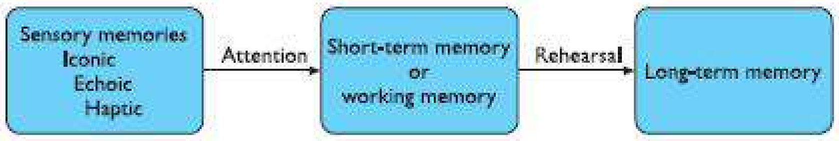


Figure 1.9 A model of the structure of memory

It allows us to repeat actions, to use language, and to use new information received via our senses. It also gives us our sense of identity, by preserving information from our past experiences. But how does our memory work? How do we remember arbitrary lists such as those generated in the memory game? Why do some people remember more easily than others? And what happens when we forget? In order to answer questions such as these, we need to understand some of the capabilities and limitations of human memory. Memory is the second part of our model of the human as an information-processing system. However, as we noted earlier, such a division is simplistic since, as we shall see, memory is associated with each level of processing. Bearing this in mind, we will consider the way in which memory is structured and the activities that take place within the system.

It is generally agreed that there are three types of memory or memory function: *sensory buffers*, *short-term memory* or *working memory*, and *long-term memory*. There is some disagreement as to whether these are three separate systems or different functions of the same system. We will not concern ourselves here with the details of this debate, which is discussed in detail by Baddeley [21], but will indicate the evidence used by both sides as we go along. For our purposes, it is sufficient to note three separate types of memory. These memories interact, with information being processed and passed between memory stores, as shown in Figure 1.9.

1.3.1 Sensory memory

The sensory memories act as buffers for stimuli received through the senses. A sensory memory exists for each sensory channel: *iconic memory* for visual stimuli, *echoic memory* for aural stimuli and *haptic memory* for touch. These memories are constantly overwritten by new information coming in on these channels. We can demonstrate the existence of iconic memory by moving a finger in front of the eye. Can you see it in more than one place at once? This indicates a persistence of the image after the stimulus has been removed. A similar effect is noticed most vividly at firework displays where moving sparklers leave a persistent image.

Information remains in iconic memory very briefly, in the order of 0.5 seconds. Similarly, the existence of echoic memory is evidenced by our ability to ascertain the direction from which a sound originates. This is due to information being received by both ears. However, since this information is received at different times, we must store the stimulus in the meantime. Echoic memory allows brief ‘play-back’ of information. Have you ever had someone ask you a question when you are reading? You ask them to repeat the question, only to realize that you know what was asked after all. This experience, too, is evidence of the existence of echoic memory. Information is passed from sensory memory into short-term memory by attention, thereby filtering the stimuli to only those which are of interest at a given time. Attention is the concentration of the mind on one out of a number of competing stimuli or thoughts. It is clear that we are able to focus our attention selectively, choosing to attend to one thing rather than another. This is due to the limited capacity of our sensory and mental processes. If we did not selectively attend to the stimuli coming into our senses, we would be overloaded. We can choose which stimuli to attend to, and this choice is governed to an extent by our *arousal*, our level of interest or need. This explains the cocktail party phenomenon mentioned earlier: we can attend to one conversation over the background noise, but we may choose to switch our attention to a conversation across the room if we hear our name mentioned. Information received by sensory memories is quickly passed into a more permanent memory store, or overwritten and lost.

1.3.2 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 356 in your head. The chances are that you will have done this calculation in stages, perhaps 56 and then 306 and added the results; or you may have used the fact that 623 and calculated 23570 followed by 370. 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. This was established in experiments by Miller [234]. Try it. Look at the following number sequence:

265397620853

Now write down as much of the sequence as you can remember. Did you get it all right? If not, how many digits could you remember? If you remembered between five and nine digits your *digit span* is average. Now try the following sequence:

44 113 245 8920

Did you recall that more easily? Here the digits are grouped or *chunked*. A generalization of the 7 2 rule is that we can remember 72 *chunks* of information. Therefore chunking information can increase the short-term memory capacity. The limited capacity of short-term memory produces a subconscious desire to create chunks, and so optimize the use of the memory. The successful formation of a chunk is known as *closure*. This process can be generalized to account for the desire to complete or close tasks held in short-term memory. If a subject fails to do this or is prevented from doing so by interference, the subject is liable to lose track of what she is doing and make consequent errors.

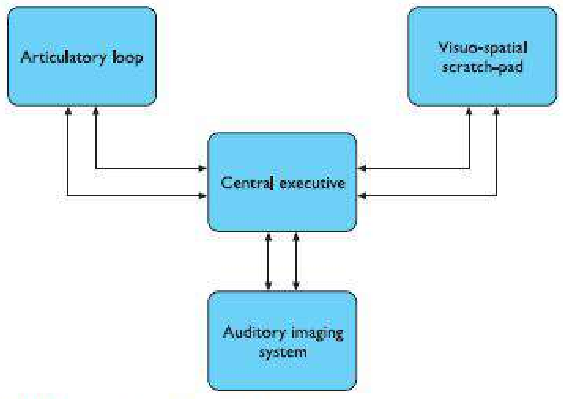


Figure 1.10 A more detailed model of short-term memory

1.3.3 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

There are 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

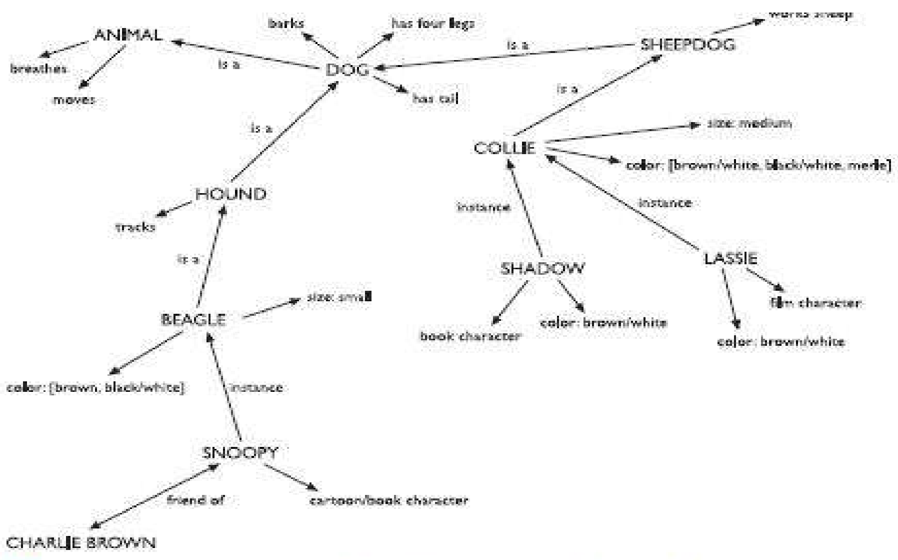


Figure 1.11 Long-term memory may store information in a semantic network

associated to each other in classes, and may inherit attributes from parent classes. This model is known as a *semantic network.* As an example, our knowledge about dogs may be stored in a network such as that shown in Figure 1.11. Specific breed attributes may be stored with each given breed, yet general dog information is stored at a higher level. This allows us to generalize about specific cases. For instance, we may not have been told that the sheepdog Shadow has four legs and a tail, but we can infer this information from our general knowledge about sheepdogs and dogs in general. Note also that there are connections within the network which link into other domains of knowledge, for example cartoon characters. This illustrates how our knowledge is organized by association. The viability of semantic networks as a model of memory organization has been demonstrated by Collins and Quillian [74]. Subjects were asked questions about different properties of related objects and their reaction times were measured. The types of question asked (taking examples from our own network) were 'Can a collie breathe?', 'Is a beagle a hound?' and 'Does a hound track?' In spite of the fact that the answers to such questions may seem obvious, subjects took longer to answer questions such as 'Can a collie breathe?' than ones such as 'Does a hound track?' Thereason for this, it is suggested, is that in the former case subjects had to search further
through the memory hierarchy to find the answer, since information is stored
at its most abstract level.

A number of other memory structures have been proposed to explain how we
represent and store different types of knowledge. Each of these represents a different

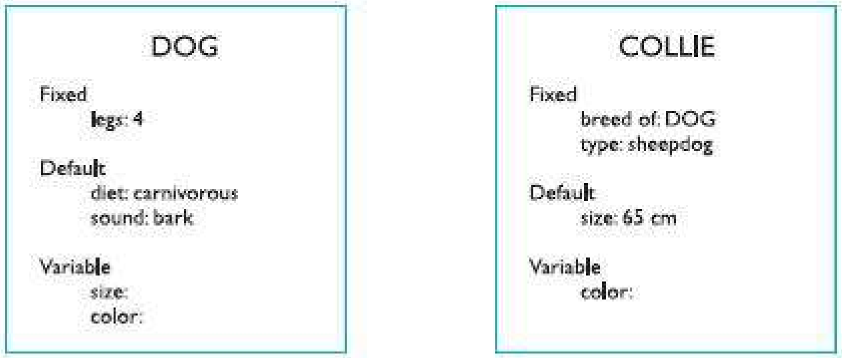
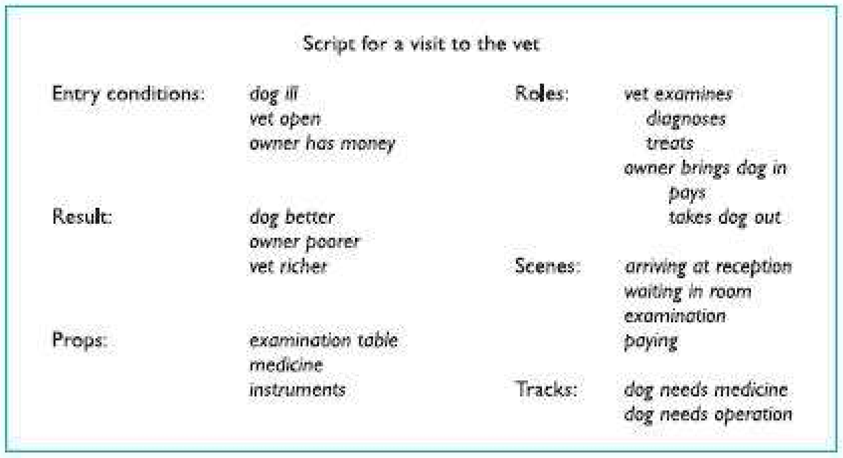


Figure 1.12 A frame-based representation of knowledge

aspect of knowledge and, as such, the models can be viewed as complementary rather than mutually exclusive. 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. 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. The fixed slots are those for which the attribute value is set, default slots represent the usual attribute value, although this may be overridden in particular instantiations (for example, the Basenji does not bark), and variable slots can be filled with particular values in a given instance. Slots can also contain procedural knowledge. Actions or operations can be associated with a slot and performed, for example, whenever the value of the slot is changed. Frames extend semantic nets to include structured, hierarchical information. They represent knowledge items in a way which makes explicit the relative importance of each piece of information. 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.



Script for a visit to the vet

Entry conditions:	*dog ill* *vet open* *owner has money*	Roles:	*vet examines* *diagnoses* *treats* *owner brings dog in* *pays* *takes dog out*
Result:	*dog better* *owner poorer* *vet richer*	Scenes:	*arriving at reception* *waiting in room* *examination* *paying*
Props:	*examination table* *medicine* *instruments*	Tracks:	*dog needs medicine* *dog needs operation*

Figure 1.13 A script for visiting the vet

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. 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

If we then meet a growling dog, the condition in the second rule is matched, and we respond by turning tail and running. (Not to be recommended by the way!)
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. In this section we will consider two categories of thinking: reasoning and problem solving. In practice these are not distinct since the activity of solving a problem may well involve reasoning and vice versa. However, the distinction is a common one and is helpful in clarifying the processes involved.

1.4.1 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*. We use each of these types of reasoning in everyday life, but they differ in significant ways.

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. One explanation for this is that people bring their world knowledge into the reasoning process. There is good reason for this. It allows us to take short cuts which make dialog and interaction between people informative but efficient. 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 [365]. You are presented with four cards as in Figure 1.14. Each card has a numberon 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 vo els, just that all vowels have even numbers.)

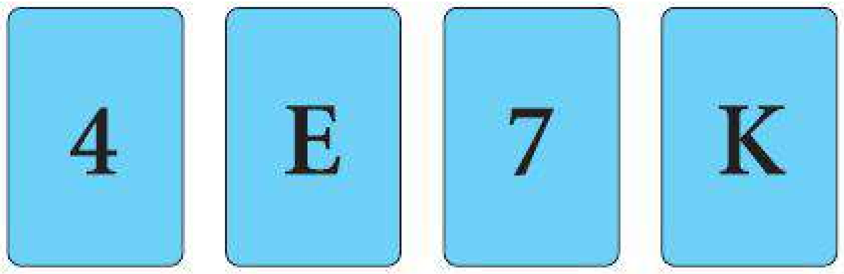


Figure 1.14 Wason's cards

Abductive reasoning

The third type of reasoning is abduction. 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. If, in fact, the event and the action are unrelated, confusion and even error often result.

1.4.2 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. We will look briefly at each of these views.

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. This explanation was considered by the Gestalt school to be insufficient to account for human problem-solving behavior. Instead, they claimed, problem 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. Indeed, 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 [202]. 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* [224]. 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. Although various solutions were proposed by subjects, few chose to use the weight of the pliers as a pendulum to 'swing' the strings together. However, when the experimenter brushed against the string, setting it in motion, this solution presented itself to subjects. Maier interpreted this as an example of productive restructuring. 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.

1.4.3 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 tha t 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. Studies of chess players by DeGroot, Chase and Simon, among others, produced some interesting observations [64, 65, 88, 89]. In all the experiments the behavior of chess masters was compared with less experienced chess players.

The first observation was that players did not consider large numbers of moves in choosing their move, nor did they look ahead more than six moves (often far fewer). Masters considered no more alternatives than the less experienced, but they took less time to make a decision and produced better moves. So what makes the difference between skilled and less skilled behavior in chess?

It appears that chess masters remember board configurations and good moves associated with them. When given actual board positions to remember, masters are much better at reconstructing the board than the less experienced. However, when given random configurations (which were unfamiliar), the groups of players were equally bad at reconstructing the positions. It seems therefore that expert players ‘chunk’ the board configuration in order to hold it in short-term memory. Expert players use larger chunks than the less experienced and can therefore remember more detail. This behavior is also seen among skilled computer programmers. They can also reconstruct programs more effectively than novices since they have the structures available to build appropriate chunks.

They acquire plans representing code to solve particular problems. When that problem is encountered in a new domain or new program they will recall that particular plan and reuse it. Another observed difference between skilled and less skilled problem solving is in the way that different problems are grouped. Novices tend to group problems according to superficial characteristics such as the objects or features common to both. Experts, on the other hand, demonstrate a deeper understanding of the problems and group them according to underlying conceptual similarities which may not be at all obvious from the problem descriptions. Each of these differences stems from a better encoding of knowledge in the expert: information structures are fine tuned at a deep level to enable efficient and accurate

retrieval. But how does this happen? How is skill such as this acquired? One model of skill acquisition is Anderson’s *ACT** model [14]. 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. For example, think of an activity at which you are skilled, perhaps

driving a car or riding a bike. Try to describe to someone the exact procedure which you go through to do this. You will find this quite diffi cult. In fact experts tend to have to rehearse their actions mentally in order to identify exactly what they do. Such skilled behavior is efficient but may cause errors when the context of the activity changes.

1.4.4 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? In order to answer the latter part of the question we must first look at what is going on when we make an error.
There are several different types of error. As we saw in the last section some errors result from changes in the context of skilled behavior. 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.

1.5 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.

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.
So what are the implications of this for design? It suggests that in situations of stress, people will be less able to cope with complex problem solving or managing difficult interfaces, whereas if people are relaxed they will be more forgiving of limitations in the design. This does not give us an excuse to design bad interfaces but does suggest that if we build interfaces that promote

positive responses – for example by using aesthetics or reward – then they are likely be more successful.

1.6 INDIVIDUAL DIFFERENCES

We should be aware of
individual differences so that we can account for them as far as possible within our designs. These differences may be long term, such as sex, physical capabilities and intellectual capabilities. Others are shorter term and include the effect of stress or fatigue on the user. Still others change through time, such as age. These differences should be taken into account in our designs. It is useful to consider, for any design decision, if there are likely to be users within the target group who will be adversely affected by our decision. At the extremes a decision may exclude a section of the user population. For example, the current emphasis on visual interfaces excludes those who are visually impaired, unless the design also makes use of the other sensory channels. On a more mundane level, designs should allow for users who are under pressure, feeling ill or distracted by other concerns: they should not push users to their perceptual or cognitive limits

1.7 PSYCHOLOGY AND THE DESIGN OF INTERACTIVE SYSTEMS

So far we have looked briefly at the way in which humans receive, process and store information, solve problems and acquire skill. But how can we apply what we have learned to designing interactive systems? Sometimes, straightforward conclusions can be drawn. For example, we can deduce that recognition is easier than recall and allow users to select commands from a set (such as a menu) rather than input them directly. However, in the majority of cases, application is not so obvious or simple. In fact, it may be dangerous, leading us to make generalizations which are not valid. In order to apply a psychological principle or result properly in design, we need to understand its context, both in terms of where it fits in the wider field of psychology and in terms of the details of the actual experiments, the measures used and the subjects involved, for example. This may appear daunting, particularly to the novice designer who wants to acknowledge the relevance of cognitive psychology but does not have the background to derive appropriate conclusions. Fortunately, principles and results from research in psychology have been distilled into guidelines for design, models to support design and techniques for evaluating design. Parts 2 and 3 of this book include discussion of a range of guidelines, models and techniques, based on cognitive psychology, which can be used to support the design process.

1.7.1 Guidelines

Throughout this chapter we have discussed the strengths and weaknesses of human cognitive and perceptual processes but, for the most part, we have avoided attempting to apply these directly to design. This is because such an attempt could only be partial and simplistic, and may give the impression that this is all psychology has to offer.However, general design principles and guidelines can be and have been derived from the theories we have discussed. Some of these are relatively straightforward: for instance, recall is assisted by the provision of retrieval cues so interfaces should incorporate recognizable cues wherever possible. Others are more complex and contextdependent

1.7.2 Models to support design

As well as guidelines and principles, psychological theory has led to the development of analytic and predictive models of user behavior. Some of these include a specific model of human problem solving, others of physical activity, and others attempt a more comprehensive view of cognition. Some predict how a typical computer user would behave in a given situation, others analyze why particular user behavior occurred. All are based on cognitive theory.

1.7.3 Techniques for evaluation

In addition to providing us with a wealth of theoretical understanding of the human user, psychology also provides a range of empirical techniques which we can employ to evaluate our designs and our systems. In order to use these effectively we need to understand the scope and benefits of each method.

THE COMPUTER

2.1 INTRODUCTION

In order to understand how humans interact with computers, we need to have an understanding of both parties in the interaction. The previous chapter explored aspects of human capabilities and behavior of which we need to be aware in the context of human–computer interaction; this chapter considers the computer and associated input–output devices and investigates how the technology influences the nature of the interaction and style of the interface. We will concentrate principally on the traditional computer but we will also look at devices that take us beyond the closed world of keyboard, mouse and screen. As well as giving us lessons about more traditional systems, these are increasingly becoming important application areas in HCI.

2.1.1 A typical computer system
Consider a typical computer setup as shown in Figure 2.1. There is the computer 'box' itself, a keyboard, a mouse and a color screen. The screen layout is shown alongside it. If we examine the interface, we can see how its various characteristics are related to the devices used. Some of this variation is driven by different hardware configurations: desktop use, laptop computers, PDAs (personal digital assistants). Partly the diversity of devices reflects the fact that there are many different types of

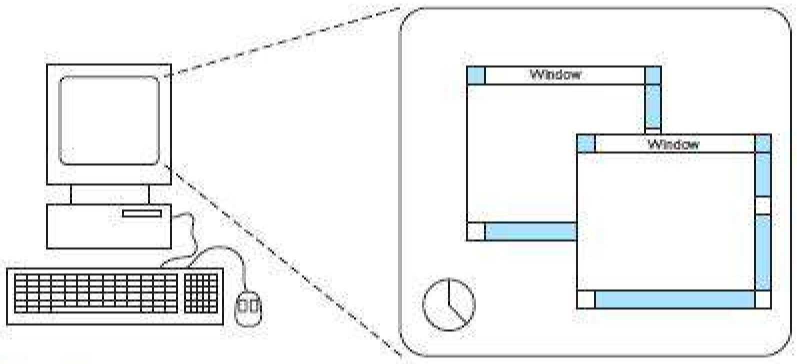


Figure 2.1 A typical computer system

data that may have to be entered into and obtained from a system, and there are also many different types of user, each with their own unique requirements.

2.1.2 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. This still continues today although now with pre-prepared electronic files or possibly machine-read forms. It is clearly the most appropriate mode for certain kinds of application, for example printing pay checks or entering the results from a questionnaire. 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!). The field of Human– Computer Interaction largely grew due to this change in interactive pace.

2.1.3 Richer interaction – everywhere, everywhen

Computers are coming out of the box! 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?

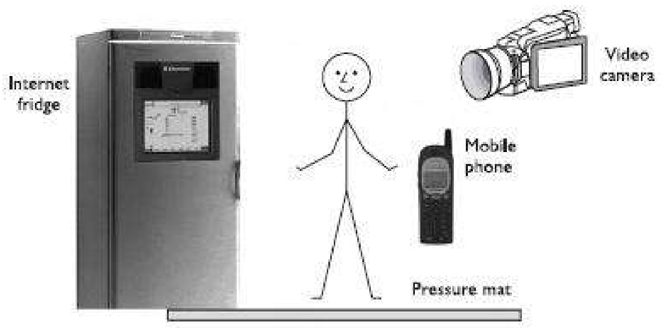


Figure 2.2 A typical computer system? Photo courtesy Electrolux

2.2 TEXT ENTRY DEVICES

2.2.1 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.

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

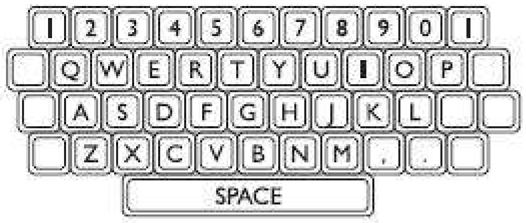


Figure 2.3 The standard QWERTY keyboard

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. Hitting a key caused an arm to shoot towards the carriage, imprinting the letter on the head on the ribbon and hence onto the paper. If two arms flew towards the paper in quick succession from nearly the same angle, they would often jam – the solution to this was to set out the keys so that common combinations of consecutive letters were placed at different ends of the keyboard, which meant that the arms would usually move from alternate sides. One appealing story relating to the key layout is that it was also important for a salesman to be able to type the word 'typewriter' quickly in order to impress potential customers: the letters are all on the top row! The electric typewriter and now the computer keyboard are not subject to the original mechanical constraints, but the QWERTY keyboard remains the dominant layout. The reason for this is social – the vast base of trained typists would be reluctant to relearn their craft, whilst the management is not prepared to accept an initial lowering of performance whilst the new skills are gained. 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. As whole populations have become keyboard users this technological inertia has probably become impossible to change.

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. 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 o n a very small keyboard and hence avoids criticisms of difficulty of use!

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.

2.2.2 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.

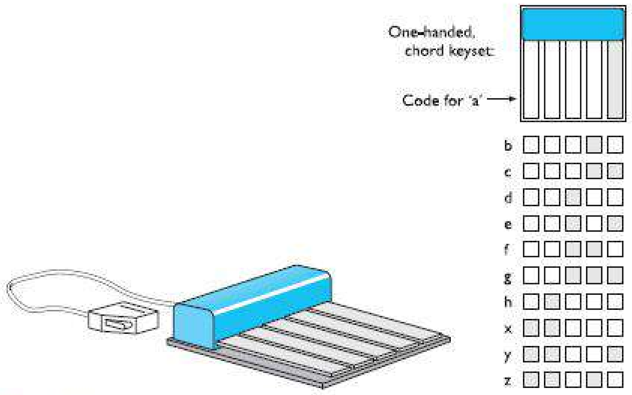
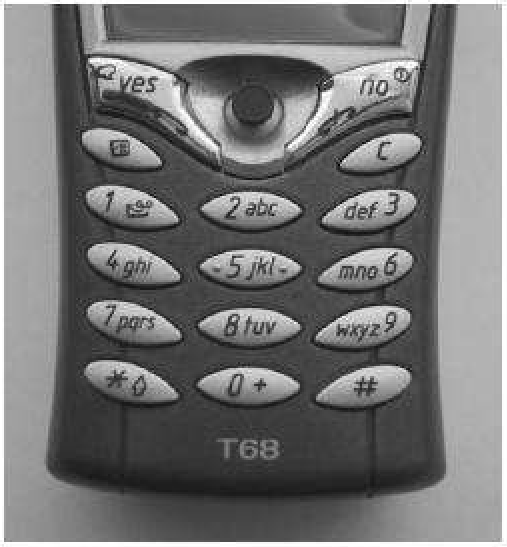


Figure 2.4 A very early chord keyboard (left) and its lettercodes (right)



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)

2.2.3 Phone pad and T9 entry

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.

2.2.4 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. Further complications arise because letters within words are shaped and often drawn very differently depending on the actual word; the context can help determine the letter's identity, but is often unable to provide enough information. Handwriting recognition is covered in more detail later in the book, in Chapter 10. More serious in many ways is the limitation on speed; it is difficult to write at more than 25 words a minute, which is no more than half the speed of a decent typist.

2.2.5 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*)! Note also that this performance is usually quoted only for a restricted vocabulary of command words. Trying to extend such systems to the level of understanding natural language, with its inherent vagueness, imprecision and pauses, opens up many more problems that have not been satisfactorily solved even for keyboard-entered natural language. 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.

2.3 POSITIONING, POINTING AND DRAWING

Central to most modern computing systems is the ability to point at something on the screen and thereby manipulate it, or perform some function. There has been a long history of such devices, in particular in *computer-aided design* (CAD), where positioning and drawing are the major activities. Pointing devices allow the user to point, position and select items, either directly or by manipulating a pointer on the screen. Many pointing devices can also be used for free-hand drawing although the skill of drawing with a mouse is very different from using a pencil. The mouse is still most common for desktop computers, but is facing challenges as laptop and handheld computing increase their market share. Indeed, these words are being typed on a laptop with a touchpad and no mouse.

2.3.1 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 (Figure 2.1). It is a small, palm-sized box housing a weighted ball – as the box is moved over the tabletop, 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. If you remove the ball occasionally to clear dust you may be able to see these rollers. 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'. In addition to detecting motion, the mouse has typically one, two or three buttons on top. These are used to indicate selection or to initiate action. Single-button mice tend to have similar functionality to multi-button mice, and achieve this by instituting different operations for a single and a double button click. A 'double-click' is when the button is pressed twice in rapid succession. Multibutton mice tend to allocate one operation to each particular button.

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 only provides information on the relative movement of the ball within the housing: it can be physically lifted up from the desktop and replaced in a different position without moving the cursor. This offers the advantage that less physical space is required for the mouse, but suffers from being less intuitive for novice users. Since the mouse sits on the desk, moving it about is easy and users suffer little arm fatigue, although the indirect nature of the medium can lead to problems with hand–eye coordination. However, a major advantage of the mouse is that the cursor itself is small, and it can be easily manipulated without obscuring the display.

2.3.2 Touchpad

Touchpads 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. The feel is very different from other input devices, but as with all devices users quickly get used to the action and become proficient. 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. If the finger moves slowly over the pad then the pad movements map to small distances on the screen. If the finger is moving quickly the same distance on the touchpad moves the cursor a long distance. For example, on the trackpad being used when writing this section a very slow movement of the finger from one side of the trackpad to the other moves the cursor less than 10% of the width of the screen. However, if the finger is moved very rapidly from side to side, the cursor moves the whole width of the screen. In fact, this form of acceleration setting is also used in other indirect positioning devices including the mouse. Fine settings of this sort of parameter makes a great difference to the 'feel' of the device.

2.3.3 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. It is fairly accurate, but is hard to draw with, as long movements are difficult. Trackballs now appear in a wide variety of sizes, the most usual being about the same as a golf ball, with a number of larger and smaller devices available. The size and 'feel' of the trackball itself affords significant differences in the usability of the device: its weight, rolling resistance and texture all contribute to the overall effect. Some of the smaller devices have been used in notebook and portable computers, but more commonly trackpads or nipples are used. They are often sold as alternatives to mice on desktop computers, especially for RSI sufferers. They are also heavily used in video games where their highly responsive behavior, including being able to spin the ball, is ideally suited to the demands of play. 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. However, if you were to try to write your signature using a thumbwheel, the limitations would be all too apparent. The appropriateness of the device depends on the task to be performed.

2.3.4 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. Another reason for their dominance of the games market is their relative familiarity to users, and their likeness to aircraft joysticks: aircraft are a favorite basis for games, leading to familiarity with the joystick that can be used for more obscure entertainment ideas. 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. More commonly a small rubber nipple projects in the center of the keyboard and acts as a tiny isometric joystick. It is usually difficult for novices to use, but this seems to be related to fine adjustment of the speed settings. Like the joystick the nipple controls the rate of movement across the screen and is thus less direct than a mouse or stylus.

2.3.5 Touch-sensitive screens (touchscreens)

Touchscreens 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 touchscreen 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 touchscreens 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.

2.3.6 Stylus and light pen

For more accurate positioning (and to avoid greasy screens), systems with touch sensitive surfaces often emply a stylus. Instead of pointing at the screen directly a small pen-like plastic stick is used to point and draw on the screen. This is particularly popular in PDAs, but they are also being used in some laptop computers. 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 touchscreen.
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. Interestingly some users of PDAs with fold-out keyboards learn to hold the stylus held outwards between their fingers so that they can type whilst holding it. As it is unattached the stylus can easily get lost, but a closed pen can be used in emergencies. Stylus, light pen and touchscreen 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.

2.3.7 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.

2.3.8 Eyegaze
Eyegaze systems allow you to control the computer by simply looking at it! 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. Such systems have been used in military applications, notably for guiding air-toair missiles to their targets, but are starting to find more peaceable uses, for disabled users and for workers in environments where it is impossible for them to use their hands. The rarity of the eyegaze is due

partly to its novelty and partly to its expense, and it is usually found only in certain domain-specific applications.

2.4 DISPLAY DEVICES

The vast majority of interactive computer systems would be unthinkable without
some sort of display screen, but many such systems do exist, though usually
in specialized applications only. Thinking beyond the traditional, systems such as
cars, hi-fis and security alarms all have different outputs from those expressible on a
screen, but in the personal computer and workstation market, screens are pervasive.

Various cursor key layouts

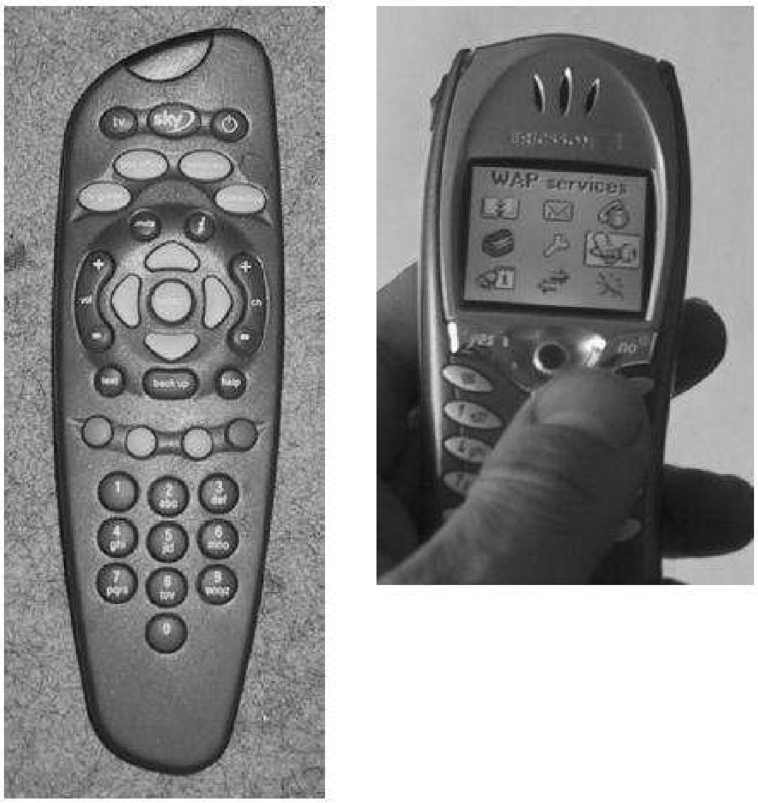


Figure 2.8 Satellite TV remote control and mobile phone. Source: Photograph left by Alan Dix with permission from British Sky Broadcasting Limited, photograph right by Alan Dix (Ericsson phone)

Liquid crystal display

If you have used a personal organizer or notebook computer, you will have seen the light, flat plastic screens. 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. Similar in principle to the digital watch, a thin layer of liquid crystal is sandwiched between two glass plates. The top plate is transparent and polarized, whilst the bottom plate is reflective. External light passes through the top plate and is polarized, which means that it only oscillates in one direction. This then passes through the crystal, reflects off the bottom plate and back to the eye, and so that cell looks white. When a voltage is applied to the crystal, via the conducting glass plates, the crystal twists. This causes it to turn the plane of polarization of the incoming light, rotating it so that it cannot return through the top plate, making the activated cell look black.

The LCD requires refreshing at the usual rates, but the relatively slow response of the crystal means that flicker is not usually noticeable. The low intensity of the light emitted from the screen, coupled with the reduced flicker, means that the LCD is less tiring to use than standard CRT ones, with reduced eyestrain. 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, light weight 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. The advent of these screens allowed small, light computers to be built, and created a large market that did not previously exist. Such computers, riding on the back of the technological wave, have opened up a different way of working for many people, who now have access to com puters when away from the office, whether out on business or at home. Working in a different location on a smaller machine with different software obviously represents a different style of interaction and so once again we can see that differences in devices may alter the human–computer interaction considerably. The growing notebook computer market fed back into an investment in developing LCD screen technology, with supertwisted crystals increasing the viewing angle dramatically. Response times have also improved so that LCD screens are now used in personal DVD players and even in home television.

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 (see Section 2.4.3 below).
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, that we discussed in Section 2.4.1. 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 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

40963120 pixels, but suffer from low contrast, low brightness and a difficulty in displaying color.

2.4.3 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 situate ed. These may be large screens where several people are expected to view or interact simultaneously, or they may be very small. Figure 2.11 shows an example of a small experimental situated display mounted by an office door to act as an electronic sticky note [70].

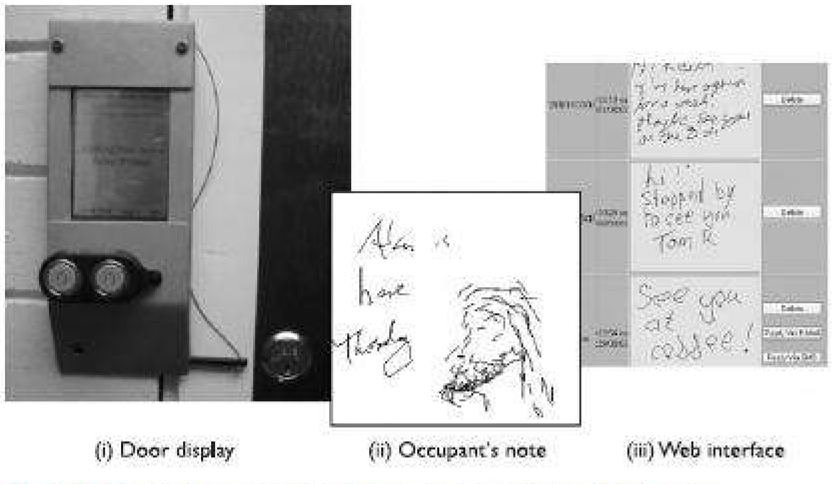


Figure 2.11 Situated door display. Source: Courtesy of Keith Cheverst

2.4.4 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. There are various technologies being investigated for this. One involves the whole surface being covered with tiny spheres, black one side, white the other. 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. Probably the first uses of these will be for large banners that can be reprogrammed or slowly animated. This is an ideal application, as it does not require very rapid updates and does not require the pixels to be small. As the technology matures, the aim is to have programmable sheets of paper that you attach to your computer to get a ‘soft’ printout that can later be changed. Perhaps one day you may be able to have a ‘soft’ book that appears just like a current book with soft pages that can be turned and skimmed, but where the contents and cover can be changed when you decide to download a new book from the net!

2.6 PHYSICAL CONTROLS, SENSORS AND SPECIAL DEVICES

2.6.1 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. There are many different things that need to be known, ranging from data from tactical systems to navigational information and aircraft status indicators.

The head-up display projects a subset of this information into the pilot’s line of vision so that the information is directly in front of her eyes. This obviates the need for large banks of information to be scanned with the corresponding lack of attention to what is happening outside, and makes the pilot’s job easier. Less important information is usually presented on a smaller number of dials and gauges in the cockpit to avoid cluttering the head-up display, and these can be monitored less often, during times of low stress.

2.6.2 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. We have discussed speech previously, but other sounds such as beeps, bongs, clanks, whistles and whirrs are all used to varying effect. As well as conveying system output, sounds offer an important level of feedback in interactive systems. 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. The advantage of auditory feedback is evident when we consider a simple device such as a doorbell. If we press it and hear nothing, we are left undecided. Should we press it again, in case we did not do it right the first time, or did it ring but we did not hear it? And if we press it again but it actually did ring, will the people in the house think we are very rude, ringing insistently? We feel awkward and a little stressed. If we were using a computer system instead of a doorbell and were faced with a similar problem, we would not enjoy the interaction and would not perform as well.

2.6.3 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.

In fact, the car gives a very good example of the power of tactile feedback. If you drive over a small bump in the road the car is sent slightly off course; however, the chances are that you will correct yourself before you are consciously aware of the bump. Within your body you have reactions that push back slightly against pressure to keep your limbs where you ‘want’ them, or move your limbs out of the way when you brush against something unexpected. These responses occur in your lower brain and are very fast, not involving any conscious effort. So, haptic devices can access very fast responses, but these responses are not fully controlled. This can be used effectively in design, but of course also with caution.

2.6.4 Physical controls

controls that can be used for a variety of purposes. In contrast, these dedicated control panels have been designed for a particular device and for a single use. This is why they differ so much. Looking first at the microwave, it has a flat plastic control panel. The buttons on the panel are pressed and ‘give’ slightly. The choice of the smooth panel is probably partly for visual design – it looks streamlined! However, there are also good practical reasons. 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 and 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 (although some washing machines do have smooth panels). It has several major

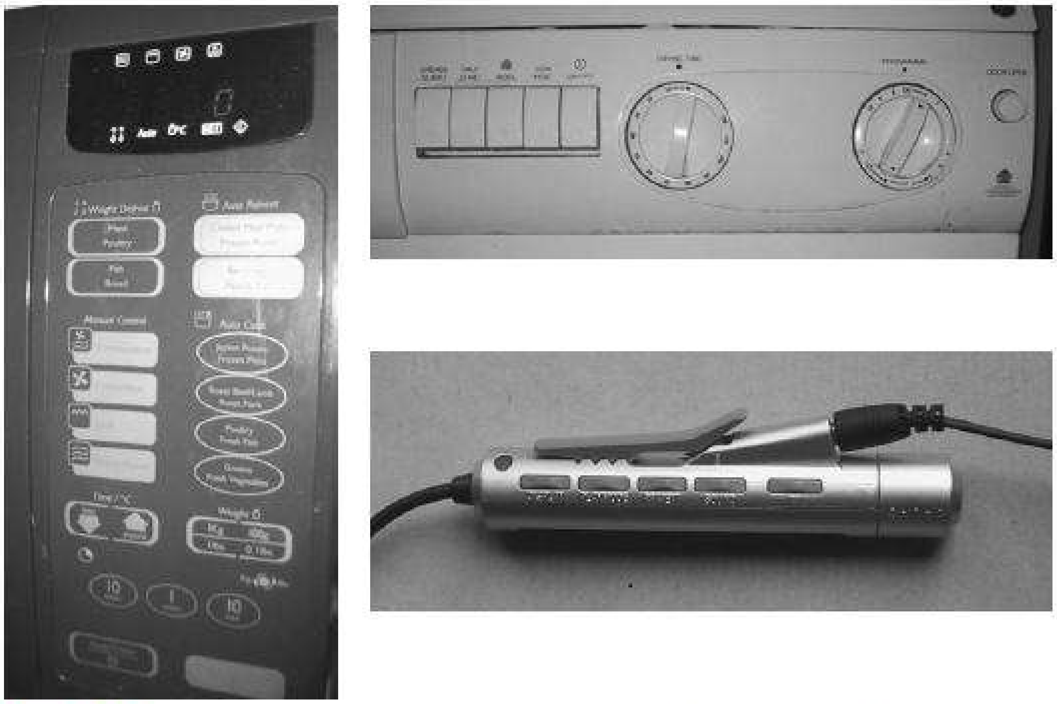


Figure 2.13 Physical controls on microwave, washing machine and MiniDisc. Source: Photograph bottom right by Alan Dix with permission from Sony (UK)

settings and the large buttons act both as control and display. Also the dials for dryer timer and the washing program act both as a means to set the desired time or program and to display the current state whilst the wash is in progress. Finally, the MiniDisc controller needs to be small and unobtrusive. It has tiny buttons, but the end control is most interesting. It twists from side to side and also can be pulled and twisted. This means the same control can be used for two different purposes. This form of multi-function control is common in small devices. We discussed the immediacy of haptic feedback and these lessons are also important at the level of creating physical devices; do keys, dials, etc., feel as if they have been pressed or turned? Getting the right level of resistance can make the device work naturally, give you feedback that you have done something, or let you know that you are controlling something. Where for some reason this is not possible, something has to be done to prevent the user getting confused, perhaps pressing buttons twice; for example, the smooth control panel of the microwave in Figure 2.13
offers no tactile feedback, but beeps for each keypress

2.7 PAPER: PRINTING AND SCANNING

Some years ago, a recurrent theme of information technology was the *paperless office*. In the paperless office, documents would be produced, dispatched, read and filed online. The only time electronic information would be committed to paper would be when it went out of the office to ordinary customers, or to other firms who were laggards in this technological race. This vision was fuelled by rocketing property prices, and the realization that the floor space for a wastepaper basket could cost thousands in rent each year. Some years on, many traditional paper files are now online, but the desire for the completely paperless office has faded. Offices still have wastepaper baskets, and extra floor space is needed for the special computer tables to house 14-inch color monitors.

2.7.1 Printing

If anything, computer systems have made it easier to produce paper documents. It is so easy to run off many copies of a letter (or book), in order to get it looking 'just right'. Older printers had a fixed set of characters available on a printhead. 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 printhead, 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,

limited only by the resolution of the dots. 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. For some mechanical printers this is slightly confused: the dots printed may be bigger than the gaps, neighboring printheads may not be able to print simultaneously and may be offset relative to one another (a diamond-shaped rather than rectangular grid). These differences do not make too much difference to the user, but mean that, given two printers at the same nominal resolution, the output of one looks better than that of the other, because it has managed the physical constraints better.

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.

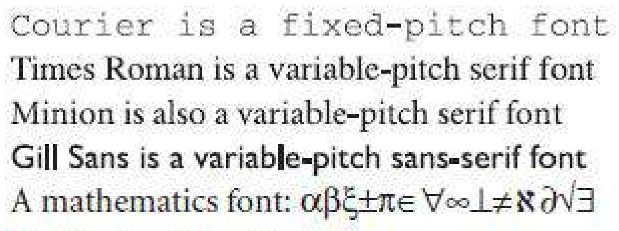
2.7.2 Fonts and page description languages

Some printers can act in a mode whereby any characters sent to them (encoded in ASCII, see Section 2.8.5) are printed, typewriter style, in a single font. Another case, simple in theory, is when you have a bitmap picture and want to print it. The dots to print are sent to the printer, and no further interpretation is needed. However, in practice, it is rarely so simple.

Many printed documents are far more complex – they incorporate text in many different fonts and many sizes, often italicized, emboldened and underlined. Within the text you will find line drawings, digitized photographs and pictures generated from ‘paint’ packages, including the ubiquitous ‘clip art’. Sometimes the computer does all the work, converting the page image into a bitmap of the right size to be sent to the printer. Alternatively, a description of the page may be sent to the printer. At the simplest level, this will include commands to set the print position on the page, and change the font size. More sophisticated printers can accept a *page description language*, the most common of which is PostScript. This is a form of programming language for printing. It includes some standard programming constructs, but also some special ones: paths

for drawing lines and curves, sophisticated character and font handling and scaled bitmaps. The idea is that the description of a page is far smaller than the associated bitmap, reducing the time taken to send the page to the printer. A bitmap version of an A4 laser printer page at 300 dpi takes 8 Mbytes; to send this down a standard serial printer cable would take 10 minutes! However, a computer in the printer has to interpret the PostScript program to print the page; this is typically faster than 10 minutes, but is still the limiting factor for many print jobs.

Text is printed in a font with a particular size and shape. The size of a font is measured in points (pt). The point is a printer’s measure and is about 1/72 of an inch. The *point size* of the font is related to its height: a 12 point font has about six lines per inch. The shape of a font is determined by its *font name*, for example Times Roman, Courier or Helvetica. Times Roman font is similar to the type of many newspapers, such as *The Times*, whereas Courier has a typewritten shape.



Courier is a fixed-pitch font
Times Roman is a variable-pitch serif font
Minion is also a variable-pitch serif font
Gill Sans is a variable-pitch sans-serif font
A mathematics font: αβξ±π∈ ∀∞⊥≠ℵ∂√∃

Figure 2.14 Examples of different fonts

Some fonts, such as Courier, are *fixed pitch*, that is each character has the same width. The alternative is a variable-pitched font, such as Times Roman or Gill Sans, where some characters, such as the ‘m’, are wider than others, such as the ‘ i’. Another characteristic of fonts is whether they are *serif* or *sans-serif*. A serif font has fine, short cross-lines at the ends of the strokes, imitating those found on cut stone lettering. A sans-serif font has square-ended strokes. In addition, there are special fonts looking like Gothic lettering or cursive script, and fonts of Greek letters and special mathematical symbols.

2.7.3 Screen and page

A common requirement of word processors and desktop publishing software is that *what you see is what you get* (see also Chapters 4 and 17), which is often called by its acronym *WYSIWYG* (pronounced whizz-ee-wig). This means that the appearance of the document on the screen should be the same as its eventual appearance on the printed page. In so far as this means that, for example, centered text is displayed centered on the screen, this is reasonable. However, this should not cloud the fact that screen and paper are very different media.

A typical screen resolution is about 72 dpi compared with a laser printer at over 600 dpi. Some packages can show magnified versions of the document in order to help in this. Most screens use an additive color model using red, green and bl ue light, whereas printers use a subtractive color model with cyan, magenta, yellow and black inks, so conversions have to be made. In addition, the sizes and aspect ratios are very different. An A4 page is about 11 inches tall by 8 wide (297210 mm), whereas a screen is often of similar dimensions, but wider than it is tall.

These differences cause problems when designing software. Should you try to make the screen image as close to the paper as possible, or should you try to make the best of each? One approach to this would be to print only what could be displayed, but that would waste the extra resolution of the printer. On the other hand, one can try to make the screen as much like paper as possible, which is the intention behind the standard use of black text on a white background, rotatable A4 displays, and tablet PCs. This is a laudable aim, but cannot get rid of all the problems.

2.7.4 Scanners and optical character recognition

Printers take electronic documents and put them on paper – *scanners* reverse this process. They start by turning the image into a bitmap, but with the aid of *optical character recognition* can convert the page right back into text. The image to be converted may be printed, but may also be a photograph or hand-drawn picture. There are two main kinds of scanner: flat-bed and hand-held. With a flat-bed scanner, the page is placed on a flat glass plate and the whole page is converted into a bitmap. A variant of the flat-bed is where sheets to be scanned are pulled through the machine, common in multi-function devices (printer/fax/copier). Many flat-bed scanners allow a small pile of sheets to be placed in a feed tray so that they can all be scanned without user intervention. Hand-held scanners are pulled over the image by hand. As the head passes over an area it is read in, yielding a bitmap strip. A roller at the ends ensures that the scanner knows how fast it is being pulled and thus how big the image is. The scanner is typically only 3 or 4 inches (80 or 100 mm) wide and may even be the size of a large pen (mainly used for scanning individual lines of text).

This means at least two or three strips must be 'glued' together by software to make a whole page image, quite a difficult process as the strips will overlap and may not be completely parallel to one another, as well as suffering from problems of different brightness and contrast. However, for desktop publishing small images such as photographs are quite common, and as long as one direction is less than the width of the scanner, they can be read in one pass. Scanners work by shining a beam of light at the page and then recording the intensity and color of the reflection. Like photocopiers, the color of the light that is shone means that some colors may appear darker than others on a monochrome scanner.
For example, if the light is pure red, then a red image will reflect the light completely

and thus not appear on the scanned image. Like printers, scanners differ in resolution, commonly between 600 and 2400 dpi, and like printers the quoted resolution needs careful interpretation. Many have a lower resolution scan head but digitally interpolate additional pixels – the same is true for some digital cameras. Monochrome scanners are typically only found in multi-function devices, but color scanners usually have monochrome modes for black and white or grayscale copying. Scanners will usually return up to 256 levels of gray or RGB (red, green, blue) color. If a pure monochrome image is required (for instance, from a printed page), then it can *threshold* the grayscale image; that is,turn all pixels darker than some particular value black, and the rest white.

2.8 MEMORY

2.8.1 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. By strict analogy, non-volatile RAM ought to be classed as LTM, but the important thing we want to emphasize is the gulf between STM and LTM in a traditional computer system. In PDAs the distinctions become more confused as the battery power means that the system is never completely off, so RAM memory effectively lasts for ever. Some also use flash memory, which is a form of silicon memory that sits between fixed content ROM (read-only memory) chips and normal RAM. Flash memory is relatively slow to write, but once written retains its content even with no power whatsoever. These are sometimes called silicon disks on PDAs. Digital cameras typically store photographs in some form of flash media and small flash-based devices are used to plug into a laptop or desktop's USB port to transfer data.

2.8.2 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, you can have as many as you have room for on your desk. Hard disks may store from under 40 Mbytes to several gigabytes (Gbytes), that is several thousand million bytes. With disks there are two access times to consider, the time taken to find the right track on the disk, and the time to read the track. The former dominates random reads, and is typically of the order of 10 ms for hard disks. The transfer rate once the track is found is then very high, perhaps several hundred kilobytes per second. 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.

2.8.3 Understanding speed and capacity

So what effect do the various capacities and speeds have on the user? Thinking of our typical personal computer system, we can summarize some typical capacities as in Table 2.1. We think first of documents. This book is about 320,000 words, or about 2 Mbytes, so it would hardly make a dent in 256 Mbytes of RAM. (This size – 2 Mbytes – is unformatted and without illustrations; the actual size of the full data files is an order of magnitude bigger, but still well within the capacity of main memory.) To take a more popular work, the Bible would use about 4.5 Mbytes. This would still consume only 2% of main memory, and disappear on a hard disk. However, it might look tight on a smaller PDA. This makes the memory look not too bad, so long as you do not intend to put your entire library online. However, many word processors
come with a dictionary and thesaurus, and there is no standard way to use the same one with several products. Together with help files and the program itself, it is not

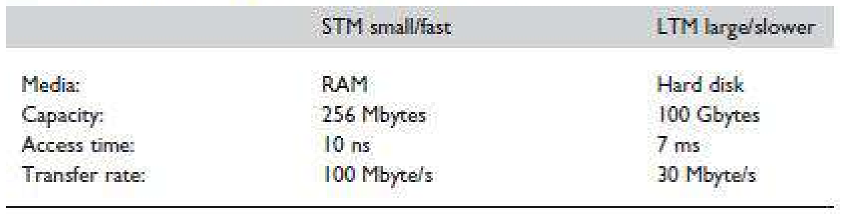


Table 2.1 Typical capacities of different storage media

	STM small/fast	LTM large/slower
Media:	RAM	Hard disk
Capacity:	256 Mbytes	100 Gbytes
Access time:	10 ns	7 ms
Transfer rate:	100 Mbyte/s	30 Mbyte/s

unusual to find each application consuming tens or even hundreds of megabytes of disk space – it is not difficult to fill a few gigabytes of disk at all! Similarly, although 256 Mbytes of RAM are enough to hold most (but not all) single programs, windowed systems will run several applications simultaneously, soon using up many megabytes. Operating systems handle this by *paging* unused bits of programs out of RAM onto disk, or even *swapping* the entire program onto disk. This makes little difference to the logical functioning of the program, but has a significant effect on interaction. If you select a window, and the relevant application happens to be currently swapped out onto the disk, it has to be swapped back in. The delay this causes can be considerable, and is both noticeable and annoying on many systems.

2.8.4 Compression

In fact, things are not quite so bad, since *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 (this book mentions ‘user’ and ‘computer’ rather frequently). Similarly, in a bitmap, if one bit is white, there is a good chance the next will be as well. Compression algorithms take advantage of this redundancy. For example, *Huffman encoding* gives short codes to frequent words [182], 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. For storage purposes this is not a problem, but when used for transmission over telephone lines or networks it can mean glitches in the video as the system catches up.
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.

2.8.5 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. 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’. There is a 16 bit extension, the UNICODE standard, which has enough room for a much larger range of characters including the Japanese Kanji character set.

The text is in different fonts and includes formatting information such as centering, page headers and footers. On the whole, the storage of formatted text is vendor specific, since virtually every application has its own file format. This is not helped by the fact that many suppliers attempt to keep their file formats secret, or update them frequently to stop others’ products being compatible. With the exception of bare ASCII, 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 the se 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.

2.8.6 Methods of access

Standard database access is by special key fields with an associated index. The user has to know the key before the system can find the information. A telephone directory is a good example of this. You can find out someone's telephone number if you know their name (the key), but you cannot find the name given the number. This is evident in the interface of many computer systems. So often, when you contact an organization, they can only help you if you give your customer number, or last order number. The usability of the system is seriously impaired by a shortsighted reliance on a single key and index. In fact, most database systems will allow multiple keys and indices, allowing you to find a record given partial information. So these problems are avoidable with only slight foresight.

There are valid reasons for not indexing on too many items. Adding extra indices adds to the size of the database, so one has to balance ease of use against storage cost. However, with ever-increasing disk sizes, this is not a good excuse for all but extreme examples. Unfortunately, brought up on lectures about algorithmic efficiency, it is easy for computer scientists to be stingy with storage. Another, more valid, reason for restricting the fields you index is privacy and security. For example, telephone companies will typically hold an online index that, given a telephone number, would return the name and address of the subscriber, but to protect the privacy of their customers, this information is not divulged to the general public.

2.9 PROCESSING AND NETWORKS

2.9.1 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! We saw one example of the former above. 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. There is typically no way for the user to tell whether the buffer is empty or not, except by interacting very slowly with the system and observing that the cursor has moved after every keypress. 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. So the user clicks again, tries something else – then, suddenly, all the buffered mouse clicks are interpreted and the screen becomes a blur of flashing windows and menus. This time, it is not so much that the response is too slow – it is fast enough when it happens – but that the response is variable. The delays due to swapping programs in and out of main memory typically cause these problems.

2.9.2 Limitations on interactive performance

There are several factors that can limit the speed of an interactive system:

Computation bound This is rare for an interactive program, but possible, for example when using find/replace in a large document. The system should be designed so that long delays are not in the middle of interaction and so that the user gets some idea of how the job is progressing. For a very long process try to give an indication of duration *before* it starts; and during processing an indication of the stage that the process has reached is helpful. This can be achieved by having a counter or slowly filling bar on the screen that indicates the amount done, or by changing the cursor to indicate that processing is occurring. Many systems notice after they have been computing for some time and then say ‘this may take some time: continue (Y/N)?’. Of course, by the time it says this the process may be nearly finished anyway!

Storage channel bound As we discussed in the previous section, the speed of memory access can interfere with interactive performance. We discussed one technique, laziness, for reducing this effect. In addition, if there is plenty of raw computation power and the system is held up solely by memory, it is possible to trade off memory against processing speed. For example, compressed data take less space to store, and is faster to read in and out, but must be compressed before storage and decompressed when retrieved. Thus faster memory access leads to increased processing time. If data is written more often than it is read, one can choose a technique that is expensive to compress but fairly simple to decompress. For many interactive systems the ability to browse quickly is very important, but users will accept delays when saving updated information.

Graphics bound For many modern interfaces, this is the most common bottleneck. It is easy to underestimate the time taken to perform what appear to be simple interface operations. Sometimes clever coding can reduce the time taken by common graphics operations, and there is tremendous variability in performance between programs running on the same hardware. Most computers include a special-purpose *graphics card* to handle many of the most common graphics operations. This is optimized for graphics operations and allows the main processor to do other work such as manipulating documents and other user data.

Network capacity Most computers are linked by networks. At the simplest this can mean using shared files on a remote machine. When accessing such files it can be the speed of the network rather than that of the memory which limits performance.

2.9.3 Networked computing

Computer systems in use today are much more powerful than they were a few years ago, which means that the standard computer on the desktop is quite capable of high-performance interaction without recourse to outside help. However, it is often the case that we use computers not in their standalone mode of operation, but linked together in networks. This brings added benefits in allowing communication between different parties, provided they are connected into the same network, as well as allowing the desktop computer to access resources remote from itself. Such networks are inherently much more powerful than the individual computers that make up the network: increased computing power and memory are only part of the story, since the effects of allowing people much more extensive, faster and easier access to information are highly significant to individuals, groups and institutions. One of the biggest changes since the first edition of this book has been the explosive growth of the internet and global connectivity. As well as fixed networks it is now no rmal to use a high bandwidth modem or wireless local area network (LAN) to connect into the internet and world wide web from home or hotel room anywhere in the world.

The effects of this on society at large can only be speculated upon at present, but there are already major effects on computer purchases and perhaps the whole face of personal computation. As more and more people buy computers principally to connect to the internet the idea of the *network computer* has arisen – a small computer with no disks whose sole purpose is to connect up to networks.

Such networked systems have an effect on interactivity, over and above any additional access to distant peripherals or information sources. Networks sometimes operate over large distances, and the transmission of information may take some appreciable time, which affects the response time of the system and hence the nature of the interactivity. There may be a noticeable delay in response, and if the user is not informed of what is going on, he may assume that his command has been ignored, or lost, and may then repeat it. This lack of feedback is an important factor in the poor performance and frustration users feel when using such systems, and can be alleviated by more sensible use of the capabilities of the desktop machine to inform users of what is happening over the network.

Another effect is that the interaction between human and machine becomes an open loop, rather than a closed one. Many people may be interacting with the machine at once, and their actions may affect the response to your own. Many users accessing a single central machine will slow its response; database updates carried out by one user may mean that the same query by another user at slightly different times may produce different results. The networked computer system, by the very nature of its dispersal, distribution and multi-user access, has been transformed from a fully predictable, deterministic system, under the total control of the user, into a nondeterministic one, with an individual user being unaware of many important things that are happening to the system as a whole. Such systems pose a particular problem since ideals of consistency, informative feedback and predictable response are violated (see Chapter 7 for more on these principles). However, the additional power and flexibility offered by networked systems means that they are likely to be with us for a long time, and these issues need to be carefully addressed in their design.

THE INTERACTION

MODELS OF INTERACTION

The interface must therefore effectively translate between them to allow the interaction to be successful. This translation can fail at a number of points and for a number of reasons. The use of models of interaction can help us to understand exactly what is going on in the interaction and identify the likely root of difficulties. They also provide us with a framework to compare different interaction styles and to consider interaction problems.

3.2.1 The terms of interaction

Traditionally, the purpose of an interactive system is to aid a user in accomplishing *goals* from some application *domain*. (Later in this book we will look at alternative interactions but this model holds for many work-oriented applications.) 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* are 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.

for the user of an interactive system in terms of the domain, goals, intentions and tasks. We can use our knowledge of tasks and goals to assess the interactive system that is designed to support them. The concepts used in the design of the system and the description of the user are separate, and so we can refer to them as distinct components, called the *System* and the *User*, respectively. The *System* and *User* are each described by means of a language that can express concepts relevant in the domain of the application. The *System*'s language we will refer to as the *core language* and the *User*'s language we will refer to as the *task language*. 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.

3.2.2 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 [265]. 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:

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.

Norman uses this model of interaction to demonstrate why some interfaces cause problems to their users. He describes these in terms of the *gulfs of execution* and the *gulfs of evaluation*. As we noted earlier, the user and the system do not use the same terms to describe the domain and goals – remember that we called the language of the system the *core language* and the language of the user the *task language*. The 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 actions allowed by the system correspond to those intended by the user, the interaction will be effective.

The interface should therefore aim to reduce this gulf.
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.

3.2.3 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 Figure 3.1. 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*.
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 Figure 3.2. The *User* begins the interact ve cycle with the formulation of a goal and a task to achieve 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

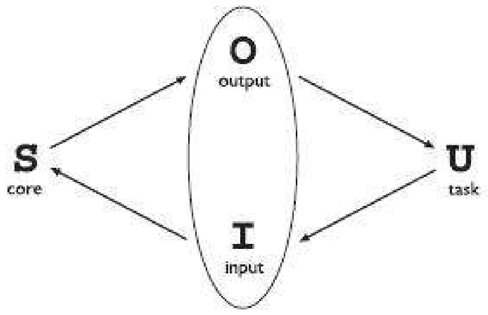


Figure 3.1 The general interaction framework

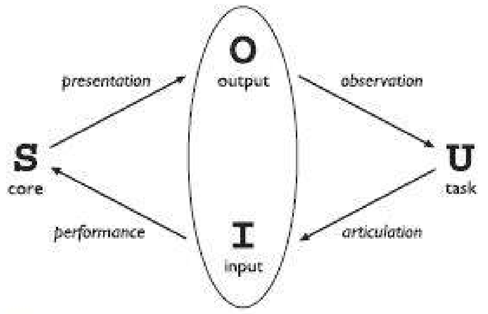


Figure 3.2 Translations between components

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. An example of a poor mapping, as pointed

out by Norman, is a large room with overhead lighting controlled by a bank of switches. It is often desirable to control the lighting so that only one section of the room is lit. We are then faced with the puzzle of determining which switch controls which lights. The result is usually repeated trials and frustration. Thi s arises from the difficulty of articulating a goal (for example, ‘Turn on the lights in the front of the room’) in an input language that consists of a linear row of switches, which may or may not be oriented to reflect the room layout. Conversely, an example of a good mapping is in virtual reality systems, where input devices such as datagloves are specifically geared towards easing articulation by making the user’s psychological notion of gesturing an act that can be directly realized at the interface. Direct manipulation interfaces, such as those found on common desktop operating systems like the Macintosh and Windows, make the articulation of some

file handling commands easier. On the other hand, some tasks, such as repetitive file renaming or launching a program whose icon is not visible, are not at all easy to articulate with such an interface. At the next stage, the responses of the *Input* are translated to stimuli for the *System*. Of interest in assessing this translation is whether the translated input language can reach as many states of the *System* as is possible using the *System* stimuli directly. For example, the remote control units for some compact disc players do not allow the user to turn the power off on the player unit; hence the off state of the player cannot be reached using the remote control’s input language. On the panel of the compact disc player, however, there is usually a button that controls the power.

The ease with which this translation from *Input* to *System* takes place is of less importance because the effort is not expended by the user. However, there can be a real effort expended by the designer and programmer. In this case, the ease of the translation is viewed in terms of the cost of implementation. Once a state transition has occurred within the *System*, the execution phase of the interaction is complete and the evaluation phase begins. The new state of the *System* must be communicated to the *User*, and this begins by translating the *System* responses to the transition into stimuli for the *Output* component.

This presentation translation must preserve the relevant system attributes from the domain in the limited expressiveness of the output devices. The ability to capture the domain concepts of the *System* within the *Output* is a question of expressiveness for this translation. For example, while writing a paper with some word-processing package, it is necessary at times to see both the immediate surrounding text where one is currently composing, say, the current paragraph, and a wider context within the whole paper that cannot be easily displayed on one screen Ultimately, the user must interpret the output to evaluate what has happened. The response from the *Output* is translated to stimuli for the *User* which trigger assessment The observation translation will address the ease and coverage of this final translation. For example, it is difficult to tell the time accurately on an unmarked analog clock, especially if it is not oriented properly. It is difficult in a command line interface to determine the result of copying and moving files in a hierarchical
file system. Developing a website using a markup language like HTML would be virtually impossible without being able to preview the output through a browser.

FRAMEWORKS AND HCI

As well as providing a means of discussing the details of a particular interaction, frameworks provide a basis for discussing other issues that relate to the interaction. The ACM SIGCHI Curriculum Development Group presents a framework similar to that presented here, and uses it to place different areas that relate to HCI [9]. In Figure 3.3 these aspects are shown as they relate to the interaction framework. In particular, the field of *ergonomics* addresses issues on the user side of the interface, covering both input and output, as well as the user's immediate context. Dialog design and interface styles can be placed particularly along the input branch of the framework, addressing both articulation and performance. However, dialog is most usually associated with the computer and so is biased to that side of the framework.

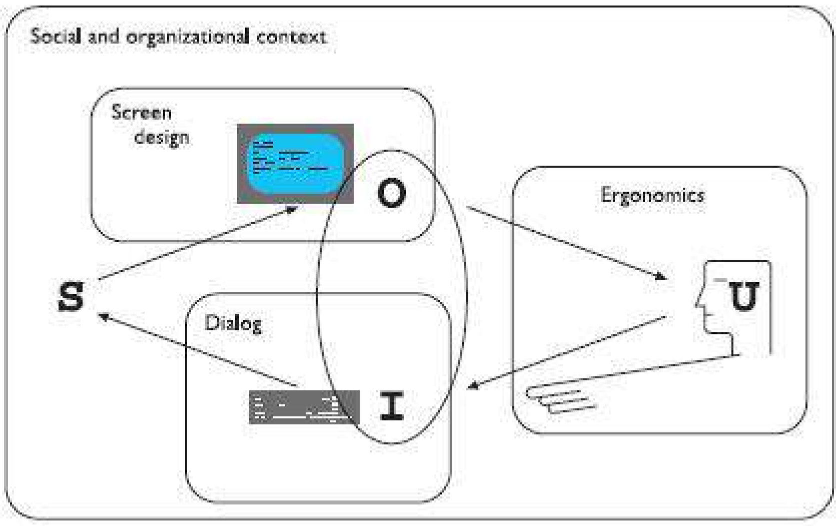


Figure 3.3 A framework for human–computer interaction. Adapted from ACM SIGCHI Curriculum Development Group [9]

Presentation and screen design relates to the output branch of the framework. The entire framework can be placed within a social and organizational context that also affects the interaction. Each of these areas has important implications for the design of interactive systems and the performance of the user.

3.3 ERGONOMICS

Ergonomics (or human factors) is traditionally the study of the physical characteristics 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, and full coverage would demand a book in its own right. Here we consider a few of the issues addressed by ergonomics as an introduction to the field. We will briefly look at the arrangement of controls and displays, the physical environment, health issues and the use of color. These are by no means exhaustive and are intended only to give an indication of the types of issues and problems addressed by ergonomics. For more information on ergonomic issues the reader is referred to the recommended reading list at the end of the chapter.

3.4.1 Arrangement of controls and displays

In Chapter 1 we considered perceptual and cognitive issues that affect the way we present information on a screen and provide control mechanisms to the user. In addition to these cognitive aspects of design, physical aspects are also important. This may not seem so important when we are considering a single user of a spreadsheet on a PC, but it becomes vital when we turn to safety-critical applications such as plant control, aviation and air traffic control. In each of these contexts, users are under pressure and are faced with a huge range of displays and controls. Here it is crucial that the physical layout of these be appropriate. Indeed, returning to the less critical PC application, inappropriate placement of controls and displays can lead to inefficiency and frustration.

For example, on one particular electronic newsreader, used by one of the authors, the command key to read articles from a newsgroup (y) is directly beside the command key to unsubscribe from a newsgroup (u) on the keyboard. This poor design frequently leads to inadvertent removal of newsgroups. Although this is recoverable it wastes time and is annoying to the user. We saw similar examples in the Introduction to this book including the MacOS X dock. We can therefore see that appropriate layout is important in all applications.
The exact organization that this will suggest will depend on the domain and the application, but possible organizations include the following:

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. Controls should be spaced to provide adequate room for the user to manoeuvre.

3.4.2 The physical environment of the interaction

As well as addressing physical issues in the layout and arrangement of the machine interface, ergonomics is concerned with the design of the work environment itself. Where will the system be used? By whom will it be used? Will users be sitting, standing or moving about? Again, this will depend largely on the domain and will be more critical in specific control and operational settings than in general computer use.

However, the physical environment in which the system is used may influence how well it is accepted and even the health and safety of its users. It should therefore be considered in all design. The first consideration here is the size of the users. Obviously this is going to vary considerably. However, in any system the smallest user should be able to reach all the controls (this may include a user in a wheelchair), and the largest user should not be cramped in the environment.

In particular, all users should be comfortably able to see critical displays. For long periods of use, the user should be seated for comfort and stability. Seating should provide back support. If required to stand, the user should have room to move around in order to reach all the controls.

3.4.3 Health issues

Perhaps we do not immediately think of computer use as a hazardous activity but we should bear in mind possible consequences of our designs on the health and safety of users. Leaving aside the obvious safety risks of poorly designed safety-critical systems (aircraft crashing, nuclear plant leaks and worse), there are a number of factors that may affect the use of more general computers. Again these are factors in the physical environment that directly affect the quality of the interaction and the user's performance:

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 activi y.

Time The time users spend using the system should also be controlled. As we saw in the previous chapter, it has been suggested that excessive use of CRT displays can be harmful to users, particularly pregnant women.

3.4.4 The use of color

In this section we have concentrated on the ergonomics of physical characteristics
of systems, including the physical environment in which they are used. However,

ergonomics has a close relationship to human psychology in that it is also concerned with the perceptual limitations of humans. For example, the use of color in displays is an ergonomics issue.
Each of these psychological phenomena leads to ergonomic guidelines; some examples are discussed below. 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.

However, we should remember that color conventions are culturally determined. For example, red is associated with danger and warnings in most western cultures, but in China it symbolizes happiness and good fortune. The color of mourning is black in some cultures and white in others. Awareness of the cultural associations of color is particularly important in designing systems and websites for a global market.

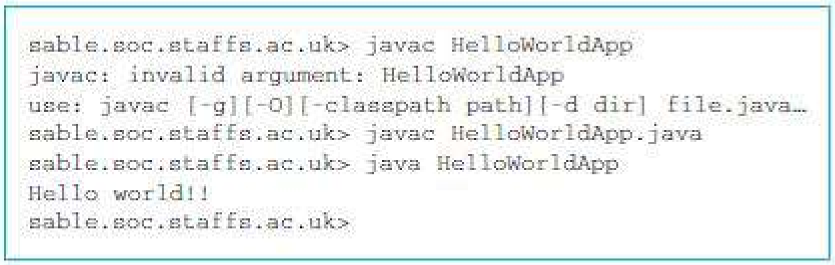
3.4.5 Ergonomics and HCI

Ergonomics is a huge area, which is distinct from HCI but sits alongside it. Its contribution to HCI is in determining constraints on the way we design systems and suggesting detailed and specific guidelines and standards. Ergonomic factors are in general well established and understood and are therefore used as the basis for standardizing hardware designs.

3.5 INTERACTION STYLES

Interaction can be seen as a dialog between the computer and the user. The choice of interface style can have a profound effect on the nature of this dialog. Dialog design is discussed in detail in Chapter 16. Here we introduce the most common interface styles and note the different effects these have on the interaction. There are a number of common interface styles including

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



```
sable.soc.staffs.ac.uk> javac HelloWorldApp
javac: invalid argument: HelloWorldApp
use: javac [-g][-O][-classpath path][-d dir] file.java…
sable.soc.staffs.ac.uk> javac HelloWorldApp.java
sable.soc.staffs.ac.uk> java HelloWorldApp
Hello world!!
sable.soc.staffs.ac.uk>
```

Figure 3.7 Command line interface

3.5.1 Command line interface

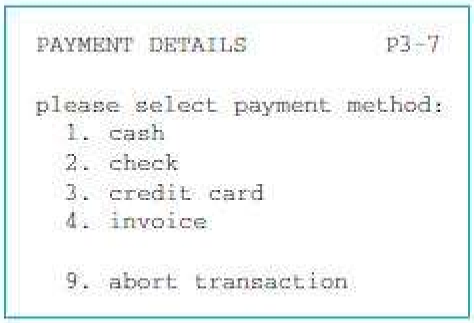
The command line interface (Figure 3.7) was the first interactive dialog style to be commonly used and, in spite of the availability of menu-driven interfaces, it is still widely used. It provides a means of expressing instructions to the computer directly, using function keys, single characters, abbreviations or whole-word commands. In some systems the command line is the only way of communicating with the system, especially for remote access using *telnet*. More commonly today it is supplementary to menu-based interfaces, providing accelerated access to the system's functionality for experienced users.

Command line interfaces are powerful in that they offer direct access to system functionality (as opposed to the hierarchical nature of menus), 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. However, this flexibility and power brings with it difficulty in use and learning.

Commands must be remembered, as no cue is provided in the command line to indicate which command is needed. They are therefore better for expert users than for novices. This problem can be alleviated a little by using consistent and meaningful commands and abbreviations. The commands used should be terms within the vocabulary of the user rather than the technician. Unfortunately, commands are often obscure and vary across systems, causing confusion to the user and increasing the overhead of learning.

3.5.2 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



```
PAYMENT DETAILS            P3-7

please select payment method:
   1. cash
   2. check
   3. credit card
   4. invoice

   9. abort transaction
```

Figure 3.8 Menu-driven interface

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 Figure 3.8), or may have a graphical component in which the menu appears within a rectangular box and choices are made, perhaps by typing the initial letter of the desired selection, or by entering the associated number, or by moving around the menu with the arrow keys. This is a restricted form of a full WIMP system, described in more detail shortly.

3.5.3 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! Natural language understanding, both of speech and written input, is the subject of much interest and research. Unfortunately, however, the ambiguity of natural language makes it very difficult for a machine to understand. Language is ambiguous at a number of levels. First, the syntax, or structure, of a phrase may not be clear. If we are given the sentence The boy hit the dog with the stick we cannot be sure whether the boy is using the stick to hit the dog or whether the dog is holding the stick when it is hit.

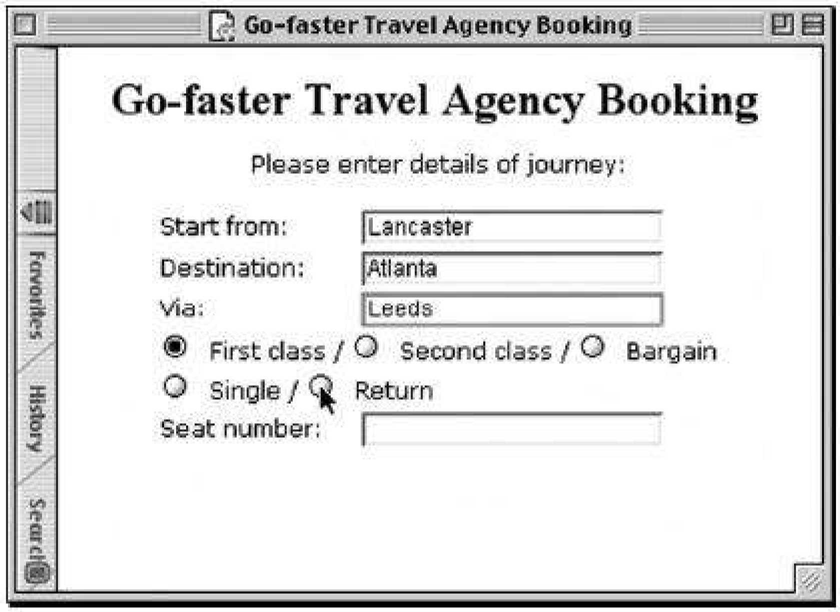
Even if a sentence’s structure is clear, we may find ambiguity in the meaning of the words used. For example, the word ‘pitch’ may refer to a sports field, a throw, a waterproofing substance or even, colloquially, a territory. We often rely on the context and our general knowledge to sort out these ambiguities. This information is difficult to provide to the machine. To complicate matters more, the use of pronouns and relative terms adds further ambiguity

3.5.4 Question/answer and query dialog

Question and answer dialog is a simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions (mainly with yes/no responses, multiple choice, or codes) and so is led through the interaction step by step. An example of this would be web questionnaires. These interfaces are easy to learn and use, but are limited in functionality and power. As such, they are appropriate for restricted domains (particularly information systems) 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 the 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 if 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 languages 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. A specialized example is the web search engine.

3.5.5 Form-fills and spreadsheets

Form-filling interfaces are used primarily for data entry but can also be useful in data retrieval applications. The user is presented with a display resembling a paper



Go-faster Travel Agency Booking

Please enter details of journey:

Start from:	Lancaster
Destination:	Atlanta
Via:	Leeds

[✓ First class] / Second class / Bargain

Single / Return

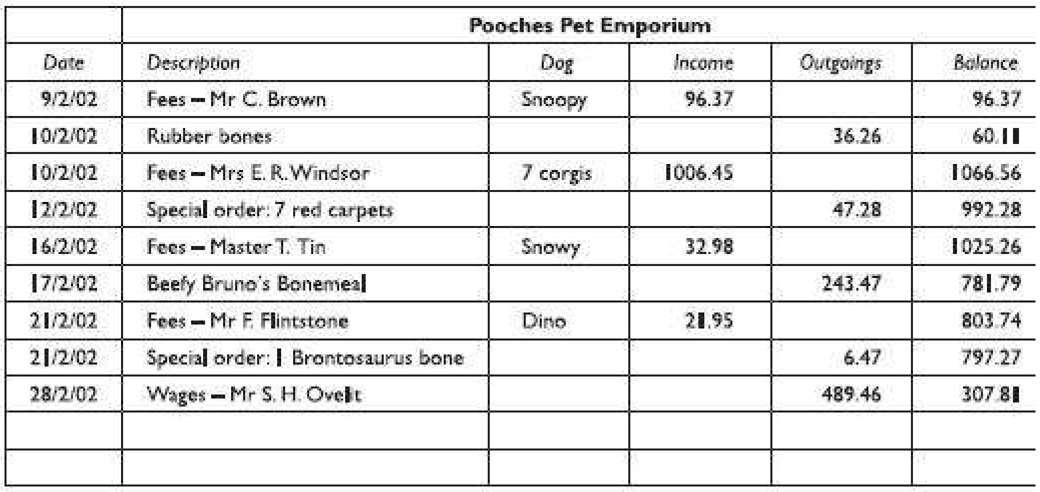
Seat number:

Figure 3.9 A typical form-filling interface. Screen shot frame reprinted by permission from Microsoft Corporation

form, with slots to fill in (see Figure 3.9). Often the form display is based upon an actual form with which the user is familiar, which makes the interface easier to use. The user works through the form, filling in appropriate values. The data are then entered into the application in the correct place. 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. The dialog style is useful primarily for data entry applications and, as it is easy to learn and use, for novice users.

However, assuming a design that allows flexible entry, form filling is also appropriate for expert users. Spreadsheets are a sophisticated variation of form filling. The spreadsheet comprises a grid of cells, each of which can contain a value or a formula (see Figure 3.10).

The formula can involve the values of other cells (for example, the total of all cells in this column). The user can enter and alter values and formulae in any order and the system will maintain consistency amongst the values displayed, ensuring that all formulae are obeyed. The user can therefore manipulate values to see the effects of changing different parameters. Spreadsheets are an attractive medium for interaction: the user is free to manipulate values at will and the distinction between input and output is blurred, making the interface more flexible and natural.



	Pooches Pet Emporium				
Date	Description	Dog	Income	Outgoings	Balance
9/2/02	Fees – Mr C. Brown	Snoopy	96.37		96.37
10/2/02	Rubber bones			36.26	60.11
10/2/02	Fees – Mrs E. R. Windsor	7 corgis	1006.45		1066.56
12/2/02	Special order: 7 red carpets			47.28	992.28
16/2/02	Fees – Master T. Tin	Snowy	32.98		1025.26
17/2/02	Beefy Bruno's Bonemeal			243.47	781.79
21/2/02	Fees – Mr F. Flintstone	Dino	21.95		803.74
21/2/02	Special order: 1 Brontosaurus bone			6.47	797.27
28/2/02	Wages – Mr S. H. Ovelt			489.46	307.81

Figure 3.10 A typical spreadsheet

3.5.6 The WIMP interface

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 (sometimes windows, icons, mice and pull-down menus), and is the default interface style for the majority of interactive computer systems in use today, especially in the PC and desktop workstation arena. Examples of WIMP interfaces include Microsoft Windows for IBM PC compatibles, MacOS for Apple Macintosh compatibles and various X Windows-based systems for UNIX.

3.5.7 Point-and-click interfaces

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. This point-and-click interface style is obviously closely related to the WIMP style. It clearly overlaps in the use of buttons, but may also include other WIMP elements. However, the philosophy is simpler and more closely tied to ideas of *hypertext*.

In addition, the point-and-click style is not tied to mouse-based interfaces, and is also extensively used in touchscreen information systems. In this case, it is often combined with a menu-driven interface.

The point-and-click style has been popularized by world wide web pages, which incorporate all the above types of point-and-click navigation: highlighted words, maps and iconic buttons.

3.5.8 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.

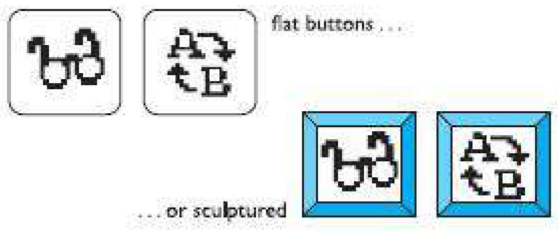


Figure 3.12 Buttons in 3D say 'press me'

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. By unstated convention, such interfaces have a light source at their top right. Where used judiciously, the raised areas are easily identifiable and can be used to highlight active areas (Figure 3.12). Unfortunately, some interfaces make indiscriminate use of sculptural effects, on every text area, border and menu, so all sense of differentiation is lost.

A more complex technique uses interfaces with 3D workspaces. The objects displayed in such systems are usually flat, but are displayed in perspective when at an angle to the viewer and shrink when they are 'further away'. Figure 3.13 shows one such systemThree-dimensional workspaces give you extra space, but in a more natural way than iconizing windows.

Finally, there are virtual reality and information visualization systems where the user can move about within a simulated 3D world.

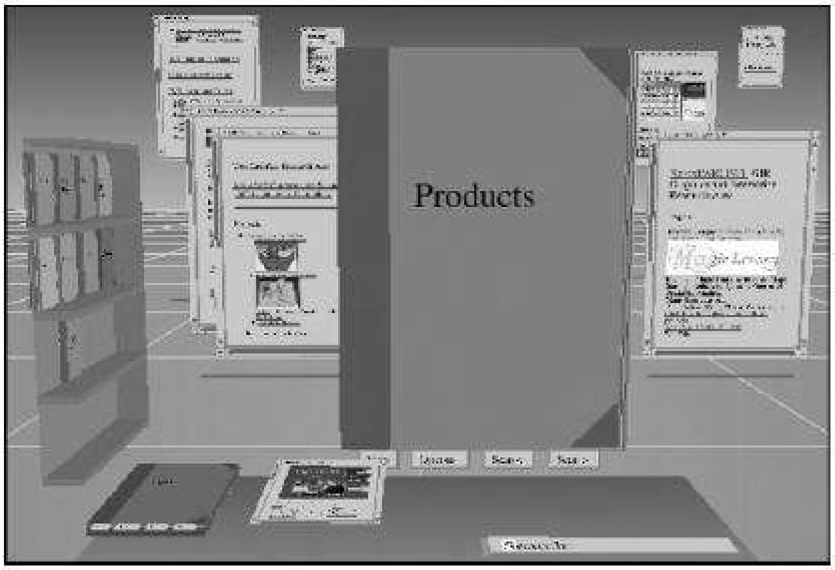


Figure 3.13 WebBook – using 3D to make more space (Card S.K., Robertson G.G. and York W. (1996). The WebBook and the Web Forager: An information workspace for the World-Wide Web. *CHI96 Conference Proceedings*, 111–17. Copyright © 1996 ACM, Inc. Reprinted by permission)

These mechanisms overlap with other interaction styles, especially the use of sculptured elements in WIMP interfaces. However, there is a distinct interaction style for 3D interfaces in that they invite us to use our tacit abilities for the real world, and translate them into the electronic world. Novice users must learn that an oval area with a word or picture in it is a button to be pressed, but a 3D button says 'push me'. Further, more complete 3D environments invite one to move within the virtual environment, rather than watch as a spectator.

ELEMENTS OF THE WIMP INTERFACE

We have already noted the four key features of the WIMP interface that give it its name – windows, icons, pointers and menus – and we will now describe these in turn. There are also many additional interaction objects and techniques commonly used in WIMP interfaces, some designed for specific purposes and others more general. We will look at buttons, toolbars, palettes and dialog boxes. Most of these elements can be seen in Figure 3.14 . There we will discover that though most modern windowing systems provide the same set of basic widgets, the 'look and feel' – how widgets are physically displayed and how users can interact with them to access their functionality – of different windowing systems and toolkits can differ drastically.

3.6.1 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

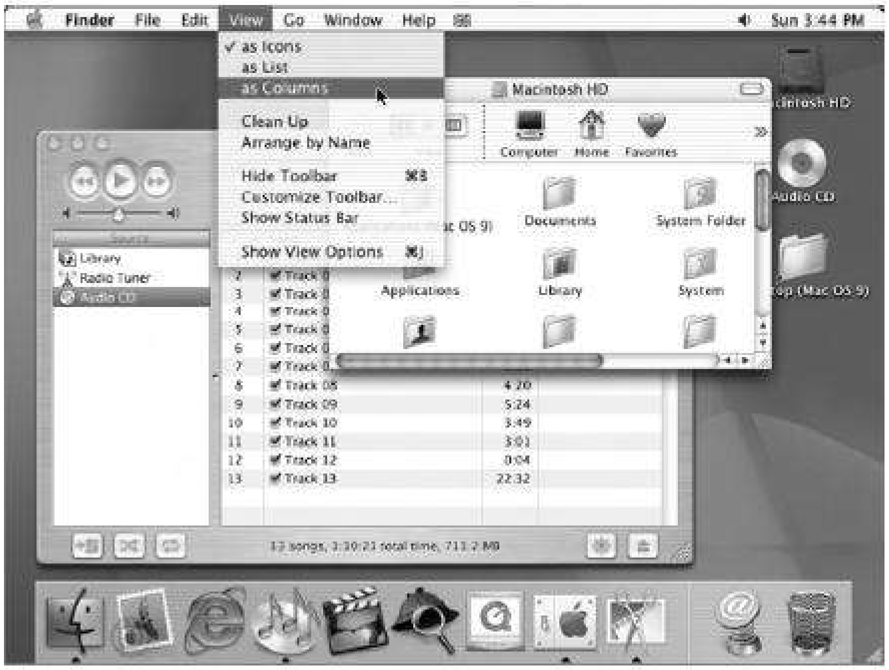
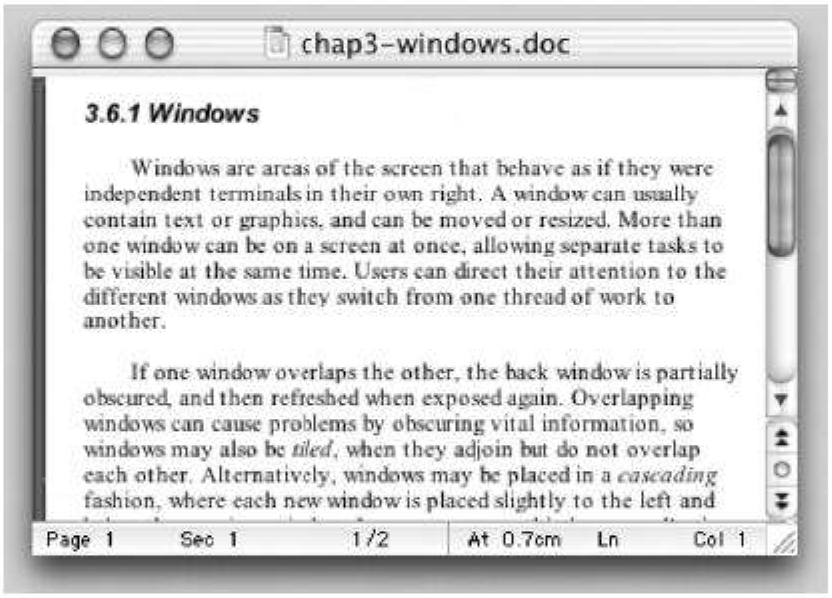


Figure 3.14 Elements of the WIMP interface – Microsoft Word 5.1 on an Apple Macintosh. Screen shot reprinted by permission from Apple Computer, Inc.

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 partia lly 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 it can be selected by the user.

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 Figure 3.15. 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.



chap3-windows.doc

3.6.1 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

Page 1 Sec 1 1/2 At 0.7cm Ln Col 1

Figure 3.15 A typical window. Screen shot reprinted by permission from Apple Computer, Inc.

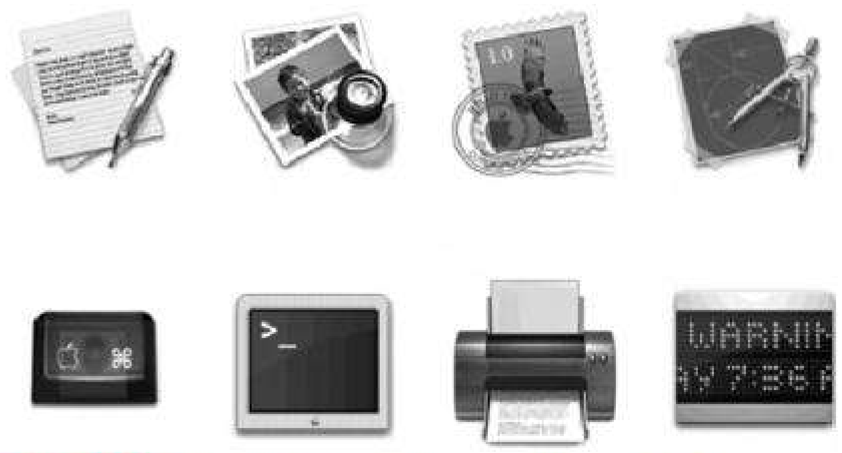


Figure 3.16 A variety of icons. Screen shot reprinted by permission from Apple Computer, Inc.

3.6.2 Icons

Windows can be closed and lost for ever, 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 c an 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 reminder to the user that he can subsequently resume the dialog by opening up the window. Figure 3.16 shows a few examples of some icons used in a typical windowing system (MacOS X). Icons can also be used to represent other aspects of the system, such as a wastebasket for throwing unwanted files into, or various disks, programs or functions that are accessible to the user. Icons can take many forms: they can be realistic representations 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.

3.6.3 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, The user is presented with a cursor on the screen that is controlled by the input device. A variety of pointer cursors are shown in Figure 3.17.

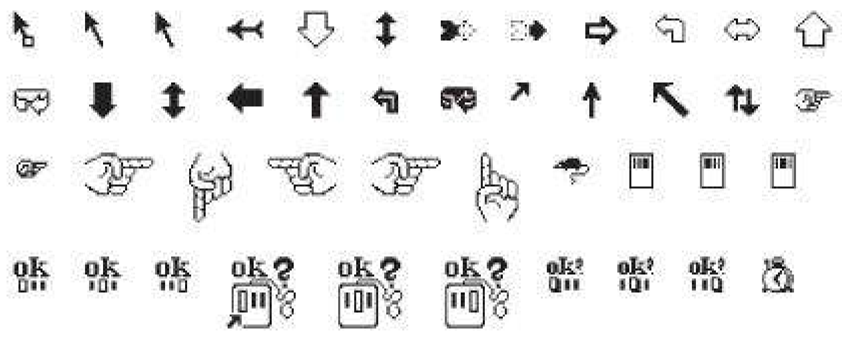


Figure 3.17 A variety of pointer cursors. Source: Sun Microsystems, Inc.

The different shapes of cursor are often used to distinguish *modes*, for example the normal pointer cursor may be an arrow, but change to cross-hairs when drawing a line. Cursors are also used to tell the user about system activity, for example a watch or hour-glass cursor may be displayed when the system is 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. For example, the three arrows at the start of Figure 3.17 each have a hot-spot at the top left, whereas the rightpointing hand on the second line has a hot-spot on its right. Sometimes the hot-spot
is not clear from the appearance of the cursor, in which case users will find it hard to click on small targets. When designing your own cursors, make sure the image has an obvious hot-spot.

3.6.4 Menus

The last main feature of windowing systems 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. 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 (by inverse video, or some similar strategy) 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.

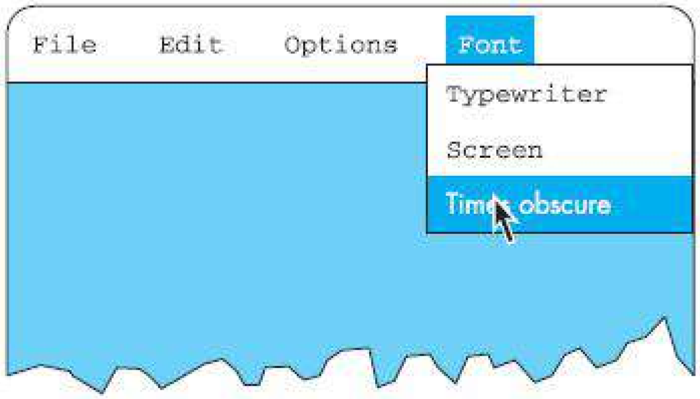


Figure 3.18 Pull-down menu

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 (Figure 3.18). Menu bars are often placed at the top of the screen (for example, MacOS) or at the top of each window (for example, Microsoft Windows). Alternatives include 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 contextsensitive 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 bar 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 are pin-up menus, in that they can be 'pinned' to the screen, staying in place until explicitly asked to go away. Pop-up menus appear when a particular region of the screen, maybe 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. we can see that it will take longer to select items near the bottom of the menu than at the top.

However, these *pie menus*, as they are known [54], take up more screen space and are therefore less common in interfaces.

3.6.5 Buttons

Buttons are individual and isolated regions within a 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. 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 from a set of mutually exclusive options, such as the size in points of the current font. These are called *radio buttons*, since the collection functions much like the old-fashioned mechanical control buttons on car radios. If a set of options is not mutually exclusive, such as font characteristics like bold, italics and underlining, 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*.

3.6.6 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 are made available, or choosing which of several predefined toolbars is displayed.

3.6.7 Palettes

In many application programs, interaction can enter 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 changes. For example, using the standard UNIX text editor vi, keystrokes can be interpreted either as operations to insert characters in the document (insert mode) or as operations to perform file manipulation (command mode). 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 or color selection in a drawing package.

3.6.8 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 system wants to save the file, a dialog box can be used to allow the user to name 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.

3.7 INTERACTIVITY

However, it is typically not used at a fine level of detail and deliberately ignores the 'semantic' level of an interface: for example, the validation of numeric information in a forms-based system.

Interactivity is also crucial in determining the 'feel' of a WIMP environment. All WIMP systems appear to have virtually the same elements: windows, icons, menus, pointers, dialog boxes, buttons, etc. However, the precise behavior of these elements differs both within a single environment and between environments. For example, we have already discussed the different behavior of pull-down and fall-down menus. These look the same, but fall-down menus are more easily invoked by accident (and not surprisingly the windowing environments that use them have largely fallen into disuse!). In fact, menus are a major difference between the MacOS and Microsoft Windows environments: in MacOS you have to keep the mouse depressed throughout menu selection; in Windows you can click on the menu bar and a pull-down menu appears and remains there until an item is selected or it is cancelled.

The exceptions to this are *pre-emptive* parts of the interface, where the system for various reasons wrests the initiative away from the user, perhaps because of a problem or because it needs information in order to continue. The major example of this is *modal dialog boxes*. It is often the case that when a dialog box appears the application will not allow you to do anything else until the dialog box has been completed or cancelled. In some cases this may simply block the application, but you can perform tasks in other applications. In other cases you can do nothing at all until the dialog box has been completed. An especially annoying example is when the dialog box asks a question, perhaps simply for confirmation of an action, but the information you need to answer is hidden by the dialog box! There are occasions when modal dialog boxes are necessary, for example when a major fault has been detected, or for certain kinds of instructional software. However, the general philosophy of modern systems suggests that one should minimize
the use of pre-emptive elements, allowing the user maximum flexibility. Interactivity is also critical in dealing with errors. We discussed slips and mistakes earlier in the chapter, and some ways to try to prevent these types of errors. The other way to deal with errors is to make sure that

the user or the system is able to tell when errors have occurred. If users can *detect* errors then they can correct them. This ability to detect and correct is important both at the small scale of button presses and keystrokes and also at the large scale. For example, if you have sent a client a letter and expect a reply, you can put in your diary a note on the day you expect a reply. If the other person forgets to reply or the letter gets lost in the post you know to send a reminder or ring when the due day passes.

THE CONTEXT OF THE INTERACTION

We have been considering the interaction between a user and a system, and how this is affected by interface design. This interaction does not occur within a vacuum. We have already noted some of the physical factors in the environment that can directly affect the quality of the interaction. This is part of the context in which the interaction takes place. But this still assumes a single user operating a single, albeit complex, machine. In reality, users work within a wider social and organizational context. This provides the wider context for the interaction, and may influence the activity and motivation of the user. Here we will confine our discussion to the influence social and organizational factors may have on the user's interaction with the system.

These may not be factors over which the designer has control. However, it is important to be aware of such influences to understand the user and the work domain fully. In order to perform well, users must be motivated. There are a number of possible sources of motivation, as well as those we have already mentioned, including fear, allegiance, ambition and self-satisfaction. The last of these is influenced by the user's perception of the quality of the work done, which leads to job satisfaction. If a system makes it difficult for the user to perform necessary tasks, or is frustrating to use, the user's job satisfaction, and consequently performance, will be reduced. The user may also lose motivation if a system is introduced that does not match the actual requirements of the job to be done. Often systems are chosen and introduced by managers rather than the users themselves. In some cases the manager's perception of the job may be based upon observation of results and not on actual activity. The system introduced may therefore impose a way of working that is unsatisfactory to the users. If this happens there may be three results: the system will be rejected, the users will be resentful and unmotivated, or the user will adapt the intended interaction to his own requirements.

3.9 EXPERIENCE, ENGAGEMENT AND FUN

3.9.1 Understanding experience

Shopping is an interesting example to consider. Most internet stores allow you to buy things, but do you go shopping? Shopping is as much about going to the shops, feeling the clothes, being with friends. You can go shopping and never intend to spend money. Shopping is not about an efficient financial transaction, it is an experience. But experience is a difficult thing to pin down; we understand the idea of a good experience, but how do we define it and even more difficult how do we designit?

This sense of flow occurs when there is a balance between anxiety and boredom. If you do something that you know you can do it is not engaging; you may do it automatically while thinking of something else, or you may simply become bored. Alternatively, if you do something completely outside your abilities you may become anxious and, if you are half way up a rock face, afraid. Flow comes when you are teetering at the edge of your abilities, stretching yourself to or a little beyond your limits.

3.9.2 Designing experience

Some of the authors were involved in the design of virtual Christmas crackers. These are rather like electronic greetings cards, but are based on crackers. For those who have not come across them, Christmas crackers are small tubes of paper between 8 and 12 inches long (20–30 cm). Inside there are a small toy, a joke or motto and a paper hat. A small strip of card is threaded through, partly coated with gunpowder. When two people at a party pull the cracker, it bursts apart with a small bang from the gunpowder and the contents spill out

the gunpowder and the contents spill out.

The virtual cracker does not attempt to fully replicate each aspect of the physical characteristics and process of pulling the cracker, but instead seeks to reproduce the experience. To do this the original crackers experience was deconstructed and each aspect of the experience produced in a similar, but sometimes different, way in the new media. Table 3.1 shows the aspects of the experience deconstructed and reconstructed in the virtual cracker. For example, the cracker contents are hidden inside; no one knows what toy or joke will be inside. Similarly, when you create a virtual cracker you normally cannot see the contents until the recipient has opened it. Even the recipient initially sees a page with just an image of the cracker; it is only after the recipient has clicked on the ‘open’ icon that the cracker slowly opens and you get to see the joke, web toy and mask.

The mask is also worth looking at. The first potential design was to have a picture of a face with a hat on it – well, it wouldn’t rank highly on excitement! The essential feature of the paper hat is that you can dress up. An iconic hat hardly does that.

Table 3.1 The crackers experience [101]

	Real cracker	Virtual cracker
Surface elements		
Design	Cheap and cheerful	Simple page/graphics
Play	Plastic toy and joke	Web toy and joke
Dressing up	Paper hat	Mask to cut out
Experienced effects		
Shared	Offered to another	Sent by email, message
Co-experience	Pulled together	Sender can't see content until opened by recipient
Excitement	Cultural connotations	Recruited expectation
Hiddenness	Contents inside	First page – no contents
Suspense	Pulling cracker	Slow . . . page change
Surprise	Bang (when it works)	WAV file (when it works)

3.9.3 Physical design and engagement

Ergonomic You cannot physically push buttons if they are too small or too close.
Physical The size or nature of the device may force certain positions or styles of control,
for example, a dial like the one on the washing machine would not fit on the MiniDisc controller; high-voltage switches cannot be as small as low-voltage ones.

Legal and safety Cooker controls must be far enough from the pans that you do not burn yourself, but also high enough to prevent small children turning them on.

Context and environment The microwave's controls are smooth to make them
easy to clean in the kitchen.
Aesthetic The controls must look good.
Economic It must not cost too much!

These constraints are themselves often contradictory and require trade-offs to be made. For example, even within the safety category front-mounted controls are better in that they can be turned on or off without putting your hands over the pans and hot steam, but back-mounted controls are further from children's grasp. The MiniDisc player is another example; it physically needs to be small, but this means there is not room for all the controls you want given the minimum size that can be manipulated. In the case of the cooker there is no obvious best solution and so different designs favor one or the other. In the case of the MiniDisc player the end
knob is multi-function. This means the knob is ergonomically big enough to turn and physically small enough to fit, but at the cost of a more complex interaction style. To add to this list of constraints there is another that makes a major impact on the ease of use and also the ability of the user to become engaged with the device, for it to become natural to use:

Fluidity The extent to which the physical structure and manipulation of the device
naturally relate to the logical functions it supports.

3.9.4 Managing value

If we want people to *want* to use a device or application we need to understand their personal values. Why should they want to use it? What value do they get from using it? Now when we say value here we don't mean monetary value, although that may be part of the story, but all the things that drive a person. For some people this may include being nice to colleagues, being ecologically friendly, being successful in their career. Whatever their personal values are, if we ask someone to do something or use something g they are only likely to do it if the value to them exceeds the cost.

This is complicated by the fact that for many systems the costs such as purchase cost, download time of a free application, learning effort are incurred up front, whereas often the returns – faster work, enjoyment of use – are seen later. In economics, businesses use a measure called 'net present value' to calculate what a future gain is worth today; because money can be invested, £100 today is worth the same as perhaps £200 in five years' time. Future gain is discounted. For human decision making, future gains are typically discounted very highly; many of us are bad at saving for tomorrow or even keeping the best bits of our dinner until last. This means that
not only must we understand people's value systems, but we must be able to offer

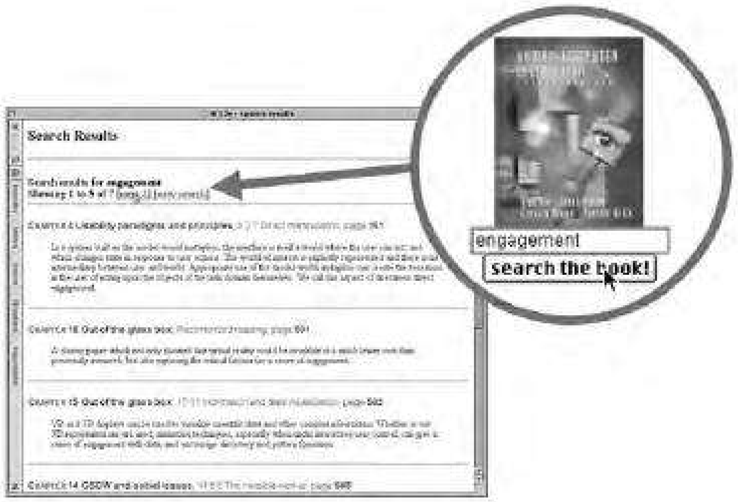


Figure 3.19 The web-based book search facility. Screen shot frame reprinted by permission from Microsoft Corporation

gains sooner as well as later, or at least produce a very good demonstration of potential future gains so that they have a *perceived* current value The latter is partly because we are all academics and researchers in the field and so want to contribute to the HCI community, but also of course we would like lots of people to buy the book. One option we thought of was to put the text online, which would be good for people without the book, but this would have less value to people who have the book (they might even be annoyed that those who hadn't paid should have access). The search mechanism was the result of this process (Figure 3.19). It gives value to those who have the book because it is a way of finding things. It is of value to those who don't because it acts as a sort of online encyclopedia of HCI.

PARADIGMS

The designer of an interactive system, then, is posed with two open questions:

1. How can an interactive system be developed to ensure its usability?
2. How can the usability of an interactive system be demonstrated or measured?

One approach to answering these questions is by means of example, in which successful interactive systems are commonly believed to enhance usability and, therefore, serve as *paradigms* for the development of future products. We believe that we now build interactive systems that are more usable than those built in the past. We also believe that there is considerable room for improvement in designing more usable systems in the future.

. The impact of technology alone, however, is not sufficient to enhance its usability. As our machines have become more powerful, the key to increased usability has come from the creative and considered application of the technology to accommodate and augment the power of the human. Paradigms for interaction have for the most part been dependent upon technological advances and their creative application to enhance interaction. In this chapter, we investigate some of the principal historical advances in interactive designs. What is important to notice here is that the techniques and designs mentioned are recognized as major improvements in interaction, though it is sometimes hard to find a consensus for the reason behind the success. It

is even harder to predict ahead what the new paradigms will be. Often new paradigms have arisen through exploratory desi gns that have then been seen, after the fact, to have created
a new base point for future design. They do not provide mutually exclusive categories, as particular systems will often incorporate ideas from more
than one of the following paradigms

PARADIGMS FOR INTERACTION

4.2.1 Time sharing

By the 1960s it was becoming apparent that the explosion of growth in computing power would be wasted if there was 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 Projects Agency (ARPA). It was Licklider's goal to finance various research centers across the United States in order to encourage new ideas about how best to apply the burgeoning computing technology.

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 (or more accurately, the programmer) 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. Rather than rely o n a model of interaction as a pre-planned activity that resulted in a complete set of instructions being laid out for the computer to follow, truly interactive exchange between programmer and computer was possible. The computer could now project itself as a dedicated partner with each individual user and the increased throughput of information between user and computer allowed the human to become a more reactive and spontaneous collaborator. Indeed, with the advent of time sharing, real human–computer interaction was now possible

4.2.2 Video display units

Sketchpad allowed a computer operator to use the computer to create, very rapidly,
sophisticated visual models on a display screen that resembled a television set. The visual patterns could be stored in the computer's memory like any other data, and could be manipulated by the computer's processor. . . . But Sketchpad was much more than a tool for creating visual displays. It was a kind of simulation language that enabled computers to translate abstractions into perceptually concrete forms. And it was a model for totally new ways of operating computers; by changing something on the display screen, it was possible, via Sketchpad, to change something in the computer's memory.

4.2.3 Programming toolkits

Douglas 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 a purposely complex technology that only the intellectually privileged were capable of manipulating. Engelbart's dedicated research team at the Stanford Research Institute in the 1960s worked towards achieving the manifesto set forth in an article published in 1963 [124]:

4.2.4 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 prog ramming language for children called LOGO. The inventor,

Seymour Papert, wanted to develop a language that was easy for children to use. He and his colleagues from MIT and elsewhere designed a computer-controlled mechanical turtle that dragged a pen along a surface to trace its path. A child could quite easily pretend they were 'inside' the turtle and direct it to trace out simple geometric shapes, such as a square or a circle. By typing in English phrases, such as Go forward or Turn left, the child/programmer could teach the turtle to draw more and more complica ted figures. By adapting the graphical programming language to a model which children could understand and use, Papert demonstrated a valuable maxim for interactive system development – no matter how powerful a system may be, it will always be more powerful if it is easier to use.

4.2.6 The metaphor

In developing the LOGO language to teach children, Papert used the metaphor of a turtle dragging its tail in the dirt. Children could quickly identify with the real-world phenomenon and that instant familiarity gave them an understanding of how they could create pictures. It is no surprise that this general teaching mechanism has been successful in introducing computer novices to relatively foreign interaction techniques. We have already seen how metaphors are used to describe the functionality of many interaction widgets, such as windows, menus, buttons and palettes. Tremendous commercial successes in computing have arisen directly from a judicious choice of metaphor. The Xerox Alto and Star were the first workstations based on the metaphor of the office desktop. The majority of the management tasks on a standard

4.2.7 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 dialog remained largely unidirectional – from user command to system command line prompt – computing was going to stay within the minority population of the hackers who revelled 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 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 [320, 321] 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:
n visibility of the objects of interest
n incremental action at the interface with rapid feedback on all actions
n reversibility of all actions, so that users are encouraged to explore without severe
penalties
n syntactic correctness of all actions, so that every user action is a legal operation
n replacement of complex command languages with actions to manipulate directly
the visible objects (and, hence, the name direct manipulation).

4.2.8 Language versus action
Whereas it is true that direct manipulation interfaces make some tasks easier to perform
correctly, it is equally true that some tasks are more difficult, if not impossible.
Contrary to popular wisdom, it is not generally true that actions speak louder than
words. The image we projected for direct manipulation was of the interface as a
replacement for the underlying system as the world of interest to the user. Actions
performed at the interface replace any need to understand their meaning at any
deeper, system level. Another image is of the interface as the interlocutor or mediator
between the user and the system. The user gives the interface instructions and it is then the responsibility of the interface to see that those instructions are carried out. The user–system communication is by means of indirect language instead of direct actions.

We can attach two meaningful interpretations to this language paradigm. The first requires that the user understands how the underlying system functions and the interface as interlocutor need not perform much translation. In fact, this interpretation of the language paradigm is similar to the kind of interaction which existed before direct manipulation interfaces were around. In a way, we have come full circle!

The second interpretation does not require the user to understand the underlying system's structure. The interface serves a more active role, as it must interpret between the intended operation as requested by the user and the possible system operations that must be invoked to satisfy that intent. Because it is more active, some people refer to the interface as an *agent* in these circumstances. We can see this kind of language paradigm at work in an information retrieval system. If you had a question about speed limits on various roads, how would you ask? The answer in this case is that you would ask the question in whatever way it comes to mind, typing in a question such as, 'What are the speed limits on different roads?' You then leave it
up to the interface agent to reinterpret your request as a legal query to the highway
code database.

4.2.9 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.

The memex was essentially a desk with the ability to produce and store a massive quantity of photographic copies of documented information. In addition to its huge storage capacity, the memex could keep track of links between parts of different documents. In this way, the stored information would resemble a vast interconnected mesh of data, similar to how many perceive information is stored in the human brain. In the context of scientific literature, where it is often very difficult to keep track of the origins and interrelations of the ever-growing body of research, a device which explicitly stored that information would be an invaluable asset.

4.2.10 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.

1. 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-modal, for humans have always used their visual and haptic (touch) 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 a greater extent on simultaneous use of multiple communication channels for both input and output. Humans quite naturally process information by simultaneous use of different channels. We point to someone and refer to them as ‘you’, and it is only by interpreting the simultaneous use of voice and touch that our directions are easily articulated and understood.

Designers have wanted to mimic this flexibility in both articulation and observation by extending the input and output expressions an interactive system will support. So, for example, we can modify a gesture made with a pointing device by speaking, indicating what operation is to be performed on the selected object.

4.2.11 Computer-supported cooperative work

The main distinction between CSCW systems and interactive systems designed for a single user is that designers 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.

A fine example of a CSCW system is electronic mail – *email* – yet another metaphor by which individuals at physically separate locations can communicate via electronic messages which work in a similar way to conventional postal systems. One user can compose a message and ‘post’ it to another user (specified by his electronic mail address). When the message arrives at the remote user’s site, he is informed that a new message has arrived in his ‘mailbox’. He can then read the message and respond as desired. Although email is modeled after conventional postal systems, its major advantage is that it is often much faster than the traditional system (jokingly referred to by email devotees as ‘snail mail’). Communication turnarounds between sites across the world are in the order of minutes, as opposed to weeks.

Electronic mail is an instance of an asynchronous CSCW system because the participants in the electronic exchange do not have to be working at the same time in order for the mail to be delivered. The reason we use email is precisely because of its asynchronous characteristics. All we need to know is that the recipient will eventually receive the message. In contrast, it might be desirable for synchronous communication, which would require the simultaneous participation of sender and recipient, as in a phone conversation.

4.2.12 The world wide web

Probably the most significant recent development in 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 (see Section 2.9) is simply a collection of computers, each linked
by any sort of data connection, 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 (*TCP/IP*) and addressing systems (*IP* addresses and *domain names*). 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 (http), a standard markup notation (such as HTML) for laying out pages of information and a global naming scheme (uniform resource locators or URLs). 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.

4.2.13 Agent-based interfaces

In the human world agents are people who work on someone’s behalf: estate agents buy and sell property for their customers, literary agents find publishers for authors, travel agents book hotels and air tickets for tourists and secret agents obtain information (secretly) for their governments. Software agents likewise act on behalf of users within the electronic world. Examples include email agents which filter your mail for you and web crawlers which search the world wide web for documents you might find interesting. Agents can perform repetitive tasks, watch and respond to events when the user is not present and even learn from the user’s own actions. Some agents simply do what they are told. For example, many email systems allow you to specify filters, simple *if then* rules, which determine the action to perform on certain kinds of mail message:

If Sender: is bank manager
Then Urgency: is high

A major problem with such agents is developing a suitable language between human and agent which allows the user to express intentions. This is especially important when the agent is going to act in the user's absence. In this case, the user may not receive feedback of any mistake until long after the effects have become irreversible; hence the instructions have to be correct, and believed to be correct.

Other agents use artificial intelligence techniques to learn based on the user's actions. An early example of this was Eager [83]. Eager watches users while they work on simple HyperCard applications. When it notices that the user is repeating similar actions a small icon appears (a smiling cat!), suggesting the next action. The user is free either to accept the suggestion or to ignore it. When the user is satisfied that Eager knows what it is doing, it can be instructed to perform all the remaining actions in a sequence.

Eager is also an example of an agent, which has a clear *embodiment*, that is, there is a representation of Eager (the cat icon) in the interface. In contrast, consider Microsoft Excel which incorporates some intelligence in its sum ( ) function. If the current cell is directly below a column of numbers, or if there is a series of numbers to the left of the current cell, the sum range defaults to be the appropriate cells. It is also clever about columns of numbers with subtotals so that they are not included twice in the overall total. As around 80% of all spreadsheet formulae are simple sums
this is a very useful feature. However, the intelligence in this is not embodied, it
is diffuse, somewhere in 'the system'.

4.2.14 Ubiquitous computing
Where does computing happen, and more importantly, where do we as users go to interact with a computer? The past 50 years of interactive computing show that we still think of computers as being confined to a box on a desk or in an office or lab. The actual form of the physical interface has been transformed from a noisy teletype terminal to a large, graphical display with a WIMP or natural language interface, but in all cases the user knows where the computer is and must walk over to it to begin interacting with it.

4.2.15 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 that 'permeate our physical environment so much that we do not notice the computers anymore', and the term ubiquitous computing encompasses a wide range from mobile devices to more pervasive environments.

UNIT II DESIGN & SOFTWARE PROCESS 9

Interactive Design basics – process – scenarios – navigation – screen design – Iteration and prototyping. HCI in software process – software life cycle – usability engineering – Prototyping in practice – design rationale. Design rules – principles, standards, guidelines, rules. Evaluation Techniques – Universal Design.

WHAT IS DESIGN?

A simple definition is:
achieving goals within constraints This does not capture everything about design, but helps to focus us on certain things:
Goals What is the purpose of the design we are intending to produce? Who is it for? Why do they want it? For example, if we are designing a wireless personal movie player, we may think about young affluent users wanting to watch the latest movies whilst on the move and download free copies, and perhaps wanting to share the experience with a few friends.

Constraints What materials must we use? What standards must we adopt? How much can it cost? How much time do we have to develop it? Are there health and safety issues? In the case of the personal movie player: does it have to withstand rain? Must we use existing video standards to download movies? Do we need to build in copyright protection?

5.2.1The golden rule of design

The designs we produce may be different, but often the raw materials are the same. This leads us to the *golden rule of design*:

Understand your materials

understand *computers*
– limitations, capacities, tools, platforms
understand *people*
– psychological, social aspects, human error.

5.3 THE PROCESS OF DESIGN

Here we'll take a simplified view of four main phases plus and iteration loop, focussed on the design of interaction (Figure 5.1).

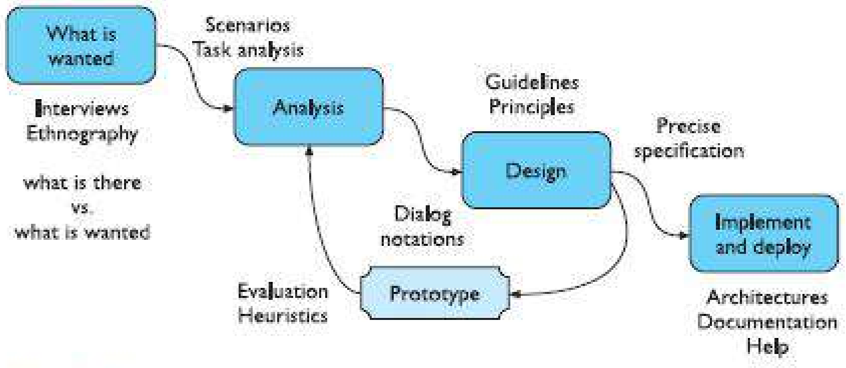


Figure 5.1 Interaction design process

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. For example, how do people currently watch movies? What sort of personal appliances do they currently use?

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. These techniques can be used both to represent the situation as it is and also the desired situation.

Design It is at this stage also where input from theoretical work is most helpful, including cognitive models, organizational issues and understanding communication.

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. Some forms of evaluation can be done using the design on paper, but it is hard to get real feedback without trying it out.

Implementation and deployment Finally, when we are happy with our design, 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.

5.5 SCENARIOS

Scenarios are stories for design: rich stories of interaction. They are perhaps the simplest design representation, but one of the most flexible and powerful. Some scenarios are quite short: 'the user intends to press the "save" button, but accidentally presses the "quit" button so loses his work'. Others are focussed more on describing the situation or context.

Communicate with others – other designers, clients or users. It is easy to misunderstand each other whilst 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

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.

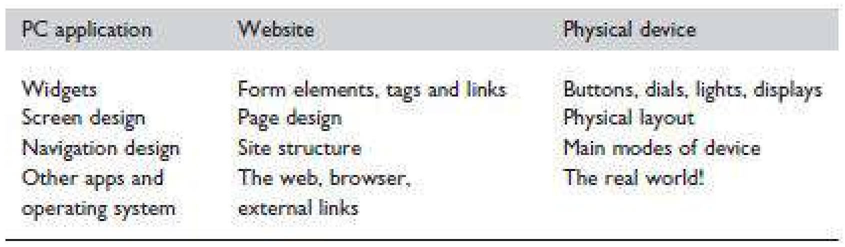
But no alternatives Real interactions have choices, some made by people, some by systems. A simple scenario does not show these alternative paths.

5.6 NAVIGATION DESIGN

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.

Table 5.1 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!

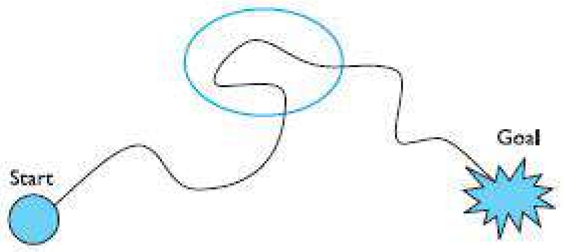
Navigation within the application You need to be able to understand what will happen when a button is pressed, to understand where you are in the interaction.

Environment The word processor has to read documents from disk, perhaps some are on remote networks. You swap between applications, perhaps cut and paste. 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 (although we may treat them differently).

Individual screens or the layout of devices will have their own structure, but this is for the next section. Here we will consider two main kinds of issue:
local structure
– looking from one screen or page out
global structure
– structure of site, movement between screens.

5.6.1 Local structure

However, in a world of partial knowledge users meander 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 (often partially formed) 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.

5.6.2 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.

For example, Figure 5.6 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.

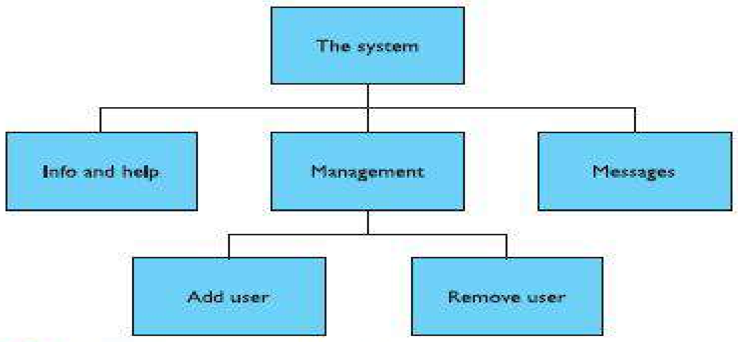


Figure 5.6 Application functional hierarchy

One of the difficulties with organizing information or system functionality is that different people have different internal structures for their knowledge, and may use different vocabulary. This is one of the places where a detailed knowledge of the intended users is essential: it is no good creating a hierarchy that the designers understand, but not the users . . . and all too commonly this is exactly what happens.

5.6.3 Global structure – dialog

Figure 5.7 shows a network diagram illustrating the main screens for adding or deleting a user from the messaging system in Figure 5.6. 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

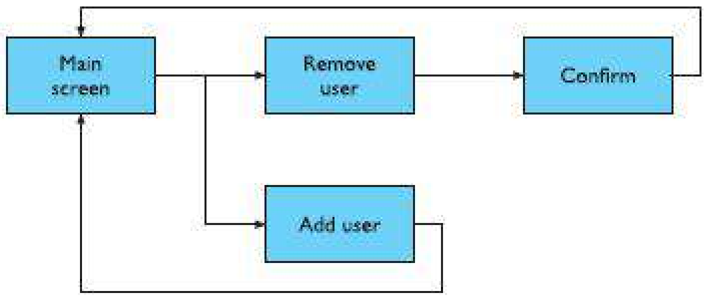


Figure 5.7 Network of screens/states

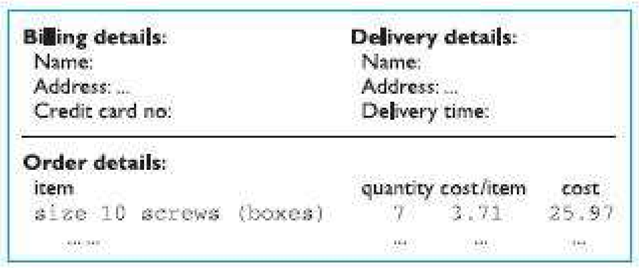
5.7 SCREEN DESIGN AND LAYOUT

The basic principles at the screen level reflect those in other areas of interaction design:

Ask What is the user doing?
Think What information is required? What comparisons may the user need to make? In what order are things likely to be needed?
Design Form follows function: let the required interactions drive the layout.

5.7.1 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.



Billing details:
Name:
Address: ...
Credit card no:

Delivery details:
Name:
Address: ...
Delivery time:

Order details:

item	quantity	cost/item	cost
size 10 screws (boxes)	7	3.71	25.97
... ...	...	...	...

Figure 5.8 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 5.8 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
If we look at Figure 5.8 again we can see that 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. Is this the right order? In general we need to think: what is the natural order for the user? 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.

Alignment
Alignment of lists is also very important. For users who read text from left to right, lists of text items should normally be aligned to the left. Numbers, however, should

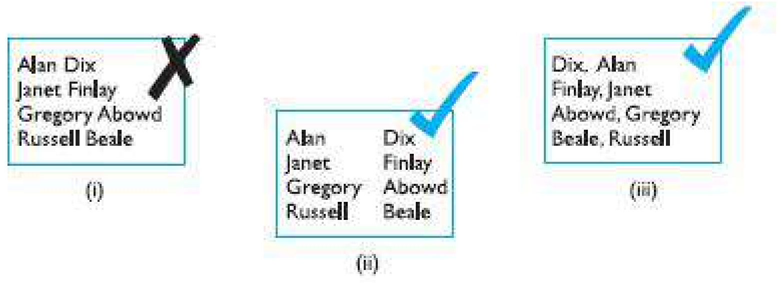
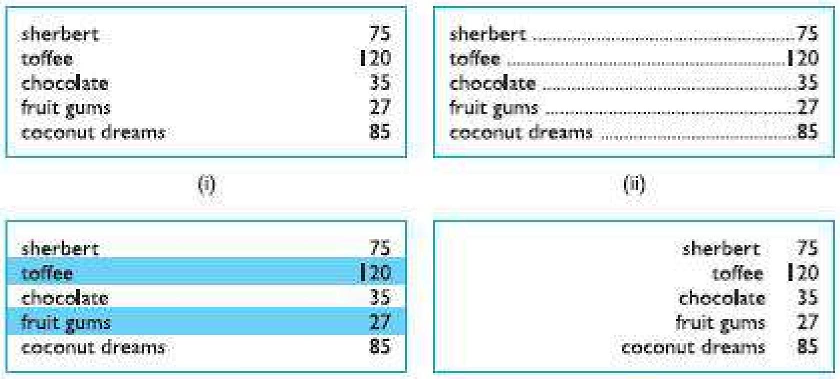


Figure 5.10 Looking up surnames

normally be aligned to the right (for integers) or at the decimal point. 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 5.11 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 grays or colors behind rows or columns. This is also a time when it may be worth breaking other



(i)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(ii)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(iii)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(iv)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

Figure 5.11 Managing multiple columns

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 (which anyway are not sorted in this figure

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. Often the 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.

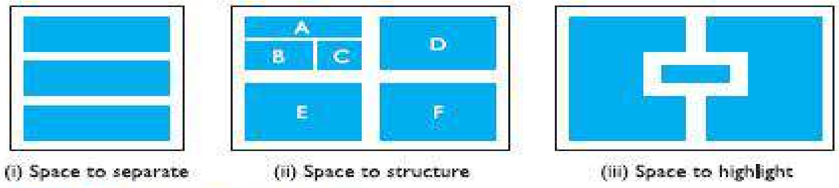


Figure 5.12 Using white space in layout

5.7.2 User action and control

Entering information

Some of the most complicated and difficult screen layouts are found in forms-based interfaces and dialog boxes. In each case the screen consists not only of information pr esented to the user, but also of places for the user to enter information or select options. Actually, many of the same layout issues for data presentation also apply to fields for data entry. Alignment is still important. It is especially common to see the text entry boxes aligned in a jagged fashion because the field names are of different lengths. This is an occasion 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.

.

Knowing what to do

Some elements of a screen are passive, simply giving you information; others are active, expecting you to fill them in, or do something to them. This is one of the reasons for platform and company style guides. If everyone designs buttons to look the same and menus to look the same, then users will be able to recognize them when they see them. However, this is not sufficient in itself. It is important that the labels and icons on menus are also clear. Again, standards can help for common actions such as save, delete or print. For more system-specific actions, one needs to follow broader principles. For example, a button says 'bold': does this represent the current *state* of a system or the *action* that will be performed if the button is pressed?

5.7.3 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 5.13. 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.

Aesthetics and utility

Remember that a pretty interface is not necessarily a good interface. Ideally, as with any well-designed item, an interface should be aesthetically pleasing. Indeed, good graphic design and attractive displays can increase users' satisfaction and thus improve productivity. However, beauty and utility may sometimes be at odds.

Making a mess of it: color and 3D

One of the worst features in many interfaces is their appalling use of color. This is partly because many monitors only support a limited range of primary colors and partly because, as with the overuse of different fonts in word processors, the designer got carried away.

Localization / internationalization
If you are working in a different country, you might see a document being word processed where the text of the document and the file names are in the local language, but all the menus and instructions are still in English. The process of making software suitable for different languages and cultures is called *localization* or *internationalization*.

A different resource database is constructed for each language, and so the program can be customized to use in a particular country by simply choosing the appropriate resource database. However, changing the language is only the simplest part of internationalization.

5.8 ITERATION AND PROTOTYPING

Prototyping is an example of what is known as a *hill-climbing* approach. Imagine you are standing somewhere in the open countryside. You walk uphill and keep going uphill as steeply as possible. Eventually you will find yourself at a hill top. This

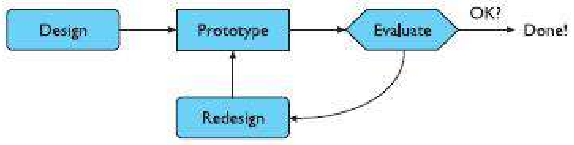


Figure 5.14 Role of prototyping

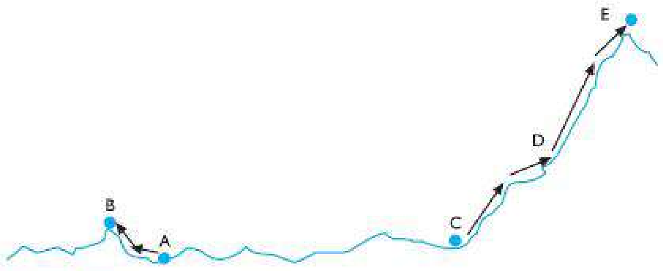


Figure 5.15 Moving little by little . . . but to where?

is exactly how iterative prototyping works: you start somewhere, evaluate it to see how to make it better, change it to make it better and then keep on doing this until it can't get any better.

From this we can see that there are two things you need in order for prototyping
methods to work:
1. To understand what is wrong and how to improve.
2. A good start point.
The first is obvious; you cannot iterate the design unless you know what must be
done to improve it. The second, however, is needed to avoid local maxima.

HCI IN THE SOFTWARE PROCESS

6.2 THE SOFTWARE LIFE CYCLE

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.

6.2.1 Activities in the life cycle

A more detailed description of the life cycle activities is depicted in Figure 6.1. The graphical representation is reminiscent of a waterfall, in which each activity naturally leads into the next.

Requirements specification

In requirements specification, the designer and customer try to capture a description of *what* the eventual system will be expected to provide. Requirements specification involves eliciting information from the customer about the work environment, or domain, in which the final product will function. Though the requirements are from the customer's perspective, if they are to be met by the

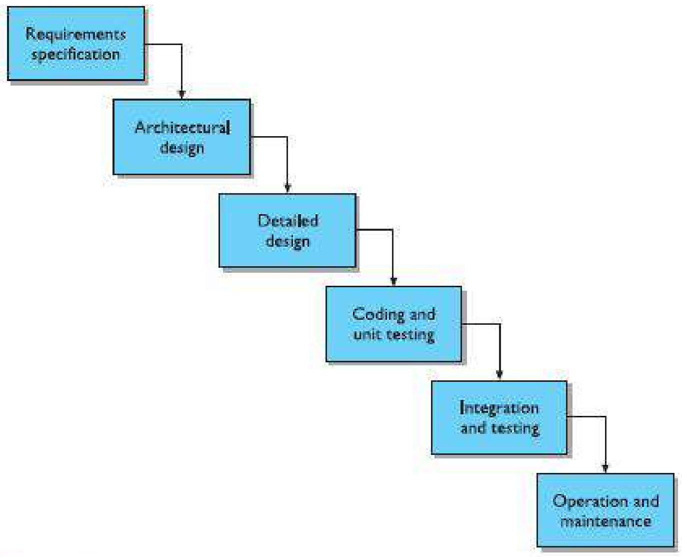


Figure 6.1 The activities in the waterfall model of the software life cycle

software product they must be formulated in a language suitable for implementation. Requirements are usually initially expressed in the native language of the customer.

Architectural design

As we mentioned, the requirements specification concentrates on what the system is supposed to do. The next activities concentrate on *how* the system provides the services expected from it. The first activity is a high-level decomposition of the system into components that can either be brought in from existing software products or be developed from scratch independently. An architectural design performs this decomposition.

There are many structured techniques that are used to assist a designer in deriving an architectural description from information in the requirements specification (such as CORE, MASCOT and HOOD).

Detailed design
The architectural design provides a decomposition of the system description that allows for isolated development of separate components which will later be integrated. For those components that are not already available for immediate integration, the designer must provide a sufficiently detailed description so that they may be implemented in some programming language.

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, according to some test criteria that were determined in earlier activities.

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 behavior and acceptable use of any shared resources.

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

6.2.2 Validation and verification
Throughout the life cycle, the design must be checked to ensure that it both satisfies the high-level requirements agreed with the customer and is also complete and internally consistent. These checks are referred to as *validation* and *verification*, respectively.

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

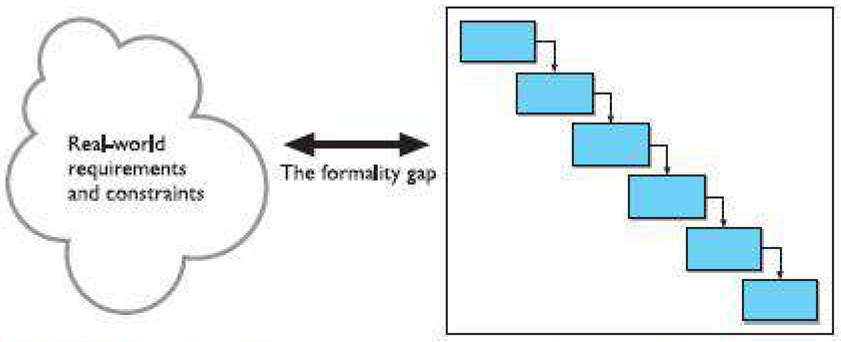


Figure 6.3 The formality gap between the real world and structured design

design notations used.

6.2.3 Management and contractual issues

The life cycle described above concentrated on the more technical features of software development. In a technical discussion, managerial issues of design, such as time constraints and economic forces, are not as important. The different activities of the life cycle are logically related to each other.

6.2.4 Interactive systems and the software life cycle

The life cycle for development we described above 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 activity both before or after it in the life cycle. We can represent this iterative relationship as in Figure 6.4, but that does not greatly enhance any understanding of the design process for interactive systems. You may ask whether it is worth the

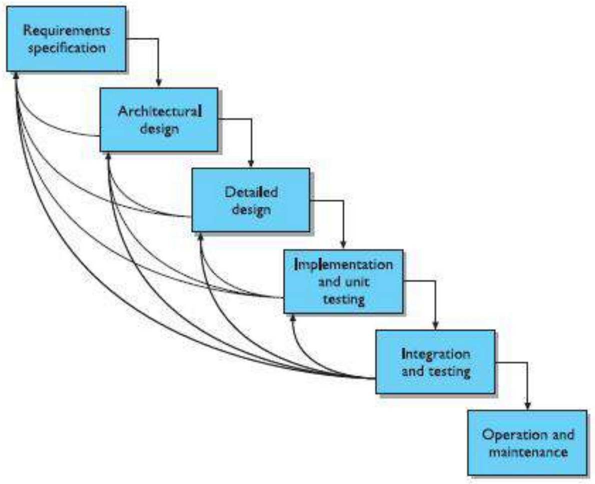

Requirements specification
Architectural design
Detailed design
Implementation and unit testing
Integration and testing
Operation and maintenance

Figure 6.4 Representing iteration in the waterfall model

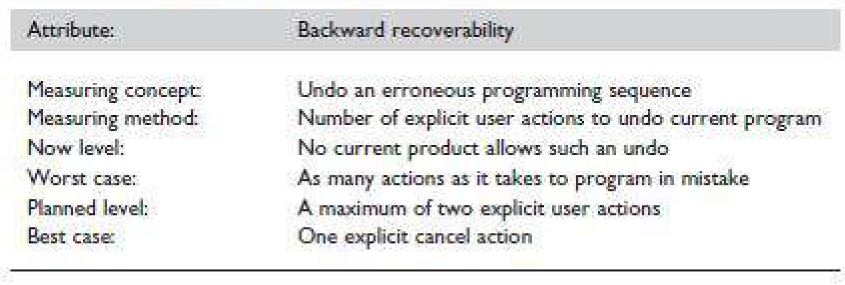
intellectual effort to understand the interactive system design

Figure 6.4 depicts how discovery in later activities can be reflected in iterations back to earlier stages. This is an admission that the requirements capture activity is not executed properly. The more serious claim we are making here is that all of the requirements for an interactive system *cannot* be determined from the start,and there are many convincing arguments to support this position. The result is that systems must be built and the interaction with users observed and evaluated in order to determine how to make them more usable.

6.3 USABILITY ENGINEERING

The ultimate test of a product's usability is based on measurements of users' experience with it. Therefore, since a user's direct experience with an interactive system is at the physical interface, focus on the actual user interface is understandable.
Various attributes of the system are suggested as gauges for testing the usability. For each attribute, six items are defined to form the usability specification of that attribute.

Table 6.1 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

Tables 6.2 and 6.3, adapted from Whiteside, Bennett and Holtzblatt [377], 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*.

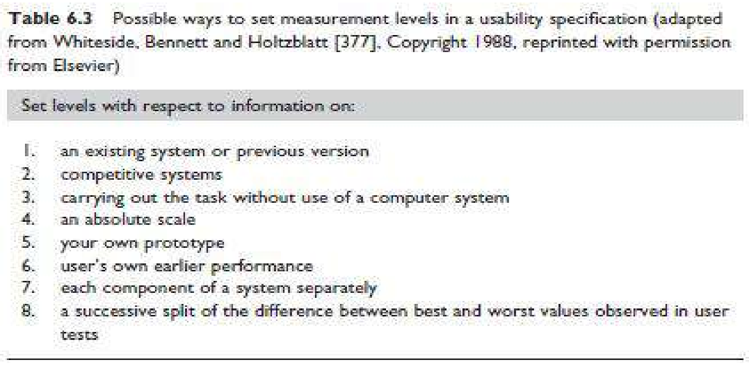


Table 6.3 Possible ways to set measurement levels in a usability specification (adapted from Whiteside, Bennett and Holtzblatt [377]. Copyright 1988, reprinted with permission from Elsevier)

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

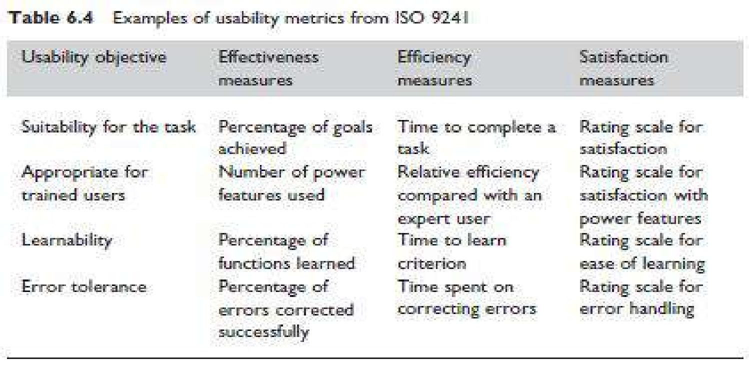


Table 6.4 Examples of usability metrics from ISO 9241

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

6.4 ITERATIVE DESIGN AND PROTOTYPING

On the technical side, iterative design is described by the use of *prototypes*, artifacts that simulate or animate some but not all features of the intended system. 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 6.5 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.

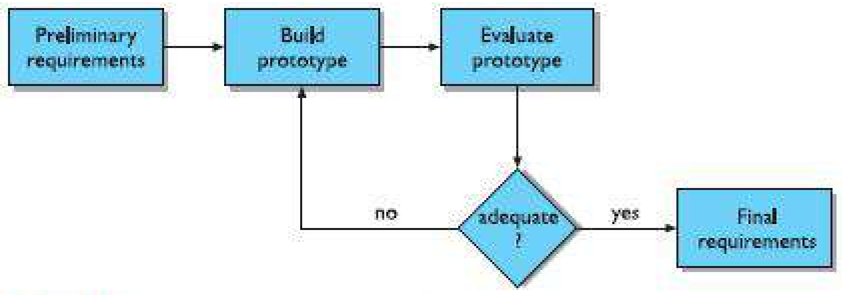


Figure 6.5 Throw-away prototyping within requirements specification

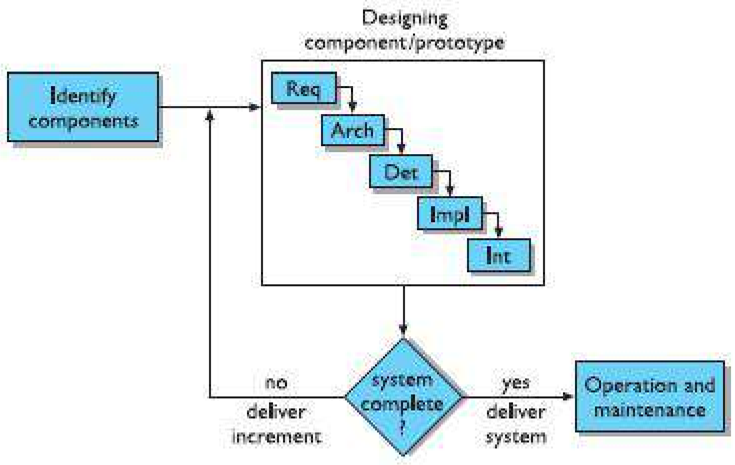


Figure 6.6 Incremental prototyping within the life cycle

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 6.6.

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 6.7. 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.

Prototypes differ according to the amount of functionality and performance they provide relative to the final product. An *animation* of requirements can involve no

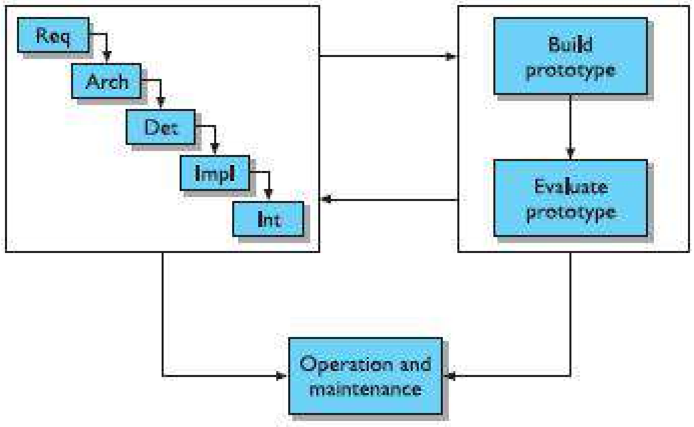


Figure 6.7 Evolutionary prototyping throughout the life cycle

real functionality, or limited functionality to simulate only a small aspect of the interactive behavior for evaluative purposes. At the other extreme, full functionality can be provided at the expense of other performance characteristics, such as speed or error tolerance.

Time Building prototypes takes 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 prototyping 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 Often the most important features of a system will be non-functional ones, 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. Prototypes and other implementations cannot form the basis for a legal contract, and so an iterative design process will still require documentation which serves as the binding agreement.

6.4.1 Techniques for prototyping

Here we will describe some of the techniques that are available for producing rapid prototypes.

Storyboards

Probably the simplest notion of a prototype is the *storyboard*, which is a graphical depiction of the outward appearance of the intended system, without any accompanying system functionality. 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.

Modern graphical drawing packages now make it possible to create storyboards with the aid of a computer instead of by hand.

Limited functionality simulations
Programming support for simulations means a designer can rapidly build graphical and textual interaction objects and attach some behavior 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 was an example of 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.

6.5 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 behavioral description.
It is beneficial to have access to the design rationale for several reasons:

In an explicit form, a design rationale provides a communication mechanism among the members of a design team so that during later stages of design and/or maintenance it is possible to understand what critical decisions were made, what alternatives were investigated (and, possibly, in what order) and the reason why one alternative was chosen over the others.

6.5.1 Process-oriented design rationale
Various *positions* are put forth as potentialresolutions 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.

6.5.2 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, it is not so important that the questions recorded are the actual questions asked during design meetings. Rather, these questions represent an agreed characterization of the

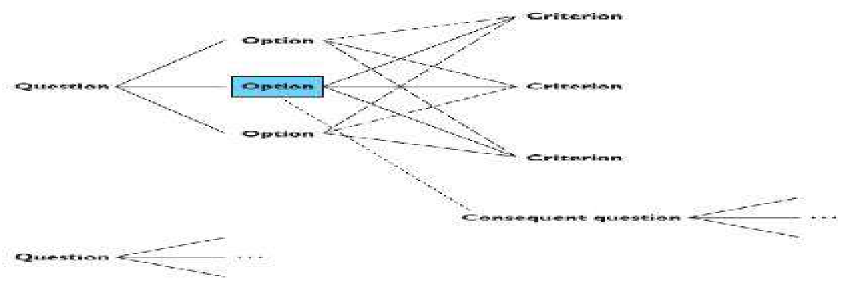


Figure 6.9 The QOC notation

issues raised based on reflection and understanding of the actual design activities. Questions in a design space analysis are therefore similar to issues in IBIS except in the way they are captured. Options provide alternative solutions to the question. They are assessed according to some criteria in order to determine the most favorable option. In Figure 6.9 an option which is favorably assessed in terms of a criterion is linked with a solid line, whereas negative links have a dashed line.

6.5.3 Psychological design rationale

The final category of design rationale tries to make explicit the psychological claims of usability inherent in any interactive system in order better to suit a product for the tasks users have. People use computers to accomplish some tasks in their particular work domain,as we have seen before. 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.

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.

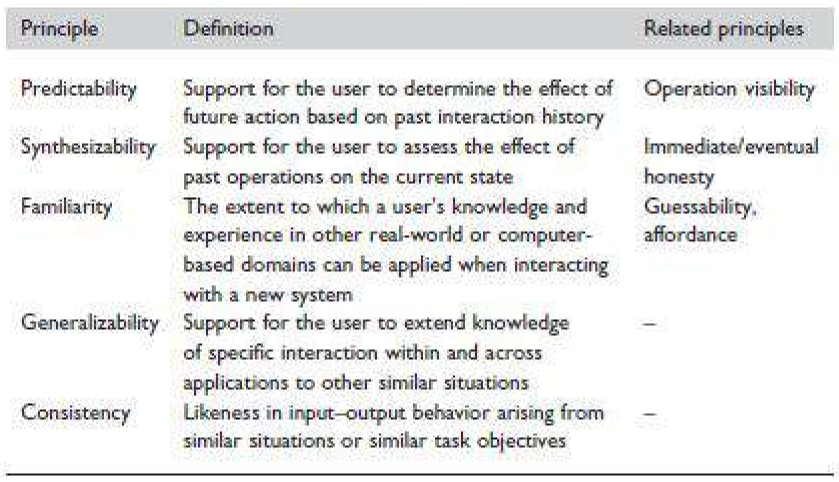
DESIGN RULES

7.2 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. Derivation of principles for interaction has usually arisen out of a need to explain why a paradigm is successful and when it might not be. Principles can provide the repeatability which paradigms in themselves cannot provide.

The principles we present are first divided into three main categories:
Learnability – the ease with which new users can begin effective interaction and achieve maximal performance.
Flexibility – the multiplicity of ways in which the user and system exchange
information.
Robustness – the level of support provided to the user in determining successful achievement and assessment of goals.

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	–

7.2.1 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 contains a summary of the specific principles that support learnability, which we will describe below.

Predictability

Except when interacting with some video games, a user does not take very well to surprises. 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.

Synthesizability

Predictability focusses on the user's ability to determine the effect of future interactions.
In building up some sort of predictive model of the system's behavior, it is important for the user to assess the consequences of previous interactions in order to formulate a model of the behavior of the system. Synthesis, therefore, is the ability of the user to assess the effect of past op erations on the current state.

Familiarity

New users of a system bring with them a wealth of experience across a wide number of application domains. This experience is obtained both through interaction in the real world and through interaction with other computer systems. 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.

Generalizability
The generalizability of an interactive system supports this activity, leading to a more complete predictive model of the system for the user. 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.

Generalization can occur within a single application or across a variety of applications. A good example of generalizability across a variety of applications can be seen in multi-windowing systems that attempt to provide cut/paste/copy operations to all applications in the same way (with varying degrees of success). Generalizability within an application can be maximized by any conscientious designer.

Consistency
Consistency relates to the likeness in behavior arising from similar situations or similar task objectives. Consistency is probably the most widely mentioned principle in the literature on user interface design. 'Be consistent!' we are constantly urged.

7.2.2 Flexibility
Flexibility refers to the multiplicity of ways in which the end-user and the system exchange information.

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. a keyboard shortcut, or saying 'open window'.

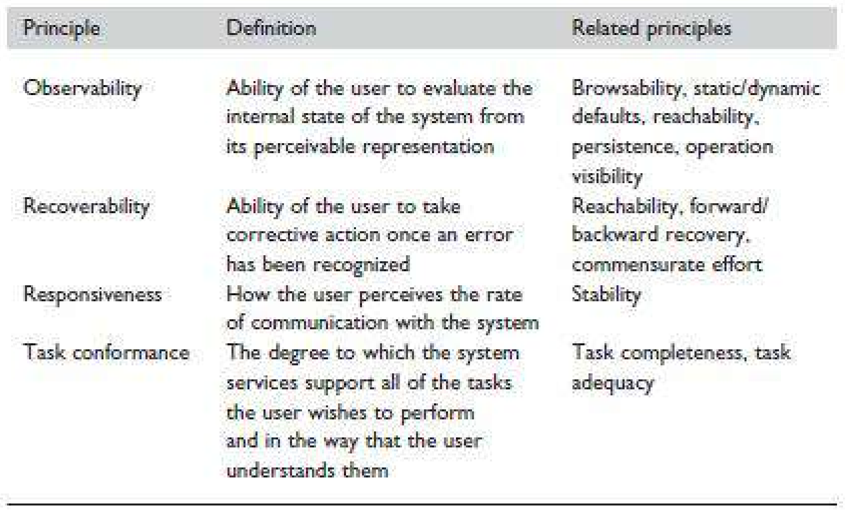
Task migratability
Task migratability concerns the transfer of control for execution of tasks between system and user. It should be possible for the user or system to pass the control of a task over to the other or promote the task from a completely internalized one to a shared and cooperative venture. 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. 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.

Customizability
Customizability is the modifiability of the user interface by the user or the system. From the system side, we are not concerned with modifications that would be attended to by a programmer actually changing the system and its interface during system maintenance. Rather, we are concerned with the automatic modification that the system would make based on its knowledge of the user.

.

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

7.2.3 Robustness
In a work or task domain, a user is engaged with a computer in order to achieve some set of goals. The robustness of that interaction covers features that support the successful achievement and assessment of the goals. Here, we describe principles that support robustness. A summary of these principles is presented in Table 7.3.

Browsability allows the user to explore the current internal state of the system via the limited view provided at the interface. Usually the complexity of the domain does not allow the interface to show all of the relevant domain concepts at once.

Reachability refers to the possibility of navigation through the observable system states.

Recoverability
Users make mistakes from which they want to recover. 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.

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.

Responsiveness
Responsiveness measures the rate of communication between the system and the user. Response time is generally defined as the duration of time needed by the system to express state changes to the user. In general, short durations and instantaneous response times are desirable.

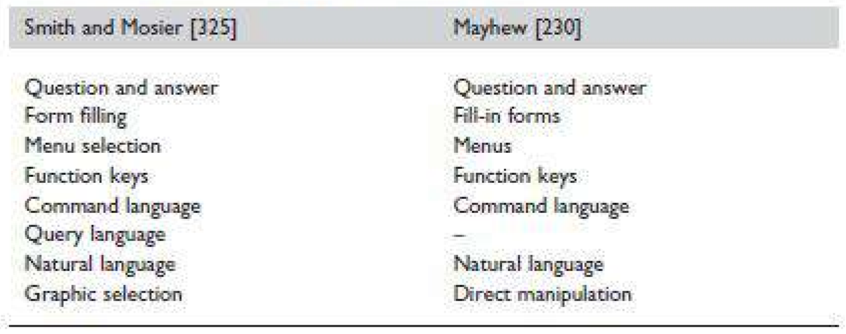
Task conformance
Since the purpose of an interactive system is to allow a user to perform various tasks in achieving certain goals within a specific application domain, we can ask whether the system supports all of the tasks of interest and whether it supports these as the user wants. *Task completeness* addresses the coverage issue and *task adequacy* addresses the user's understanding of the tasks.

7.3 STANDARDS

7.4 GUIDELINES

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

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

7.5 GOLDEN RULES AND HEURISTICS

7.5.1 Shneiderman'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.

1. *Strive for consistency* in action sequences, layout, terminology, command use and so on.
2. *Enable frequent users to use shortcuts*, such as abbreviations, special key sequences and macros, to perform regular, familiar actions more quickly.
3. *Offer informative feedback* for every user action, at a level appropriate to the magnitude of the action.
4. *Design dialogs to yield closure* so that the user knows when they have completed a task.
5. *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.
6. *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.
7. *Support internal locus of control* so that the user is in control of the system, which responds to his actions.
8. *Reduce short-term memory load* by keeping displays simple, consolidating multiple page displays and providing time for learning action sequences.

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

Things, he summarizes user-centered design using the following seven principles:
1. *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.

2. *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.

3. *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.

4. *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.

5. *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. A simple 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.

6. *Design for error*. To err is human, so anticipate the errors the user could make and design recovery into the system.

7. *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

9.1 WHAT IS EVALUATION?

Evaluation should not be thought of as a single phase in the design process (still less as an activity tacked on the end of the process if time permits). Ideally, evaluation should occur throughout the design life cycle, with the results of the evaluation feeding back into modifications to the design. Clearly, it is not usually possible to perform extensive experimental testing continuously throughout the design, but analytic and informal techniques can and should be used.

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, and to identify any specific problems with the system.

9.3 EVALUATION THROUGH EXPERT ANALYSIS

9.3.1 Cognitive walkthrough

Cognitive walkthrough was originally proposed and later revised by Polson and colleagues [294, 376] as an attempt to introduce psychological theory into the informal and subjective walkthrough technique. The origin of the cognitive walkthrough approach to evaluation is the code walkthrough familiar in software engineering. Walkthroughs require a detailed review of a sequence of actions.

To do a walkthrough (the term walkthrough from now on refers to the cognitive walkthrough, and not to any other kind of walkthrough), you need four things:

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

1. **Is the effect of the action the same as the user's goal at that point?** Each user action will have a specific effect within the system. Is this effect the same as what the user is trying to achieve at this point? For example, if the effect of the action is to save a document, is 'saving a document' what the user wants to do?

2. **Will users see that the action is available?** Will users see the button or menu item, for example, that is used to produce the action? This is *not* asking whether they will recognize that

the button is the one they want. This is merely asking whether it is visible to them at the time when they will need to use it. Instances where the answer to this question might be ‘no’ are, for example, where a VCR remote control has a covered panel of buttons or where a menu item is hidden away in a submenu.
3. **Once users have found the correct action, will they know it is the one they need?**
This complements the previous question. It is one thing for a button or menu item to be visible, but will the user recognize that it is the one he is looking for to complete his task? Where the previous question was about the visibility of the action, this one is about whether its meaning and effect is clear.
4. **After the action is taken, will users understand the feedback they get?** If you now assume that the user did manage to achieve the correct action, will he know that he has done so? Will the feedback given be sufficient confirmation of what has actually happened? This is the completion of the execution–evaluation interaction cycle In order to determine if they have accomplished their goal, users need appropriate feedback.

9.3.2 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. Heuristic evaluation can be performed on a design specification so it is useful for evaluating early design. But it can also be used on prototypes, storyboards and fully functioning systems. It is therefore a flexible, relatively cheap approach. Hence it is often considered a *discount usability* technique.

The evaluator also assesses the severity of each usability problem, based on four factors: how common is the problem, how easy is it for the user to overcome, will it be a one-off problem or a persistent one, and how seriously will the problem be perceived? These can be combined into an overall severity 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)
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** Dialogs should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialog competes with the relevant units of information and diminishes their relative visibility.
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, focussed 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.

9.3.3 Model-based evaluation

A third expert-based approach is the use of models. Certain cognitive and design models provide a means of combining design specification and evaluation into the same framework. These are discussed in detail in Chapter 12. 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 provide predictions of the time users will take to perform low-level physical tasks.
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.

9.3.4 Using previous studies in evaluation

Experimental psychology and human–computer interaction between them possess a wealth of experimental results and empirical evidence. Some of this is specific to a particular domain, but much deals with more generic issues and applies in a variety of situations. Examples of such issues are the usability of different menu types, the recall of command names, and the choice of icons.

9.4 EVALUATION THROUGH USER PARTICIPATION

The techniques we have considered so far concentrate on evaluating a design or system through analysis by the designer, or an expert evaluator, rather than testing with actual users. However, useful as these techniques are for filtering and refining the design, they are not a replacement for actual usability testing with the people for whom the system is intended: the users. In this section we will look at a number of different approaches to evaluation through user participation. These include empirical or experimental methods, observational methods, query techniques, and methods that use physiological monitoring, such as eye tracking and measures of heart rate and skin conductance.

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

9.4.1 Styles of evaluation

Before we consider some of the techniques that are available for evaluation with users, we will distinguish between 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 (although the 'lab' may simply be a quiet room). 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 – for example, filing cabinets, wall calendars, books or interruptions – and the unnatural situation may mean that one accurately records a situation that never arises in the real world

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 (though difficult even in the field).

9.4.2 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.
Any experiment has the same basic form. The evaluator chooses a hypothesis to test, which can be determined by measuring some attribute of participant behavior. A number of experimental

conditions are considered which differ only in the values of certain controlled variables. Any changes in the behavioral measures are attributed to the different conditions. Within this basic form there are a number of factors that are important to the overall reliability of the experiment, which must be considered carefully in experimental design. These include the participants chosen, the variables tested and manipulated, and the hypothesis tested.

Participants

The choice of participants is vital to the success of any experiment. In evaluation experiments, participants should be chosen to match the expected user population as closely as possible. Ideally, this will involve experimental testing with the actual users but this is not always 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. Their experience with computers in general, and with systems related to that being tested, should be similar, as should their experience or knowledge of the task domain. It is no good testing an interface designed to be used by the general public on a participant set made up of computer science undergraduates: they are simply not representative of the intended user population.

A second issue relating to the participant set is the sample size chosen. Often this is something that is determined by pragmatic considerations: the availability of participants is limited or resources are scarce. However, the sample size must be large enough to be considered to be representative of the population, taking into account the design of the experiment and the statistical methods chosen.

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).
Independent variables are those elements of the experiment that are manipulated to produce different conditions for comparison. Examples of independent variables in evaluation experiments are interface style, level of help, number of menu items and icon design. Each of these variables can be given a number of different values; each value that is used in an experiment is known as a *level* of the variable. So, for example, an experiment that wants to test whether search speed improves as the number of menu items decreases may consider menus with five, seven, and ten items. Here the independent variable, number of menu items, has three levels.
More complex experiments may have more than one independent variable. For example, in the above experiment, we may suspect that the speed of the user's response depends not only on the number of menu items but also on the choice of commands used on the menu. In this case there are two independent variables.

If there were two sets of command names (that is, two levels), we would require
six experimental conditions to investigate all the possibilities (three levels of menu
size × two levels of command names). Dependent variables, on the other hand, are the variables that can be measured in the experiment, their value is 'dependent' on the changes made to the independent variable. In the example given above, this would be the speed of menu selection.

The dependent variable must be measurable in some way, it must be affected by the independent variable, and, as far as possible, unaffected by other factors. Common choices of dependent variable in evaluation experiments are the time taken to complete a task, the number of errors made, user preference and the quality of the user's performance. Obviously, some of these are easier to measure objectively than others.

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. This is done by disproving the null hypothesis, which states that there is no difference in the dependent variable between the levels of the independent variable. The statistical measures described below produce values that can be compared with various levels of significance. If a result is significant it shows, at the given level of certainty, that the differences measured would not have occurred by chance (that is, that the null hypothesis is incorrect).

Experimental design
In order to produce reliable and generalizable results, an experiment must be carefully designed. We have already looked at a number of the factors that the experimenter must consider in the design, namely the participants, the independent and dependent variables, and the hypothesis. The first phase in experimental design then is to choose the hypothesis: to decide exactly what it is you are trying to demonstrate. In doing this you are likely to clarify the independent and dependent variables, in that you will have identified what you are going to manipulate and what change you expect. If your hypothesis does not clearly identify these variables then you need to rethink it.

The second experimental design is within-subjects (or *repeated measures*). Here each user performs under each different condition. This design can suffer from transfer of learning effects, but this can be lessened if the order in which the conditions are tackled is varied between users, for example, group A do first condition followed by second and group B do second condition followed by first. Within-subjects is less costly than between-subjects, since fewer users are required, and it can be particularly effective where learning is involved. There is also less chance of effects from variation between participants.

Statistical measures
The first two rules of statistical analysis are to *look* at the data and to *save* the data. It is easy to carry out statistical tests blindly when a glance at a graph, histogram or table of results would be more instructive. In particular, looking at the data can expose *outliers*, single data items that are very different from the rest. Outliers are often the result of a transcription error or a freak event not connected to the experiment. For example, we notice that one participant took three times as long as everyone else to do a task. We investigate and discover that the participant had been suffering from flu on the day of the experiment. Clearly, if the participant's data were included it would bias the results.

Saving the data is important, as we may later want to try a different analysis method. It is all too common for an experimenter to take some averages or otherwise tabulate results, and then throw away the original data. At worst, the remaining statistics can be useless for statistical purposes, and, at best, we have lost the ability to trace back odd results to the original data, as, for example, we want to do for outliers.

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. A continuous variable can be rendered discrete by clumping it into classes, for example we could divide heights into short (<5 ft (1.5 m)), medium (5–6 ft (1.5–1.8 m)) and tall (>6 ft (1.8 m)).
In many interface experiments we will be testing one design against another. In these
cases the independent variable is usually discrete. 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

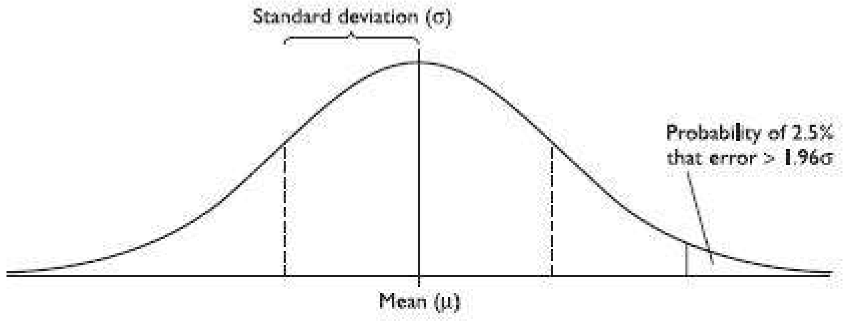
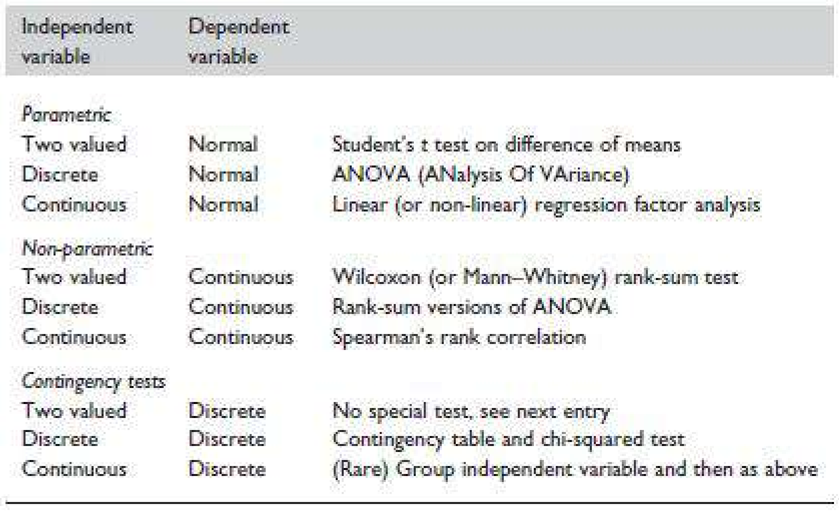


Figure 9.2 Histogram of normally distributed errors

form the well-known bell-shaped graph (Figure 9.2). Happily, many of these tests are fairly *robust*, that is they give reasonable results even when the data are not precisely normal.

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

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

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

Is there a difference? For example, is one system better than another? Techniques that address this are called *hypothesis testing*. The answers to this question are not simply yes/no, but of the form: 'we are 99% certain that selection from menus of five items is faster than that from menus of seven items'.
How big is the difference? For example, 'selection from five items is 260 ms faster than from seven items'. This is called *point estimation*, often obtained by averages.

9.4.3 Observational techniques

A popular way to gather information about actual use of a system is to observe users interacting with it. Usually they are asked to complete a set of predetermined tasks, although, if observation is being carried out in their place of work, they may be observed going about their normal duties. The evaluator watches and records the users' actions (using a variety of techniques – see below). Simple observation isseldom sufficient to determine how well the system meets the users' requirements since it does not always give insight into the their decision processes or attitude.
Consequently users are asked to elaborate their actions by 'thinking aloud'. In this section we consider some of the techniques used to evaluate systems by observing
user behavior.

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. Think aloud has the advantage of simplicity; it requires little expertise to perform (though can be tricky to analyze fully) and can provide useful insight into problems with an interface. It can also be employed to observe how the system is actually used.

It can be used for evaluation throughout the design process, using paper or simulated mock-ups for the earlier stages. However, the information provided is often subjective and may be selective, depending on the tasks provided. The process of observation can alter the way that people perform tasks and so provide a biased view. The very act of describing what you are doing often changes the way you do it – like the joke about the centipede who was asked how he walked . . .
This more relaxed view of the think aloud process has a number of advantages:
n the process is less constrained and therefore easier to learn to use by the evaluator
n the user is encouraged to criticize the system
n the evaluator can clarify points of confusion at the time they occur and so maximize the effectiveness of the approach for identifying problem areas. largely dependent on the effectiveness of the recording method and subsequent analysis. The record of an evaluation session of this type is known as a *protocol*, and there are a number of methods from which to choose.

Protocol analysis
Methods for recording user actions include the following:
Paper and pencil This is primitive, but cheap, and allows the analyst to note interpretations
and extraneous events as they occur. However, it is hard to get detailed information, as it is limited by the analyst's writing speed. Coding schemes for frequent activities, developed during preliminary studies, can improve the rate of recording substantially, but can take some time to develop. A variation of paper and pencil is the use of a notebook computer for direct entry, but then one is limited to the analyst's typing speed, and one loses the flexibility of paper for writing styles, quick diagrams and spatial layout. If this is the only recording facility available then a specific note-taker, separate from the evaluator, is recommended.

Audio recording This is useful if the user is actively 'thinking aloud'. However, it may be difficult to record sufficient information to identify exact actions in later analysis, and it can be difficult to match an audio recording to some other form of protocol (such as a handwritten script).

Video recording This has the advantage that we can see *what* the participant is doing (*as long as* the participant stays within the range of the camera). Choosing suitable camera positions and viewing angles so that you get sufficient detail and yet keep the participant in view is difficult. Alternatively, one has to ask the participant not to move, which may not be appropriate for studying normal behavior! For single-user computer-based tasks, one typically uses two video cameras, one looking at the computer screen and one with a wider focus including
the user's face and hands. The former camera may not be necessary if the computer system is being logged.

Computer logging It is relatively easy to get a system automatically to record user actions at a keystroke level, particularly if this facility has been considered early in the design. It can be more

difficult with proprietary software where source code is not available (although some software now provides built-in logging and playback facilities). Obviously, computer logging only tells us what the user is doing on the system, but this may be sufficient for some purposes. Keystroke data are also 'semantics free' in that they only tell us about the lowest-level actions, not why they were performed or how they are structured (although slight pauses and gaps can give clues).
Technical problems with it are that the sheer volume of data can become unmanageable without automatic analysis, and that one often has to be careful to restore the state of the system (file contents, etc.) before replaying the logs.

User notebooks The participants themselves can be asked to keep logs of activity/ problems. This will obviously be at a very coarse level – at most, records every few minutes and, more likely, hourly or less. It also gives us 'interpreted' records, which have advantages and problems. The technique is especially useful in longitudinal studies, and also where we want a log of unusual or infrequent tasks and problems.

9.4.4 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. They embody the philosophy that states that the best way to find out how a system meets user requirements is to 'ask the user'. They can be used in evaluation and more widely to collect information about user requirements and tasks. The advantage of such methods is that they get the user's viewpoint directly and may reveal issues that have not been considered by the designer. In addition, they are relatively simple and cheap to administer. However, the information gained is necessarily subjective, and may be a 'rationalized' account of events rather than a wholly accurate one. Also, it may be difficult to get accurate feedback about alternative designs if the user has not experienced them, which limits the scope of the information that can be gleaned. However, the methods provide useful supplementary material to other methods. There are two main types of query technique: interviews and questionnaires.

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. Interviews can be effective for high-level evaluation, particularly in eliciting information about user preferences, impressions and attitudes. They may also reveal problems that have not been anticipated by the designer or that have not occurred under observation. When used in conjunction with observation they are a useful means of clarifying an event (compare the post-task walkthrough).

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. However, it can be used to reach a wider participant group, it

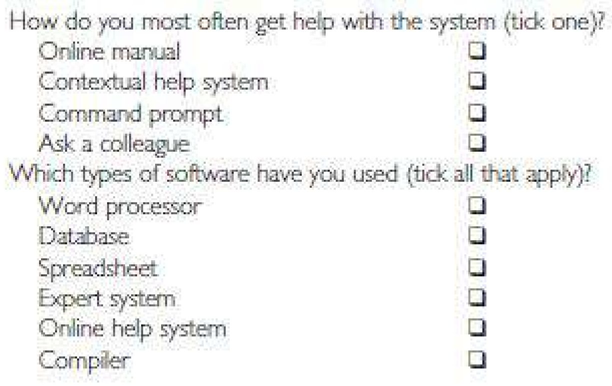
takes less time to administer, and it can be analyzed more rigorously. It can also be administered at various points in the d esign process, including during requirements capture, task analysis and evaluation, in order to get information on the user's needs, preferences and experience. Given that the evaluator is not likely to be directly involved in the completion of the questionnaire, it is vital that it is well designed. The first thing that the evaluator must establish is the purpose of the questionnaire: what information is sought? It is also useful to decide at this stage how the questionnaire responses are to be analyzed. For example, do you want specific, measurable feedback on particular interface features, or do you want the user's impression of using the interface? There are a number of styles of question that can be included in the questionnaire.
These include the following:

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 (see below).

Open-ended These ask the user to provide his own unprompted opinion on a question, for example 'Can you suggest any improvements to the interface?'. They are useful for gathering general subjective information but are difficult to analyze in any rigorous way, or to compare, and can only be viewed as supplementary. They are also most likely to be missed out by time-conscious respondents! However, they may identify errors or make suggestions that have not been considered by the designer. A special case of this type is where the user is asked for
factual information, for example how many commands were used. **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
The granularity of the scale varies: a coarse scale (say, from 1 to 3) gives a clear indication of the meaning of the numbers (disagree, neutral and agree). However, it gives no room for varying levels of agreement, and users may therefore be tempted to give neutral responses to statements that they do not feel strongly about but with which they mildly disagree or agree. A very fine scale (say 1 to 10) suffers from the opposite problem: the numbers become difficult to interpret in a consistent way. One user will undoubtedly interpret the scale differently from another. A middle ground is therefore advisable. Scales of 1 to 5 or 1 to 7 have been used effectively. They are fine enough to allow users to differentiate adequately but still retain clarity in meaning. It can help to provide an indication of the meaning of intermediate scalar values. Odd-numbered scales are used most often but it is possible to use even-numbered scales (e.g. 1–6) if the 'neutral'
option is not wanted. This does not allow for fence sitting – except decisively by selecting 31/2!).

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)?



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

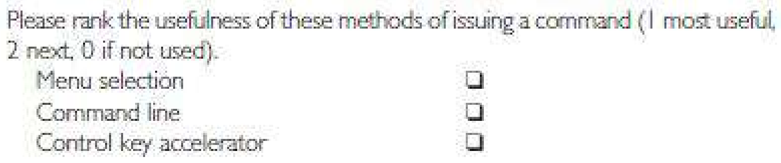
Online manual	☐
Contextual help system	☐
Command prompt	☐
Ask a colleague	☐

Which types of software have you used (tick all that apply)?

Word processor	☐
Database	☐
Spreadsheet	☐
Expert system	☐
Online help system	☐
Compiler	☐

These are particularly useful for gathering information on a user's previous experience. A special case of this type is where the offered choices are 'yes' or 'no'.

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	☐
Command line	☐
Control key accelerator	☐

These question types are all useful for different purposes, as we have noted. However, in order to reduce the burden of effort on the respondent, and so e ncourage a high response rate amongst users, it is best to use closed questions, such as scalar, ranked or multi-choice, as much as possible. These provide the user with alternative responses and so reduce the effort required. They also have the advantage of being easier to analyze. Responses can be analyzed in a number of ways, from determining simple percentages for each response, to looking at correlations and factor analysis. For more detail on available methods the reader is referred to the recommended reading list at the end of the chapter.

9.4.5 Evaluation through monitoring physiological responses

One of the problems with most evaluation techniques is that we are reliant on observation and the users telling us what they are doing and how they are feeling. What if we were able to measure these things directly? Interest has grown recently in the use of what is sometimes 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.

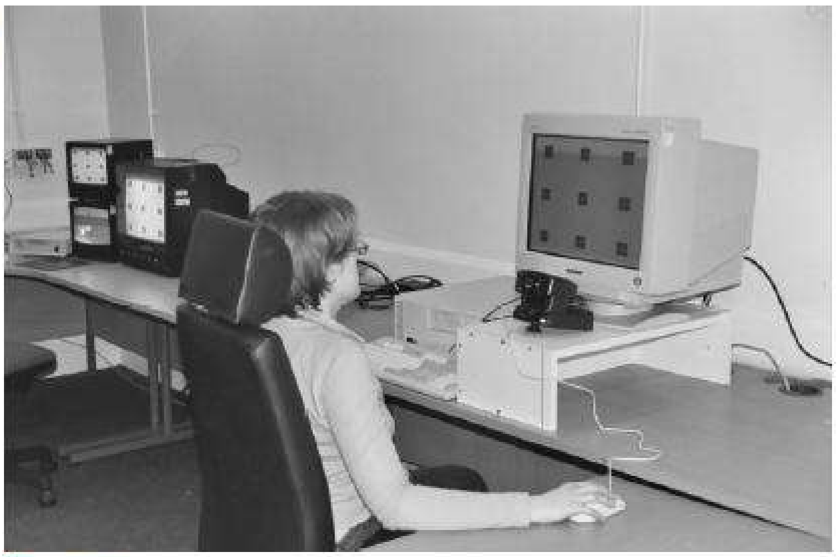


Figure 9.5 Eye-tracking equipment. Source: Courtesy of J. A. Renshaw

Eye tracking for usability evaluation

Eye tracking has been possible for many years, but recent improvements in hardware and software have made it more viable as an approach to measuring usability. The original eye trackers required highly invasive procedures where eye caps were attached to the cornea under anaesthetic. Modern systems vary: some use a head-mounted camera to monitor the eye, but the most sophisticated do not involve any contact between the equipment and the participant, with the camera and light sources mounted in desk units (see Figures 9.5, 9.6) [112]. Furthermore, there have been rapid improvements in the software available both for the control of eye-tracking equipment and the analysis and visualization of the large volumes of data it produces.

Eye movements are believed to reflect the amount of cognitive processing a display requires and, therefore, how easy or difficult it is to process [150]. So measuring not only where people look, but also their patterns of eye movement, may tell us which areas of a screen they are finding easy or difficult to understand. Eye movement measurements are based on fixations, where the eye retains a stable position for a period of time, and saccades, where there is rapid ballistic eye movement from one point of interest to another. 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.

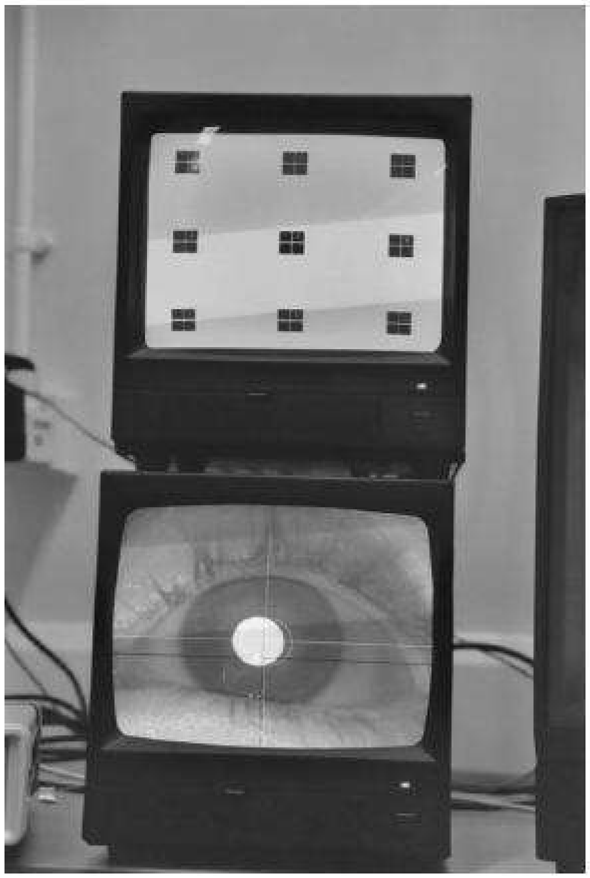


Figure 9.6 Calibrating the eye tracker. Source: Courtesy of J. A. Renshaw

Scan path indicating areas of interest, search strategy and cognitive load. Moving straight to a target with a short fixation at the target is the optimal scan path but plotting scan paths and fixations can indicate what people look at, how often and for how long. Eye tracking for usability is still very new and equipment is prohibitively expensive for everyday use. However, it is a promising technique for providing insights into what really attracts the eye in website design and where problem areas are in system use. More research is needed to interpret accurately the meaning of the various eye movement measurements, as well as to develop more accessible and robust equipment. But, given the potential for gathering new data measurements relatively unobtrusively, it is likely that eye tracking will become part of the standard equipment for usability laboratories in the coming few years.

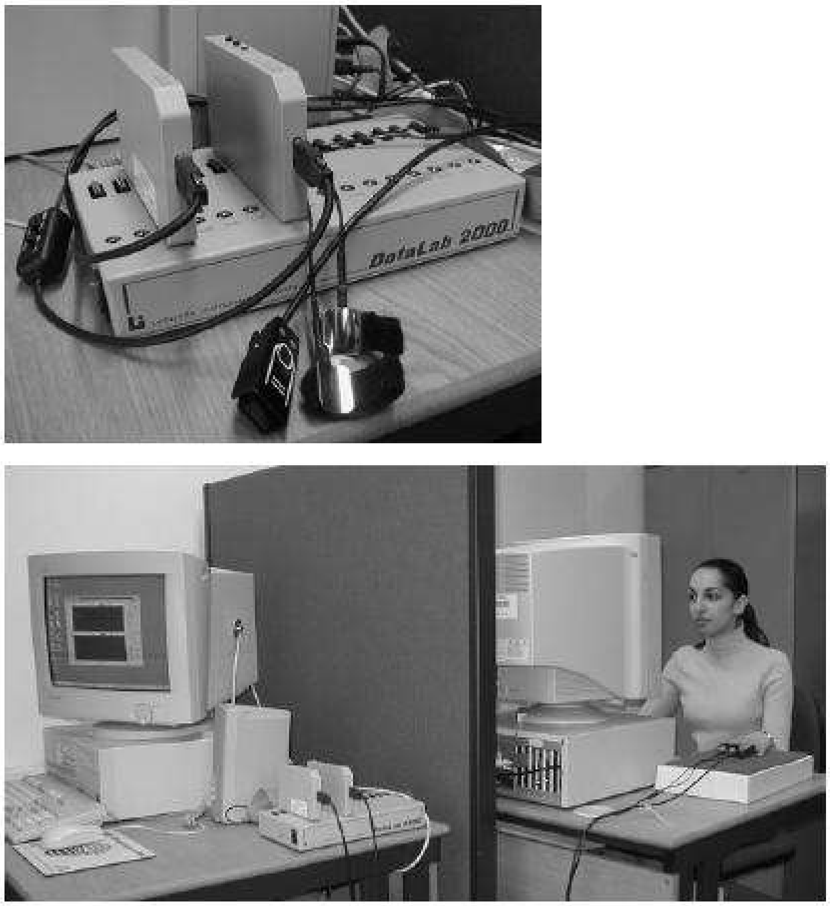


Figure 9.7 Data Lab Psychophysiology equipment showing some of the sensors (above) and a typical experimental arrangement (below) with sensors attached to the participant's fingers and the monitoring software displayed on the evaluator's machine.
Source: Courtesy of Dr R. D. Ward

Physiological measurements

Physiological measurement involves attaching various probes and sensors to the user (see Figure 9.7). These measure a number of factors:

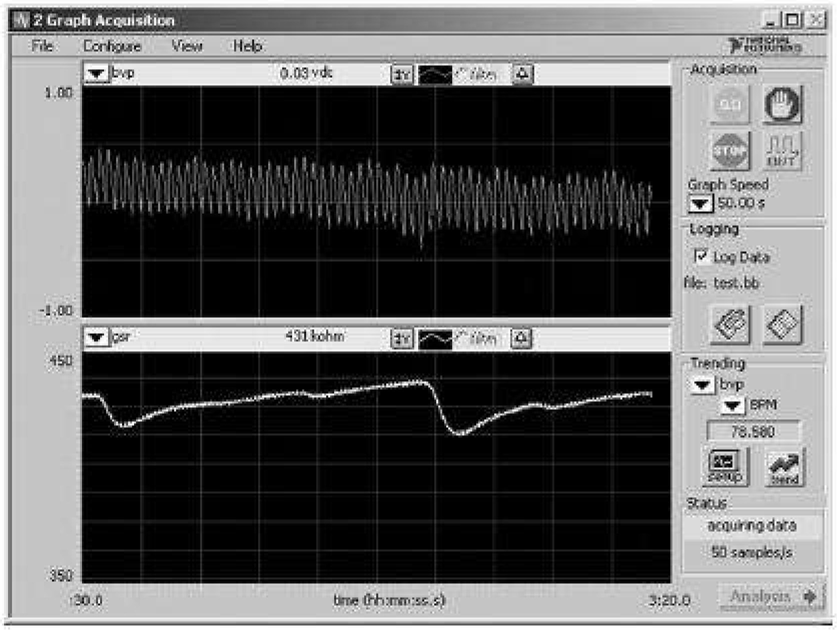


Figure 9.8 Output of monitoring pulse rate (above) and skin conductivity (below). Source: Screen shot courtesy of Dr R. D. Ward; frame source: National Instruments BioBench software

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.

Figure 9.8 illustrates the output obtained from such measurements.

One of the problems with applying these measurements to interaction events is that it is not clear what the relationship between these events and measurements might be. For example, if increased pulse rate is observed during an interactive task, does that indicate frustration with the interface or stress at being unable to complete the task? How will physiological changes differ in response to discrete events or to continuous interface use? Is it possible to map patterns of physiological measurement to specific emotional states?

These are still research questions but the approach is interesting, as it offers a potential means of objectively capturing information about the user's emotional

CHOOSING AN EVALUATION METHOD

9.5.1 Factors distinguishing evaluation techniques

We can identify at least eight factors that distinguish different evaluation techniques and therefore help us to make an appropriate choice. These are:

n the stage in the cycle at which the evaluation is carried out

n the style of evaluation

n the level of subjectivity or objectivity of the technique
n the type of measures provided
n the information provided
n the immediacy of the response
n the level of interference implied
n the resources required.

Design vs. implementation

The first factor to affect our choice of evaluation method is the stage in the design process at which evaluation is required . This may be anything from a paper mock-up to a full implementation, but it is something concrete that can be tested. Evaluation of a design, on the other hand, precedes this stage and seeks instead to provide information to feed the development of the physical artifact. Roughly speaking, evaluation at the design stage needs to be quick and cheap so might involve design experts only and be analytic, whereas evaluation of the implementation needs to be more comprehensive, so brings in users as participants. There are, of course, exceptions to this: participatory design (see Chapter 13) involves users throughout the design process, and techniques such as cognitive walkthrough are expert based and analytic but can be used to evaluate implementations as well as designs.

Early evaluation, whether of a design or an early prototype or mock-up, will bring the greatest pay-off since problems can be easily resolved at this stage. As more commitment is made to a particular design in the implementation, it bec omes increasingly difficult for changes to be made, no matter what the evaluation suggests. Ironically, the most resources are often ploughed into late evaluations. This is less profitable and should be avoided, although obviously some evaluation with users is required with a complete, or almost complete, system, since some elements (such as system performance) will only be evident in the working system.

Laboratory vs. field studies

We have already discussed the pros and cons of these two styles of evaluation. Laboratory studies allow controlled experimentation and observation while losing something of the naturalness of the user's environment. Field studies retain the latter but do not allow control over user activity. Ideally the design process should include both styles of evaluation, probably with laboratory studies dominating the early stages and field studies conducted with the new implementation.

Subjective vs. objective

Evaluation techniques also vary according to their objectivity – some techniques rely heavily on the interpretation of the evaluator, others would provide similar information for anyone correctly carrying out the procedure. The more subjective techniques, such as cognitive walkthrough or think aloud, rely to a large extent on the knowledge and expertise of the evaluator, who must recognize problems and understand what the user is doing. They can be powerful if used correctly and will provide information that may not be available from more objective methods.
However, the problem of evaluator bias should be recognized and avoided. One way to decrease the possibility of bias is to use more than one evaluator. Objective techniques, on the other hand, should produce repeatable results, which are not dependent on the persuasion of the particular

evaluator. Controlled experiments are an example of an objective measure. These avoid bias and provide comparable results but may not reveal the unexpected problem or give detailed feedback on user experience. Ideally, both objective and subjective approaches should be used. The extent to which the results are dependent on the subjective response of the user also varies. At one extreme is asking for the user's opinions, which is highly subjective; at the other is measuring physiological changes in the body, which are outside the user's control.

Qualitative vs. quantitative measures

The type of measurement provided by the evaluation technique is also an important consideration. There are two main types: *quantitative measurement* and *qualitative measurement*. The former is usually numeric and can be easily analyzed using statistical techniques. The latter is non-numeric and is therefore more difficult to analyze, but can provide important detail that cannot be determined from numbers. The type of measure is related to the subjectivity or objectivity of the technique, with subjective techniques tending to provide qualitative measures and objective techniques,quantitative measures. A common example of this is in questionnaires where qualitative information is being sought (for example, user preferences) but a quantitative scale is used. This is also common in experimental design where factors such as the quality of the user's performance are used as dependent variables, and measured on
a quantitative scale.

Information provided

The level of information required from an evaluation may also vary. The information required by an evaluator at any stage of the design process may range from low-level information to enable a design decision to be made (for example, w hich font is most readable) to higher-level information, such as 'Is the system usable?'. Some evaluation techniques, such as controlled experiments, are excellent at providing low-level information – an experiment can be designed to measure a particular aspect of the interface. Higher-level information can be gathered using questionnaire and interview techniques, which provide a more general impression of the user's view of the system.

Immediacy of response

Another factor distinguishing evaluation techniques is the immediacy of the response they provide. Some methods, such as think aloud, record the user's behavior at the time of the interaction itself. Others, such as post-task walkthrough, rely on the user's recollection of events. Such recollection is liable to suffer from bias in recall and reconstruction, with users interpreting events according to their preconceptions

Intrusiveness

Related to the immediacy of the response is the intrusiveness of the technique itself. Certain techniques, particularly those that produce immediate measurements, are obvious to the user during the interaction and therefore run the risk of influencing the way the user behaves. Sensitive activity on the part of the evaluator can help to reduce this but cannot remove it altogether. Most immediate evaluation techniques are intrusive, with the exception of automatic system logging. Unfortunately, this is limited in the information that it can provide and is difficult to interpret.

Resources

The final consideration when selecting an evaluation technique is the availability of resources. Resources to consider include equipment, time, money, participants, expertise of evaluator and context. Some decisions are forced by resource limitations: it is not possible to produce a video protocol without access to a video camera (and probably editing facilities as well). However, other decisions are not so clear cut. For example, time and money may be limited, forcing a choice between two possible evaluations. In these circumstances, the evaluator must decide which evaluation tactic will produce the most effective and useful information for the system under consideration. It may be possible to use results from other people's experiments to avoid having to conduct new experiments.

Some techniques are more reliant on evaluator expertise than others, for example the more formal analytic techniques. If evaluator expertise is limited, it may be more practical to use more simple heuristic methods than methods that require understanding of user goal structures and so on. Finally, the context in which evaluation can occur will influence what can be done. For practical reasons it may not be possible to gain access to the intended users of a system (if it is a general system, for example). Or it may not be feasible to test the system in its intended environment (for example, a system for a space station or a defence system). In these circumstances simulations must be used.

9.5.2 A classification of evaluation techniques

Using the factors discussed in the previous section we can classify the evaluation techniques we have considered in this chapter. This allows us to identify the techniques that most closely fit our requirements. Table 9.4 shows the classification for

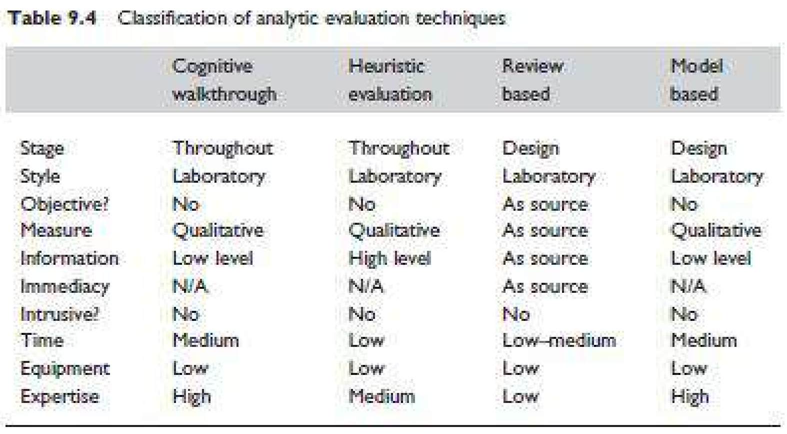


Table 9.4 Classification of analytic evaluation techniques

	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

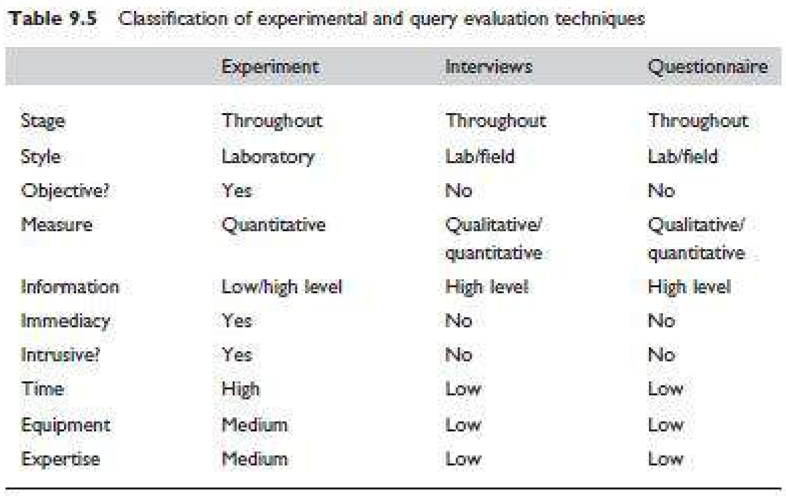


Table 9.5 Classification of experimental and query evaluation techniques

	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

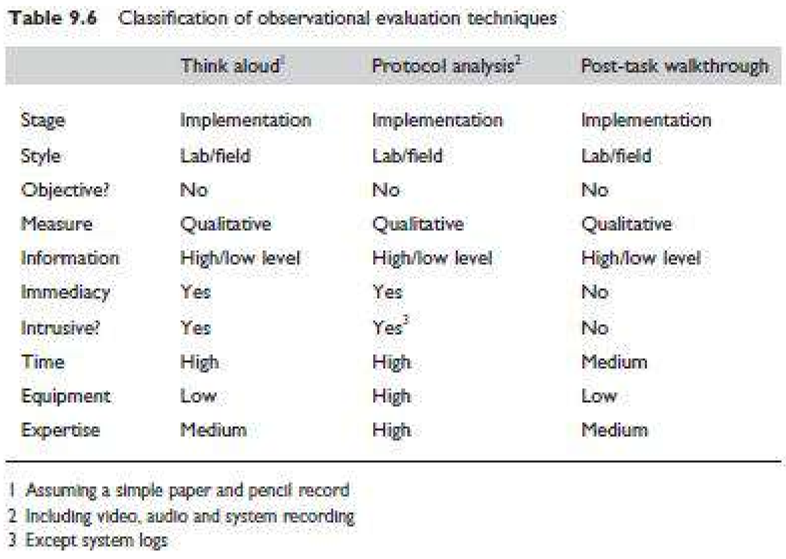


Table 9.6 Classification of observational evaluation techniques

	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

analytic techniques, Table 9.5 for experimental and query techniques, Table 9.6 for observational and Table 9.7 for monitoring techniques. The classification is intended as a rough guide only – some of the techniques do not fit easily into such a classification since their use can vary considerably.

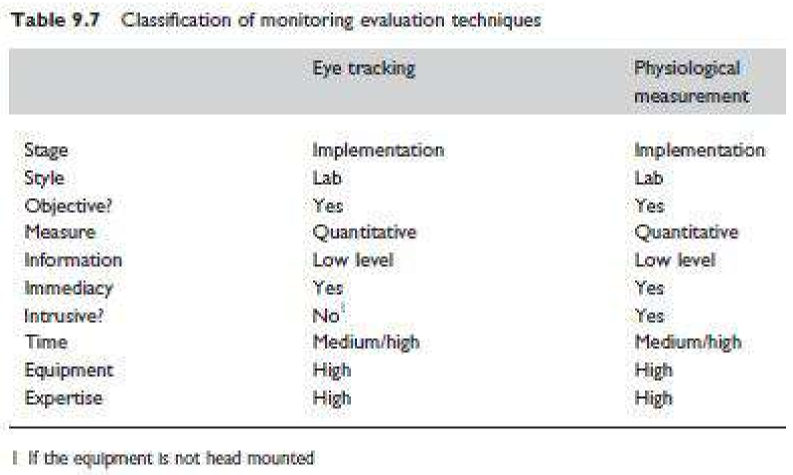


Table 9.7 Classification of monitoring evaluation techniques

	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

10 UNIVERSAL DESIGN

UNIVERSAL DESIGN PRINCIPLES

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. Wherever possible, access should be the same for all; where identical use is not possible, equivalent use should be supported. Where appropriate, security, privacy and safety provision should be available to all.

Principle two is *flexibility in use*: the design allows for a range of ability and preference, through choice of methods of use and adaptivity 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. It should provide prompting and feedback as far as possible.

Principle four is *perceptible information*: the design should provide effective communication of information regardless of the environmental conditions or the user's abilities. Redundancy of presentation is important: information should be represented in different forms or modes (e.g. graphic, verbal, text, touch). Essential information should be emphasized and differentiated clearly from the peripheral content. 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. Important elements should be on the line of sight for both seated and standing users. 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. These seven principles give us a good starting point in considering universal design. They are not all equally applicable to all situations, of course. For example, principles six and seven would be vital in designing an information booth but less important in designing word-processing software. But they provide a useful checklist of considerations for designers, together with guidelines on how each principle can be achieved.

10.3 MULTI-MODAL INTERACTION

As we have seen in the previous section, 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*.

10.3.1 Sound in the interface

Sound is an important contributor to usability. There is experimental evidence to suggest that the addition of audio confirmation of modes, in the form of changes in keyclicks, reduces errors [237]. Video games offer further evidence, since experts tend to score less well when the sound is turned off than when it is on; they pick up vital clues and information from the sound while concentrating their visual attention on different things.

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. However, in spite of this, the auditory channel is comparatively little used in standard interfaces, and where it is used it is often peripheral to the interaction. Information provision is predominantly visual. There is a danger that this will overload the visual channel, demanding that the user attend to too many things at once and select appropriate information from a mass of detail in the display.

Reliance on visual information forces attention to remain focussed on the screen, and the persistence of visual information means that even detail that is quickly out of date may remain on display after it is required, cluttering the screen further. It also presents significant problems for people with visual impairment, whose access to applications can be severely restricted by solely visual output. More widespread effective use of sound in the interface would alleviate these problems. There are two types of sound that we could use: speech and non-speech.

Speech in the interface

Language is rich and complex. We learn speech naturally as children 'by example' – by listening to and mimicking the speech of those around us. This process seems so effortless that we often do not appreciate its complex structures, and it is not until we attempt to learn a new language later in life, or to make explicit the rules of the one we speak, that the difficulties inherent in language understanding become apparent. This complexity makes speech recognition and synthesis by computer very difficult.

Structure of speech If we are fully to appreciate the problems involved with the computer-based recognition and generation of speech, we need first to understand the basic structure of speech. We will use English to illustrate but most other languages have similar issues. Even being able to decompose sentences into their basic parts does not mean that we can then understand them: the syntax (structure) only serves as a standard foundation upon which the semantics (meaning) is based. We are rarely aware of the complex structure of speech, and concentrate on extracting the meaning from the sentences we hear, rather than decomposing the sounds into their constituent parts.

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. The complexity of language is one barrier to success, but there are other, more practical, problems also associated with the automatic recognition of the spoken word. Background noise can interfere with the input, masking or distorting the information, while speakers can introduce redundant or meaningless noises into the information stream by repeating themselves, pausing or using 'continuation' noises such as 'ummm' and 'errr' to fill in gaps in their usual speech.

Variations between individuals also cause problems; people have unique voices, and systems that are successful are tuned to be sensitive to minute variations in tone and frequency of the speaker's voice – new speakers present different inflections to the system, which then fails to perform as well. A more serious problem is caused by regional accents, which vary considerably. This strong variation upsets the trained response of the recognition system. More serious still is the problem posed by different languages: everything from phonemes up can be different.

Speech synthesis Complementary to speech recognition is speech synthesis. The notion of being able to converse naturally with a computer is an appealing one for many users, especially those who do not regard themselves as computer literate, since it reflects their natural, daily medium of expression and communication. However, there are as many problems in speech synthesis as there are in recognition. The most difficult problem is that we are highly sensitive to variations and intonation in speech, and are therefore intolerant of imperfections in synthesized speech. We are so used to hearing natural speech that we find it difficult to adjust to the monotonic,
non-prosodic tones that synthesized speech can produce. In fact, 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. 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.

Non-speech sound

We have considered the use of speech in the interface, but non-speech sounds can offer a number of advantages. As speech is serial, we have to listen to most of a sentence before we understand what is being said. Non-speech sounds can often be assimilated much more quickly. Speech is language dependent – a speech-based system requires translation for it to be used for another language group. The meaning of non-speech sounds can be learned regardless of language. Speech requires the user's attention. Non-speech sound can make use of the phenomenon of auditory adaptation: background sounds are ignored unless they change or cease. However, a
disadvantage is that non-speech sounds have to be learned, whereas the meaning of a spoken message is obvious (at least to a user who is familiar with the language used).

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. Users of early home computers with their noisy power supplies, and computer operators listening to the chatter of the printer and the spinning of disks and tape drives, both report that they are able to tell what stage a process is at by the characteristic sounds that are made.

Non-speech sound can also be used to provide a second representation of actions and objects in the interface to support the visual mode and provide confirmation for the user. It can be used for navigation round a system, either giving redundant supporting information to the sighted user or providing the primary source of information for the visually impaired. Experiments on auditory navigation [290] have demonstrated that auditory clues are adequate for a user to locate up to eight targets on a screen with reasonable speed and accuracy. This suggests that there is little reason for ignoring the role of sound in interfaces on the grounds that it may be too vague or inaccurate. But what kind of non-speech sounds should we use in the interface? There are two
alternatives: using sounds that occur naturally in the world and using more abstract generated sounds. We will consider an example of each type.

Auditory icons *Auditory icons* [141] use natural sounds to represent different types of objects and actions in the interface. The SonicFinder [142] for the Macintosh was developed from these ideas, to enhance the interface through redundancy. Natural sounds are used because people recognize the source of a sound and its behavior rather than timbre and pitch [364]. For example, a noise will be identified as glass breaking or a hollow pipe being tapped. Such recognition is quite sophisticated: we can identify not only the source of a sound (e.g. tapping a pipe) but characteristics of the sound source (e.g. whether the pipe is hollow or solid).

Non-speech sounds such as this can convey a lot of meaning very economically. A file arrives in a mailbox and, being a large file, it makes a weighty sound. If it is a text file it makes a rustling noise, whereas a compiled program may make a metallic clang. The sound can be muffled or clear, indicating whether the mailbox is hidden by other windows or not, while the direction of the sound would indicate the position of the mailbox icon on the screen. If the sound then echoes, as it would in a large, empty room, the system load is low. All this information can be presented in a second or so.

Earcons

An alternative to using natural sounds is to devise synthetic sounds. *Earcons* [36] 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. In this way, earcons can be hierarchically structured to represent menus. Earcons are easily grouped and refined owing to their compositional and hierarchical nature, but they require learning to associate with a specific task in the interface since there is an

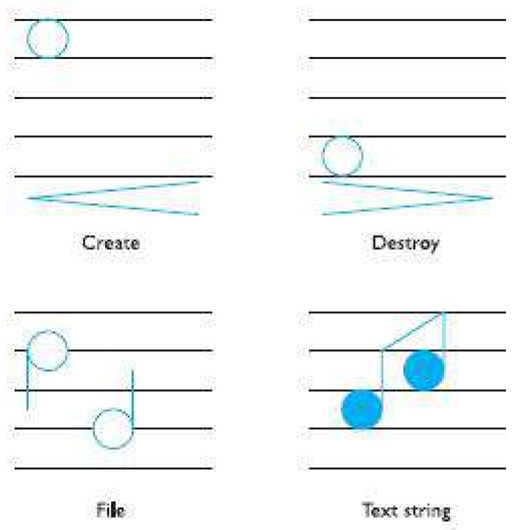


Figure 10.2 Earcons (after Blattner [36], reprinted by permission of Lawrence Erlbaum Associates, Inc.)

arbitrary mapping. Conversely, auditory icons have a semantic relationship with the function that they represent, but can suffer from there being no appropriate sound for some actions.

Earcons provide a structured approach to designing sound for the interface, but can users learn the sounds adequately, and what factors influence their use? Evidence suggests that people can learn to recognize earcons, and that the most important element in distinguishing different sounds is timbre, the characteristic quality of the sound produced by different instruments and voices [47]. Other factors such as pitch, rhythm and register should be used to supplement timbre in creating distinctive sets of musical earcons. Interestingly, the user's musical ability appears to have little effect on his ability to remember earcons: users were able to identify around 80% of earcons from hierarchically ordered sets of 30 or more, regardless of their musical background

[45]. It is also possible to create compound earcons by playing sounds in parallel as well as serially. This obviously reduces the time taken to hear the sound but does not affect the user's accuracy [45].

10.3.2 Touch in the interface

We have already considered the importance of touch in our interaction with our environment, in Chapter 1. Touch is the only sense that can be used to both send and receive information. Although it is not yet widely used in interacting with computers, there is a significant research effort in this area and commercial applications are becoming available.

The use of touch in the interface is known as *haptic interaction*. Haptics is a generic term relating to touch, but it can be roughly divided into two areas: cutaneous perception, which is concerned with tactile sensations through the skin; and kinesthetics, which is the perception of movement and position. Both are useful in interaction but they require different technologies.

In this section, we will look in a little more detail at some of the different types of haptic devices and consider, in particular, the role of haptics in universal design. As we will see, touch can provide both a primary source of information for users with visual impairments and a richer multi-modal experience for sighted users. 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.

Electronic braille displays benefit from two factors: a well-established tactile notation (braille) and a user group with expertise in using this notation. But can similar techniques be used to provide more generic tactile feedback, such as to display g raphics? The problem with using raised pins for this type of display is the resolution required. Braille requires only six or eight pins; a graphical display would require many more, which raises the problem of fitting the necessary number of fast actuators (to move the pins) into a few cubic centimeters. This presents a serious engineering challenge.

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. One of the most commonly used examples is the PHANTOM range, from SensAble Technologies. The PHANTOM provides three-dimensional

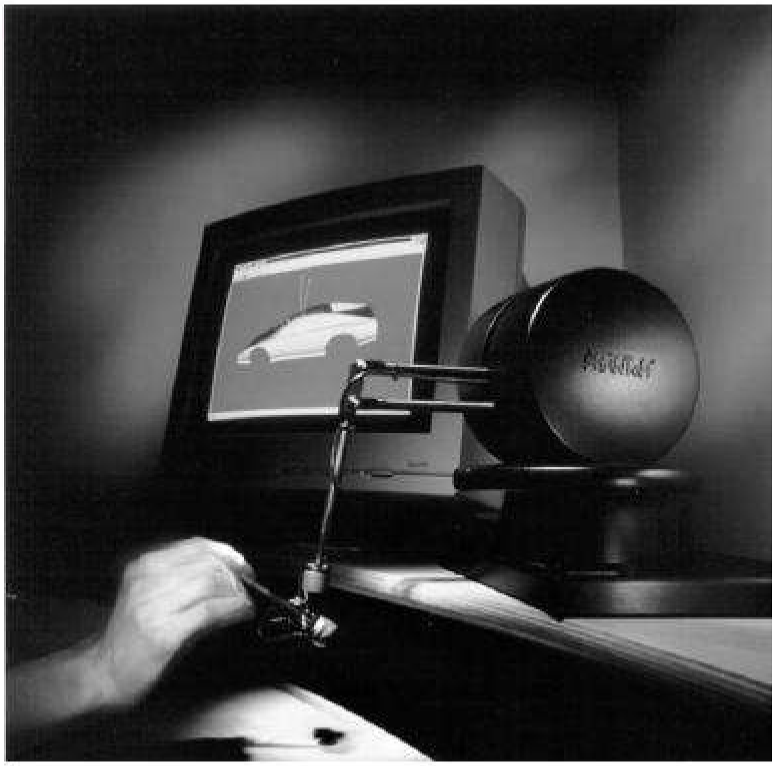


Figure 10.3 A PHANTOM Premium 1.5 haptic device. Source: Courtesy of SensAble Technologies

force feedback, allowing users to touch virtual objects. As well as offering the functionality of the mouse, in addition, the user's movement is monitored by optical sensors on the device, and these, together with models of the virtual objects, are used to calculate the forces applied back to the user. The user therefore can feel the outline and resistance of objects, their texture and position. This type of device has potential application for simulations and training situations where touch is important, such as medicine. It can also be used to provide a haptic 'image' of an interface, providing the user with information about the objects and their functionality based on how they feel. This offers another channel of information, which enhances the richness of the interaction and makes the design more universal.

10.3.3 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 handwriting appears to offer both textual and graphical input using the same tools.

Technology for handwriting recognition

The major piece of technology used to capture handwriting is the digitizing tablet, explained in more detail in Chapter 2. 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.

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.

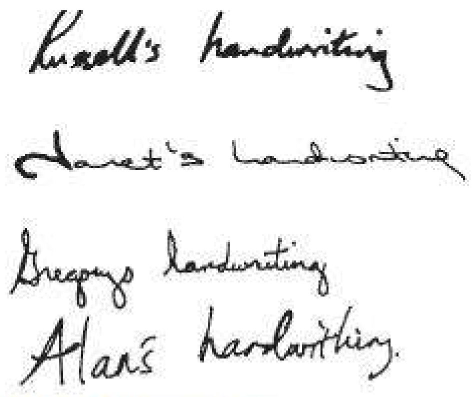

Russell's handwriting
Janet's handwriting
Gregory's handwriting
Alan's handwriting.

Figure 10.4 Handwriting varies considerably

These problems are reminiscent of those already discussed in speech recognition, and indeed the recognition problem is not dissimilar. The equivalent of co-articulation is also prevalent in handwriting, since different letters are written differently according to the preceding and successive ones. This causes problems for recognition systems, which work by trying to identify the lines that contain text, and then to segment the digitized image into separate characters. This is so difficult to achieve reliably that there are no systems in use today that are good at general cursive script recognition.

10.3.4 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 interpretation of the sampled data is very difficult, since segmenting the gestures causes problems. A team from Toronto [131] has produced a gesture recognition system that translates hand movements into synthesized speech, using five neural networks working in parallel to learn and then interpret different parts of the inputs. The Media Room at MIT uses a different approach in order to incorporate gestures into the interaction. The Media Room has one wall that acts as a large screen, with smaller touchscreens on either side of the user, who sits in a central chair. The user can navigate through information using the touchscreens, or by joystick, or by voice. Gestures are incorporated by using a position-sensing cube attached to a wristband worn by the user. The *put that there* system uses this gestural information coupled with speech recognition to allow the user to indicate what should be moved where by pointing at it. This is a much more natural form of interaction than having to specify verbally what it is that has to be moved and describing where it has to go, as well has having the advantage of conciseness. Such a short, simple verbal statement is much more easily interpreted by the speech recognition system than a long and complex one, with the resolution of ambiguity done by interpreting the other mode of interaction, the gesture. Each modality supports the other.

10.4 DESIGNING FOR DIVERSITY

10.4.1 Designing for users with disabilities

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.

Visual impairment

The sensory impairment that has attracted the most attention from researchers, perhaps because it is potentially also one of the most debilitating as far as interaction is concerned, is 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. We have already considered these in Section 10.3 so we will summarize only briefly here.

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 – a sure sign of good universal design!

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.

Speech impairment
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. Facilities to allow turn-taking protocols to be established also help natural communication [256]. Speech synthesis also needs to be rapid to reflect natural conversational pace, so responses can be pre-programmed and selected using a single switch.

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. Consistent navigation structure and clear signposting cues are also important to people with dyslexia. Color coding information can help in some cases and provision of graphical information to support textual can make the meaning of text easier to grasp.

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*:
1. Social interaction – problems in relating to others in a meaningful way or responding appropriately to social situations.
2. Communication – problems in understanding verbal and textual language including the use of gestures and expressions.
3. 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. The use of computers to support people with autism in this way is still a new research area and it is likely that new software and tools will develop in the next few years.

10.4.2 Designing for different age groups

We have considered how people differ along a range of sensory, physical and cognitive abilities. However, there are other areas of diversity that impact upon the way we design interfaces. One of these is age. In particular, older people and children have specific needs when it comes to interactive technology.

Older people

The proportion of older people in the population is growing steadily. Contrary to popular stereotyping, there is no evidence that older people are averse to using new technologies, so this group represents a major and growing market for interactive applications. People are living longer, have more leisure time and disposable income, and older people have increased independence. These factors have all led to an increase in older users. But 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.
Some older users, while not averse to using technology, may lack familiarity with it and fear learning. They may find the terminology used in manuals and training books difficult to follow and alien (words like ‘monitor’ and ‘boot’, for example, may have a completely different meaning to an older person than a young person). Interests and concerns may also be different from younger users.

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, however, 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.

Children are included in an *intergenerational design team* that focusses on understanding and analyzing context. Team members, including children, use a range of sketching and note-taking techniques to record their observations. Paper prototyping, using art tools familiar to children, enables both adults and children to participate in building and refining prototype designs on an equal footing. The approach has been used effectively to develop a range of new technologies for children.

10.4.3 Designing for cultural differences

The final area of diversity we will consider is cultural difference. Cultural difference is often used synonymously with national differences but this is too simplistic. Whilst there are clearly important national cultural differences, such as those we saw in Chapter 5, other factors such as age, gender, race, sexuality, class, religion and political persuasion, may all influence an individual's response to a system. This is particularly the case when considering websites where often the explicit intention is to design for a particular culture or subculture.

Another area where diversity can cause misunderstanding is in the use of gesture. Recently, one of the authors was teaching a new class of international students and was disconcerted to see one sitting in the front row, smiling and shaking his head. After the lecture this student came and asked a question. Every time the author asked the student if he understood the explanation, he shook his head, so further explanation ensued, much to the frustration of the student! It was only after a few minutes that it became clear: the student was from India and his gestural convention was to shake his head in agreement, the opposite of the European interpretation of the gesture. Use of gesture is quite common in video and animation and care must be taken with differences such as this. As interactions begin to incorporate gesture in virtual reality and avatars, issues such as this will become even more significant.

UNIT III MODELS AND THEORIES

Cognitive models –Socio-Organizational issues and stake holder requirements – Communication and collaboration models-Hypertext, Multimedia and WWW.

INTRODUCTION

One way to classify the models is in respect to how well they describe features of the competence and performance of the user. Quoting from Simon [323]: Competence models tend to be ones that can predict legal behaviour sequences but generally do this without reference to whether they could actually be executed by users. In contrast, performance models not only describe what the necessary behaviour sequences are but usually describe both what the user needs to know and how this is employed in actual task execution.
The presentation of the cognitive models in this chapter follows this classification scheme, divided into the following categories: n hierarchical representation of the user's task and goal structure
n linguistic and grammatical models
n physical and
device-level models.

GOAL AND TASK HIERARCHIES

The GOMS model of Card, Moran and Newell is an acronym for Goals, Operators, Methods and Selection [56]. A GOMS description consists of these four elements:
Goals These are the user's goals, describing what the user wants to achieve. Further, in GOMS the goals are taken to represent a 'memory point' for the user, from which he can evaluate what should be done and to which he may return should any errors occur.
Operators These are the lowest level of analysis. They are the basic actions that the user must perform in order to use the system. They may affect the system (for example, press the 'X' key) or only the user's mental state (for example, read the dialog box). There is still a degree of flexibility about the granularity of operators; we may take the command level 'issue the SELECT command' or be more primitive: 'move mouse to menu bar, press center mouse button . . .'.
Methods As we have already noted, there are typically several ways in which a goal can be split into subgoals. For instance, in a certain window manager a currently selected window can be closed to an icon either by selecting the 'CLOSE' option from a pop-up menu, or by hitting the 'L7' function key. In GOMS these two goal decompositions are referred to as methods, so we have the CLOSE-METHOD and the L7-METHOD:

GOAL: ICONIZE-WINDOW . [select GOAL: USE-CLOSE-METHOD . . MOVE-MOUSE-TO-WINDOW-HEADER . . POP-UP-MENU . . CLICK-OVER-CLOSE-OPTION GOAL: USE-L7-METHOD . . PRESS-L7-KEY]
The dots are used to indicate the hierarchical level of goals.
Selection From the above snippet we see the use of the word select where the choice of methods arises. GOMS does not leave this as a random choice, but attempts to predict which methods will be used. This typically depends both on the particular user and on the state of the system and details about the goals. For instance, a user, Sam, never uses the L7-METHOD, except for one game, 'blocks', where the mouse needs to be used in the game until the very moment the key is

pressed. GOMS captures this in a selection rule for Sam: User Sam: Rule 1: Use the CLOSE-METHOD unless another rule applies. Rule 2: If the application is 'blocks' use the L7-METHOD. The goal hierarchies described in a GOMS analysis are almost wholly below the level of the unit task defined earlier. A typical GOMS analysis would therefore consist of a single high-level goal, which is then decomposed into a sequence of unit tasks, all of which can be further decomposed down to the level of basic operators: GOAL: EDIT-MANUSCRIPT . GOAL: EDIT-UNIT-TASK repeat until no more unit tasks

Create a GOMS description of the task of photocopying an article from a journal. Discuss the issue of closure (see Chapter 1) in terms of your GOMS description. Answer One possible GOMS description of the goal hierarchy for this task is given below. Answers will vary depending on assumptions about the photocopier used as the model for the exercise. In this example, we will assume that the article is to be copied one page at a time and that a cover over the imaging surface of the copier has to be in place before the actual copy can be made.

```
GOAL: PHOTOCOPY-PAPER
.            GOAL: LOCATE-ARTICLE .
             GOAL: PHOTOCOPY-PAGE repeat until no more pages . .
            GOAL: ORIENT-PAGE . . .
             OPEN-COVER . . . SELECT-PAGE . . .
            POSITION-PAGE . . .
CLOSE-COVER . .
            GOAL: PRESS-COPY-BUTTON . .
            GOAL: VERIFY-COPY . . .
            LOCATE-OUT-TRAY . . .
             EXAMINE-COPY .
            GOAL: COLLECT-COPY . .
            LOCATE-OUT-TRAY
            . REMOVE-COPY (outer goal satisfied!) .
GOAL: RETRIEVE-JOURNAL . .
             OPEN-COVER . .
REMOVE-JOURNAL . . CLOSE-COVER
```

12.2.2 Cognitive complexity theory

Cognitive complexity theory, introduced by Kieras and Polson [199], begins with the basic premises of goal decomposition from GOMS and enriches the model to provide more predictive power. CCT has two parallel descriptions: 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. We introduced production rules in Chapter 1 and we further describe their use in CCT below. For the system grammar, CCT uses generalized transition networks, a form of state transition network

The production rules are a sequence of rules:

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

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 'cognitivecomplexity theory'. This is a reasonably frequent typing error and so we assume that we have developed good procedures to perform the task. We consider a fragment of the associated CCT production rules.

```
(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-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 (LOOKTEXT 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.

12.2.3 Problems and extensions of goal hierarchies

On the positive side, the conceptual framework of goal hierarchies and user goal stacks can be used to express interface issues, not directly addressed by the notations above. For instance, we can use this to examine in more detail the closure problem with early automated teller machines (ATMs) mentioned in the Design Focus box in Chapter 1, Section 1.3.2. These early ATMs gave the customers the money before returning their cards. Unfortunately, this led to many customers leaving their cards behind. This was despite on-screen messages telling them to wait. This is referred to as a problem of closure. The user's principal goal is to get money; when that goal is satisfied, the user does not complete or close the various subtasks which still remain open:

```
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. However, it is not always easy to predict when the user will consider a goal to have been satisfied.

LINGUISTIC MODELS

12.3.1 BNF Representative of the linguistic approach is Reisner's use of Backus–Naur Form (BNF) rules to describe the dialog grammar [301]. This views the dialog at a purely syntactic level, ignoring the semantics of the language. 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

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, and 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 lastpoint. 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 positionmouse (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

12.3.2 Task–action grammar

Measures based upon BNF have been criticized as not 'cognitive' enough. They ignore the advantages of consistency both in the language's structure and in its use of command names and letters. Task–action grammar (TAG) [284] attempts to deal with some of these problems by including elements such as parametrized grammar rules to emphasize consistency and encoding the user's world knowledge (for example, up is the opposite of down).

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
```

Measures based upon BNF could not distinguish between these consistent commands and an inconsistent alternative – say if ln 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’
| ‘ln’ + filenames + directory
```

This captures the consistency of the commands and closely resembles the original textual description. One would imagine that a measure of the complexity of the language based on the TAG description would be better at predicting actual learning and performance than a simple BNF one

THE CHALLENGE OF DISPLAY-BASED SYSTEMS

Goal hierarchy methods have different problems, as more display-oriented systems encourage less structured methods for goal achievement. Instead of having well-defined plans, the user is seen as performing a more exploratory task, recognizing fruitful directions and backing out of others. Typically, even when this exploratory style is used at one level, we can see within it and around it more goal-oriented methods.

So, for example, we might consider the high-level goal structure

```
WRITE_LETTER
 FIND_SIMILAR_LETTER
 COPY_IT
 EDIT_COPY
```

However, the task of finding a similar letter would be exploratory, searching through folders, etc. Such recognition-based searching is extremely difficult to represent as a goal structure. Similarly, the actual editing would depend very much on non-planned activities: ‘ah yes, I want to reuse that bit, but I’ll have to change that’. If we then drop to a lower level again, goal hierarchies become more applicable. For instance, during the editing stage we might have the ‘delete a word’ subdialog:

```
DELETE_WORD . SELECT_WORD . .
 MOVE_MOUSE_TO_WORD_START . .
 DEPRESS_MOUSE_BUTTON . .
MOVE_MOUSE_TO_WORD_END . .
RELEASE_MOUSE_BUTTON .
CLICK_ON_DELETE . .
 MOVE_MOUSE_TO_DELETE_ICON . .
CLICK_MOUSE_BUTTON
```

These problems have been one of the factors behind the growing popularity of situated action [334] and distributed cognition [208, 185] in HCI (see also Chapter 14). Both approaches emphasize the way in which actions are contingent upon events and determined by context, rather than being pre-planned. At one extreme, protagonists of these approaches seem to deny any planned actions or long-term goals. On the other hand, traditional cognitive modelers are modeling display-based cognition using production rules and similar methods, which include sensory data within the models.

PHYSICAL AND DEVICE MODELS

12.5.1 Keystroke-level mode
Compared with the deep cognitive understanding required to describe problemsolving activities, the human motor system is well understood. KLM (Keystroke-Level Model [55]) uses this understanding as a basis for detailed predictions about user performance. It is aimed at unit tasks within interaction – the execution of simple command sequences, typically taking no more than 20 seconds. Examples of this would be using a search and replace feature, or changing the font of a word. It does not extend to complex actions such as producing a diagram.
The task is split into two phases:
acquisition of the task, when the user builds a mental representation of the task;
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, and can be thought of as a very low-level GOMS model where the method is given. The 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
R 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 tK, then the total time doing keystrokes is
TK = 2tK
Similar calculations for the rest of the operators give a total time of
Texecute = TK + TB + TP + TH + TD + TM + TR
= 2tK + 2tB + tP + 3tH + 0 + 2tM + 0
In this example, the system response time was zero. However, if the user had to wait for the system then the appropriate time would be added. In many typing tasks, the user can type ahead anyway and thus there is no need to add response times. Where needed, the response time can be measured by observing the system

12.5.2 Three-state model
Buxton has developed a simple model of input devices [53], 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 12.1). 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 12.2).
A touchscreen 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 touchscreen 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.

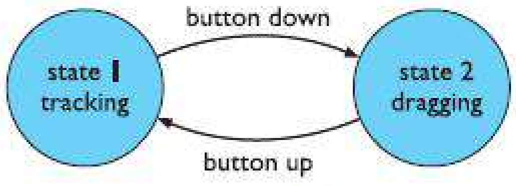


Figure 12.1 Mouse transitions: states 1 and 2

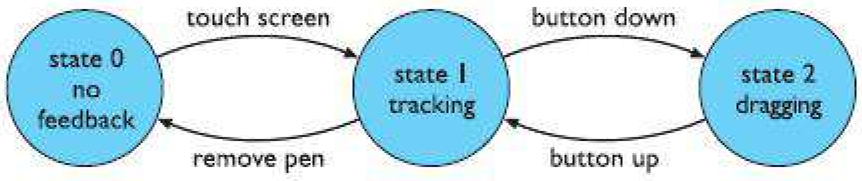


Figure 12.2 Light pen transitions: three states

The only additional complexity is if the device has several buttons, in which case wewould have one state for each button: 2left, 2middle, 2right.One use of this classification is to look at

different pointing tasks, such as iconselection or line drawing, and see what state 0–1–2 behavior they require. We can
then see whether a particular device can support the required task. If we have to usean inadequate device, it is possible to use keyboard keys to add device states. Forexample, with a touchscreen, we may minate the escape key to be the 'virtual'mouse button whilst the user's finger is on the screen. Although the mixing of keyboardand mouse keys is normally a bad habit, it is obviously necessary on occasions.At first, the model appears to characterize the states of the device by the inputs \available to the system. So, from this perspective, state 0 is clearly differentfrom states 1 and 2. However, if we look at the state 1–2 transaction, we see that itis symmetric with respect to the two states. In principle, there is no reason why aprogram should not decide to do simple mouse tracking whilst in state 2 and dragthings about in state 1. That is, there is no reason until you want to type something!The way we can tell state 1 from state 2 is by the activity of the *user*. State 2 requiresa button to be pressed, whereas state 1 is one of relative relaxation (whilst still requiringhand–eye coordination for mouse movement). There is a similar difference in
tension between state 0 and state 1.It is well known that Fitts' law has different timing constants for different devices.Recall that Fitts' law says that the time taken to move to a target of size S at a distance

D is:$a + b \log2(D/S + 1)$

The constants a and b depend on the particular pointing device used and the skill ofthe user with that device. However, the insight given by the three-state model is thatthese constants also depend on the device state. In addition to the timing, the finalaccuracy may be affected.These observations are fairly obvious for state 0–1 devices. With a touchscreen, orlight pen, a cursor will often appear under the finger or pen when it comes in contactwith the screen. The accuracy with which you can move the cursor around willbe far greater than the accuracy with which you can point in the first place. Also it isreasonable to expect that the Fitts' law constant will be different, although not soobvious which will be faster.

ORGANIZATIONAL ISSUES

13.2.1 Cooperation or conflict?

The term 'computer-supported *cooperative* work' (CSCW) seems to assume that groups will be acting in a cooperative manner. This is obviously true to some extent; even opposing football teams cooperate to the extent that they keep (largely) within the rules of the game, but their cooperation only goes so far. People in organizations
and groups have conflicting goals, and systems that ignore this are likely to fail spectacularly. Imagine that an organization is already highly computerized, the different departments all have their own systems and the board decides that an integrated information system is needed. The production manager can now look directly at stocks
when planning the week's work, and the marketing department can consult the sales department's contact list to send out marketing questionnaires. All is rosy and the company will clearly run more efficiently – or will it?

The storekeeper always used to understate stock levels slightly in order to keep an emergency supply, or sometimes inflate the quoted levels when a delivery was due from a reliable supplier. Also, requests for stock information allowed the storekeeper to keep track of future demands and hence plan future orders. The storekeeper has now lost a sense of control and important sources of information.

Members of the sales department are also unhappy: their contacts are their livelihood. The last thing they want is someone from marketing blundering in and spoiling a relationship with a customer built up over many years. Some of these people may resort to subverting the system, keeping 'sanitized' information online, but the real information in personal files. The system gradually loses respect as the data it holds is incorrect, morale in the organization suffers and productivity drops. The board gets worried and meets to consider upgrading the computer system!

Before installing a new computer system, whether explicitly 'coop erative' or not, one must identify the *stakeholders* who will be affected by it. These are not just the immediate users, but anyone whose jobs will be altered, who supplies or gains information from it, or whose power or influence within the organization will increase or decrease. It will frequently be the case that the formal 'client' who orders the system falls very low on the list of those affected. Be very wary of changes that take power, influence or control from some stakeholders without returning something tangible in their place.

13.2.2 Changing power structures

The identification of stakeholders will uncover information transfer and power relationships that cut across the organizational structure. Indeed, all organizations have these informal networks that support both social and functional contacts. However, the official lines of authority and information tend to flow up and down through line management. The physical layout of an organization often reflects the formal hierarchy: each department is on a different floor, with sections working in the same area of an office. If someone from sales wants to talk to someone from marketing then one of them must walk to the other's office. Their respective supervisors can monitor the contact. Furthermore, the physical proximity of colleagues can foster a sense of departmental loyalty. An email system has no such barriers; it is as easy to 'chat' to someone in another department as in your own. This challenges the mediating and controlling role of the line managers.

Furthermore, in face-to-face conversation, the manager can easily exert influence over a subordinate: both know their relative positions and this is reflected in the patterns of conversation and in other non-verbal cues. Email messages lose much of this sense of presence and it is more difficult for a manager to exercise authority. The 'levelling' effect even makes it possible for subordinates to direct messages 'diagonally' across the hierarchy, to their manager's peers, or, even worse, to their manager's manager!

Many organizations are moving toward flatter management structures anyway, so from a strategic viewpoint these effects may be acceptable. But can the organization cope with a disaffected junior management during the transition? For other organizations the effects may be less welcome and the system dropped or heavily regulated.

In one case, an email system was introduced and was agreed to be functioning well, but the management, feeling a loss of control and suspicion over their subordinates' communications, introduced logging so that all email messages could be monitored. The system quickly fell into disuse. Logging of email is becoming more widespread
with employers using it to identify cases of system 'abuse' by employees. But such activity must be handled carefully: it is as likely to backfire on the management by reducing the productive use of email as it is to have the desired effect.

13.2.3 The invisible worker

The ability to work and collaborate at a distance can allow functional groups to be distributed over different sites. This can take the form of cross-functional neighbourhood centers, where workers from different departments do their jobs in electronic contact with their functional colleagues. Alternatively, distributed groupware can allow the true home-based teleworker to operate on similar terms to an office-based equivalent. The ecological and economic advantages of such working practices are now becoming well established, and it seems that communications and
CSCW technology can overcome many of the traditional barriers. presence can be a problem. When the time comes for promotion, the present employee may seem more worthy than the distant one – not because of any objective
criteria, but because presence increases perceived worth.

13.2.4 Who benefits?

One frequent reason for the failure of information systems is that the people who get the benefits from the system are not the same as those who do the work. One example, which we discuss in more detail in Chapter 19, is structured message systems such as Lens. In these systems the sender has to do work in putting information into
fields appropriately, but it is the recipient who benefits. Another example is shared calendars. The beneficiary of the system is a manager who uses the system to arrange meeting times, but whose personal secretary does the work of keeping the calendar up to date. Subordinates are less likely to have secretarial support, yet must keep up the calendar with little perceived benefit. Of course, chaos results when a meeting is automatically arranged and the subordinates may have to rearrange commitments that have not been recorded on the system. The manager may force use by edict or the system may simply fall into disuse. Many such groupware systems are introduced on a 'see if it works' basis, and so the latter option is more likely.

13.2.5 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*. Take an electronic conferencing system. If there is plenty of discussion of relevant topics then there are obvious advantages to subscribing and reading the contributions. However, when considering writing a contribution, the effort of doing so may outweigh any benefits. The total benefit of the system for each user outweighs the costs, but for any particular decision the balance is overturned. A few free riders in a conference system are often not a problem, as the danger is more likely from too much activity. In addition, in electronic conferences the patterns of activity and silence may reflect other factors such as expertise. However, it is easy for the number of free riders gradually to increase and the system slide into disuse.

It is hard to enforce equal use, except by restrictive schemes such as round-robin contributions (everyone contributes something however short). In the real world, such problems are often solved by social pressure, and the free rider reacts to the collective censure of the group. Increasing the *visibility* of participants' contributions might also help these social mechanisms. For example, one could display an activity meter showing the number of contributions from each subscriber.

13.2.6 Critical mass

Another issue related to the free rider problem is the need to develop a *critical mass*. When telephones were only in public places, their use as a form of pervasive interpersonal communication was limited. However, once a large number of people have telephones in their homes it becomes worthwhile paying to have a telephone installed. In cost/benefit terms, the early subscribers probably have a smaller benefit than the cost. Only when the number of subscribers increases beyond the critical mass does the benefit *for all* dominate the cost (see Figure 13.1). The situation for conferencing systems and email is, of course, very similar. We can learn something from the lessons of the telephone system and other successful technologies (but remember, telephones took the best part of 100 years
to become pervasive in affluent countries). The telephone was useful for subgroups before it became beneficial for all. Even when only a small proportion of the population had personal telephones, they still formed a significant proportion of their social group, so these cliques of use could grow gradually over time.

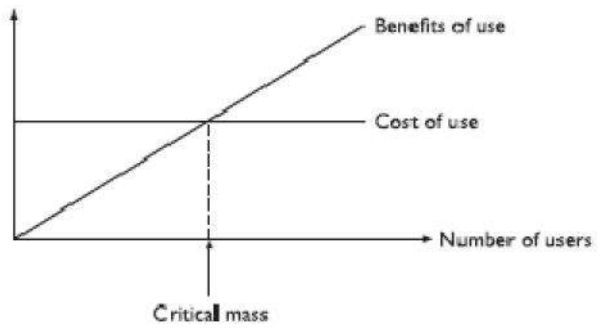


Figure 13.1 Cost/benefit of system use

13.2.7 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.

13.2.8 Evaluating the benefits

We have seen several problems that can arise from the mismatch between information systems and organizational and social factors. Let us assume that we have a system in place – and it has not fallen apart at the seams. Everyone seems happy with it and there are no secret resentments. Now it is time to count the cost – it was an expensive system to buy and install, but was it worth it? This is an almost impossible question to answer. The benefits from cooperative systems, especially organization-wide systems such as email or electronic conferencing, are in terms of job satisfaction or more fluid information flow. Some, such as the *video wall* (see Chapter 19), are expected primarily to help social contact within the organization.

CAPTURING REQUIREMENTS

13.3.1 Who are the stakeholders?

Understanding stakeholders is key to many of the approaches to requirements capture, since in an organizational setting it is not simply the end-user who is affected by the introduction of new technology. Imagine that a new billing system is to be introduced by a local gas supplier. Who will be affected by this decision? Obviously, the people who are responsible for producing and sending out bills – they will be the ones using the system directly. But where do they get the information from to produce the bills? To whom do they send the bills? Who determines the level of charging and on what grounds? Who stands to make a profit from increased revenue? Who will suffer if customers choose to switch supplier due to the improved service? Meter readers, customers, managers, regulators, shareholders and competitors are all stakeholders in the system. We need approaches that will capture the
complexity of their concerns, which may be in conflict with each other. A stakeholder, therefore, can be defined as anyone who is affected by the success or failure of the system. It can be useful to distinguish different categories of stakeholder, and the following categorization from the CUSTOM approach (see [200]) is
helpful for this:

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.

13.3.2 Socio-technical models

Socio-technical models for interactive systems are therefore concerned with technical, social, organizational and human aspects of design. They recognize the fact that technology is not developed in isolation but as part of a wider organizational environment. It is important to consider social and technical issues side by side so that human issues are not overruled by technical considerations.

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:

n The problem being addressed: there is a need to understand why the technology is being proposed and what problem it is intended to solve.
n The stakeholders affected, including primary, secondary, tertiary and facilitating, together with their objectives, goals and tasks.
n The workgroups within the organization, both formal and informal.
n The changes or transformations that will be supported.
n The proposed technology and how it will work within the organization.
n External constraints and influences and performance measures.

CUSTOM methodology
CUSTOM is a socio-technical methodology designed to be practical to use in small organizations [200]. It is based on the User Skills and Task Match (USTM) approach, developed to allow design teams to understand and fully document user requirements [219]. CUSTOM focusses on establishing stakeholder requirements:all stakeholders are considered, not just the end-users.
It is applied at the initial stage of design when a *product opportunity* has been identified, so the emphasis is on capturing requirements. It is a forms-based methodology, providing a set of questions to apply at each of its stages.

There are six key stages to carry out in a CUSTOM analysis:

1.Describe the organizational context, including its primary goals, physical characteristics, political and economic background.
2. 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. For example, CUSTOM addresses issues such as stakeholder motivation, disincentives, knowledge, skills, power and influence within the organization, daily tasks and so on.
3. 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.
4. 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.
5. 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. For example, if a stakeholder is identified as currently lacking a particular skill that is required in the proposed system then a need
for training is identified.
6. Consolidate and check stakeholder requirements. Here the stakeholder needs list is checked against the criteria determined at earlier stages.

Open System Task Analysis (OSTA)

OSTA [116] is an alternative socio-technical approach, which attempts to describe what happens when a technical system is introduced into an organizational work environment. Like CUSTOM, OSTA specifies both social and technical aspects of the system. However, whereas in CUSTOM these aspects are framed in terms of stakeholder perspectives, in OSTA they are captured through a focus on tasks.

OSTA has eight main stages:

1. The primary task which the technology must support is identified in terms of users' goals.
2. Task inputs to the system are identified. These may have different sources and forms that may constrain the design.
3. The external environment into which the system will be introduced is described, including physical, economic and political aspects.
4. The transformation processes within the system are described in terms of actions performed on or with objects.
5. The social system is analyzed, considering existing work-groups and relationships within and external to the organization.
6. The technical system is described in terms of its configuration and integration with other systems.
7. Performance satisfaction criteria are established, indicating the social and technical requirements of the system.
8. The new technical system is specified.

13.3.3 Soft systems methodology

The socio-technical models we have looked at focus on identifying requirements from both human and technical perspectives, but they assume a technological solution is being proposed. Soft systems methodology (SSM) arises from the same tradition but takes a view of the organization as a system of which technology and people are components. There is no assumption of a particular solution: the emphasis is rather on understanding the situation fully.

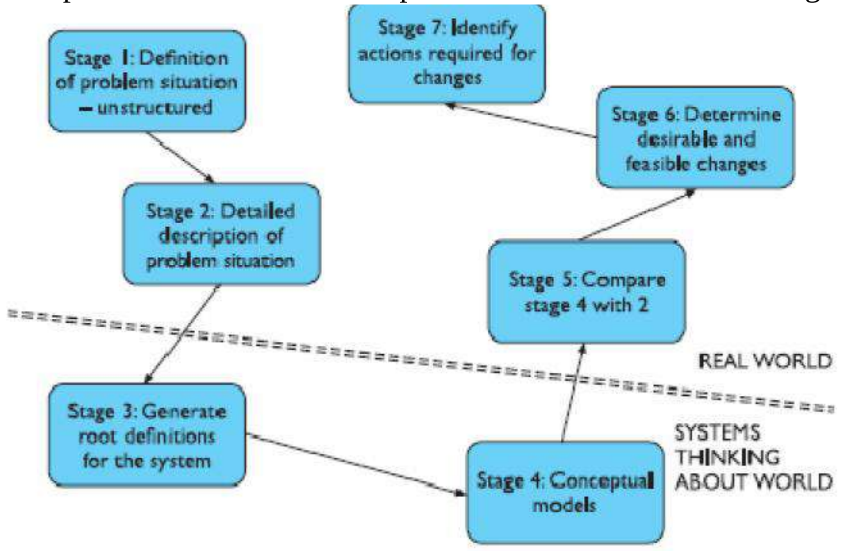

Stage 1: Definition of problem situation – unstructured
Stage 2: Detailed description of problem situation
Stage 3: Generate root definitions for the system
Stage 4: Conceptual models
Stage 5: Compare stage 4 with 2
Stage 6: Determine desirable and feasible changes
Stage 7: Identify actions required for changes
REAL WORLD
SYSTEMS THINKING ABOUT WORLD

Figure 13.2 The seven stages of soft systems methodology. (Adapted from Checkland [66], p.163)

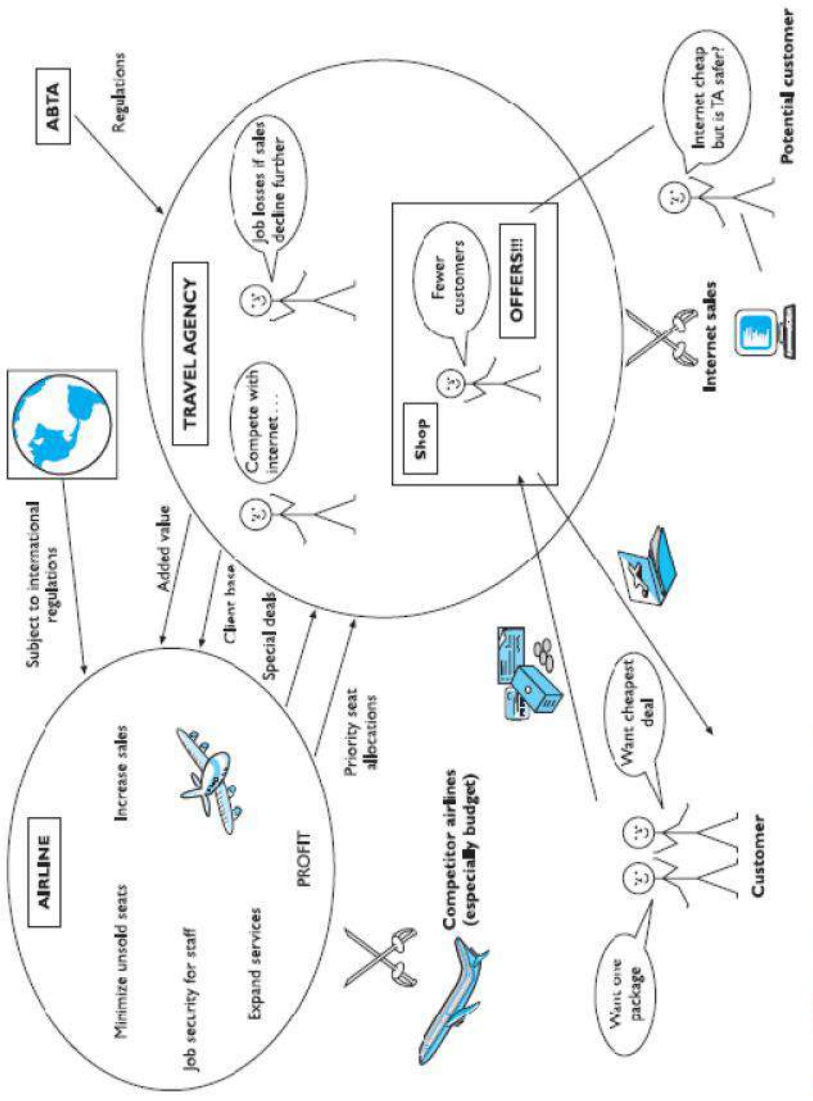


Figure 13.3 A rich picture of a travel agency

At the next stage in SSM we move from the real world to the systems world and attempt to generate *root definitions* for the system, which define the essence of what the system is about. There may be several root definitions of a system, representing different stakeholder perspectives, for example. Root definitions are described in terms of specific elements, summarized using the acronym, CATWOE:

Clients – those who receive output or benefit from the system.

Actors – those who perform activities within the system.

Transformations – the changes that are effected by the system. This is a critical part of the root definition as it leads to the activities that need to be included in the next stage. These 'transform' the inputs of the system into the required outputs.

Weltanschauung – (from the German) meaning 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.

13.3.4 Participatory design

Participatory design is a philosophy that encompasses the whole design cycle. It is design in the workplace, where the user is involved not only as an experimental subject or as someone to be consulted when necessary but as a member of the design team. Users are therefore active collaborators in the design process, rather than passive participants whose involvement is entirely governed by the designer. The argument is that users are experts in the work context and a design can only be effective within that context if these experts are allowed to contribute actively to the design process. In addition, introduction of a new system is liable to change the work context and organizational processes, and will only be accepted if these changes are acceptable to the user. Participatory design therefore aims to refine system requirements iteratively through a design process in which the user is actively involved.

Effective Technical and Human Implementation of Computer-based Systems (ETHICS)

ETHICS [243] 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. In the 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 decisionmaking 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:

1. *Make the case for change*. Change for its own sake is inappropriate. If a case cannot be made for changing the current situation then the process ends and the system remains as it is.
2. *Identify system boundaries*. This focusses on the context of the current system and its interactions with other systems, in terms of business, existing technology, and internal and external organizational elements. How will the change impact upon each of these?
3. *Describe the existing system*, including a full analysis of inputs and outputs and the various other activities supported, such as operations, control and coordination.
4. *Define key objectives*, identifying the purpose and function of each area of the organization.
5. *Define key tasks*: what tasks need to be performed to meet these objectives?
6. *Define key information needs*, including those identified by analysis of the existing system and those highlighted by definition of key tasks.

7. *Diagnose efficiency needs*, those elements in the system that cause it to underperform 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.
8. *Diagnose job satisfaction needs*, with a view to increasing job satisfaction where it is low.
9. *Analyze likely future changes*, whether in technology, external constraints (such as legal requirements), economic climate or stakeholder attitudes. This is necessary to ensure that the system is flexible enough to cope with change.
10. *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).

13.3.5 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 encultured, to understand the situation from within its own cultural framework. Culture here means that of the particular workgroup 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.

COMMUNICATION AND COLLABORATION MODELS
FACE-TO-FACE COMMUNICATION

Face-to-face contact is the most primitive form of communication – primitive, that is, in terms of technology. If, on the other hand, we consider the style of communication, the interplay between different channels and productivity, we instead find that face-to-face is the most sophisticated communication mechanism available.

14.2.1 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. An example of these problems concerns personal space. When we converse with one another we tend to stand with our heads a fairly constant distance apart. If people start to converse at opposite ends of a room, they will quickly move toward one another until they are a few feet apart. The exact distance depends somewhat on context; a high level of noise may make people come closer just to be heard. However, even in crowded rooms, conversants will dip their heads toward one another whilst speaking and then straighten up to restore their personal distance.

Direction is also important. We can accept people closer to us if they are at our sides or behind than if we are facing them. Because of this, passengers on tube trains, forced to be close, will incline their faces at an angle to one another whilst talking. Personal space also differs across cultures: North Americans get closer than Britons, and southern Europeans and Arabs closer still. This can cause considerable problems during cross-cultural meetings. Imagine a Briton, Eustace Warbuck- Smyth, and an American, Bud Sterton, conversing. After a few minutes Eustace is bent backwards over a table, trying to maintain his personal distance, whilst Bud stands almost knee to knee trying to get close enough. Eustace feels Bud is either rather aggressive, or possibly over-friendly. Bud, on the other hand, feels Eustace is rather distant and uninterested. Unless the situation gets extreme, or the participants are trained in non-verbal skills, they will be unaware of why they feel uncomfortable.

14.2.2 Eye contact and gaze
Long-term gazing into one another's eyes is usually reserved for lovers. However, normal conversation uses eye contact extensively, if not as intently. Our eyes tell us whether our colleague is listening or not; they can convey interest, confusion or boredom. Sporadic direct eye contact (both looking at one another's eyes) is important in establishing a sense of engagement and social presence. People who look away when you look at them may seem shifty and appear to be hiding something. Furthermore, relative frequency of eye contact and who 'gives way' from direct eye contact is closely linked to authority and power.

14.2.3 Gestures and body language
In a similar but more direct way, we use our hands to indicate items of interest. This may be conscious and deliberate as we point to the item, or may be a slight wave of the hand or alignment of the body. Again, a video connection may not be sufficient to allow our colleagues to read our movements. This can be a serious problem since our conversation is full of expressions such as 'let's move this one there', where the 'this' and 'there' are indicated by gestures (or eyegaze). This is called *deictic reference*

14.2.4 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: Alison: Do you fancy that film . . . *er*. . . 'The Green' *um* . . . it starts at eight.
Brian: Great!
Alison has asked Brian whether he wants to go to the cinema (or possibly to watch the television at home). She is a bit vague about the film, but Brian obviously does not mind! However, if we had listened to the conversation more closely and watched Alison and Brian we would have seen more exchanges. As Alison says 'that film *er*. . .', she looks at Brian. From the quizzical look on his face he obviously does not know which film she is talking about. She begins to expand 'The Green *um* . . .', and light dawns; she can see it in his eyes and he probably makes a small affirmative sound 'uh huh'.

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. Imagine making no response as someone talks to you, no little 'yes'es, no nods or raised eyebrows. You could answer questions and speak in turn, but not use back channels. It is likely that your colleague would soon become very uncomfortable, possibly rambling on with ever more detailed explanations, looking for some sign of understanding:2

Do you fancy that film . . . *er*. . . 'The Green' *um* . . . the one with Charles Dermot in . . . you know with that song, *er*and the black cat on the poster . . . *uhh*These 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 (such as the telephone) 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 (and perhaps you have been going on a bit). 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.

14.2.5 Turn-taking

As well as giving confirmation to the speaker that you understand, and indications when you do not, back channels can be used to interrupt politely. Starting to speak in the middle of someone's utterance can be rude, but one can say something like 'well uh' accompanied by a slight raising of the hands and a general tensing of the body and screwing of the eyes. This tells the speaker that you would like to interrupt, allowing a graceful transition. In this case, the listener *requested* the floor.

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.

CONVERSATION

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.

14.3.1 Basic conversational structure

Imagine we have a transcript of a conversation, recalling from Chapter 9 that the production of such a transcript is not a simple task. For example, a slightly different version of Alison and Brian's conversation may look like this:

Alison: Do you fancy that film?
Brian: The *uh* (*500 ms*) with the black cat – 'The Green whatsit'?
Alison: Yeah, go at *uh* . . . (*looks at watch – 1.2 s*) . . . 20 to?
Brian: Sure.
This transcript is quite heavily annotated with the lengths of pauses and even Alison's action of looking at her watch. However, it certainly lacks the wealth of gesture and back channel activity that were present during the actual conversation. Transcripts may be less well documented, perhaps dropping the pause timings, or more detailed, adding more actions, where people were looking and some back channelling. Whilst thinking about the structure of conversation, the transcript above is sufficient.
As we have noted previously, the most basic conversational structure is *turntaking*. 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.

14.3.2 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. We can identify two types of context within conversation:

internal context – dependence on earlier utterances. For example, when Brian says 'masses' in the last transcript, this is meaningful in the light of Alison's question 'and lots of chocolate?'. This in turn is interpreted in the context of Brian's original offer of gateau.

external context – dependence on the environment. For example, if Brian had said simply 'do you want one?', this could have meant a slice of gateau, or, if he had been holding a bottle, a glass of wine, or, if accompanied by a clenched fist, a punch on the nose

14.3.3 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.

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!
Alison began the conversation with the *topic* of roses. Brian shifts to the related, but distinct, topic of greenfly. However, for some reason Alison has missed this shift in focus, so when she makes her second utterance, her focus and Brian's differ, leading to the *breakdown* in communication. The last two utterances are a recovery which re-establishes a shared *dialog focus*.

Another way of classifying utterances is by their relation to the task in hand. At one extreme the utterance may have no direct relevance at all, either a digression or purely social. Looking at the task-related conversation, the utterances can be classified into three kinds [335]:

substantive directly relevant to the development of the topic;
annotative points of clarification, elaborations, etc.;
procedural talking about the process of collaboration itself.
In addition, the procedural utterances may be related to the structure of collaboration itself, or may be about the technology supporting the collaboration. The latter is usually in response to a breakdown where the technology has intruded into the communication.

Alison and Brian are now discussing the best way to get to the cinema. Alison is

using a whiteboard to draw a map.
1. Alison: You go along this road until you get to the river.
2. Brian: Do you stop before the river or after you cross it?
3. Alison: Before.
4. Brian: Draw the river in blue and the roads black . . .
5. Alison: So, you turn right beside the river.
6. Brian: Past the pub.
7. Alison: Yeah . . . Is there another black pen, this one's gone dry?

14.3.4 Breakdown and repair

At a lower level, we may see breakdown due to incorrectly read gestures or eyegaze, and through missed or inappropriate back channel responses. For instance, in Section 14.2.5, we described the problems in turn-taking during satellite-based video conferences. It may be difficult to interpret just where a breakdown occurred, as the breakdown may take some time to come to light, and be apparent at a different level from which it began. Alison and Brian are enjoying a day out at a country park:

Alison: Isn't that beautiful?
She points at a stag standing beside a large tree; Brian sees the tree.
Brian: The symmetry of the branches.
Alison: How some people can dislike them I can't understand.
Brian: Yes, the rangers ought to cull those deer, they strip the bark terribly
in winter.
Alison: (*Silence*)

The breakdown began with a confused gesture, but led to a divergence of dialog focus. Unfortunately, Brian's remark about the branching (of the tree) could be interpreted in terms of Alison's focus (the stag's antlers) and thus the breakdown did not become apparent until Brian had well and truly put his foot in it. Happily, most breakdowns are detected more quickly, but the deeper the breakdown, and the longer it lasts, the more difficult it is to recover.

14.3.5 Constructing a shared understanding

We have seen that human conversation is in itself inherently ambiguous, relying on context and shared understanding between the parties to disambiguate the utterances. In some spheres, such as legal contracts, the precise meaning out of context becomes very important and thus highly stylized language is used to reduce ambiguity. However, even the legal profession depends on a large body of shared knowledge and understanding about legal terms, case law, etc. Similarly, a book, such as this, attempts to use less ambiguous language and only commonly available knowledge. The major difference between a book and conversation is that the latter is interactive. The shared knowledge used in a book is static, whereas that used during a conversation is dynamic, as the participants increase their understanding of one another and as they shift their focus from topic to topic.

14.3.6 Speech act theory

A particular form of conversational analysis, *speech act theory*, has been both influential and controversial in CSCW. Not only is it an analytic technique, but it has been used as the guiding force behind the design of a commercial system, Coordinator. Speech act theory has origins going back over 25 years, but was popularized by Winograd and Flores in the design of Coordinator [381].

The basic premise of speech act theory is that utterances can be characterized by what they *do*. If you say ‘I’m hungry’, this has a certain *propositional meaning* – that you are feeling hungry. However, depending on who is talking and to whom, this may also carry the meaning ‘get me some food’ – the intent of the statement is to evoke an action on the part of the hearer. Speech act theory concerns itself with the way utterances interact with the actions of the participants.
Some speech acts actually cause a significant effect by the act of being said. The classic example is when a minister says ‘I pronounce you husband and wife’. This is not simply a statement that the minister is making about the couple. The act of saying the words changes the state of the couple. Other acts include promises by the speaker to do something and requests that the hearer do something. These basic acts are called *illocutionary points*.

Individual speech acts can contribute to a conversation. The basic structure of conversations can then be seen as instances of generic conversations. One example of such a generic structure is a *conversation for action* (*CfA*). This is shown as a state diagram in Figure 14.1. It represents the stages two participants go through in

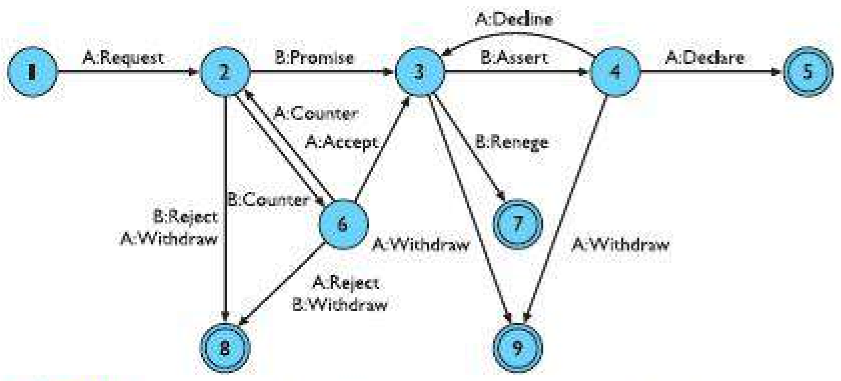


Figure 14.1 Conversation for action. Source: *Understanding Computers and Cognition: A New Foundation for Design* by Terry Winograd/Fernando Flores, © 1986. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ

initiating an external action that one of them should perform. There are two variants, the one shown representing a conversation where the first speaker (A) is requesting that the other participant (B) does something. The other, similar, variant is where the first speaker begins with an offer. The numbered circles in Figure 14.1 are ‘states’ of the conversation, and the labeled arcs represent the speech acts, which move the conversation from state to state. Note that the speech acts are named slightly differently in different sources (by the same author even!), but the structure of a CfA is the same. The simplest route
through the diagram is through states 1–5.
Alison: Have you got the market survey on chocolate mousse?

Brian: Sure.
Rummages in filing cabinet and hands it to Alison
Brian: There you are.
Alison: Thanks.

Alison makes a *request* for the survey (although it is phrased as a question). Brian *promises* to fulfill the request ('sure'). After he feels he has done so (by handing it to Alison), Brian *asserts* that the request has been fulfilled ('there you are') and Alison *declares* her satisfaction that Brian has completed her request ('thanks'). More complex routes may involve some negotiating between the parties. For example, the conversation might have begun:

Alison: Have you got the market survey on chocolate mousse?
Brian: I've only got the summary figures.
Alison: That'll do.
In this Alison's *request* is met by a *counter* from Brian, that is Brian attempts to modify Alison's request. This brings us to state 6 in the diagram. Alison then *accepts* Brian's counter, bringing the conversation back to state 3. The network has some nodes marked with a double circle. These are the completion nodes, and at these points neither party expects any more acts by the other as part of this conversation. So the fragment above which left Alison and Brian in state 3 must continue. Of these completion nodes only state 5 represents conclusions where the request has been satisfied.

For example, Alison's initial request could have been answered with 'it's confidential' (meaning 'you can't have it'). This is the action of Brian *rejecting* Alison's request, leaving the conversation in state 8 and complete. 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.

TEXT-BASED COMMUNICATION

For *asynchronous* groupware (and even some synchronous systems), the major form of direct communication is text based. There are exceptions to this, for instance voice messaging systems and answerphones, and other media may be used in addition to text such as graphics, voice annotation or even video clips. But despite these, text is still the dominant medium. Text-based communication is familiar to most people, in that they will have written and received letters. However, the style of letter writing and that of face-toface communication are very different. The text-based communication in groupware systems is acting as a speech substitute, and, thus, there are some problems adapting between the two media.

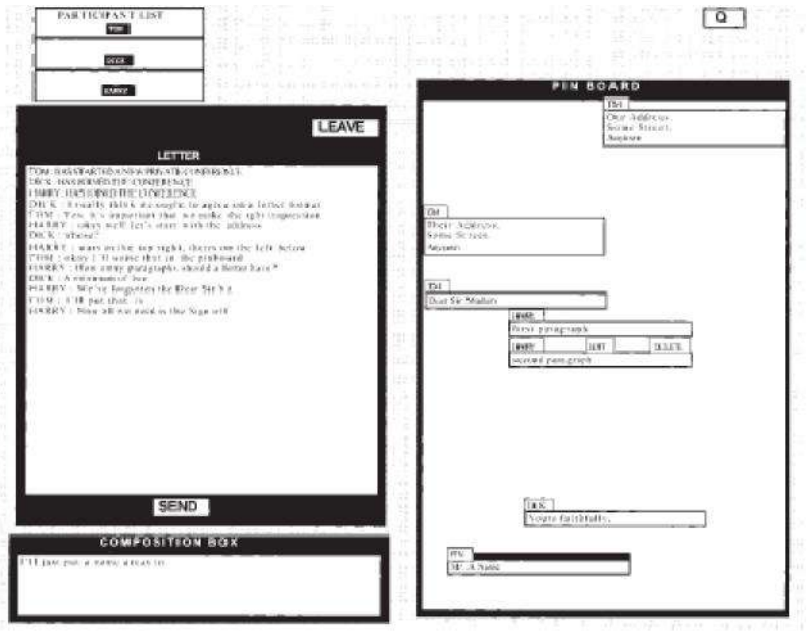
There are four types of textual communication in current groupware:
discrete – directed message as in email. There is no explicit connection between different messages, except in so far as the text of the message refers to a previous one.

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.

14.4.1 Back channels and affective state

One of the most profound differences between face-to-face and text-based communication is the lack of fine-grained channels. Much of the coordination of face-to-face conversation depends on back channels and interpretation of the listener's expressions. Text-based communication loses these back channels completely. Consider the effect of this on even a two-party conversation.

Where the



TOM : HAS STARTED A NEW PRIVATE CONFERENCE
DICK : HAS JOINED THE CONFERENCE
HARRY : HAS JOINED THE CONFERENCE
DICK : I really think we ought to agree on a letter format
TOM : Yes, it's important that we make the right impression
HARRY : okay well let's start with the address
DICK : where?
HARRY : ours on the top right, theirs on the left below
TOM : okay I'll write that in the pinboard
HARRY : How many paragraphs should a letter have?
DICK : A minimum of two
HARRY : We've forgotten the Dear Sir bit
TOM : I'll put that in
HARRY : Now all we need is the sign off

I'll just put a name [illegible] in

Figure 14.2 Conferencer screen shot showing text transcript and pin-board

speaker would pause to seek back channel confirmation or to offer the floor, the text 'speaker' must either continue regardless, or finish the message, effectively passing the turn. One consequence of the lack of interruptions and more measured pace of interaction is that the utterances are more grammatical than speech – but still *not* Queen's English! 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!

14.4.2 Grounding constraints

In Section 14.3.5, we discussed the process by which conversants obtain common ground. This grounding process is linked strongly with the types of channels through which the conversants communicate. Clark and Brennan [71] describe the properties 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. For example, simultaneity in face-to-face conversation allows back channel responses. Even where, say, two participants can see each other's typed messages as they are produced, the nature of typing makes it all but impossible to type your message whilst looking for your colleague's 'back channel' response.

In a text-based system, different participants can compose simultaneously, but they lack cotemporality. As we saw, even if the messages appear as they are produced, they will not be read in real time. In addition, the messages may only be delivered when complete and even then may be delayed by slow communications networks.

Linear transcripts obviously have some idea of sequence, but this is confused by the overlap and interleaving caused by the lack of cotemporality and simultaneity. Consider this typical interchange during the use of the York Conferencer system:

1. Bethan: How many should be in the group?
2. Rowena: Maybe this could be one of the four strongest reasons?
3. Rowena: Please clarify what you mean.
4. Bethan: I agree.
5. Rowena: Hang on.
6. Rowena: Bethan what did you mean?

Rowena and Bethan composed their first utterances simultaneously. When Rowena looks up to the transcript area, she sees Bethan's message and does not understand it, so she enters the canned phrase 'Please clarify what you mean' which is generated by a button marked 'Clarify'. Simultaneously, Bethan reads Rowena's message (2) and hits her canned phrase button 'Agree'.

Rowena is then confused about what Bethan means by ‘I agree’ as the preceding message was her request for clarification.
In a spoken conversation, Rowena and Bethan would have quickly corrected themselves if they began to speak at once, and the linearity would have reflected a *common* experience. The trouble is that the participants in the text-based conference each experienced the messages in a different order:
Rowena: 2 1 3 4 5 6
Bethan: 1 2 4 3 5 6

14.4.3 Turn-taking

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 cotemporality of the audio channel allows the other participants to negotiate the turn. Basically, whoever s peaks 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.

14.4.4 Context and deixis
We have seen how important context is in ordinary speech. 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’).
In

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, theobjects 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 methodsof linking the conversation to its context, either by embedding it within the objectsas annotations or by having hypertext links between the conversation and the object.

The trouble does not end with external context; there are also problems withdeictic reference to internal context. In speech, the context is intimately connectedto 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.

1. Alison: Brian’s got some lovely roses.
2. Brian: I’m afraid they’re covered in greenfly.
3. Clarise: I’ve seen them, they’re beautiful.

Brian and Clarise both reply to Alison’s message at the same time. However, inthe 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 companionplanting?’ Of course, the breakdown quickly becomes apparent in this case. Theproblem is not so much that people cannot recover from such breakdowns, as in theextra burden the recovery puts on the participants. If these messages are being sent,say, between continents, network delays and time differences may limit exchangesto once a day. Even one or two messages recovering from breakdown are then amajor disaster.

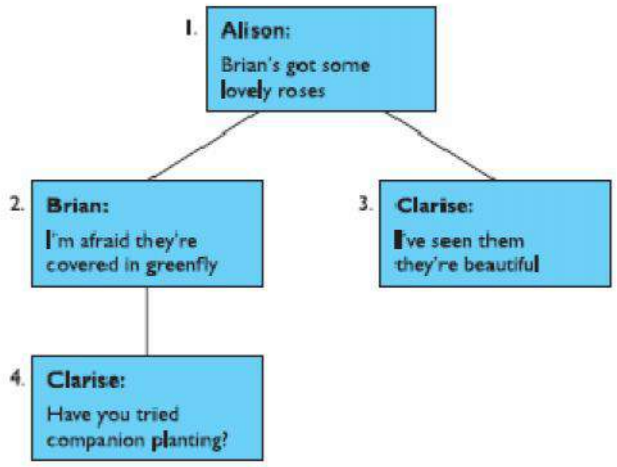

1. Alison:
Brian's got some lovely roses
2. Brian:
I'm afraid they're covered in greenfly
3. Clarise:
I've seen them they're beautiful
4. Clarise:
Have you tried companion planting?

Figure 14.3 Hypertext conversation structure

14.5.1 Group dynamics

Whereas organizational relationships such as supervisor/supervisee are relativelystable, the roles and relationships within a group may change dramatically within thelifetime of a task and even within a single work session. For example, studies of jointauthoring have found that roles such as author, co-author and commentator changethroughout the lifetime of a document [254, 295]. This means that systems, such asco-authoring systems, which use a formal concept of *role*, must allow these roles tochange together with the socially defined roles.Even the naming of roles can cause problems. A person may be an author of abook or paper, but never write the words in it, acting instead as a source of ideas andcomments. A particular case of this is the biographical story where the individualconcerned and a professional writer co-author the book, but only the professional author writes. A co-authoring system such as Quilt would call the non-writingauthor a 'commentator' or a 'reviewer', but *not* an 'author'. One can imagine someof the social friction such naming will cause.

Within the microcosm of group interaction, authority roles can be entirelyinverted. For example, if the managing director of a coal mining company visits thecoal face, he should act under the authority of the supervisor at the face, for his ownsafety and that of the mine. These inversions can cause problems even in computerfreesituations – it is hard for the supervisor to say 'No' to the MD. But, if a systemdemands an explicit controlling role, it is even harder for the manager to relinquishthis explicit role, even if in the context the subordinate should be in control.

14.5.2 Physical layout

In Section 14.2, we discussed the importance of eyegaze and gesture in face-to-facecommunication and how these help to mediate turn-taking. In particular, we notedin Section 14.2.3 that we must ensure that monitors do not block the participants'views of one another. In general, the physical layout of a room has a profound effectupon the working relationship of those in it. This is particularly obvious for meetingrooms, but should be considered in any group-working environment.As well as being unobtrusive, the orientation of computing equipment can affectgroup working. If we wish to encourage conversation, as we do in a meeting room,the participants must be encouraged to look toward one another.

Meeting roomshave a natural focus toward the screen at the front of the room, but inward-facing terminals can counteract this focus and thus encourage eye contact [226].The designers of Capture Lab, an eight-person meeting room, considered all thesefeatures and many other subtle effects. However, the users still had some difficultyin adapting to the *power positions* in the electronic meeting room. At first sight, theelectronic meeting room is not unlike a normal conference room. If the sharedscreen is a whiteboard or an overhead projector, then the most powerful position is
toward the front of the room (seats 1 or 6 in Figure 14.7). Managers would normallytake this seat as they can then easily move to the whiteboard or overhead projectorto point out some item and draw the group's attention.

14.5.3 Distributed cognition

In Chapter 1, we discussed human cognition, but the emphasis was, as in all traditional
psychology, upon the activity *within* the person's head. A school of thinkinghas recently developed which regards thinking as happening not just within the head,but in the external relationships with things in the world and with other people. Thisviewpoint is called *distributed cognition* [208, 185].

In fact, this viewpoint is not as radical as it first appears. Traditional viewstalk about the movement of information between *working memory* and *long-termmemory*: it is not so difficult then to regard bits of paper, books and computersystems as extensions to these internal memory systems. Similarly, many models ofhuman cognition regard the mind as a set of interacting subsystems (see Chapter 12):the step to regarding several people as involved in joint thinking is not difficult.

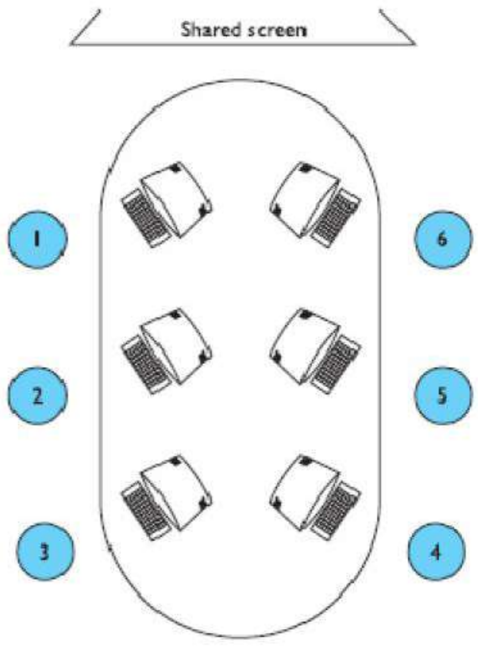


Figure 14.7 Meeting room layout

HYPERTEXT, MULTIMEDIA AND THE WORLD WIDE WEB
UNDERSTANDING HYPERTEXT
21.2.1 Hypertext definition – text, hypertext and multimedia

Although pictures and sculpture came first, it is text – the written word – that iscommonly seen as the defining point of civilization; pre-literate and oral societiesare often regarded as pre-history simply because they have no extant story. Hieroglyphics,Babylonian mud tablets, the Book of Kells, the Caxton press: these are thestepping-stones toward our current information-centric society.All these traditional texts share a common linear nature. Aristotle in his *Poetics* said that a story must have a beginning, a middle and an end, and even postmodernnon-linear narrative is actually written in a linear fashion even though the eventsmay not be causally connected.

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 creaturesin linear time. We are natural story-tellers and natural story-hearers. This iswhy, in Chapter 5, we found that scenarios were so powerful. The skill of the authoris in producing an experience for the reader that introduces new events and newconcepts so that they fit meaningfully into what has gone before.

There are many different ways of traversing the network, and so there are manydifferent ways of reading a hypertext document – the intention is that the user is able

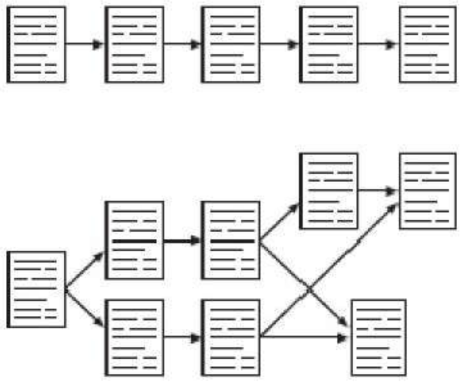


Figure 21.1 Typical structures of linear text and hypertext

to read it in the way that suits him best. Links can exist at the end of pages, with theuser choosing which one to follow, or can be embedded within the document itself.

For example, in an online manual, all the technical words may be linked directly totheir definitions in the glossary. Simply clicking on an unknown word takes the userto the relevant place in the glossary. Another unknown word encountered there canalso be traced back to its definition and then the user can easily return to his originalplace in the manual. The positions of these links are known as *hot-spots* since theyrespond to mouse clicks. Hot-spots can also be embedded within diagrams, picturesor maps, allowing the user to focus his attention on aspects that interest him.

Animation

Animation is the term given to the addition of motion to images, making themmove, alter and change in time. A simple example of animation in an interface is inthe form of a clock. Digital clocks can flick by the seconds, whilst others imitateSalvador Dali and bend and warp one numeral into the next. Analog clocks havemoving hour and minute hands, with an optional second hand sweeping round theclock face. Such a desktop accessory is found in a lot of interface setups, and theadditional processing time required to produce such effects is no longer a major
factor.

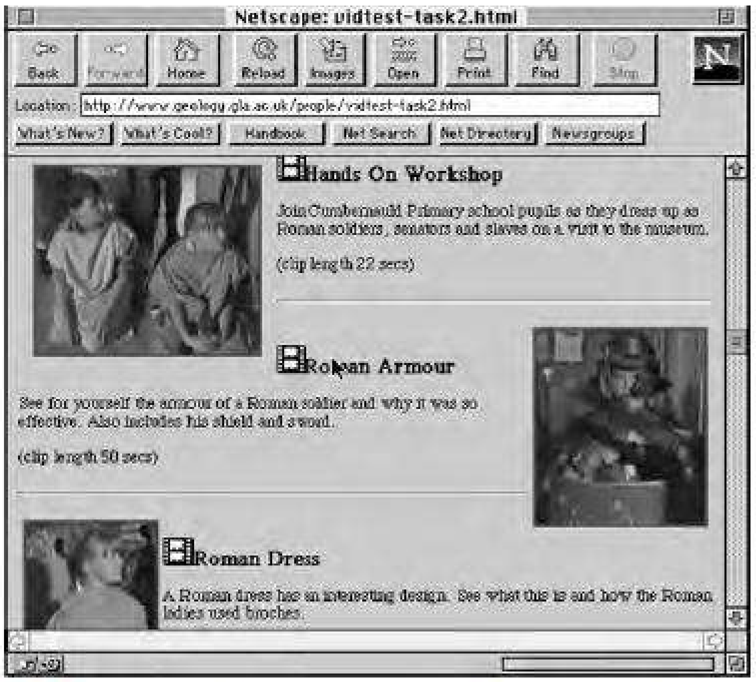
Another common use for animation in current windowing systems is to animatethe cursor. Instead of simply having a basic pointer always on the screen, many interfacesnow use the typical 16 16 bitmap that makes up the cursor to indicate morecomplex information. We saw in Chapter 3 that there are a number of different staticcursors in use, but animation takes this one stage further by adding motion to theimages. This is usually done to indicate that some process is in progress, to confirmto the user that something is actually happening. Animating the cursor means that
messages do not need to be printed out to a window, making it a neat and conciseway of presenting the desired information. On the Macintosh, work in progress isindicated by a watch icon, with the hands moving round and round, or by a spinningdisk. One system uses a stick

person, apparently doing weightlifting, to show thatheavy work is in progress, whilst another has an hourglass trickling down.

Video and audio

In a media dominated world, there are strong arguments for using video or audiomaterial as part of hypertext systems whether for education, entertainment or reference.Both audio and video material are expensive and time consuming to produce,but increasingly even home-PC systems include video and audio editing as standard.For example, the iMac includes a suite for editing video and burning DVDs.

Combined with digital video cameras these bring the production of audio/videomaterial into the reach of many. Furthermore, standard formats such as QuickTimeallow this material to be embedded in web pages for easy distribution. Of course,quality video production requires extensive experience, but then so does text!



Hands On Workshop

Join Cumbernauld Primary school pupils as they dress up as Roman soldiers, senators and slaves on a visit to the museum.

(clip length 22 secs)

Roman Armour

See for yourself the armour of a Roman soldier and why it was so effective. Also includes his shield and sword.

(clip length 50 secs)

Roman Dress

A Roman dress has an interesting design. See what this is and how the Roman ladies used broches.

Figure 21.2 Hunterian Museum – learning about the Romans. Source [194a] reprinted with permission of Springer-Verlag; Netscape browser window © 2002 Netscape Communications Corporation, used by permission

Audio material may be stored as sound samples, that is digital recordings, whichcan include instruments, voice or other sounds. As we are often more sensitiveto poor sound than poor video images, and sounds are, by definition, constantlychanging, it can be harder to *compress* sound than video. MP3, used for much webdownloadable music, recruits knowledge of the auditory system to achieve its highcompression. For example, if there are two simultaneous sounds, one a lot louderthan the other, the quieter one can be played with less fidelity because its details are masked by the louder sound. For pure instrumental music, MIDI files simply recordwhich instrument must play and the duration, pitch and loudness of each note –
basically a digital form of the sheet music.

Any sound in the interface can, of course, be potentially annoying, especially inopen-plan working environments, and so audio should be used with care and itshould be easy to mute. As these words are being written, the current author's wifeis struggling to turn off an annoying musical backing to a web page she is viewing!

Moving pictures are excellent at conveying information, and exert a quite hypnotichold over us; note the success of television. It is fair to say that many designpractices will have to be updated to make the most of the possibilities that thesetechniques offer; moreover, it is also the case that people are unsure how to get themost out of such technology. The techniques required to gain maximum benefitfrom moving images are very different from those that are used for static or minimalmotion displays, and designers do not have enough experience to start applying the
relevant technology at the relevant time. It may well be that computer interfacedesigners will have to study the techniques of the film makers and cartoonists beforethey start to discover the real benefits that these techniques can provide

21.2.3 Delivery technology

On the computer

Some hypertexts, in particular help systems, are downloaded or installed permanentlyon a computer. This has the advantage of instant access and such applicationsneed not use a standard viewer but may include their own bespoke browsing software.However, with media-rich hypertexts containing substantial graphics, videoand audio clips it may be impractical to store everything on hard disk. Also, for copyrightprotection, some systems will deliberately not allow themselves to be copiedfrom their original distribution media.

Many hypermedia systems are supplied on CD-ROM. This has the advantage ofreasonably large capacity (650–700 Mbytes), but access is slower than with installedsystems. For highly dynamic material, such as educational media, a special player isinstalled; alternatively, material such as software documentation may use a standardformat such as web pages.

On the web

Of course, 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 textcan be augmented through the use of hypertext links to other documents, whilegraphics can easily be incorporated as pictures, photographs, icons, page dividingbars, or backgrounds. Pages can also have hypertext links embedded into differentregions, which take the user to a different page or graphic if they are clicked on;these are known as active image maps. T

hese features allow web pages to becomeinteractive, acting as the interface to the information as well as its holder. Dynamicmaterial in the form of movies and sounds is also available to the designer; all thesefeatures push web page design well away from the conventional paper-based kind.However, the fact that the web has many more technological features than a book
doesn't mean that web pages are necessarily better than their paper counterparts.Also, the fact that a web page is packed full of features doesn't imply that it gets itsmessage across effectively. Designing web pages is a developing art, and should beviewed in much the same way as designing any other interactive system. Good pageshave been developed with the reader as the

focus, and act as effective interactiontools or presentation tools to allow the user to obtain the information he is looking
for most effectively.

21.2.4 Application areas

There are many applications of hypermedia, too many to describe in detail here.
However, it is worth noting the type of domains in which hypermedia systems have
proved successful, looking briefly at some example systems.

Rapid prototyping

Although now lacking the wealth of features expected of a hypermedia system,HyperCard on Macintosh computers has been very influential as a basis for experimentalhypertext systems. HyperCard uses the metaphor of a card index, aroundwhich the user can navigate. Each card can hold text, diagrams, photographs,

Cards may also contain forward and backward buttons and a home icon, to allow theuser to move sequentially and start from scratch respectively. HyperCard can be usedfor a range of applications including information management and teaching.However, HyperCard’s simple scripting language and easy to produce graphicalinterfaces meant it was also used extensively as a *rapid prototyping tool* for generatinginteractive systems. In fact, HyperCard stacks for both single-user and networkedapplications are available from the book website.

Help and documentation

Hypermedia systems are ideally suited to online manuals and other help systemapplications (see Chapter 10). They allow user-oriented access to the information,and support browsing. In addition the information can be organized hierarchically,with successive selections providing more detailed information. This supports thevarying needs that users have, such as quick reference, usage information, full detailsand so on. Many commercial help systems use hypermedia-style help. Good examplesare the Sun Guide system, HyperCard help and Microsoft Windows help (used bymany Windows applications).

Education and e-learning

Hypertext and hypermedia are used extensively in educational settings, as they allowvaried subjects to be related to each other in numerous ways so that the learner caninvestigate the links between different areas. In contrast to *computer-aided learning*(*CAL*) packages, hypermedia allows a student-controlled learning process.

An early example of a hypermedia learning environment that inspired many subsequentsystems was Intermedia [385]. This is a hypermedia system built and usedat Brown University to support teaching in subjects as varied as English literatureand biology. The system includes text, diagrams, photos and so on. Both learners andteachers can add information and links, giving students access to each other’s opinionsas well as those of their tutors. A map provides an overall view of the information fordirect access and navigation, with links providing browsing facilities in the normal
way. Intermedia has been successfully used for university-level teaching, and can be
seen as a forerunner of the educational resources now facilitated by the web.

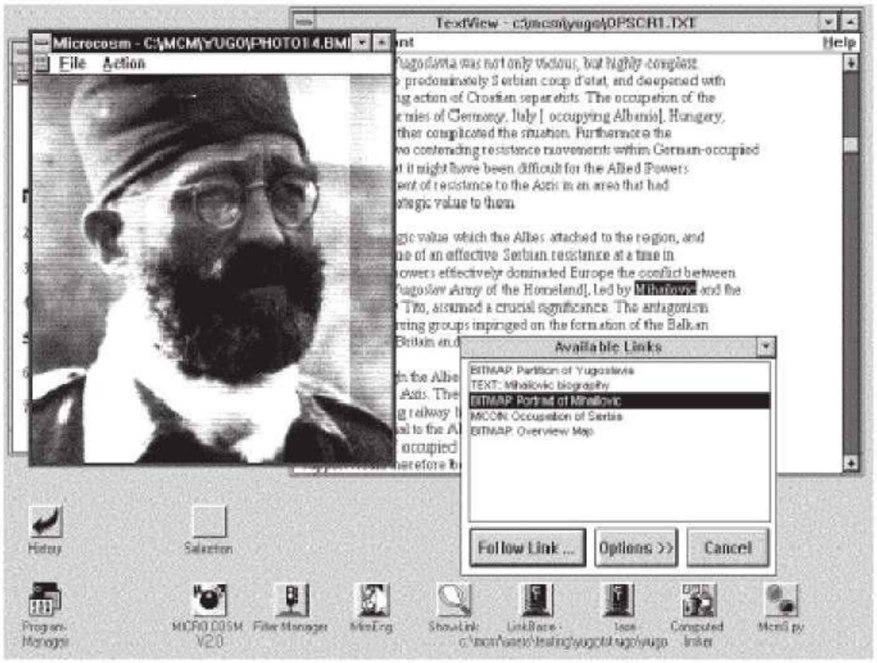
E-commerce
For some companies the web is simply another sales opportunity. Many readers willhave used online stores such as Amazon or bought from an auction site such as eBay.Hypertext's use of hierarchies, links, images and so on, makes it ideal for displayingcertain kinds of product. Actual buying and selling requires not only security at thelevel of the networks, websites, etc., but also trust. When you walk into a shop youcan see the person you are dealing with, and the fact that it is physically there today givesyou confidence that it will be there tomorrow if anything goes wrong.

FINDING THINGS

21.3.1 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 'lostin 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 beingdescribed, the writer knows what the reader has already seen. In a hypertext, thereader 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 anyassumption 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 mayhave a hierarchical or other structure, the user may navigate by hyperlinks that moveacross this main structure. It is easy to lose track of where you are and where youhave been.

The solution to the former issue is to design the information better. The solution tothe latter is to give users better ways of understanding where they are and of navigatingin the hypertext. To say 'the solution' is disingenuous – there is no simple 'solution'.If we want to provide information that allows complex, unplanned, non-linearaccess, there will probably always be problems.

21.3.2 Designing structure
We discussed the importance of good structural design in Chapter 5. Some of the task
analysis techniques presented in Chapter 15 may be useful in giving a task-oriented


Microcosm - C:\MCM\YUGO\PHOTO14.BMP

TextView - c:\mcm\yugo\OPSCR1.TXT

Yugoslavia was not only vicious, but highly complex
predominately Serbian coup d'etat, and deepened with
g action of Croatian separatists. The occupation of the
rmies of Germany, Italy [occupying Albania], Hungary,
ther complicated the situation. Furthermore the
wo contending resistance movements within German-occupied
t it might have been difficult for the Allied Powers
ent of resistance to the Axis in an area that had
ategic value to them.

gic value which the Allies attached to the region, and
e of an effective Serbian resistance at a time in
owers effectively dominated Europe the conflict between
Yugoslav Army of the Homeland], led by Mihailovic and the
Tito, assumed a crucial significance. The antagonism
ring groups impinged on the formation of the Balkan
Britain and

n the Allie
Axis. The
g railway l
al to the Al
occupied
herefore b

Available Links

BITMAP: Partition of Yugoslavia
TEXT: Mihailovic biography
BITMAP: Portrait of Mihailovic
MCCIN: Occupation of Serbia
BITMAP: Overview Map

Figure 21.4 Microcosm: an open hypermedia environment shown running a teaching application entitled 'The civil war in Yugoslavia 1941–45' (developed by the Departments of Electronics and Computer Science and History at the University of Southampton and used by permission)

or knowledge-orientated breakdown of a hypertext. In some areas there may be preexisting
understood structures to mirror; for example, the faculty and departmentalstructure of a university, or the main disciplines (circulatory, neurological, etc.)within medicine.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 touse 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 carcomponents in terms of physical location.

If multiple structures are used, you have to consider what to do about the commonmaterial. For example, if we examine a car hypermedia text under 'enginecompartment' and get to the fuel pump, this would also appear in the functionalview under 'fuel system'. Such common elements may be replicated. This has theadvantage that the material can be presented in ways that make sense given theircontext, but it can also lead to inconsistencies.

Alternatively we may make links across the hierarchy at some level; for example,the engine compartment may have a diagram of the engine with a labeled arrowsaying 'fuel pump (fuel system)', which takes you to the description of the pump inthe fuel system part of the hypertext.

Notice, too, the importance of making linksthat go to different parts of the hypertext very clear, following the ‘knowing whereyou are going’ principle from Chapter 5 (Section 5.6.1).
An in-between solution would be to have a small dedicated description of thefuel pump in the context of the engine compartment: perhaps describing how it isphysically connected to the engine block and how the piping is routed. This couldthen be linked to a more extensive explanation of its function in the fuel systemsection, using a ‘see also’ link.

21.3.3 Making navigation easier

No matter how well designed the site structure is, there will still be problems: becausethe user does not understand the structure; or because the user has individual needsthat 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 thecurrent 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 thehypertext; 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 canbe woven into the layout of the document; for example, some sites have an outlinerstylesidebar listing the main sections and drilling down to the current location.This acts both as an indication of where you are in the site (like the breadcrumbsdiscussed in Chapter 5) and as a constant reminder of the overall site structure.

Once information has been retrieved, a paper version is often needed. Printing adocument requires the pages to be in a particular order, but hypertext does notsupport 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 linearlylinked pages. In general, you should not rely on the print facility of a browser as thisis printing a page designed for on-screen viewing. You may notice websites offeringprinter friendly pages. These may be in a different format such as PDF, or maysimply be web pages without sidebar navigation aids, etc.

21.3.4 History, bookmarks and external links

Hypertext viewers and web browsers usually have some sort of history mechanismto 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 backbutton may be used where a user has followed a hyperlink and then decided it wasto the wrong place, or alternatively, when browsing back and forth from a centralpage 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 [63]. Other studies have shown frequent revisiting of pages during a single
browsing session [341].

Although the back button is used extensively, it is used relatively little to go backmore then one step. For error correction this makes sense, but for general revisitingone might think that moving

back several steps would be common. Possibly, onereason for this is confusion about the meaning of the back button; indeed a formalcomparison of back and history mechanisms in four different hypertext and webbrowsers found that the operation of back and history were subtly different in
each [104].

Framed websites are particularly difficult. The material you want to bookmark orlink to may be one of the frame content pages, but the URL you can see or bookmarkis that of the overall frameset. This may either discourage linking to the site or, ifcircumvented, 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 withinthe frameset. Search engines, too, may generate links to individual frames in a frameset.

Many web style guides heavily discourage the use of frames for this reason. If thesite is designed using a development tool that supports page templates, or is beingdynamically generated, there is rarely any need for frames as most of the effects canbe obtained using other page formatting.

21.3.5 Indices, directories and search

As well as 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 nota complete list of all words in a document. If this were the case then the index for thisbook would be as big as the rest of the book! The words in an index are chosenbecause they are significant key phrases or words with a domain meaning, and notevery 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 thepaper index you have to physically look up the page after finding the word in theindex, whereas in an electronic index the links are ‘live’ so you can simply click tothe content.

On the web an index would be very big (!); however, directory services such asYahoo! (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 simplyan alphabetical list of keywords, web directories give a hierarchical categorization tosites. The categorization is done either by self-submission, or, in the case of qualitydirectories 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 byusing 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 bemore 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 onthe underlying data rather than the generated pages.

WEB TECHNOLOGY AND ISSUES

21.4.1 Basics

The web consists of a set of protocols built on top of the *internet* that, in theory, allowmultimedia documents to be created and read from any connected computer in theworld. The web supports hypertext, graphics, sound and movies, and, to structureand describe the information, uses a language called *HTML* (hypertext markuplanguage) or in some cases, XML (extensible markup language). HTML is a markuplanguage that allows hypertext links, images, sounds and movies to be embeddedinto 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 aremany browsers, and each can interpret HTML in subtly different ways, or supportdifferent levels of functionality, which means that a web page viewed through onebrowser can look very different from the same page viewed through another. Theweb requires no particular multimedia capabilities from the machines that runthe 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*, *MicrosoftInternet Explorer* and *Opera*. These offer a graphical interface to the document, controlledby the mouse. Hypertext links are shown by highlighting the text that acts asthe link in an alternative color, and are activated by clicking on the link. A furthercolor is used to indicate a link that has already been visited. Hypertext links can alsobe embedded into regions within an image.

Although the browser contains most of the functionality required to view a webdocument, supporting text and graphics in an integrated package, special file formatsand media, including some movie formats, may require additional plug-ins or helperapplications.

As well as publishing personal, corporate and governmental information, the webis used as a source of entertainment, an advertising medium, a communication environment,and more. The vast and ever-increasing quantity of information availableon the web certainly exacerbates the user's 'lost in hyperspace' problems (Section21.3.1). But increased familiarity with hypertext, and better web page design, are aidingthe situation at least as much as technological efforts to create maps and indexesof the data. Another problem is that of information overload: multimedia images,gigabytes of graphics and mountains of text swamp the reader in a glitzy but unmoderated
world, in which the fact that almost anyone can make anything available leads
to the gemstones often being lost amongst the slag.

21.4.2 Web servers and web clients

Whereas a conventional PC program runs and is displayed on one computer, theweb is *distributed*. Different parts of it run on different computers, often in differentcountries of the world. They are linked, of course, by the internet, an enormousglobal computer network (see also Chapter 2, Section 2.9.3).

The pages are stored on web servers that may be on a company's own premisesor in special data centers. Because they are networked, the webmaster for a site canupload pages to the server from wherever she is. For example, the web pages forwww.hcibook.com are stored in a data center several thousand miles from where anyof the authors live!

Your machine, the PC running the web browser, is called a *client* because it wantsthe pages from the servers. When you click on a link your web browser works outthe full URL of the page it needs: say 'http://www.hcibook.com/e3/authors.htl'. Itsplits this into parts. The first part is the protocol 'http' which says how it talks to theserver (other alternatives include 'ftp'). The second part 'www.hcibook.com' is thehost name, that is the name of the web server containing the requested page. The lastpart '/e3/authors.html' gives the particular file on the site. The browser then establishesea 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, whichthen finds the requested html file (or image, or other file type) and returns it to thebrowser, which then displays it to you.

21.4.3 Network issues

The fact that the web is networked raises a series of issues that can impact onusability.
Network capacity is called *bandwidth*. This is a measure of the amount ofinformation that can pass down the channel in a given time. For example, a typicalmodem speed is 56 kbs – that is 56 kilobits per second. This equates to about6000 characters per second. This sounds fine until you realize that images may takemany tens or hundreds of characters (bytes) to encode . . . this is why many haverenamed the web the 'world wide wait'!

However, bandwidth is not the only important measure. There is also the time ittakes for a message to get across the network from your machine to the web serverand back. This delay is called *latency*. Latency is caused by several factors – the finitespeed of electrical or optical signals (no faster than the speed of light), and delays

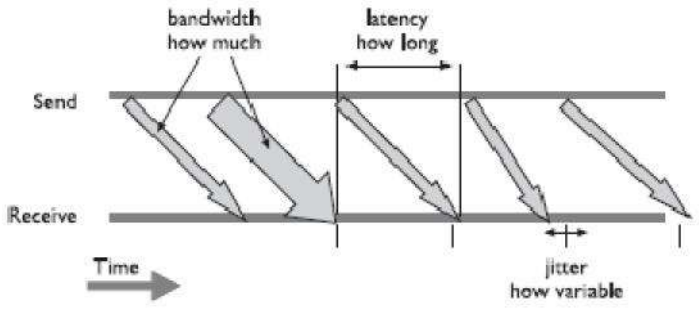


Figure 21.7 Bandwidth, latency and jitter

at routers along the way that take messages from one computer network and passthem on. This latency may not always be the same, varying with the exact routethrough 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).As well as the underlying latency and jitter of the network, each layer of network
software adds its own. The underlying internet protocols are *lossy*, that is messagescan be lost. When this happens, higher level software (the TCP – transmission controlprotocol – layer) notices and resends messages to give reliable communication.

These losses mean that the average delays increase, but also that the jitter increases –
a lost and resent message takes more time than one that gets there first time.There may also be an appreciable amount of time setting up a new connection,which may outweigh the time taken to actually send data. This is particularly a problemwith sites containing many small images.

STATIC WEB CONTENT

21.5.1

21.5.1 The message and the medium

One thing is often forgotten when web pages are created. It is of vital importance,
and hence will be discussed first.It is content.

Many people assume that because they can make information available on theweb, they should. Unfortunately, because it is very easy to publish information,much less care is taken with the actual content. Material may be nonsense, it may beincorrect, it may not read well, or be incomplete, or inane.Excellent page design can make useless material look attractive, but it still remainsuseless material. On the other hand, poor design can mean that excellent materialis never seen by potential readers, as they have become bored, or intolerant of themedium, or confused, or for a host of other reasons have aborted their attempts todownload and view the information.

Pages do have to look immediately interestingand attractive if people are to spend time, effort and, because of the communicationcosts, money, in viewing them; the user-centered nature of the medium makes thisimperative. This is in marked contrast to television or cinema or other dynamic
media, which are not under any direct user control, where information is presentedto a passive audience. With web documents, people have actually to want to see theinformation, and make an effort to retrieve it, which clearly must have an influenceon design.

21.5.2 Text

Because web pages are displayed on many different machines, there are only asmall set of fonts that can be guaranteed to be available: a standard font and a typewriterfont (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 orComic Sans are available on most web platforms. The difficult thing is to balancefine 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 creatematerial that will lay out passably on all platforms. But these offer a fairly coarse levelof control. The size and boldness of the heading should be chosen carefully; forexample, 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 wordprocessor or DTP package. However, care must be taken. For example, many pagesspecify fixed point sizes that may not display well on different platforms and canause 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 ofthe page will have problems with color, either because they are using older machineswith a limited color palette, or because they have some form of color blindness.Color, when used, should not be the only cue available. Users also bring a deeprootedemotional interpretation to colors; as we have seen, in some cultures, red isassociated with danger and anger, whilst green is regarded as go, or safe. Blue can bea cool color, orange a warm one, and so on.

Links usually change color once they have been accessed, providing cues to theuser about what material they have already explored. This means that two distinctbut still suitable colors need to be associated with each link, so that the system isacceptable whether or not the links have been activated. Note, too, that consistentuse of color can help the user understand the role of particular elements more intuitively,whereas color used for no clear purpose is often distracting.

21.5.3 Graphics
Obtaining graphics
There are a number of sites on the web that contain archives of graphical images,icons, backgrounds and so on. There are also paint and image manipulation packagesavailable 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 useshould be carefully thought out. Graphical images take longer to load than text, andthis may become a problem. Text uses 8 bits to represent a character: some roughcalculations show that approximately 2000 characters represent about a screenful ofinformation, and so 16,000 bits (2 K) are required. For graphics, one pixel may use8 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 toappear! Users become bored with operations that take a long time to complete, and
are unlikely to wait for ages while a page appears.

The GIF format also allows *animated GIFs*. These are a sort of mini-slide showor 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 overusedcan lead to very 'noisy'

pages.Active image maps are pictures with defined ‘hot’ areas, which, when clicked, executea particular script, usually calling another web page or graphic. The careful useof such maps can transform an interface, but there is an overhead to pay in loadingthe map and calling and running the script, and this should be considered carefullyby a page designer. Another characteristic of image maps is that there is rarely anyindication of which areas are active and which are not, and it can be difficult for users
to ensure that they have visited all the active regions. For accessing spatially organizedinformation, image maps are very suitable (for example, in a tourist informationsystem, the area about which information is available can be represented well),
but for information that does not have any clear spatial analog, their use is more
questionable.

Icons

Icons often appear on web pages, and while there are many available to choose from, they should be used with care. On web pages, icons are typically used in one of two ways. They are either visual cues, associating some small picture with different parts of the text (for example, some pages have icon-sized characters that appear next to instructions). Alternatively, they are used in much the same way as in a standard WIMP interface to represent aspects of the functionality of the underlying pages.

In this latter case, they must represent their associated functionality in either a concreteor an abstract form. This means that the design of the individual icon has to becarefully thought out, as a lot of information may have to be represented in a smallarea of screen estate. However, icons are rarely seen on their own, and when placednext to their neighbors, the whole effect has to be pleasing rather than disruptiveand garish. Therefore, the group of icons has to be designed together, with a coherentand recognizable style. The picture is broader than this, however: other applicationsalso use icons, which has its advantages and disadvantages. One advantage is
that certain icons are already associated with specific functionality (for example,a picture of a floppy disk to represent ‘save’). Disadvantages are that it restrictsthe individuality and style icon sets can show, and may mean that icons designedfor one purpose are misunderstood by users because they have seen somethingsimilar in another context. It is therefore vital that time is spent in examining the wayicons are used in other systems, before importing them into web pages or designingnew ones.

Icons sometimes appear for no apparent reason at all, when thepage creator has decided that as graphics are supported, a few shouldbe used. One interesting example is the icon that is a copy of theroadworks sign, used in conjunction with text saying something like‘This page still under construction!’.This is an interesting social effect brought on by the ease of web publishing –
incomplete (sometimes even non-existent!) information can be made immediatelyavailable. It’s like buying a map of the world nowadays and finding ‘here be dragons’around the edges because the geographer could not be bothered to draw all thecountries in. Most pages can be designed properly before they are made available,structured and presented in a complete and coherent way, allowing for extensionsand updates from the beginning.

There are times when the disclaimer ‘under construction’ has its uses: when criticallyimportant information becomes available, publishing it may well be moreimportant than presenting it (much like older maps with their dragons – if maps hadonly been printed once the whole world had been explored, civilization would bevery different today). There is, too, a sense in which web pages can be continually‘under construction’, changing, evolving and growing, because of their dynamic
nature and the ease with which they can be updated, but this does not obviate thedesigner’s responsibility to create pages that have both form and content.

Graphics and color
Using many different colors within graphics may well result in the browsers for oldermachines running out of entries in the colormap, with unpredictable consequences.This is often problematical as the browser may be running in tandem with othercolor 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 unlikelyto be a problem as home machines are often relatively recent. However, manybusinesses 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 iscomputer access it may well be through older or second-hand machines.

If universal access is required it is therefore still wise, where possible, to restrictimages to a limited number of colors, taken from the standard 216 color web palette,and to reduce complex color images to simpler approximations. Reducing thenumber of colors used also allows the depth of the images to be reduced; a changefrom 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 graphicsas they do to text.

21.5.4 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 andvideo clips, as they usually require some sort of multimedia capability on behalfof the host machine in order to be able to digitize sound and capture and digitizevideo. Video suffers from the same problems as graphics, magnified by an order ofmagnitude or two; it can take extremely large amounts of time for a video segmentto 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 allreceiving machines have the capability to play video, or sound, and so it is unwise fora designer to rely on these dynamic media to convey information without replicatingit elsewhere.

The use of sound and video moves page design further away from the typesetterand toward the sound engineer and cinematographer; the integration of thesecinematic media with the enhanced textual capabilities offered by the web is a newdomain, in which the techniques that work and those that fail have not yet been fullyexplored, let alone understood.

The need to download movies and sound (see Figure 21.8) puts sharp limits onthe length of clip that can be shown. Streaming media over the internet, such asRealVideo, RealAudio and CuSeeMe, allow potentially unlimited sources. As wellas longer prepared clips, these techniques allow live transmission (e.g. live radiobroadcasts over RealAudio) and long recorded sequences for asynchronous communication.An excellent use of the latter is the eClass project, introduced in Section

20.2.2, 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.

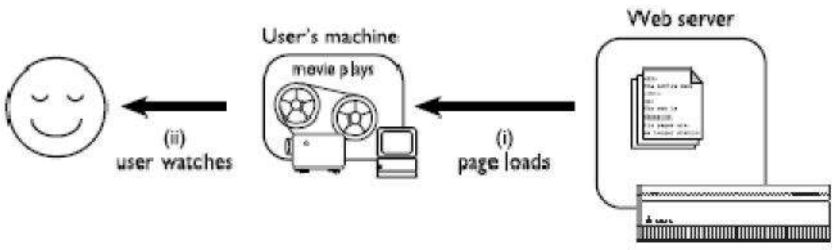


Figure 21.8 Animated GIF or movie needs to download completely

DYNAMIC WEB CONTENT

21.6.1 The active web

In the early days, the web was simply a collection of (largely text) pages linkedtogether. The material was static or slowly changing and much of it authored andupdated by hand. Some pages were generated on the fly, in particular the gatewaysinto ftp servers and to gophers, which were so important in adding 'free' content tothe web [97]. However, even here the user's model was still of a static repository ofinformation. Web surfers may not have always known where they were, but they hada pretty good idea of what they were seeing and that if they came back it would bethe same.

It was a pleasant, if somewhat boring world, but from a usability viewpoint it waswonderful – a consistent interface to terabytes of information. Who could ask formore? Indeed, this is one of the key arguments Nielsen brings against frames-richsites in his famous alertbox, *Why frames suck (most of the time)* [263] – frames breakthis simple user model and hence cause trouble. Nielsen calls for a new richer modelfor the web, which preserves the simplicity of the old model, but which can accommodateand guide the development of new features.

21.6.2 What happens where

When considering dynamic material on the web we need to take the external, user'sviewpoint and ask *what* is changing: media, presentation or actual data; by *whom*: bythe 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-browsingclient, 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 majorimpact on the pace of interaction, both *feedback*, how fast users see the effects of their
own actions, and *feedthrough*, 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 ischanging simply because it is the fundamental nature of the medium. It may bethe presentation or view the user has of the underlying content; for example, sortingby different categories or choosing text-only views for blind users. A specialform 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 contentchanges.

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 principalsources of change are the site author and the user. However, in complex sites usersmay see each other's changes – feedthrough.

How often? Finally, what is the pace of change? Months, days, or while you watch?We'll use the 'what changes?' categories as we examine alternatives and trade-offs inmore detail below. But first we also need to look at the technological constraints.

Technology and security

The fundamental question here is where 'computation' is happening. If pages arechanging, there must be some form of 'computation' of those changes. Where doesit 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 withdynamic 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 orone 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 thesame machine as they came from. This means that databases accessed from clientsideJDBC (Java database connectivity) must run on the web server (see below).

Another machine Although the pages are *delivered* from the web server, they maybe *constructed* elsewhere. For hand-produced pages, this will usually be on thepage author's desktop PC. For generated pages, this may be a PC or a centraldatabase server.
People Of course, as noted earlier, the process of production and update may even involve people!

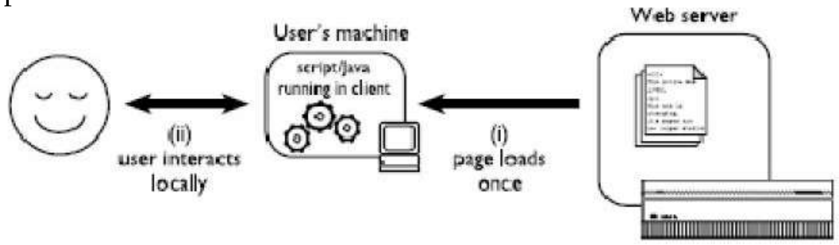


Figure 21.9 Java applet or JavaScript running locally

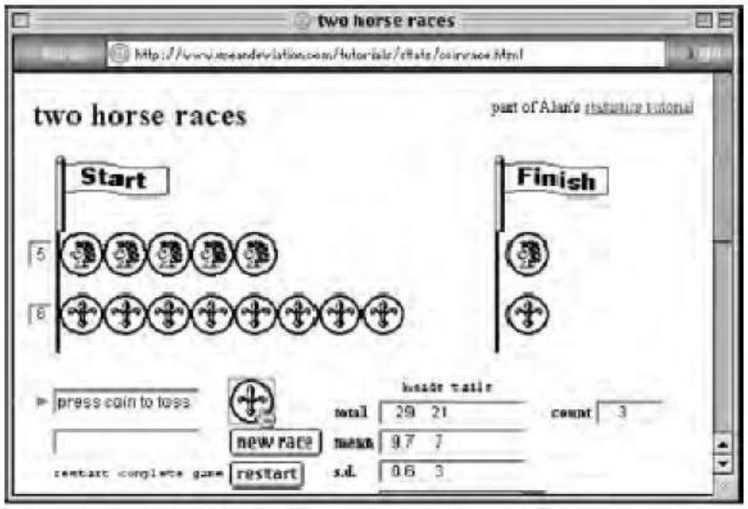


Figure 21.10 Simulated coin tossing using JavaScript. Screen shot frame reprinted by permission from Microsoft Corporation

21.6.4 Search

Some user-driven interaction can be accommodated at the client end, but not all.Consider search engines. It would be foolish to download several megabytes ofinformation so that a Java applet can search it online! Instead, all common websearch pages work by submitting forms to the server where CGI programs performthe searches and return results.

21.6.5 Automatic generation

It was evident in the earliest days of the web that a key problem for the future wouldbe maintenance. In the first rush of enthusiasm, individuals and organizationsproduced extensive and fascinating sites built largely of hand-crafted HTML. Notsurprisingly, vast areas of the web are not just static but in perpetual stasis. Websurfing sometimes seems not so much a watersport, but an exercise in archaeology.From the beginning it was clear that websites would eventually need to be createdfrom databases of content combined with some form of templates or layout description.

However, at that stage there were no tools available and those who saw thedatabase future used a variety of ad hoc methods.Happily, there are now a (sometimes bewildering) array of products for automatingweb production from existing and bespoke databases. These includevendor-specific products such as Oracle Web Server and Domino (for publishingLotus Notes), and also more general techniques such as using SQL (structured querylanguage) or JDBC to access databases from CGI scripts or even from running Javaapplets.
Database-generated websites have many advantages. They make use of existingdata sources. They guarantee consistency of different views of the data within the siteand between the site and the corporate data. They allow easy generation of tables ofcontents, indices, and inter-page links. They separate content and layout.The most dynamic way to get database content on the web is by accessing adatabase directly from a running applet.

The interface can then have any lookand-feel that can be programmed in Java and can allow very rapid interactionwith the user. The Java applet can establish an internet connection back to the webserver to access data files using HTTP (as if they were web pages), it can connectto a bespoke server (e.g. for chat type applications) or it can use standard databaseaccess methods. The latter would normally use JDBC, the Java database accesspackage. Using JDBC the applet can issue complex SQL queries back to the databasemeaning that some of the most complex work happens there (Figure 21.13).

In all cases, the Java security model built into most web browsers means that theapplet can only connect back to the machine from which it came. This means thatthe database server must run on the *same* machine as the web server.most browsers support forms, but some still do not support Java or scripting in aconsistent manner. The web search engine for this book works in this way. The user'skeywords 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 bookare returned (Figure 21.12). This also reminds us of another reason for not downloadingall the text to the user's machine – security; we don't want to distribute thefull electronic text for free!

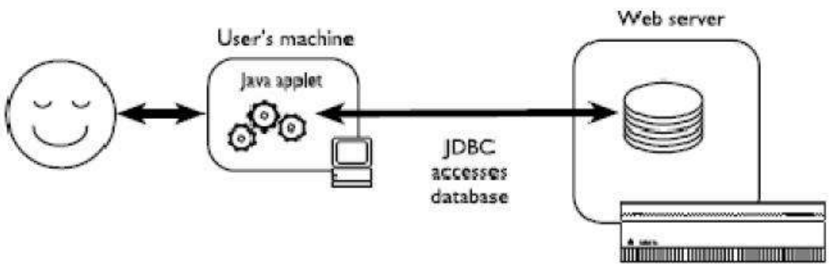


Figure 21.13 Java applet accesses database using JDBC

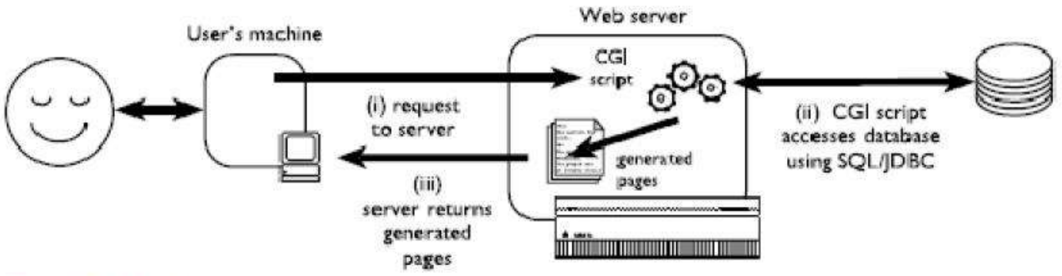

User's machine
Web server
CGI script
(i) request to server
(ii) CGI script accesses database using SQL/JDBC
generated pages
(iii) server returns generated pages

Figure 21.14 CGI script accesses database

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 21.14). 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 alimitation, but is at least consistent and means that it will work with virtually anybrowser. Java applets can offer more rapid surface interaction, but both have to waitfor the actual data to move between server and client. Of course, the pages generatedby a CGI script can themselves contain JavaScript or Java applets for local interaction,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 separatemachine (using standard database remote access methods or even a Java/JDBCCGI program), thus making the system more secure. However, the database cannotbe entirely secure – if the web-server machine is compromised the CGI scripts canbe altered to corrupt or modify the database! The special vendor-specific web serversare probably more secure as they don't require a standard web server to be running.

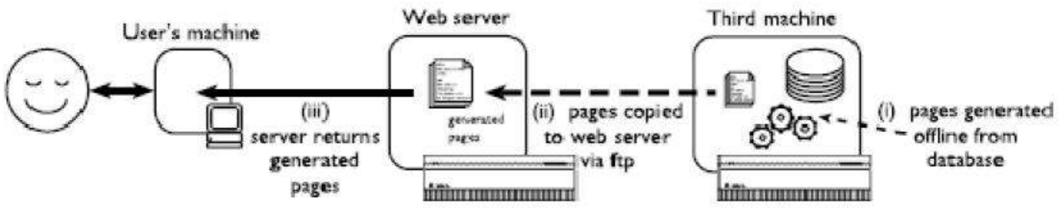
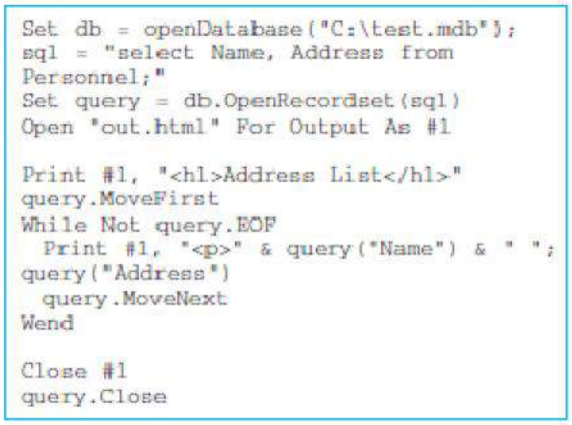

User's machine
Web server
Third machine
(iii) server returns generated pages
generated pages
(ii) pages copied to web server via ftp
(i) pages generated offline from database

Figure 21.15 Batch pre-generation of web pages

21.6.6 Batch generation

A low-tech but very secure solution is to generate pages offline from a databaseand then upload them to the web server (Figure 21.15). Many of Alan's earliest webpages were generated in this way from HyperCard stacks.This is certainly a simple solution as it separates out the task of page generationfrom that of page delivery. Pages can be generated directly using many standard database packages such as Access or HyperCard. Alternatively, standalone programsin languages such as Visual Basic, Java or C can access a database and output HTMLpages. These programs can run on a central computer, or on your own PC. Thegenerating program simply produces a set of HTML pages on your own disk that canbe checked locally and then copied onto the web server using ftp or shared networkdisks. Many people think that this will be difficult, but in reality it is remarkablyeasy, as you can use the tools you are used to – if you can create a text file you can
create HTML. In fact, the snippet of Visual Basic in Figure 21.16 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
```

Figure 21.16 Visual Basic code to generate a web page

21.6.7 Dynamic content

The mechanisms we have been discussing manage the feedthrough when the databaseis updated by some non-web means. Perhaps the most 'active' web pages are thosewhere 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/JDBCmethod or the CGI method, the same mechanisms can as easily be used to updateas to access the database. The feedback of changes to the user is thus effectively
instantaneous – 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 itchange from free to booked on the web page.

This sort of web application opens up many additional problems. You may needto add some form of login or authentication. If credit card numbers are supplied youneed to ensure that the web server is secure. Also, without care it is easy to designsolutions that accidentally book multiple seats if the user presses the back button andends up on what appears to be a simple confirmation screen.

If we consider an estate agent's web page, with houses for sale and potentialbuyers, the situation is rather different. The pace of change is slow; house purchasestake place over days, weeks and months. A solution that automatically marked ahouse as sold would neither be necessary nor desirable! In this case a socio-technicalsolution using low-tech database generation would probably be sufficient. The webpage can have a contact telephone number, email address or message form. Queriesvia these channels (as well as non-web-based queries) come to the estate agent who

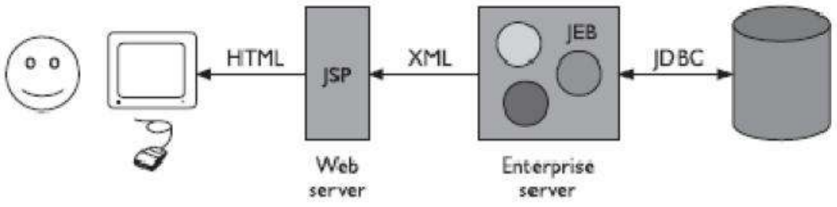


Figure 21.17 *n*-tier architecture

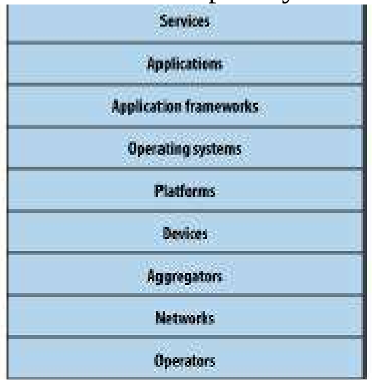
is

responsible for deciding when to mark the house 'sold'. There is a continuousloop between web user and the database, but it involves human as well as automaticprocesses.Going in the direction of greater complexity, many business applications operatean *n*-tier web architecture. This involves multiple layers of software where the outerlayers are concerned more with the user interface and the inner layers more withbusiness functionality. Figure 21.17 shows this using several web standards. The userinteracts through a web browser with a web server. The pages are generated usingJava Servlet Pages (JSP). To generate the page the servlets connect to Java Enterprise
Beans (JEB) on an enterprise server. These are components that encapsulate 'businesslogic'. For example, in a banking system this could include rules on whether aparticular transaction is allowed. These Java Enterprise Beans draw their data fromthe corporate database using JDBC connections.

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.

Mobile Ecosystem :

Think of the mobile ecosystem instead as a system of layers, as shown in Figure 2-1.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 whether we expressly put them there.

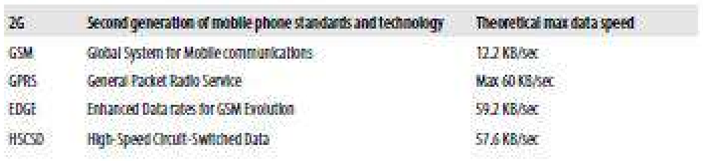


Operators
The base layer in the mobile ecosystem is the *operator*. Operators go by many names, depending on what part of the world you happen to be in or who you are talking to. Operators can be referred to as Mobile Network Operators (MNOs); mobile service providers, wireless carriers, or simply carriers; mobile phone operators; or cellular companies. In the mobile community, we officially refer to them as operators, thoughin the United States, there is a tendency to call them *carriers*. Operators are what essentially make the entire mobile ecosystem work. They are the gatekeepers to the kingdom. They install cellular towers, operate the cellular network,make services (such as the Internet) available for mobile subscribers, and they oftenmaintain relationships with the subscribers, handling billing and support, and offeringsubsidized device sales and a network of retail stores. The operator's role in the ecosystem is to create and maintain a specific set of wireless services over a reliable cellular network.

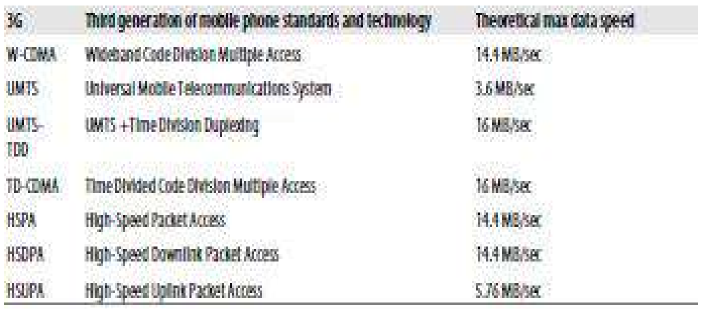
Networks
Operators operate wireless networks. Remember that cellular technology is just a radiothat receives a signal from an antenna. The type of radio and antenna determines thecapability of the network and the services you can enable on it.You'll notice that the vast majority of networks around the world use the GSM standard(see Table 2-2 for an explanation of these acronyms), using GPRS or GPRS EDGE for2G data and UMTS or HSDPA for 3G. We also have CDMA (Code Division MultipleAccess) and its 2.5G hybrid CDMA2000, which offers greater coverage than its morewidely adopted rival. So in places like the United States or China, where people

aremore spread out, CDMA is a great technology. It uses fewer towers, giving subscribersfewer options as they roam networks.

Table 2-2. GSM mobile network evolutions



2G	Second generation of mobile phone standards and technology	Theoretical max data speed
GSM	Global System for Mobile communications	12.2 KB/sec
GPRS	General Packet Radio Service	Max 60 KB/sec
EDGE	Enhanced Data rates for GSM Evolution	59.2 KB/sec
HSCSD	High-Speed Circuit-Switched Data	57.6 KB/sec



3G	Third generation of mobile phone standards and technology	Theoretical max data speed
W-CDMA	Wideband Code Division Multiple Access	14.4 MB/sec
UMTS	Universal Mobile Telecommunications System	3.6 MB/sec
UMTS-TDD	UMTS + Time Division Duplexing	16 MB/sec
TD-CDMA	Time Divided Code Division Multiple Access	16 MB/sec
HSPA	High-Speed Packet Access	14.4 MB/sec
HSDPA	High-Speed Downlink Packet Access	14.4 MB/sec
HSUPA	High-Speed Uplink Packet Access	5.76 MB/sec

Platforms

A mobile platform's primary duty is to provide access to the devices. To run softwareand services on each of these devices, you need a *platform*, or a core programminglanguage in which all of your software is written. Like all software platforms, these aresplit into three categories: licensed, proprietary, and open source.

Licensed

Licensed platforms are sold to device makers for nonexclusive distribution on devices.The goal is to create a common platform of development Application ProgrammingInterfaces (APIs) that work similarly across multiple devices with the least possibleeffort required to adapt for device differences, although this is hardly reality. Followingare the licensed platforms:

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 platformand provides a collection of Java APIs for the development of software for resourceconstraineddevices such as phones.

Binary Runtime Environment for Wireless (BREW)

BREW is a licensed platform created by Qualcomm for mobile devices, mostly forthe U.S. market. It is an interface-independent platform that runs a variety of application frameworks, such as C/C++, Java, and Flash Lite.

Windows Mobile

Windows Mobile is a licensable and compact version of the Windows operatingsystem, combined with a suite of basic applications for mobile devices that is basedon the Microsoft Win32 API.

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.

Application Frameworks

Often, the first layer the developer can access is the application framework or API released by one of the companies mentioned already. The first layer that you have anycontrol over is the choice of application framework.Application frameworks often run on top of operating systems, sharing core services
such as communications, messaging, graphics, location, security, authentication, andmany others.

Java

Applications written in the Java ME framework can often be deployed across the majorityof Java-based devices, but given the diversity of device screen size and processorpower, cross-device deployment can be a challenge.Most Java applications are purchased and distributed through the operator, but they
can also be downloaded and installed via cable or over the air.

S60

The S60 platform, formerly known as Series 60, is the application platform for devicesthat run the Symbian OS. S60 is often associated with Nokia devices—Nokia owns theplatform—but it also runs on several non-Nokia devices. S60 is an open source framework.S60 applications can be created in Java, the Symbian C++ framework, or even FlashLite.

BREW

Applications written in the BREW application framework can be deployed across themajority of BREW-based devices, with slightly less cross-device adaption than otherframeworks.However BREW applications must go through a costly and timely certification processand can be distributed only through an operator.

Flash Lite

Adobe Flash Lite is an application framework that uses the Flash Lite and ActionScriptframeworks to create vector-based applications. Flash Lite applications can be runwithin the Flash Lite Player, which is available in a handful of devices around the world.Flash Lite is a promising and powerful platform, but there has been some difficulygetting it on devices. A distribution service for applications written in Flash Lite is long

Windows Mobile

Applications written using the Win32 API can be deployed across the majority of WindowsMobile-based devices. Like Java, Windows Mobile applications can be downloadedand installed over the air or loaded via a cable-connected computer.

Cocoa Touch
Cocoa Touch is the API used to create native applications for the iPhone and iPodtouch. Cocoa Touch applications must be submitted and certified by Apple beforebeing included in the App Store. Once in the App Store, applications can be purchased,downloaded, and installed over the air or via a cable-connected computer.

Android SDK
The Android SDK allows developers to create native applications for any device thatruns the Android platform. By using the Android SDK, developers can write applicationsin C/C++ or use a Java virtual machine included in the OS that allows the creationof applications with Java, which is more common in the mobile ecosystem.

Web Runtimes (WRTs)
Nokia, Opera, and Yahoo! provide various Web Runtimes, or WRTs. These are meantto be miniframeworks, based on web standards, to create mobile widgets. Both Opera'sand Nokia's WRTs meet the W3C-recommended specifications for mobile widgets.Although WRTs are very interesting and provide access to some device functions usingmobile web principles, I've found them to be more complex than just creating a simplemobile web app, as they force the developer to code within an SDK rather than justcode a simple web app. And based on the number of mobile web apps written for theiPhone versus the number written for other, more full-featured WRTs, I don't thinkI'm alone in thinking this. Nonetheless, it is a move in the right direction.

WebKit
With Palm's introduction of webOS, a mobile platform based on WebKit, and givenits predominance as a mobile browser included in mobile platforms like the iPhone,Android, and S60, and that the vast majority of mobile web apps are written specificallyfor WebKit, I believe we can now refer to WebKit as a mobile framework in its ownright.WebKit is a browser technology, so applications can be created simply by using webtechnologies such as HTML, CSS, and JavaScript. WebKit also supports a number ofrecommended standards not yet implemented in many desktop browsers.Applications can be run and tested in any WebKit browser, desktop, or mobile device.

Types of Mobile Applications
Mobile Application Medium Types
The *mobile medium type* is the type of application framework or mobile technology thatpresents content or information to the user. It is a technical approach regarding whichtype of medium to use; this decision is determined by the impact it will have on theuser experience. The technical capabilities and capacity of the publisher also factor intowhich approach to take.Earlier I discussed the common mobile platforms in terms of how they factor in thelarger mobile ecosystem. Now we will look deeper into each of these platforms from amore tactical perspective, unpacking them, so to speak, to see what is inside.Figure 6-1 illustrates the spectrum

of mobile media; it starts with the basic text-basedexperiences and moves on to the more immersive experiences.

SMS

The most basic mobile application you can create is an SMS application. Although itmight seem odd to consider text messages applications, they are nonetheless a designed experience. Given the ubiquity of devices that support SMS, these applications can be useful tools when integrated with other mobile application types. Typically, the user sends a single keyword to a five-digit short code in order to return information or a link to premium content. For example, sending the keyword “freebie”to a hypothetical short code “12345” might return a text message with a coupon codethat could be redeemed at a retail location, or it could include a link to a free ringtone.SMS applications can be both “free,” meaning that there is no additional charge beyond the text message fees an operator charges, or “premium,” meaning that you are chargedan additional fee in exchange for access to premium content.

Pros

The pros of SMS applications include:

• They work on any mobile device nearly instantaneously.
• They’re useful for sending timely alerts to the user.
• They can be incorporated into any web or mobile application.
• They can be simple to set up and manage.

Cons

The cons of SMS applications include:

• They’re limited to 160 characters.
• They provide a limited text-based experience.
• They can be very expensive.

Mobile Websites

As you might expect, a mobile website is a website designed specifically for mobiledevices, not to be confused with viewing a site made for desktop browsers on a mobilebrowser. Mobile websites are characterized by their simple “drill-down” architecture,or the simple presentation of navigation links that take you to a page a level deeper, asshown in Figure 6-3.

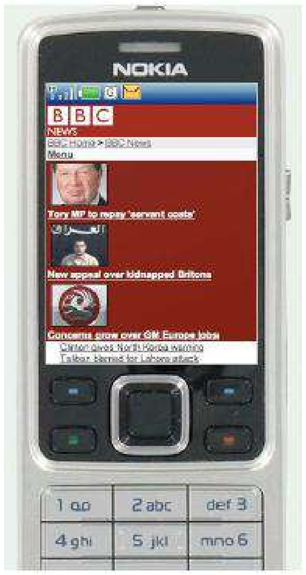

NOKIA
BBC
NEWS
Tory MP to repay 'servant costs'
New appeal over kidnapped Britons
Concerns grow over GM Europe jobs
Clinton gives North Korea warning
Taliban blamed for Lahore attack

Figure 6-3. An example of a mobile website

Mobile websites often have a simple design and are typically informational in nature,offering few—if any—of the interactive elements you might expect from a desktop site.Mobile websites have made up the majority of what we consider the mobile web forthe past decade, starting with the early WML-based sites (not much more than a list oflinks) and moving to today's websites, with a richer experience that more closely resemblesthe visual aesthetic users have come to expect with web content.

Pros

The pros of mobile websites are:

• They are easy to create, maintain, and publish.

• They can use all the same tools and techniques you might already use for desktop sites.

• Nearly all mobile devices can view mobile websites.

Cons

The cons of mobile websites are:

• They can be difficult to support across multiple devices.

• They offer users a limited experience.

• Most mobile websites are simply desktop content reformatted for mobile devices.

• They can load pages slowly, due to network latency.

Mobile Web Widgets

The ever-trusty Wikipedia defines a web widget this way:

A portable chunk of code that can be installed and executed within any separate HTMLbased 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. I think one reason for all the confusionaround what is a mobile web widget is that this definition can also encompassany web application that runs in a browser. Opera Widgets, Nokia Web RunTime(WRT), Yahoo! Blueprint, and Adobe Flash Lite are all examples of widget platformsthat work on a number of mobile handsets. Using a basic knowledge of HTML (or vector graphics in the case of Flash), you can create compelling user experiences thattap into device features and, in many cases, can run while the device is offline.

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.

Mobile Web Applications

Mobile web applications are mobile applications that do not need to be installed orcompiled 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 webbrowser. By "application-like" experience, that they do not use the drill-downor page metaphors in which a click equals a refresh of the content in view. Web applicationsallow users to interact with content in real time, where a click or touch performsan action within the current view.The history of how mobile web applications came to be so commonplace is interesting,and is one that I think can give us an understanding of how future mobile trends canbe assessed and understood. Shortly after the explosion of Web2.0, web applicationslike Facebook, Flickr, and Google Reader hit desktop browsers, and there was discussionof how to bring those same web applications to mobile devices. The Web 2.0movement brought user-centered design principles to the desktop web, and those sameprinciples were sorely needed in the mobile web space as well.With the introduction of the first iPhone, we saw a cataclysmic change across the board.Using WebKit, the iPhone could render web applications not optimized for mobiledevices as perfectly usable, including DHTML- and Ajax-powered content. Developersquickly got on board, creating mobile web applications optimized mostly for the iPhone(Figure 6-5). The combination of a high-profile devce with an incredibly powerfulmobile web browser and a quickly increasing catalog of nicely optimized experiences created the perfect storm the community had been waiting for.

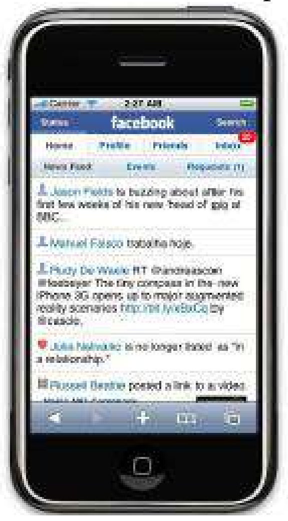


Figure 6-5. The Facebook mobile web app

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, location lookup, filesystem access, camera, and so on.

Native Applications
The next mobile application medium is the oldest and the most common; it is referred

to as native applications, which is actually a misnomer because a mobile web app ormobile web widget can target the native features of the device as well. These applicationsactually should be called “platform applications,” as they have to be developedand compiled for each mobile platform.These native or platform applications are built specifically for devices that run the platformin question. The most common of all platforms is Java ME (formerly J2ME). Intheory, a device written as a Java ME MIDlet should work on the vast majority of featurephones sold around the world. The reality is that even an application written as a JavaME MIDlet still requires some adaptation and testing for each device it is deployed on.

Figure 6-6. A native application in the iPhone

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.

Games
The final mobile medium is games, the most popular of all media available to mobiledevices. Technically games are really just native applications that use the similar platformSDKs to create immersive experiences (Figure 6-7). But I treat them differentlyfrom native applications for two reasons: they cannot be easily duplicated with webtechnologies, and porting them to multiple mobile platforms is a bit easier than typicalplatform-based applications.

Although you can do many things with a powerful mobile web browser, creating animmersive gaming experience is not one of them—at least not yet. Seeing as how wehave yet to see these types of gaming experiences appear on the desktop using standardweb technologies, I believe we are still a few years out from seeing them on mobiledevices. Adobe's Flash and the SVG (scalable vector graphics) standard are the onlyway to do it on the Web now, and will likely be how it is done on mobile devices in thefuture, the primary obstacle being the performance of the device in dealing with vector graphics.The reason games are relatively easy to port ("relatively" being the key word), is that the bulk of the gaming experience is in the graphics and actually uses very little of the device APIs. The game mechanics are the only thing that needs to adapted to the various shops that can quickly take a game written in one language and port it to another.These differences, in my mind, are what make mobile games stand apart from all otherapplication genres—their capability to be unique and difficult to duplicate in anotherapplication type, though the game itself is relatively easy to port. Looking at this modelfor other application areas—namely, the mobile web—could provide helpful insight into how we create the future of mobile web applications.

Figure 6-7. An example game for the iPhone

Pros

The pros of game applications are:

• 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.

Mobile Information Architecture

What Is Information Architecture?

Before we get into the specifics of mobile information architecture, let's first talk aboutexactly what information is. I can think of no better definition than the seminal O'Reillybook *Information Architecture for the World Wide Web* (*http://oreilly.com/catalog/*

9780596527341/) by Morville and Rosenfeld, otherwise known in information architect circles as the “polar bear” book. This definition outlines the following:• 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 find ability • An emerging discipline and community of practice focused on bringing principlesof design and architecture to the digital landscapeSimilar to how mobile technology has many facets, so does information architecture, as it is often used as an umbrella term to describe several unique disciplines, includingthe following:

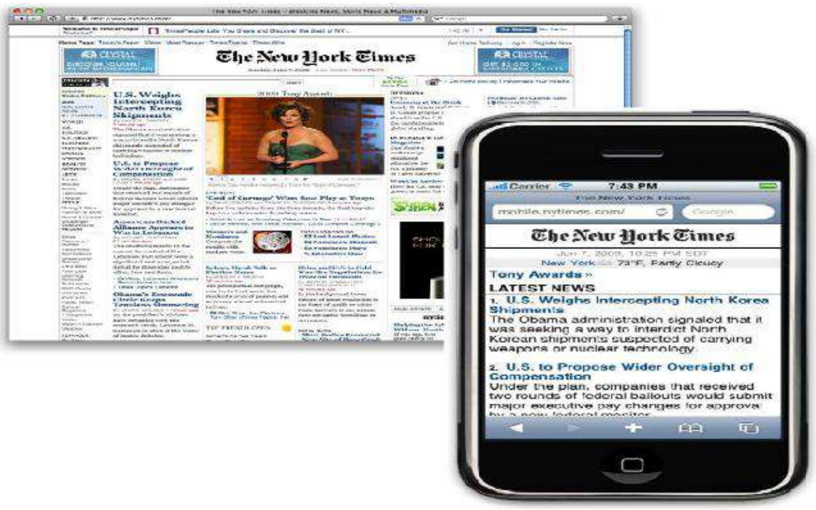
Information architecture
The organization of data within an informational space. In other words, how theuser will get to information or perform tasks within a website or application.
Interaction design
The design of how the user can participate with the information present, either in a direct or indirect way, meaning how the user will interact with the website ofapplication to create a more meaningful experience and accomplish her goals.

Information design
The visual layout of information or how the user will assess meaning and directiongiven the information presented to him.

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.

Interface design
The design of the visual paradigms used to create action or understanding.

Mobile Information Architecture
Although information architecture has become a common discipline in the web industry,unfortunately, the mobile industry—like software—has only a handful of specialized mobile information architects. Although mobile information architecture is hardlya discipline in its own right, it certainly ought to be. This is not because it is so dissimilarfrom its desktop cousin, but because of context, added technical constraints, andneeding to display on a smaller screen as much information as we would on a desktop.For example, if we look at the front page of *http://www.nytimes.com* as seen from adesktop web browser compared to how it may render in a mobile browser (Figure7-2), we see a content-heavy site that works well on the desktop, and is designedto present the maximum amount of information above the “fold” or where the screen cuts off the content. However, in the mobile browser, the text is far too small to beuseful.


The New York Times
7:43 PM
mobile.nytimes.com/
Tony Awards »
LATEST NEWS
1. U.S. Weighs Intercepting North Korea Shipments
2. U.S. to Propose Wider Oversight of Compensation

The role of a mobile information architect would be to interpret this content to themobile context. Do you use the same structure, or sections? Do you present the sameinformation above the fold? If so, how should that be prioritized? How does the usernavigate to other areas? Do you use the same visual and interaction paradigms, or inventnew ones? And if you do start to invent new paradigms, will you lose the visual characteristicsof what users expect? These are only some of the questions you have to ask yourself when starting to createa mobile information architecture. As you can see in Figure 7-3 there are several differentways that the *New York Times* has been interpreted for the mobile context.

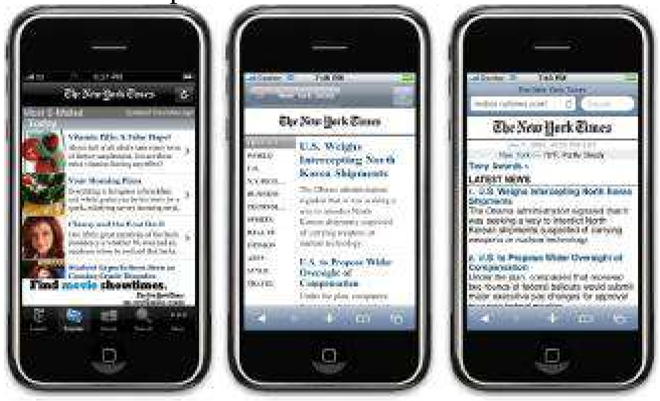


Figure 7-3. The many mobile experiences of the New York Times

Keeping It SimpleWhen thinking about your mobile information architecture, you want to keep it assimple as possible.Support your defined goalsIf something doesn't support the defined goals, lose it. Go back to your user goals andneeds, 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 you understand that, it is a simple process of reverseengineeringthe path from where they want to be to where they are starting. Cut out everything else—your site or application doesn't need it. For example, to get some news and information on a mobile device, you need to first ask what the goal is. What is the need you are trying to fill? Then you need to apply context. Where are your 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, you get your informationarchitecture.Clear, simple labels Good trigger labels, the words we use to describe each link or action, are crucial inMobile. Words like "products" or "services" aren't good trigger labels. They don't tellus anything about that content or what we can expect. Now, I would argue that goodtrigger labels are crucial in the Web as well, that we've become lazy and we assume somuch about the user that we ignore the use of good trigger labels. Users have a muchhigher threshold of pain when clicking about on a desktop site or application, huntingand pecking for tasty morsels. Mobile performs short, to-the-point, get-it-quick, andget-out types of tasks. What is convenient on the desktop might be a deal breaker onmobile.Keep all your labels short and descriptive, and never try to be clever with the wordsyou use to evoke action. The worst sin is to introduce branding or marketing into yourinformation architecture; this will just serve to confuse and distract your users. Executiveslove to use the words they use internally to external communications on websitesand applications, but these words have no meaning outside of your company walls. Ifthe user is just trying to get music, don't call it "My Music," "My MP3s," or somethingmade up that only strokes our corporate egos, such as "AudioJams™"—just call it"Music."Site MapsThe first deliverable we use to define mobile information architecture is the site map.Site maps are a classic information architecture deliverable. They visually represent therelationship of content to other content and provide a map for how the user will travelthrough the informational space, as shown in Figure 7-4.Mobile site maps aren't that dissimilar from site maps we might use on the Web. Butthere are a few tips specific to mobile that we want to consider.

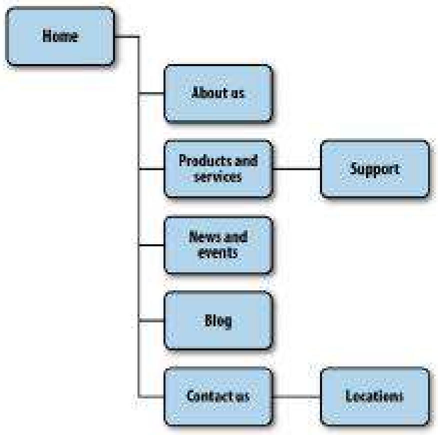


Figure 7-4. An example mobile site map

Limit opportunities for mistakes

In Figure 7-5, you 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.

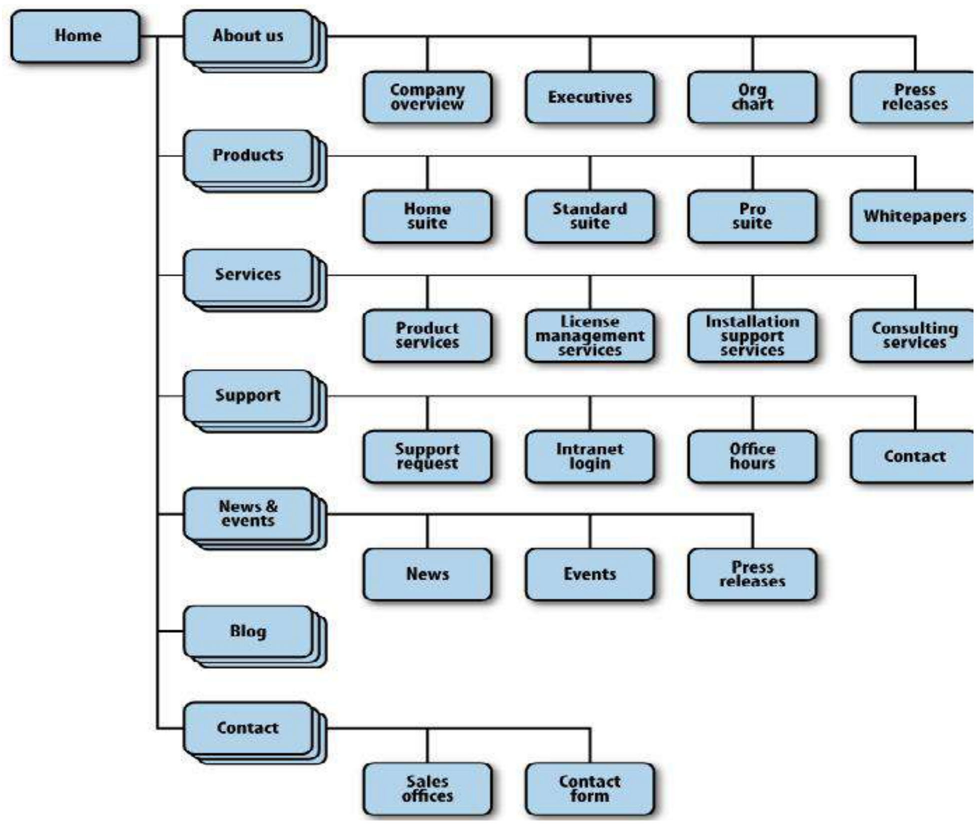


Figure 7-5. An example of a bad mobile information architecture that was designed with desktop usersin mind rather than mobile users

But in mobile, we cannot make this assumption. In the mobile context, tasks are shortand users have limited time to perform them. And with mobile websites, we can'tassume that the users have access to a reliable broadband connection that allows themto quickly go back to the previous page. In addition, the users more often than not haveto pay for each page view in data charges. So not only do they pay cash for viewing thewrong page by mistake, they pay to again download the page they started from: we can't assume that pages will be cached properly.
Confirm the path by teasing content

After the users have selected a path, it isn't always clear whether they are getting towhere they need to be. Information-heavy sites and applications often employ nestedor drill-down architectures, forcing the user to select category after category to get totheir target. To reduce risking the user's time and money, we want to make sure wepresent enough information for the user to wade through our information architecturesuccessfully. On the Web, we take these risks very lightly, but with mobile, we mustgive our users a helping hand. We do this by teasing content within each category—that is, providing at least one content item per category.In order to make sense of a vast inventory of content, we have to group, subgroup, andsometimes 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 applicationwith content, the dreaded “Page 1 of 157” appears. What user would ever sit there with a mobile device and page through 157 pages of ringtones? What user wouldpage through five pages of content?On an early site I worked on, users would flip through a few pages of content, then giveup or go back and visit another area. We could see a direct relationship to the numberof 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 7-6, you can see in a constrained screen that teasing the first few items of thepage provides the user with a much more intuitive interface, immediately indicatingwhat type of content the user can expect.

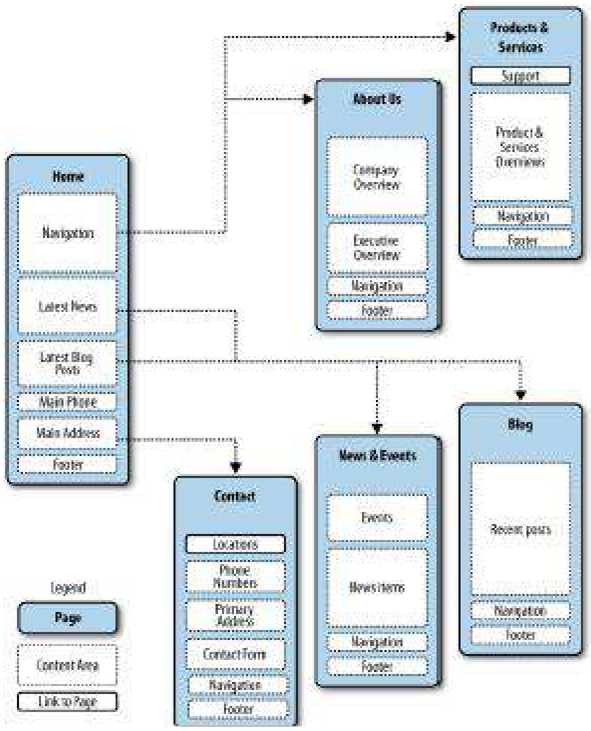
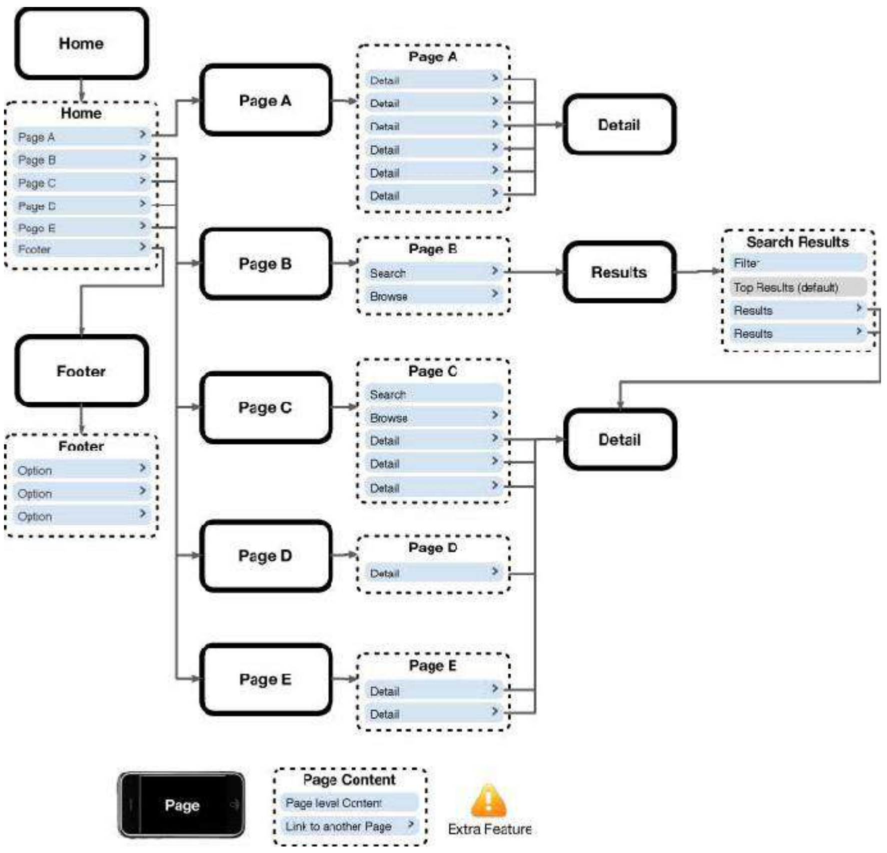


Figure 7-6. Teasing content to confirm the user's expectations of the content within

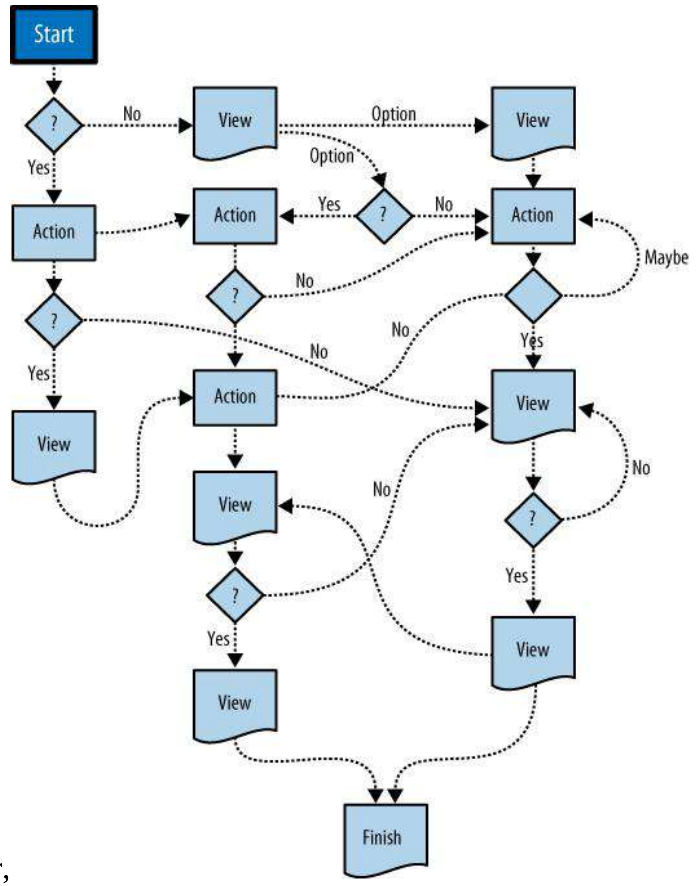
Clickstreams

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 datagathered from server logs. Clickstreams are usually historical, used to see the flaws inyour information architecture, typically using heat-mapping or simple percentages toshow where your users are going. I've always found them to be a useful tool for rearchitectinglarge websites. However, information architecture in mobile is more like software than it is the Web,meaning that you 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 pathusers will take gives you a holistic or bird's-eye view of your mobile information architecture,just as a road map does. When you can

see all the paths next to each otherand take a step back, you start to see shortcuts and how you can get users to their goal
faster or easier, as shown in Figure 7-7.



Now the business analyst in you is probably saying, "Just create user or process flows," such as the esoteric diagram shown in Figure 7-8, which is made up of boxes anddiamonds that look more like circuit board diagrams than an information architecture.If that is what your team prefers, then by all means, flow away. Personally, I like topresent all of my information architecture deliverables from the perspective of the



user,

Figure 7-8. An example process flow diagram

using the same metaphors she will use to make her way through my information architecture—in this case, either a screen or page metaphor.A good architect's job is to create a map of user goals, not map out every technicalcontingency or edge case. Too often, process flows go down a slippery slope of addingevery project requirement, bogging down the user experience with unnecessary distractions,rather than focusing on streamlining the experience. Remember, 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. Anything that distracts us from that goal is just a distraction.

Wireframes

The next information architecture tool at our disposal is wireframes. *Wireframes* are away to lay out information on the page, also referred to as *information design*. Site mapsshow how our content is organized in our informational space; wireframes show howthe user will directly interact with it. Wireframes are like the peanut butter to the sitemap jelly in our information architecture sandwich. It's the stuff that sticks. Wireframeslike the one in Figure 7-9 serve to make our information space tangible and useful.

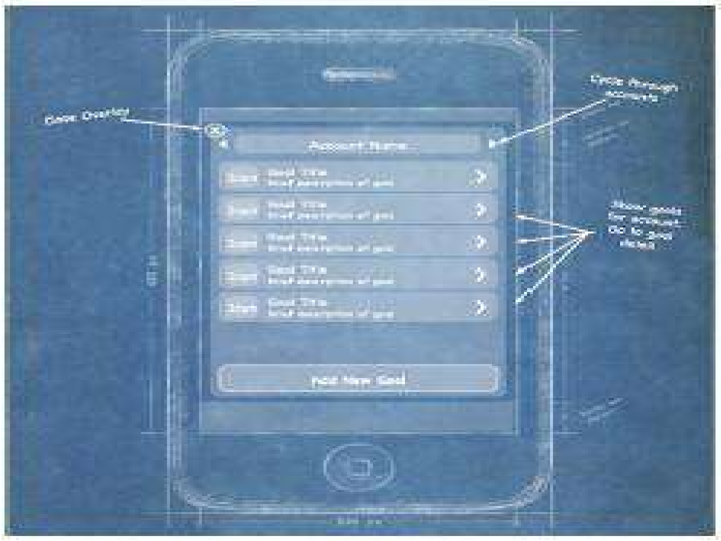


Figure 7-10. Using annotations to indicate the desired interactions of the site or application

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 theexperience. How do we lay out our navigation? What visual or interaction metaphorswill we use to evoke action? What are the best ways to communicate and show informationin the assumed context of the user? These questions and many more are answeredwith wireframes. Although I've found wireframes to be one of the most valuable information deliverablesto communicate my vision for how a site or app will work, the challenge is that adiagram on a piece of paper doesn't go a long way toward describing how the interactionswill work. Most common are what I call "in-place" interactions, or areas wherethe 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 copious amountsof annotation, describing each content area in as much length as you can fit in the margins of the page, as shown in Figure 7-10.

At this point, I highly recommend that you get some feedback from either others onyour project or my most trusted reviewer, my wife. Well, not *my* wife, but someoneyou know and trust—and the less technical, the better. Have her review your work asyou iterate through ideas. Describe what problems you are trying to solve and ask herwhat she is thinking. It has been an invaluable process for me over the years, not specificallyfor the feedback I receive—though you do get the occasional strokes ofgenius—but for forcing me to communicate what I've done and verbalize my thoughts.You'd be surprised how often you get an idea in your head that you think is brilliant,but once you say it out loud, it just sounds absurd.

What Is Mobile 2.0?

In autumn 2006, I was asked to help design and build a website for the first Mobile 2.0 conference, happening a few days before O'Reilly's Web 2.0 Summit. The event wasput together by several Mobile Monday organizers. Mobile Monday is a series of mobilesocial gatherings that happen all over the world on the first Monday of the month. Theto attend the Web 2.0 Summit in order to discuss the future of mobile development.

By the time the event arrived, I had become frustrated with the term Web 2.0. By thistime, it had devolved from a great idea to jargon that people would casually toss about.Everyone wanted “Web 2.0,” but no one understood what it meant. Needless to say,I was skeptical about anything labeled 2.0. So I went to the first Mobile 2.0, curious about exactly what my fellow speakers thought Mobile 2.0 actually meant.By the end of the day, I had a loosely defined picture of what Mobile 2.0 was, and forthe first time in a long time I was excited about the future of mobile development. Isaw a groundswell from the people at the conferences to finally overcome the industrybureaucracy and technical problems that had gone unsolved for too long. I saw notonly what mobile technology could be, but people willing to make it happen.In an interesting twist of foreshadowing, the first Mobile 2.0 event occurred only a fewmonths before the announcement of the first iPhone—a device that would come to symbolize many of the things we talked about that day.

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 mobileweb has always been viewed as a second-class citizen within the mobile ecosystem, formany 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 mobileweb by its users to levels never seen before. The spur of new mobile web apps createdjust for the iPhone doubled the number of mobile websites available in under a year.

Mobile Web Applications Are the Future

Creating mobile web applications instead of mobile software applications has remainedan 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 marketquickly and at little cost. They see the success of small iterative development cycles andwant to apply this to mobile development, something that is not that feasible in thetraditional mobile ecosystem.Developers have been keen for years to shift away from the costly mobile applicationsthat are difficult to publish through the mobile service provider, require massive testingcycles 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 thismodel that an app sold through a marketplace can earn huge profits if promoted correctly.Being *promoted correctly* is the key phrase. What gets promoted and why is anebulous process with no guarantees. One thing is certainly true: 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 providesany 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 onmodern 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 your app 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.

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 you wanted a good experience, you built a native app.

Descriptions within the industry range from the honest "the mobile user experience is utterly horrid," to the sales pitch of "look at these cool things you can do," to the optimistic "the mobile user experience is the future!" These polar attitudes toward the mobile user experience are somewhat ironic, given that the mobile user experience waslargely ignored for close to a decade. People in mobile treat the user experience like achicken-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 you to use their models of user experience, meaning that youhave to work in the constraints of the API. This is good for creating consistent experiencesfor that platform, but these experiences don't translate to others. For example,would you take an iPhone app design and put it on an Android device? The user experiencefor these devices is similar but still remains different.

Mobile Widgets Are the Next Big Thing

At many Mobile 2.0 events, I've heard a lot of buzz about mobile widgets, though noone can tell me how mobile widgets would define a mobile widget, or how they are different from mobile web apps. The consensus seems to be that the solution for the challenges with the mobile web is to create a series of "small webs" targeted at a specificuser or task. Though I couldn't figure out the problem being solved with these widgets,I had to admit that they looked pretty cool.Don't get me wrong. I believe that the concept of small network-enabled applicationsis very promising, but the mobile industry tends to take promising ideas like this, inflateexpectations to unsustainable levels, then abandon them at the first sign of trouble orsacrifice them for the next big thing, whichever happens first.

Carrier Is the New "C" Word

I noticed a strong tendency over the years for people in the mobile industry to avoid uttering the word "carrier." Even the more European equivalent term "operator" seems to be on the decline. To give you an example of how much carriers are hated: you can have entire conversations with people who work for the big operators and even they will avoid using the term in conversation.It is almost like when you get a bunch of mobile experts in a room for a day: they wantto pretend that operators don't exist at least for one day. Maybe they prefer to see afuture with no mobile service providers at all. I think more likely the case is that the industry has finally figured out that very few can make a profit when your business relies on carriers. Though the “C” word isn’t uttered often, it still is the 800-poundgorilla in the room. 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

For years, I’ve sat on the line between the mobile community and the web community. They have treated each other almost like rivals. I’ve been frustrated with both sides,but 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, you have some incredibly intelligent people who have been innovating amazing products under insane constraints for years. On the web side, you 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. Onthe 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.”Don’t get me wrong—I think that the mobile guys are very smart and great people; some of them I consider to be close friends. They have to deal with really hard problems that would make a web professional give up to go serve coffee. But I’m a very patient and tolerant person, and I have to admit that these same people are some of the most difficult people I’ve ever worked with.

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 oftoday’s market don’t view themselves as consumers, but rather as creators. But before we get into that, let’s back up for a minute.The web is about content. Sure, there are programming languages, APIs, and other technical underpinnings, but what do you do when you open a web browser? You 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, givingindividuals 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, microbloggig, 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 portabledevices 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.

Elements of the Mobile Design

Context

Make sure you do your
homework to answer the following 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 your design. Treat these uestions as a checklist to your design from start to finish. They can provide not only great inspiration for design challenges, but justification for your design decisions later.

Message

Another design element is your message, or what you are trying to say about your site or plication visually. One might also call it the “branding,” although I see branding and messaging as two different things. Your message is the overall mental impression you create explicitly hrough visual design. I like to think of it as the holistic or at times instinctual reaction someone will have to your design. If you take a step back, and look at a design from a distance, what is your impression? Or conversely, look at a design for 30 seconds, and then put it down. What words would you use to describe the experience?

Branding shouldn’t be confused with messaging. Branding is the impression your company
name and logo gives—essentially, your 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 your 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?”

A “heavy” design with use of dark colors and lots of graphics will tell the user to expect something more immersive.

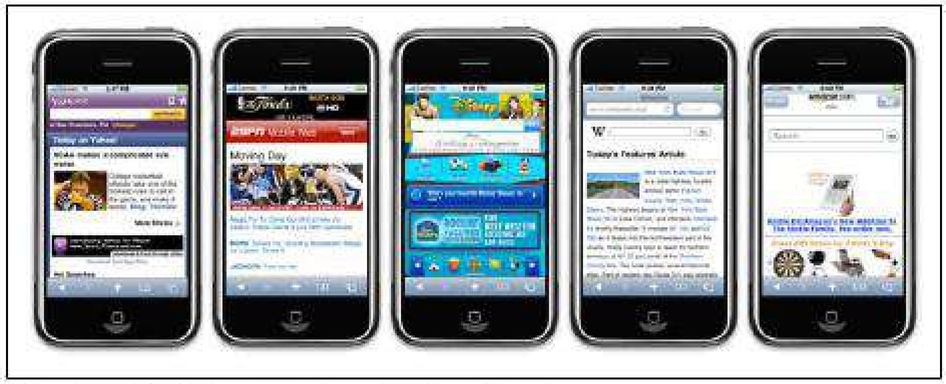


Figure 8-4. What is the message for each of these designs?

Which of the following designs provide a message? What do they say to you?

Yahoo!

Yahoo! sort of delivers a message. This app provides a clean interface, putting a focus on search and location, using color to separate it from the news content. But I'm not exactly sure what it is saying. Words you might use to describe the message are crisp, clean, and sharp.

ESPN

The ESPN site clearly is missing a message. It is heavily text-based, trying to put a lot of content above the fold, but doesn't exactly deliver a message of any kind. If you took out the ESPN logo, you likely would have indifferent expectations of this site; it could be about anything, as the design doesn't help set expectations for the user in any way. Words you might use to describe the message: bold, cluttered, and content-heavy.

Disney

Disney creates a message with its design. It gives you a lot to look at—probably too much—but it clearly tries to say that the company is about characters for a younger audience. Words you might use to describe the message: bold, busy, and disorienting.

Wikipedia

The Wikipedia design clearly establishes a message. With a prominent search and text-heavy layout featuring an article, you know what you are getting with this design. Words you might use to describe the message: clean, minimal, and text-heavy.

Amazon

Amazon sort of creates a message. Although there are some wasted opportunities above the fold with the odd ad placement, you can see that it is mostly about products (which is improved even more if you scroll down).

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?

I think of 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.

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 detailed Human 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 that Apple 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.

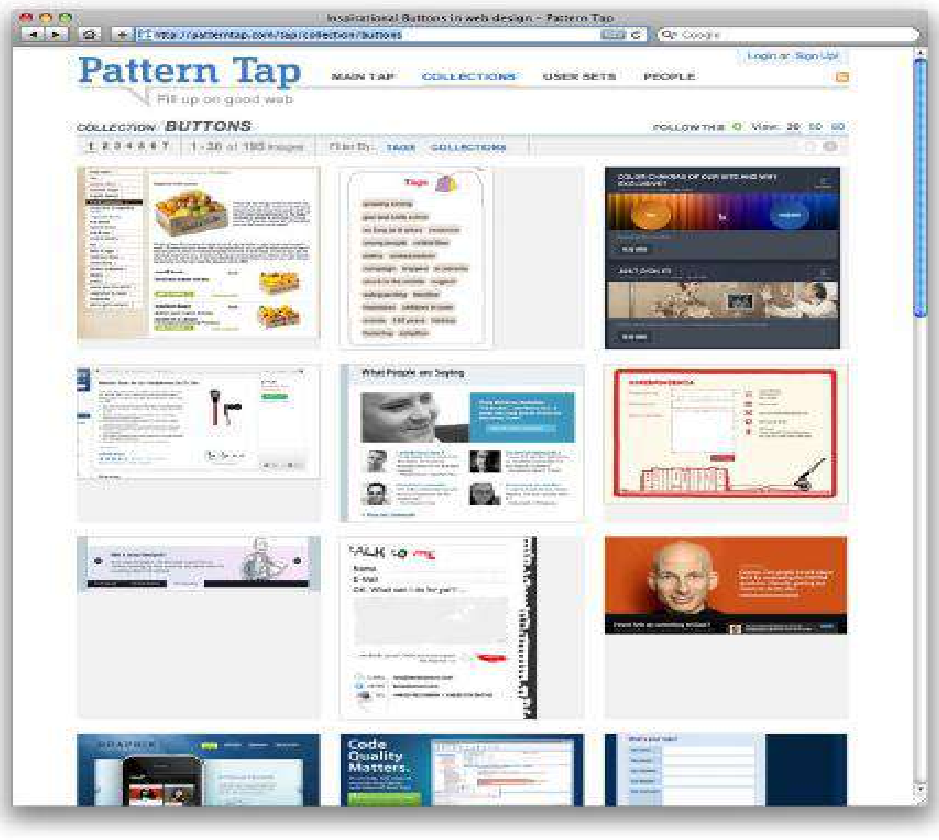


Figure 8-5. Pattern Tap shows a number of user interface patterns that help to establish look and feel

Different layouts for different devices

The second part of layout design is how to visually represent content. In mobile design, the primary content element you deal with the is navigation. Whether you 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 you support.

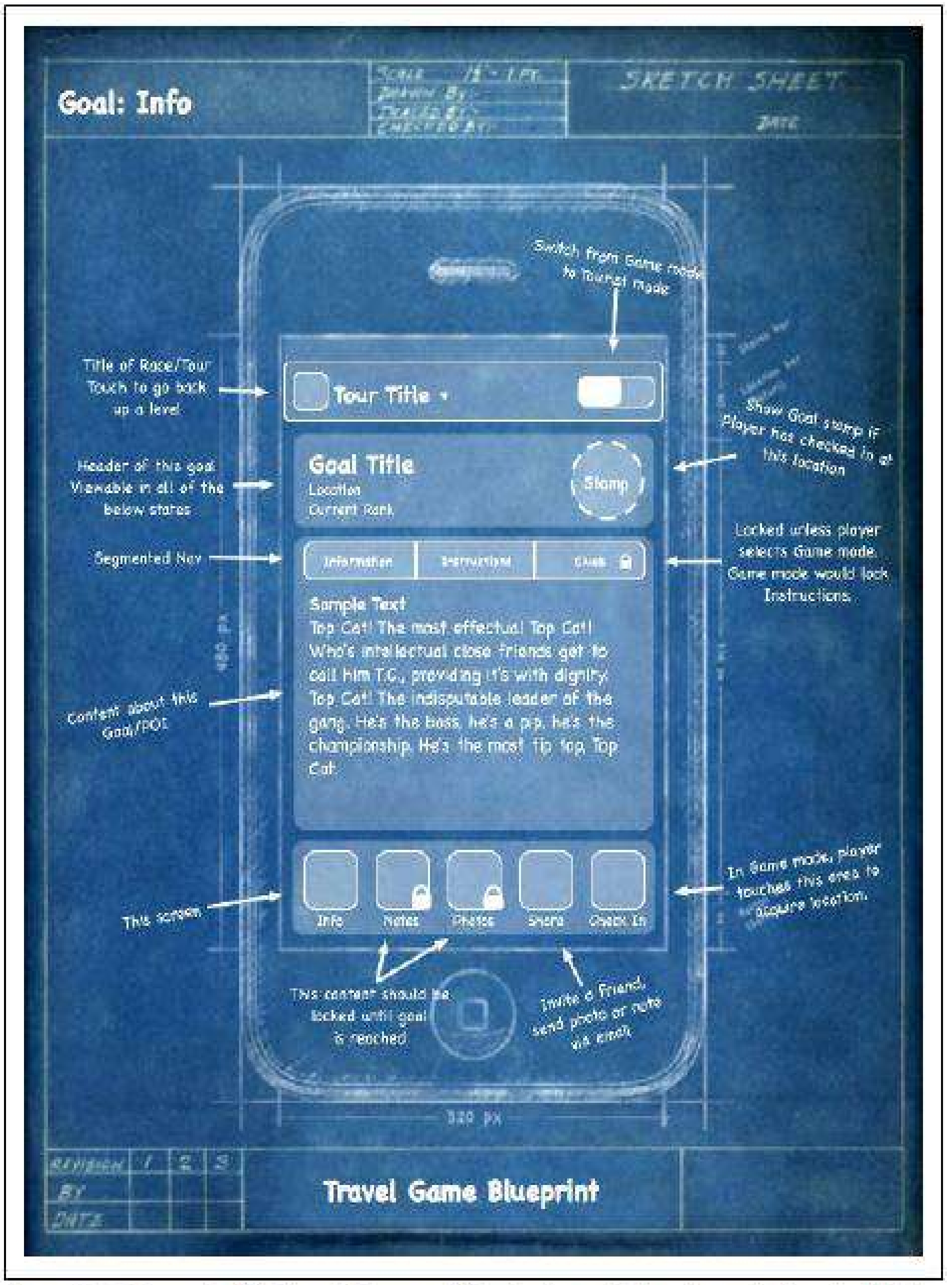

Goal: Info
SKETCH SHEET
Switch from Game mode to Tourist mode
Title of Race/Tour Touch to go back up a level
Tour Title
Header of this goal Viewable in all of the below states
Goal Title
Location
Current Rank
Stamp
Show Goal stamp if Player has checked in at this location
Segmented Nav
Information
Instructions
Clues
Locked unless player selects Game mode. Game mode would lock Instructions.
Sample Text
Top Cat! The most effectual Top Cat! Who's intellectual close friends get to call him T.C., providing it's with dignity. Top Cat! The indisputable leader of the gang. He's the boss, he's a pip, he's the championship. He's the most tip top, Top Cat.
Content about this Goal/POI
This screen
Info
Notes
Photos
Share
Check In
In Game mode, player touches this area to acquire location.
This content should be locked until goal is reached
Invite a Friend, send photo or note via email
320 px
480 px
REVISION 1 2 3
BY
DATE
Travel Game Blueprint

Figure 8-7. Using a low-fidelity wireframe to define the layout design element before visual design begins

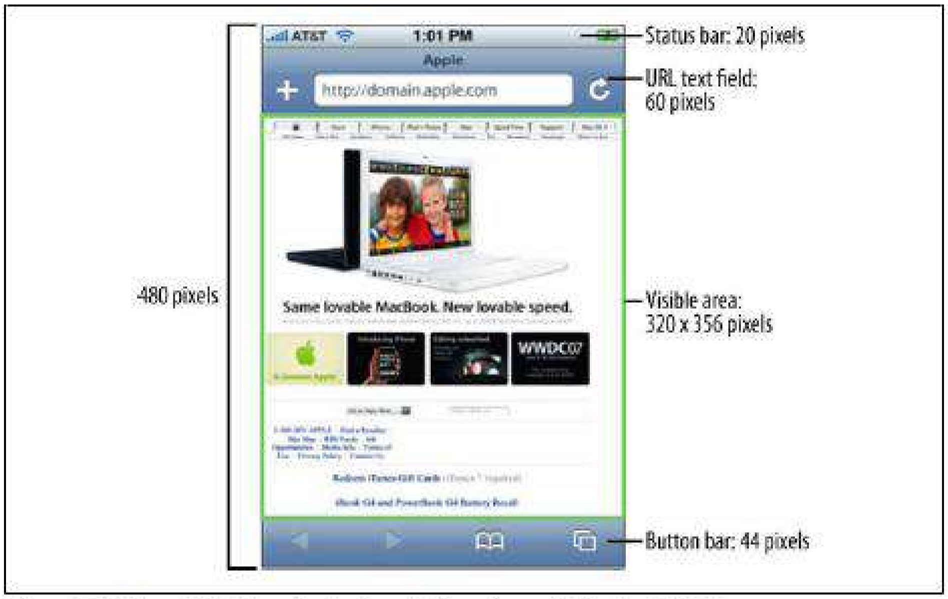


Figure 8-8. iPhone HIG, showing the layout dimensions of Safari on the iPhone

This is the opposite of the scroll navigation type, where the device's D-pad is used to go 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 8-9, you can actually see by the bold outline that the first item selected on the screen is the link around the logo.

When dealing with scroll navigation, you also have to make the choice of whether to display navigation horizontally or vertically. Visually, horizontally makes a bit more sense, but when you 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 my advice is just to try and keep it as simple as possible.

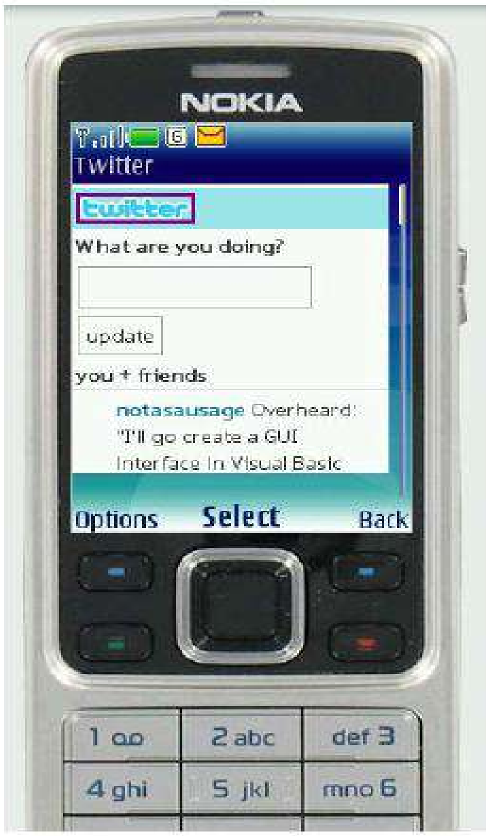


Figure 8-9. Example layout of a scroll-based application, where the user had to press the D-pad past each link to scroll the page

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 that mobile screens were available only in black and white (well, technically, it was black on a green screen). These days, we have nearly the entire spectrum of colors to choose from for mobile designs.

The most common obstacle you 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.

For an example of posterization, the technical term for when the gradation of tone is replaced with regions of fewer tones, see in Figure 8-10 how dramatically the color depth can affect the quality of a photo or gradient, producing banding in several parts in the image.

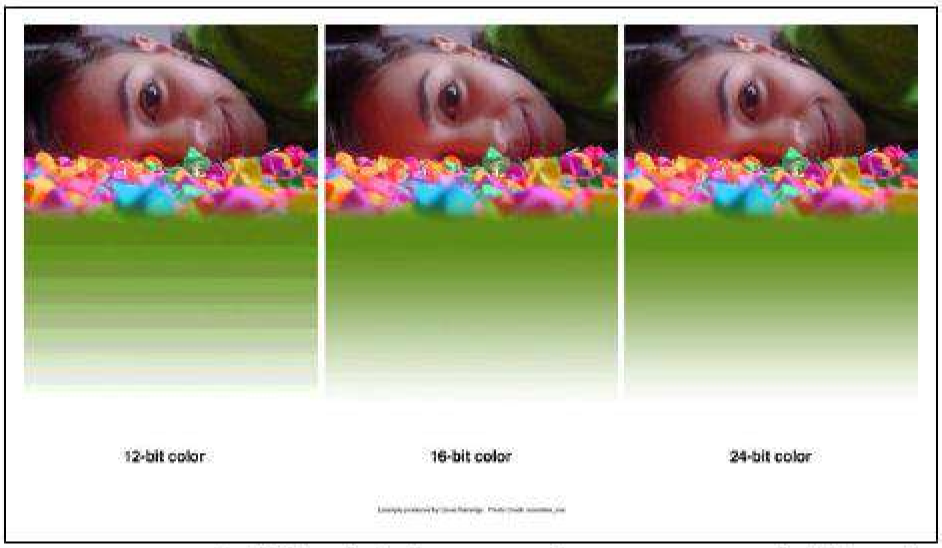


Figure 8-10. An example of different levels of posterization that can occur across multiple device color depths

Table 8-1. Supported colors and 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
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

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 personal ways. Using the right colors can be useful for delivering the right message and setting expectations.

One of the examples I 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 8-2 provides some of the characteristics of various colors that naturally evoke certain emotions in people.

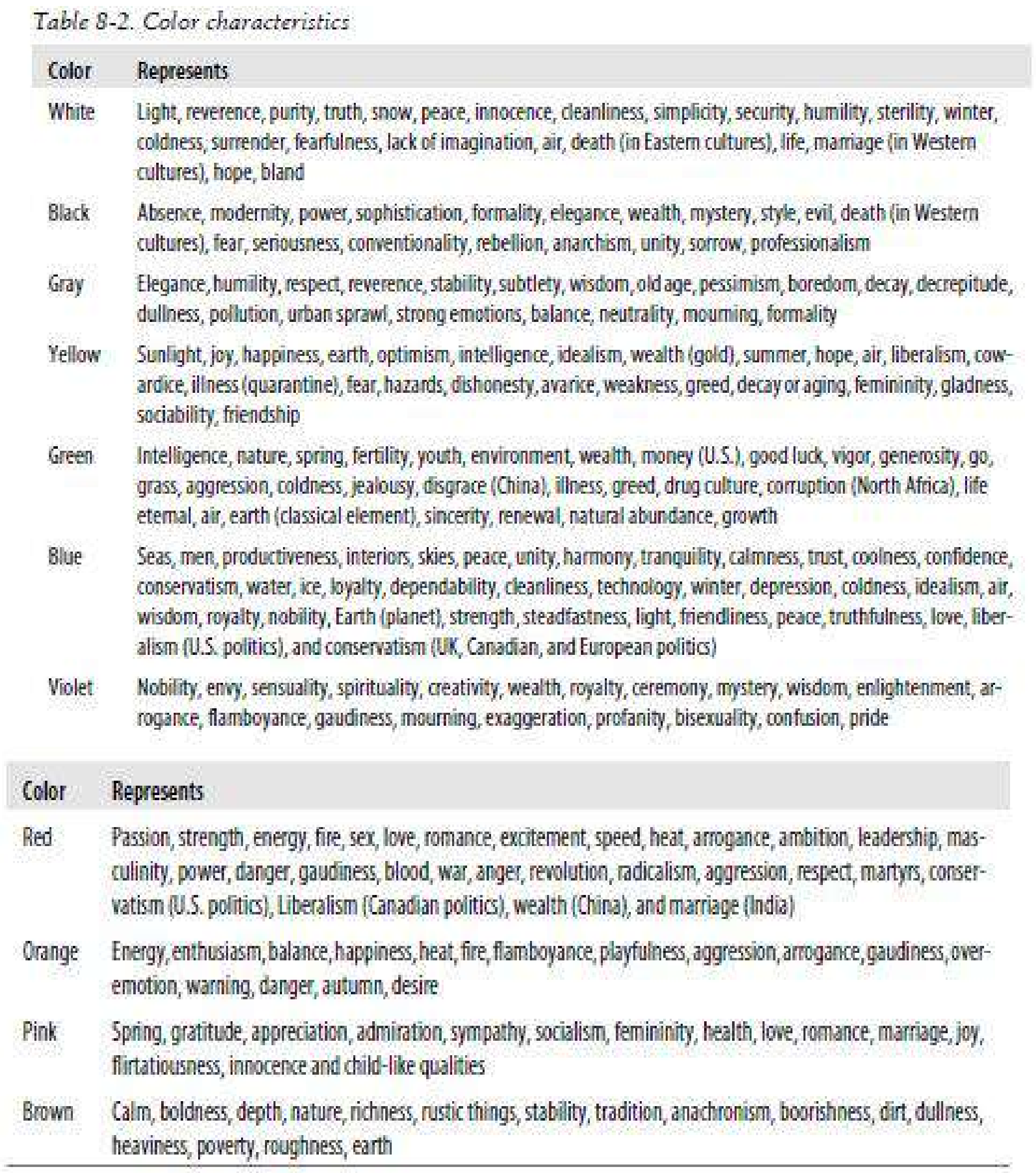


Table 8-2. Color characteristics

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

Color palettes

Defining color palettes can be useful for maintaining a consistent use of color in your 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. I've found that I use three basic ways to define a color palette:

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 I select using a color wheel.

Adaptive

An adaptive palette is one in which you 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, I use an adaptive palette to make sure that my colors are consistent with the target mobile platform.

Inspired

This is a design that is created from the great pieces of design you might see online, as shown in Figure 8-11, or offline, in which a picture of the design might inspire you. This could be anything from an old poster in an alley, a business card, or some packaging. When I sit down with a new design, I thumb through some of materials to create an inspired palette. Like with the adaptive palette, you actually extract the colors from the source image, though you should never ever use the source material in a design.

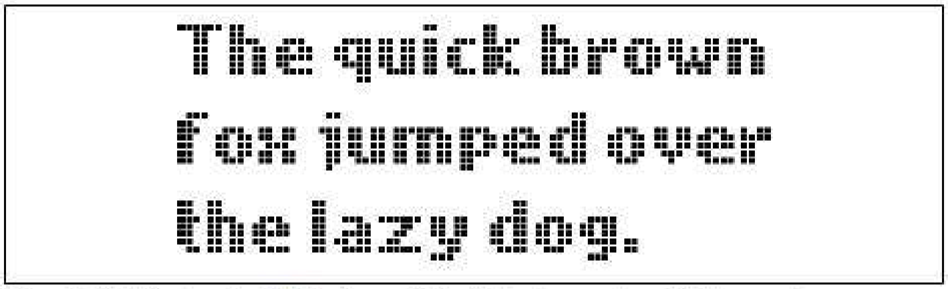


Figure 8-12. What most mobile designers think of when it comes to mobile typography

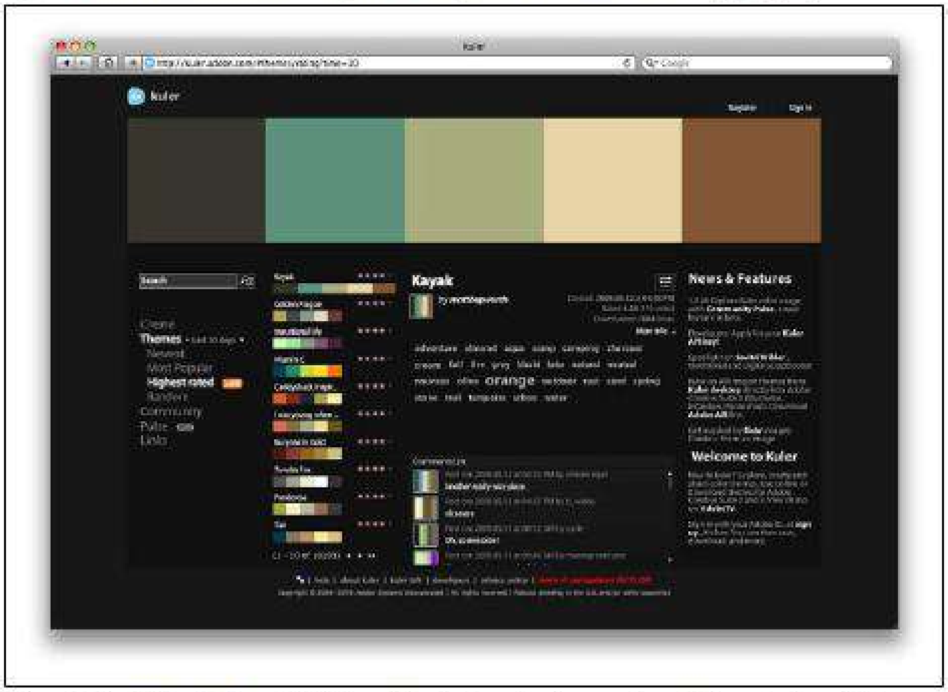


Figure 8-11. Adobe Kuler, a site that enables designers to share and use different color palettes

Typography

The sixth element of mobile design is typography, which in the past would bring to mind 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 you could use (Figure 8-12), and that was the device font. The only control over the presentation was the size.

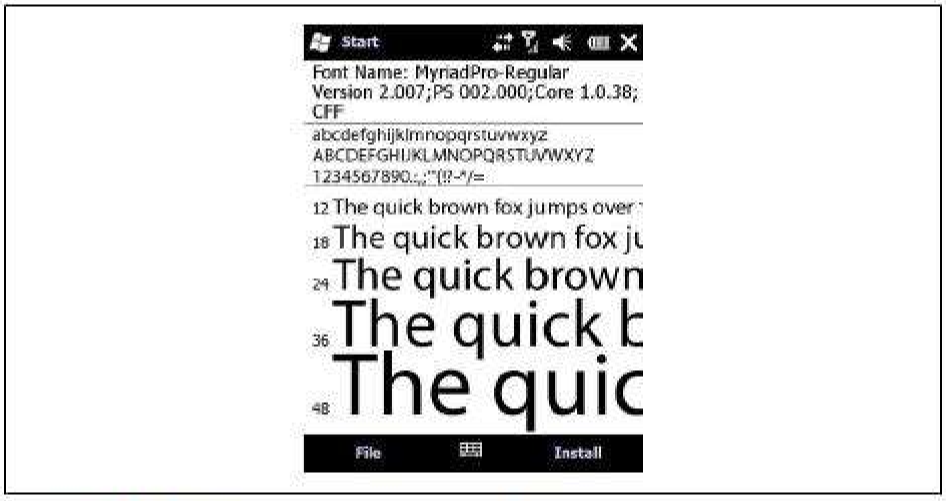

Start
Font Name: MyriadPro-Regular
Version 2.007;PS 002.000;Core 1.0.38;
CFF
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890.:,;"(!?-*/=
12 The quick brown fox jumps over
18 The quick brown fox ju
24 The quick brown
36 The quick b
48 The quic
File
Install

Figure 8-14. Microsoft ClearType using subpixels to display sharp text

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 8-13, 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.

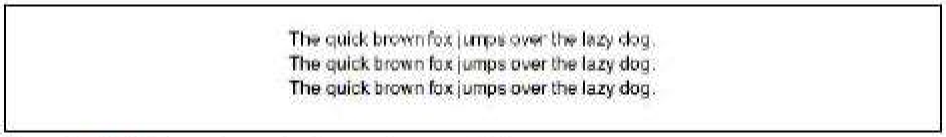

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

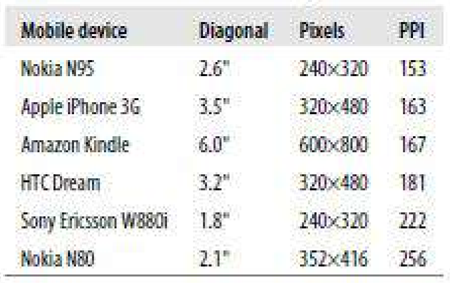
Figure 8-13. Different ways text can render on mobile screens

The Microsoft Windows Mobile platform uses the subpixel technique with its Clear-Type technology, as shown in Figure 8-14.

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 ("I have a 15.4-inch laptop screen") or their pixel dimensions, or resolution ("The resolution of my laptop is 1440×900 pixels"). 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 for my 15.4-inch laptop 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 screen appears 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 8-3 provides the dimensions and PPI for a few mobile devices.

Table 8-3. Dimensions and PPI for some 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

Type options

Fortunately, today's mobile devices have a few more options than a single typeface, but the 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, you have a few variations of serif, sans-serif, and monospace fonts, and depending on the platform, maybe a few custom fonts (Figure 8-15).

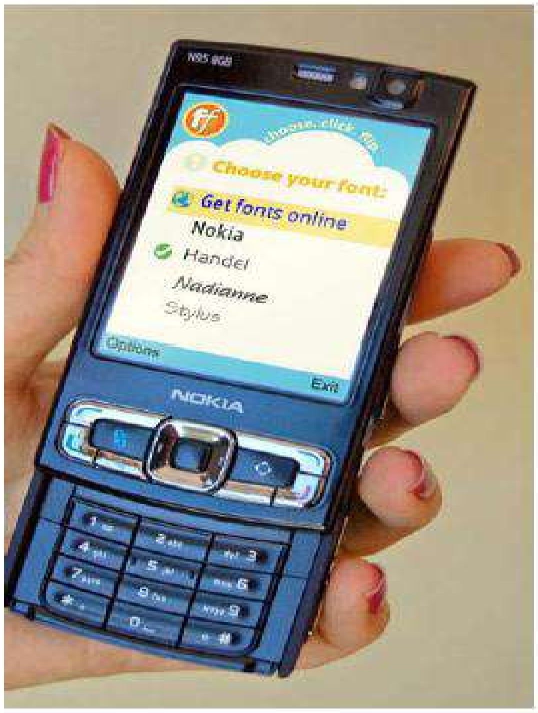


Figure 8-15. Options in typography increase as the devices become more sophisticated

Therefore, when creating mobile designs for either web or native experiences, my advice
is to stick with either the default device font, or web-safe fonts—your basic serif variants
like Times New Roman and Georgia or sans-serif typefaces like Helvetica, Arial, or Verdana.

Font replacement

The ability to use typefaces that are not already loaded on the device varies from model to model and your chosen platform. Some device APIs will allow you to load a typeface into your native application. Some mobile web browsers support various forms of font replacement; the two most common are sIFR and Cufon. sIFR uses Flash to replace

HTML text with a Flash representation of the text, but the device of course 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 8-16. This can be done by following these six simple rules:

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.

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.

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.

Leave space on the right and left of each line; don't crowd the screen

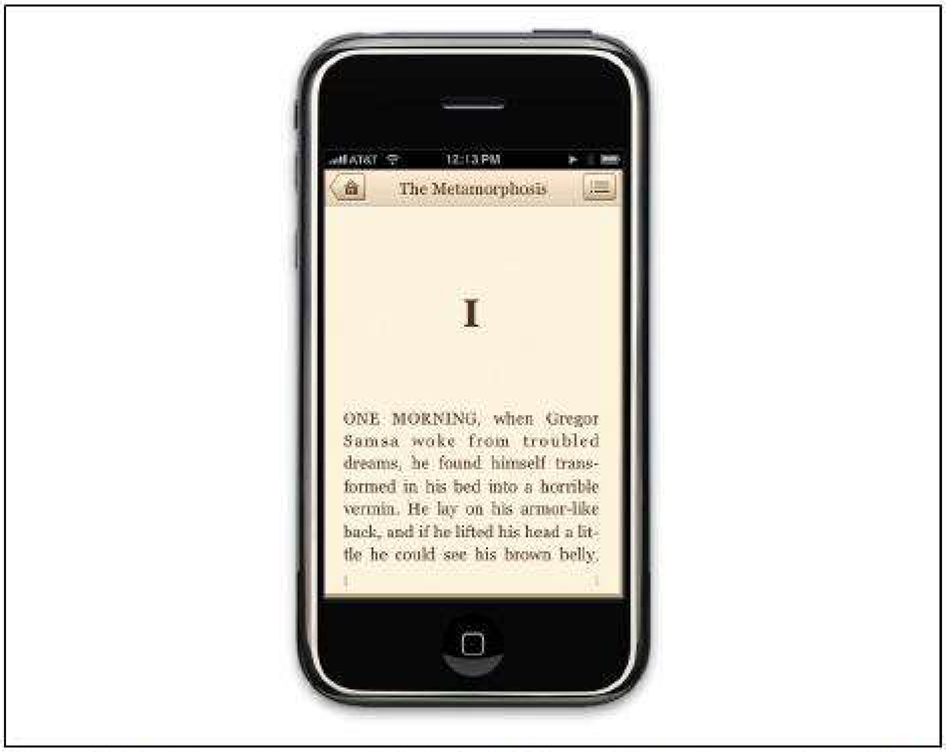
Most mobile frameworks give you full access to the screen, meaning that you normally need to provide some spacing between the right and left side of the screen's edge and your text—not much, typically about three to four character widths.

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.

Use short paragraphs

Like on the Web, keep paragraphs short, using no more than two to three sentences per paragraph.



I

ONE MORNING, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armor-like back, and if he lifted his head a little he could see his brown belly,

Figure 8-16. Classics, an iPhone application designed with readability and typography in mind

Graphics

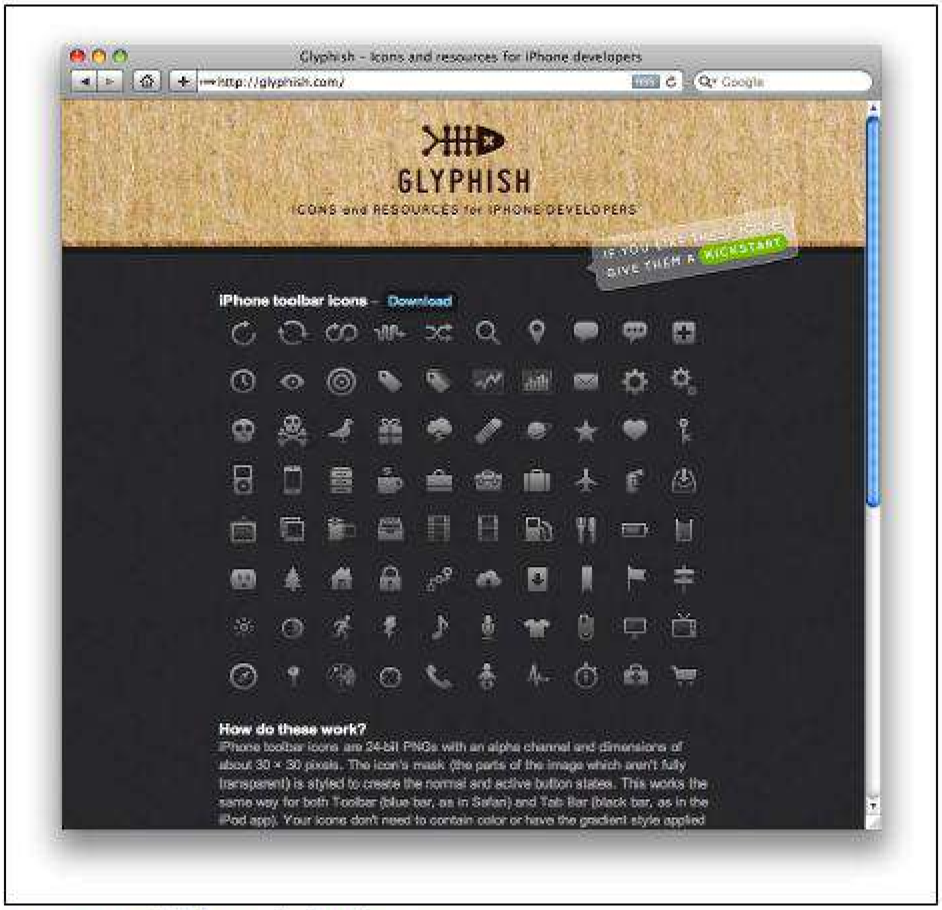
The final design element is graphics, or the images that are used to establish or aid a visual experience. Graphics can be used to supplement the look and feel, or as content displayed inline with the text.

For example, in Figure 8-17, you 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.

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 8-18, you can see some helpful icons that clearly communicate an idea and some perplexing icons that leave you scratching your head.


Glyphish – Icons and resources for iPhone developers
http://glyphish.com/
Google
GLYPHISH
ICONS and RESOURCES for IPHONE DEVELOPERS
IF YOU LIKE THESE ICONS GIVE THEM A KICKSTART
iPhone toolbar icons – Download

How do these work?
iPhone toolbar icons are 24-bit PNGs with an alpha channel and dimensions of about 30 × 30 pixels. The icon's mask (the parts of the image which aren't fully transparent) is styled to create the normal and active button states. This works the same way for both Toolbar (blue bar, as in Safari) and Tab Bar (black bar, as in the iPod app). Your icons don't need to contain color or have the gradient style applied

Figure 8-18. 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 you 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 you can encounter issues regarding how that image will display in a flexible UI—for example, when the device orientation is changed. In Figure 8-19, you can see how the pig graphic is designed to be positioned to the right regardless of the device orientation.

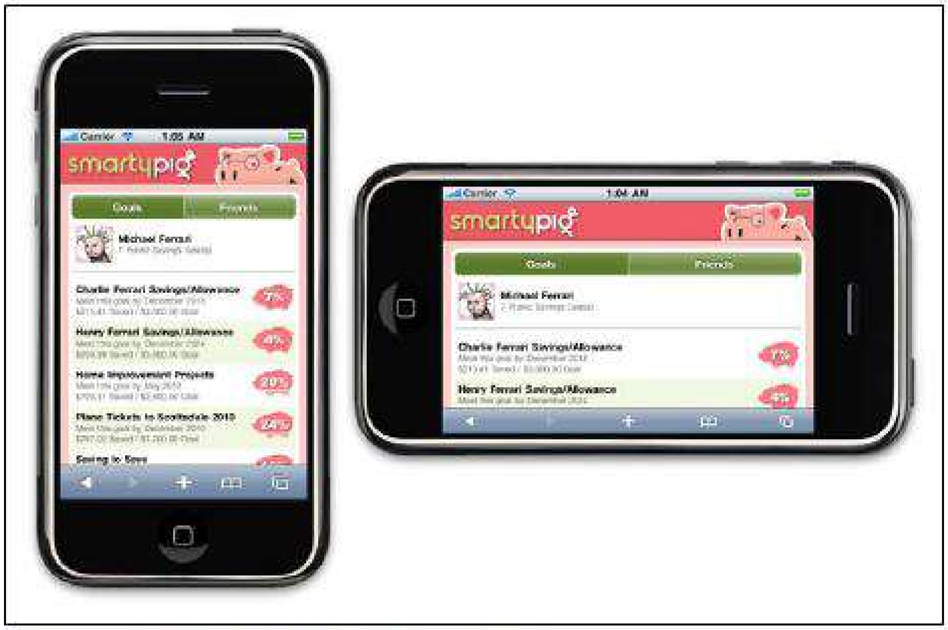

smartypig
Goals
Friends
Michael Ferrari
Charlie Ferrari Savings/Allowance
Henry Ferrari Savings/Allowance
Home Improvement Projects
Plane Tickets to Scottsdale 2010

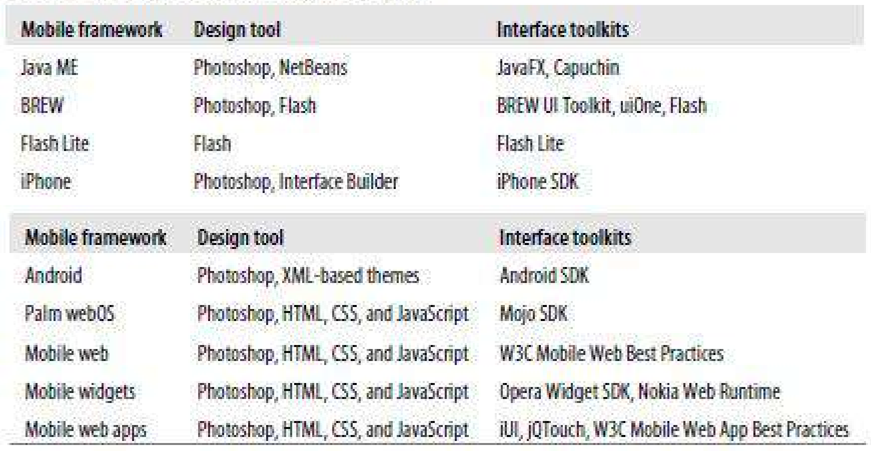
Figure 8-19. Using graphics in multiple device orientations

Mobile Design Tools

As I mentioned earlier, 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 8-4, you can see each of the design tools and what interface toolkits are available for it.

Table 8-4. Design tools and interface toolkits



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

Designing for the Right Device

Now is the time to ask, “What device suits this design best? What market niche would appreciate it most? What devices are the most popular within that niche?” The days of tent-poles are gone. Focus instead on getting your best possible experience to the market that will appreciate it most. It might not be the largest or best long-term market, but what you will learn from the best possible scenario will tell you volumes about your mobile product’s potential for success or failure. You will learn which devices you need to design for, what users really want, and how well your design works in the mobile context.

This knowledge will help you develop your porting and/or adaptation strategy, the most expensive and riskiest part of the mobile equation. For example, if you know that 30 percent of your users have iPhones, then that is a market you can exploit to your advantage. iPhone users consume more mobile content and products than the average mobile user. This platform has an easy-to-learn framework and excellent documentation, for both web and native products, and an excellent display and performance means. Although iPhone users might not be the majority of your market, the ability to create the best possible design and get it in front of those users presents the least expensive product to produce with the lowest risk.

Designing for Different Screen Sizes

Mobile devices come in all shapes and sizes. Choice is great for consumers, but bad for design. It can be incredibly difficult to create that best possible experience for a plethora of different screen sizes. For example, your typical feature phone might only be 140 pixels wide, whereas your higher-end smartphone might be three to four times wider.

Landscape or portrait? Fixed width or fluid? Do you use one column or two? These are common questions that come up when thinking about your design on multiple screen sizes. The bad news is that there is no simple answer. How you design each screen of content depends on the scope of devices you look to support, your content, and what type of experience you are looking to provide. The good news is that the vast majority of mobile device screens share the same vertical or portrait orientation, even though they vary greatly in dimension, as shown in Figure 8-20.

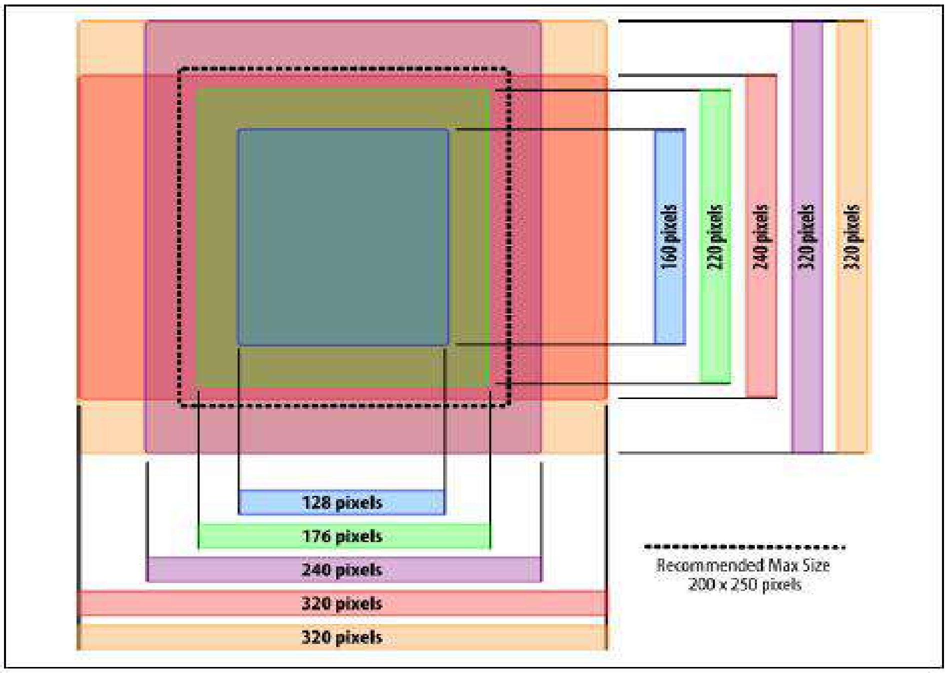


Figure 8-20. Comparing the various screen sizes

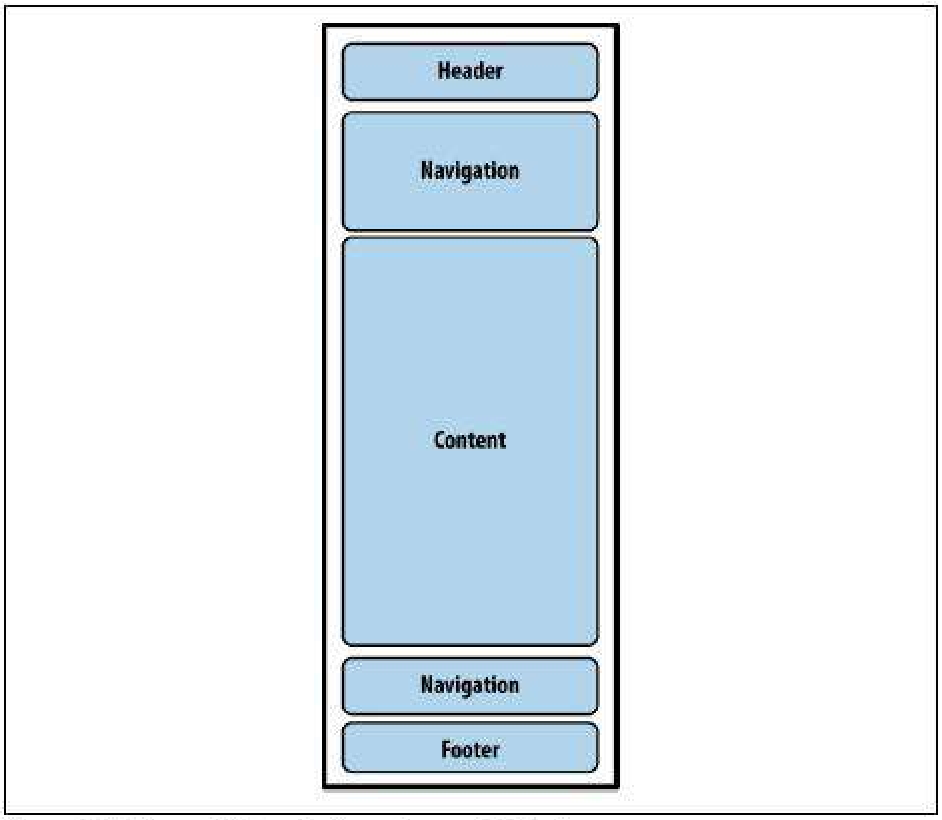


Figure 8-21. The typical flow of information on mobile devices

The greatest challenge to creating a design that works well on multiple screen sizes is filling the width. For content-heavy sites and applications, the width of mobile devices is almost the perfect readability, presenting not too many words per line of text. The problem is when you have to present a number of tasks or actions. The easiest and most compatible way is to present a stacked list of links or buttons, basically one action per line. It isn't the most effective use of space, but presenting too many actions on the horizontal axis quickly clutters the design—not to mention that it is more difficult to adapt to other devices.

Unfortunately, it isn't always reasonable to implement fluid or flexible designs that stretch to fit the width of the screen. Although most mobile web browsers and device framework APIs enable it in principle, its execution across multiple devices is a little anticlimatic. Mobile websites usually employ a fixed-width layout for the lowest common denominator, and native applications are often resized for multiple screen sizes during development.

As devices get larger, denser screens, you will see an increase in the use of touch, forcing the size of content to increase to fingertip size—typically 40 pixels wide and 40 pixels tall (Figure 8-22). This actually solves part of the horizontal axis problem, simply by making content larger for larger screens. Ironically, you can fit almost the same amount of usable content in an iPhone as you can a lower-end device.

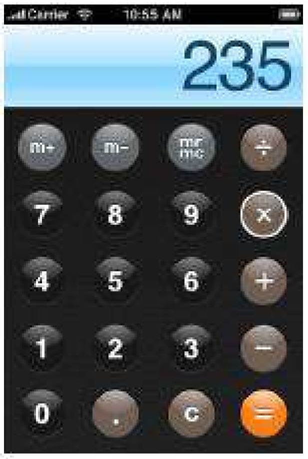


Figure 8-22. The iPhone calculator application uses common fingertip-size controls

UNIT V WEB INTERFACE DESIGN 9
Designing Web Interfaces – Drag & Drop, Direct Selection, Contextual Tools, Overlays, Inlays and Virtual Pages, Process Flow. Case Studies.

Drag & Drop

Interesting Moments
At first blush, drag and drop seems simple. Just grab an object and drop it somewhere.
But, as always, the devil is in the details. There are a number of individual states at which interaction is possible.
• How will users know what is draggable?
• What does it mean to drag and drop an object?
• Where can you drop an object, and where is it not valid to drop an object?
• What visual affordance will be used to indicate draggability?

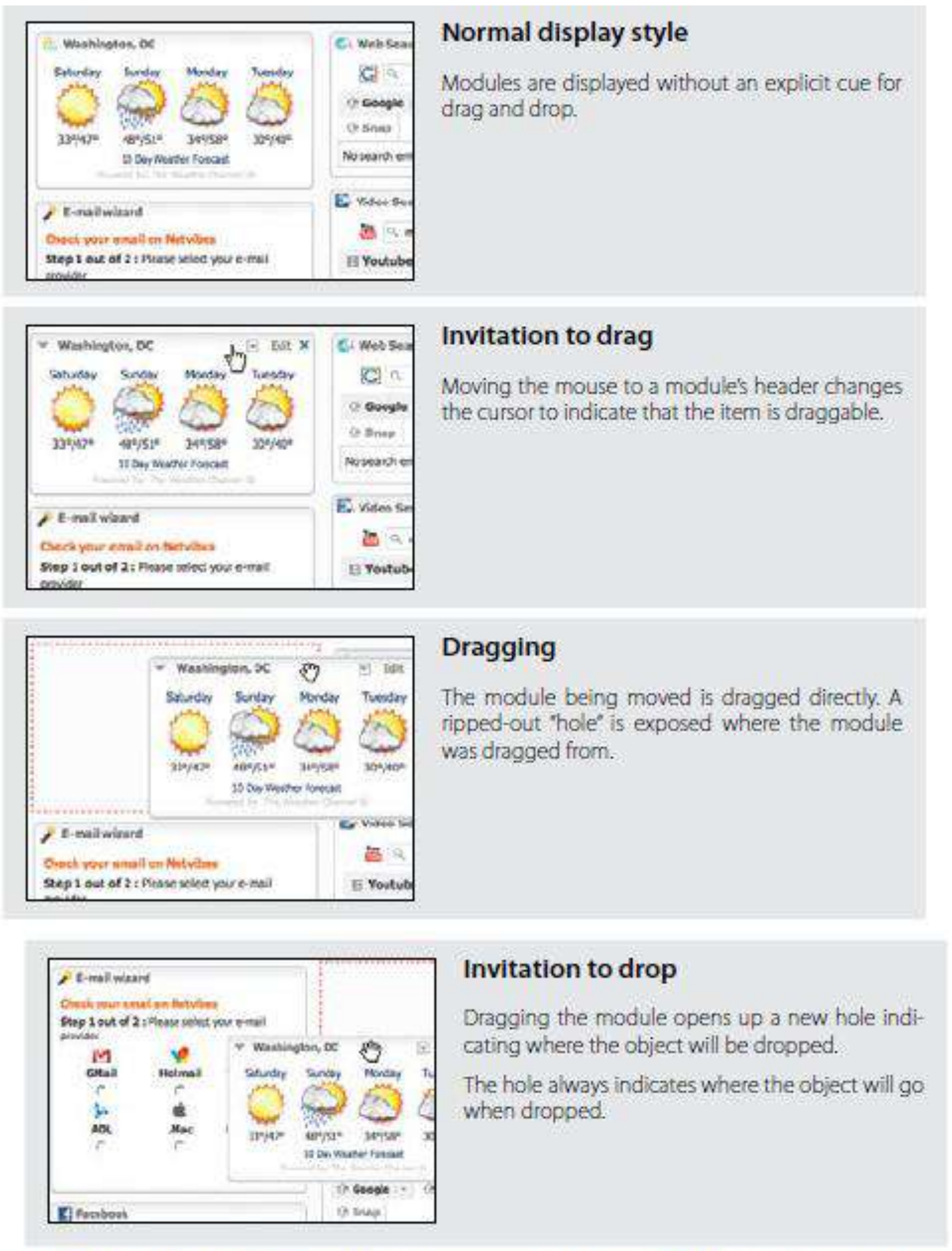
During drag, how w • ill valid and invalid drop targets be signified?
• Do you drag the actual object?
• Or do you drag just a ghost of the object?
• Or is it a thumbnail representation that gets dragged?
• What visual feedback should be used during the drag and drop interaction?
What makes it challenging is that there are a lot of events during drag and drop that can be used as opportunities for feedback to the user. Additionally, there are a number of elements on 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:
Page Load
Before any interaction occurs, you can pre-signify the availability of drag and drop.
For example, you could display a tip on the page to indicate draggability.
Mouse Hover
The mouse pointer hovers over an object that is draggable.
Mouse Down
The user holds down the mouse button on the draggable object.
Drag Initiated
After the mouse drag starts (usually some threshold—3 pixels).
Drag Leaves Original Location
After the drag object is pulled from its location or object that contains it.
Drag Re-Enters Original Location
When the object re-enters the original location.
Drag Enters Valid Target
Dragging over a valid drop target.
Drag Exits Valid Target
Dragging back out of a valid drop target.
Drag Enters Specific Invalid Target
Dragging over an invalid drop target.
Drag Is Over No Specific Target

Dragging over neither a valid or invalid target. Do you treat all areas outside of valid
Drag Hovers Over Invalid Target
User pauses over an invalid target without dropping the object. Do you care? Will you want additional feedback as to why it is not a valid target?
Drop Accepted
Drop occurs over a valid target and drop has been accepted.
Drop Rejected
Drop occurs over an invalid target and drop has been rejected. Do you zoom back the dropped object?
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 you can visually manipulate a number of *actors*. The page elements available include:
• Page (e.g., static messaging on the page)
• Cursor
• Tool Tip
• Drag Object (or some portion of the drag object, e.g., title area of a module)
• Drag Object's Parent Container
• Drop Target

Purpose of Drag and Drop
Drag and drop can be a powerful idiom if used correctly. Specifically it is useful for:
Drag and Drop Module
Rearranging modules on a page.
Drag and Drop List
Rearranging lists.
Drag and Drop Object
Changing relationships between objects.
Drag and Drop Action
Invoking actions on a dropped object.
Drag and Drop Collection
Maintaining collections through drag and drop.
Drag and Drop Module
One of the most useful purposes of drag and drop is to allow the user to directly place objects where she wants them on the page. A typical pattern is Drag and Drop Modules on a page. Netvibes provides a good example of this interaction pattern (Figure 2-3).



Normal display style

Modules are displayed without an explicit cue for drag and drop.

Invitation to drag

Moving the mouse to a module's header changes the cursor to indicate that the item is draggable.

Dragging

The module being moved is dragged directly. A ripped-out "hole" is exposed where the module was dragged from.

Invitation to drop

Dragging the module opens up a new hole indicating where the object will be dropped.

The hole always indicates where the object will go when dropped.

Figure 2-3. *Netvibes allows modules to be arranged directly via drag and drop; the hole cues what will happen when a module is dropped*

Placeholder target

Netvibes uses a placeholder (hole with dashed outline) as the drop target. The idea (illustrated in Figure 2-5) is to always position a hole in the spot where the drop would occur. When module 1 starts dragging, it gets "ripped" out of the spot. In its place is the placeholder target (dashed outline). As 1 gets dragged to the spot between 3 and 4, the placeholder target jumps to fill in this spot as 4 moves out of the way.

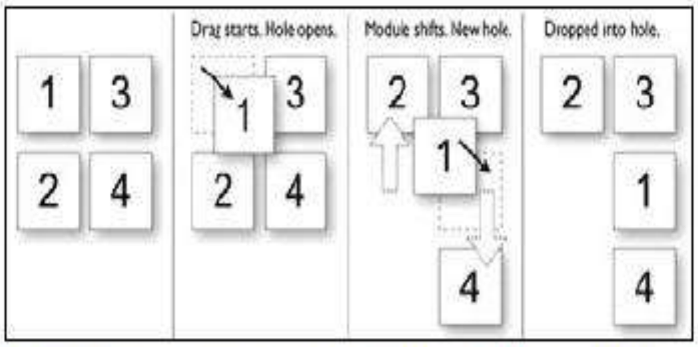


Figure 2-5. A placeholder target always shows where the dragged module will end after the drop; module 1 is being dragged from the upper right to the position between modules 3 and 4

The hole serves as a placeholder and always marks the spot that the dragged module will land when dropped. It also previews what the page will look like (in relation to the other modules) if the drop occurs there. For module drag and drop, the other modules only slide 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 understand what is actually happening. This issue is compounded when modules look similar. The user starts dragging the modules around and quickly gets confused about what just got moved. One way to resolve this is to provide a quick animated transition as the modules move. It is important, however, that any animated transitions not get in the way of the normal interaction. In Chapter 11, we will discuss timing of transitions in detail.

There is a point in Figure 2-5 where the placeholder shifts to a new location. What determines placeholder targeting? In other words, what determines where the user is intending to place the dragged object? 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's new location.

Boundary-based placement. Since most sites that use placeholder targeting drag the module in its original size, targeting is determined by the boundaries of the dragged object and the boundaries of the dragged-over object. The mouse position is usually ignored because modules are only draggable in the title (a small region). Both Netvibes and iGoogle take the boundary-based approach. But, interestingly, they calculate the position of their placeholders differently.

In Netvibes, the placeholder changes position only after the dragged module's title bar has moved beyond the dragged-over module's title bar. In practice, this means if you are moving a small module to be positioned above a large module, you have to move it to the very top of the large module. In Figure 2-6 you have to drag the small "To Do List" module all the way to the top of the "Blog Directory" module before the placeholder changes position.

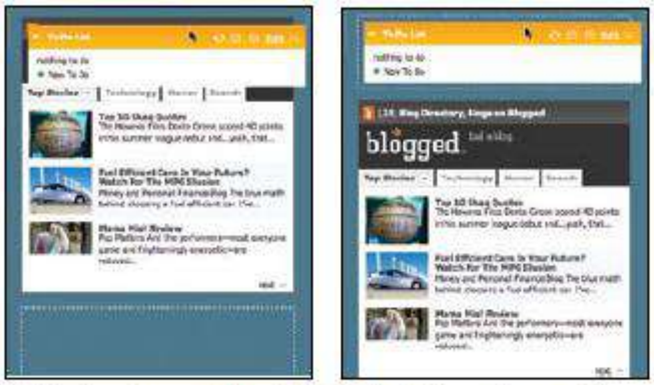


Figure 2-6. *In Netvibes, dragging a small module to be placed above a large module requires dragging a large distance; the "To Do List" has to be dragged to the top of the "Blog Directory" module*

In contrast, moving the small module below the large module actually requires less drag distance since you only have to get the title bar of the small module below the title bar of the large module (Figure 2-7).

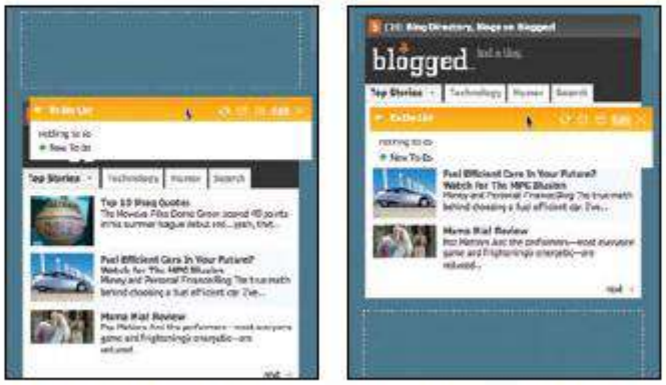


Figure 2-7. *Dragging a small module below a large module requires a smaller drag distance; since the targeting is based on the header of the dragged-over module, the drag distance in this scenario is less than in the previous figure*

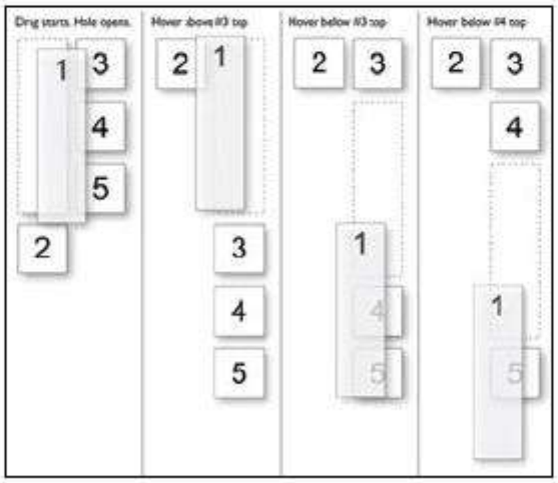


Figure 2-8. *The Netvibes approach requires the dragged object's title to be placed above or below a module before the placement position changes; this results in inconsistent drag distances*

A more desirable approach is that taken by iGoogle. Instead of basing the drag on the *title bar*, iGoogle calculates the placeholder targeting on the dragged-over object's *midpoint*. In Figure 2-9, the stock market module is very large (the module just above the moon phase module).

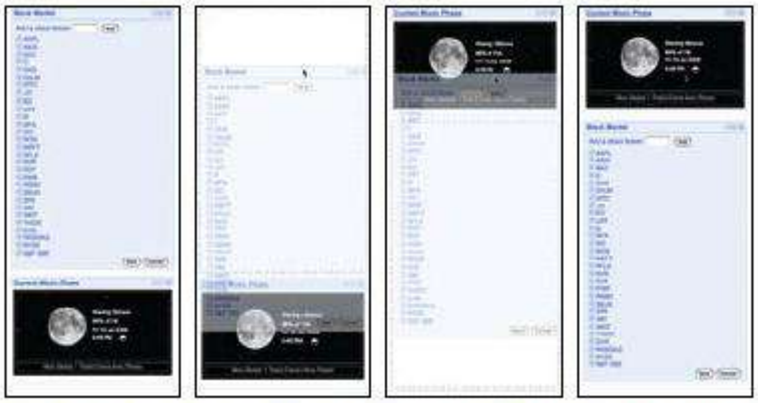


Figure 2-9. *When dragging a module downward, iGoogle moves the placeholder when the bottom of the dragged module crosses the midpoint of the object being dragged over; the distance to accomplish a move is less than in the Netvibes approach*

With the Netvibes approach, you would have to drag the stock module's title below the moon phase module's title. iGoogle instead moves the placeholder when the *bottom* of the dragged module (stock module) crosses the midpoint of the dragged over module (moon phase 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 (Figure 2-10).

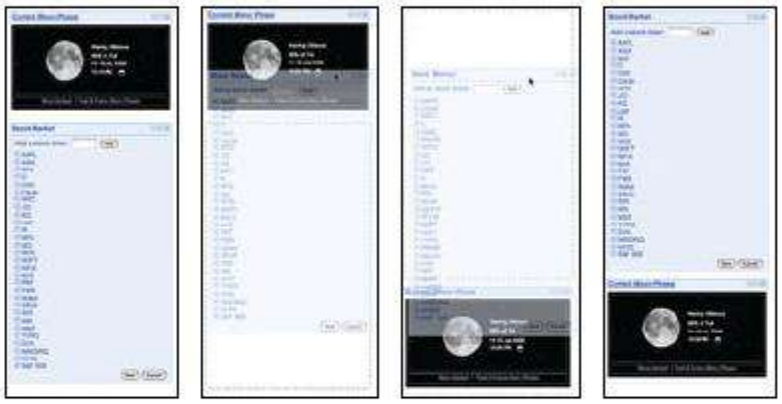


Figure 2-10. *When dragging a module upward, iGoogle moves the placeholder when the top of the dragged module crosses the midpoint of the object being dragged over; dragging modules up or down requires the same effort, unlike in the Netvibes example*

As Figure 2-11 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.

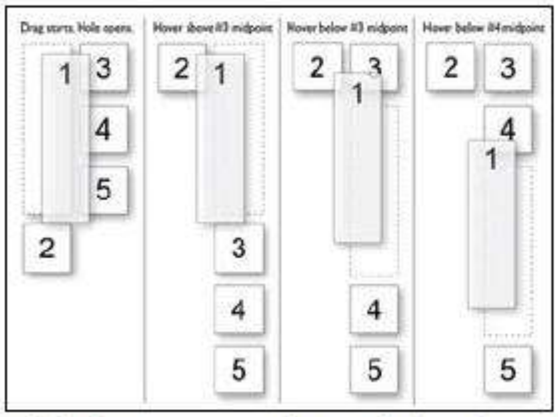


Figure 2-11. *To create the best drag experience, use the original midpoint location of the module being dragged over to determine where to drop the dragged module: module 1 is being dragged into the position just below module 4*

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 (see Figure 2-13).

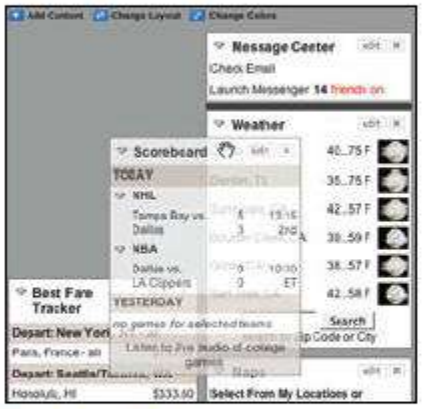


Figure 2-13. *My Yahoo! uses the insertion bar approach*

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 2-14. 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.

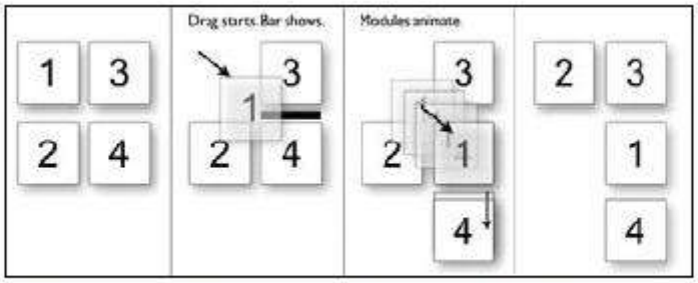
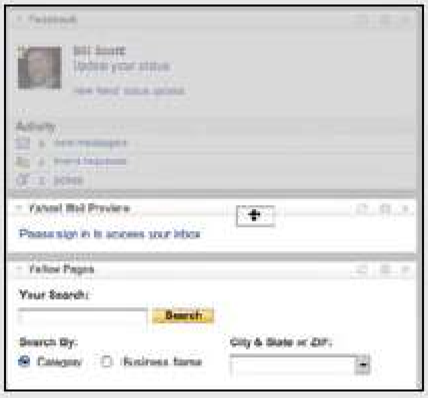


Figure 2-14. *Using an insertion bar keeps the page stable during dragging and makes it clear how things get rearranged when the module is dropped*

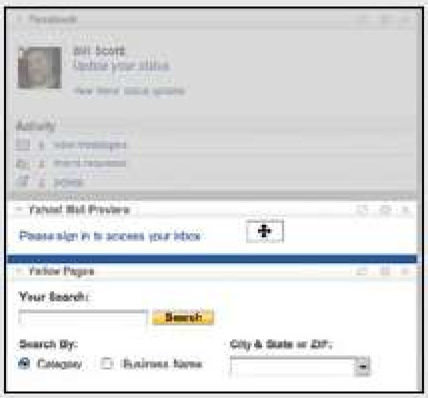
Unlike with the placeholder target, the dragged module 1 is usually represented with a slightly transparent version of the module (also known as *ghosting*). This is the approach shown in Figure 2-13 in an earlier version of My Yahoo!. In the most current version, fullsize module dragging has been replaced with a thumbnail representation (the small gray outline being dragged in Figure 2-15). This is somewhat unfortunate since the small gray outline is not very visible.



Dragging a module down

Note the gray outline being dragged. The small gray rectangle represents the dragged module (the Facebook module).

Since the drag representation is above the mid-point of the Yahoo! Mail Preview module, no change is indicated for a drop.



Insertion bar appears

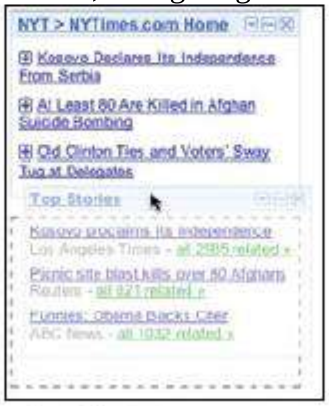
The dragged module representation is now below the Yahoo! Mail Preview module's midpoint.

The insertion bar is rendered to show that the Facebook module will be placed just below the Mail Preview module if dropped.

Figure 2-15. *My Yahoo! uses a small gray rectangle to represent the dragged module*

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.

Drag rendering
How should the dragged object be represented? Should it be rendered with a slight transparency (ghost)? Or should it be shown fully opaque? Should a thumbnail representation be used instead? As shown earlier, My Yahoo! uses a small gray rectangle to represent a module (Figure 2-15). Netvibes represents the dragged module in full size as opaque (shown back in Figure 2-3), while iGoogle uses partial transparency (Figure 2-17). The transparency (ghosting) effect ommunicates 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.


NYT > NYTimes.com Home

⊞ Kosovo Declares Its Independence From Serbia

⊞ At Least 80 Are Killed in Afghan Suicide Bombing

⊞ Old Clinton Ties and Voters' Sway Tug at Delegates

Top Stories

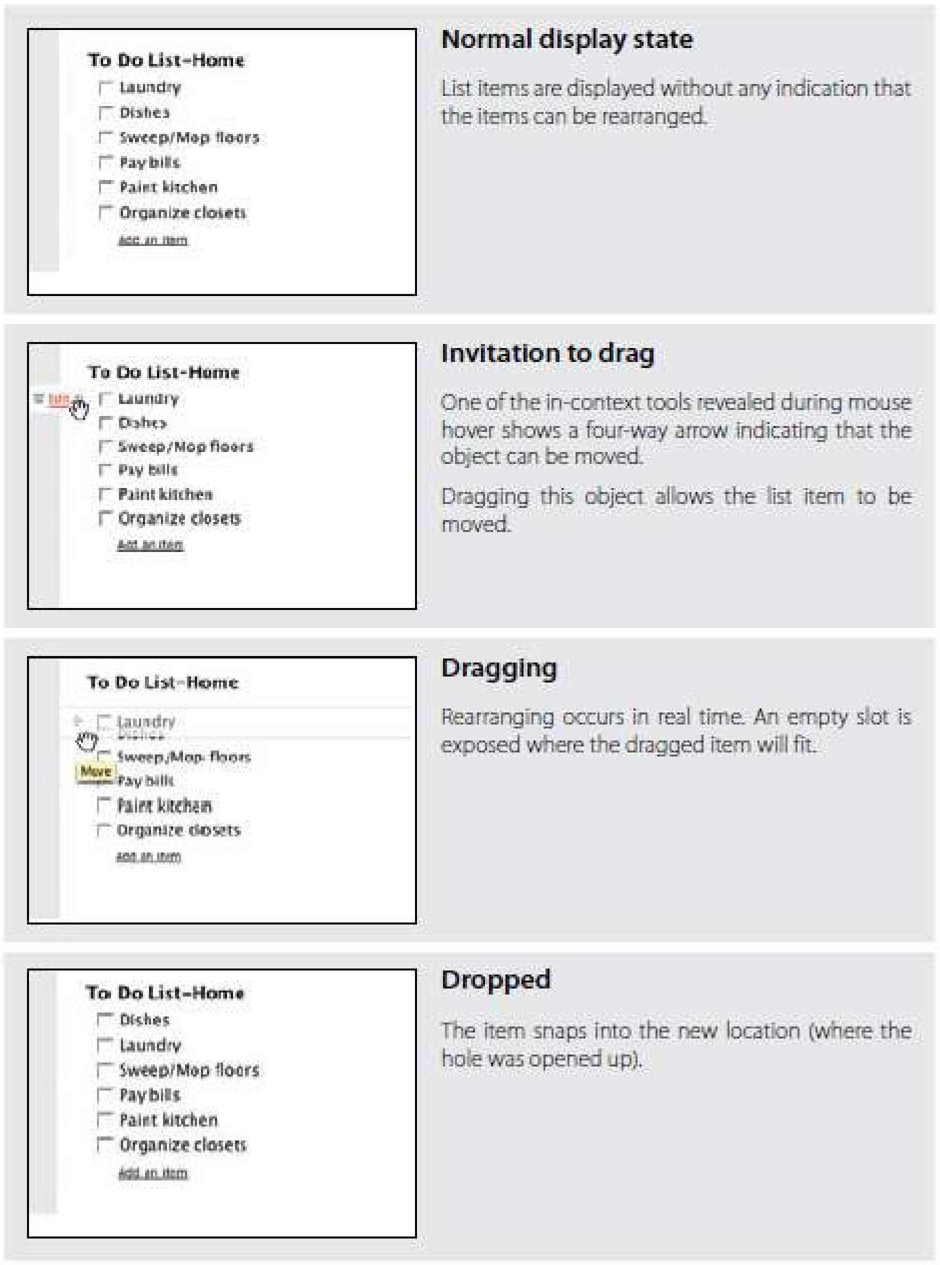
Kosovo proclaims its independence
Los Angeles Times - all 2985 related »

Picnic site blast kills over 80 Afghans
Reuters - all 821 related »

Funnies: Obama Backs Cher
ABC News - all 1032 related »

Figure 2-17. *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*

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 2-18).



Normal display state

List items are displayed without any indication that the items can be rearranged.

Invitation to drag

One of the in-context tools revealed during mouse hover shows a four-way arrow indicating that the object can be moved.

Dragging this object allows the list item to be moved.

Dragging

Rearranging occurs in real time. An empty slot is exposed where the dragged item will fit.

Dropped

The item snaps into the new location (where the hole was opened up).

Figure 2-18. *Backpackit allows to-do lists to be rearranged directly via drag and drop*

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, you 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 (Figure 2-19).

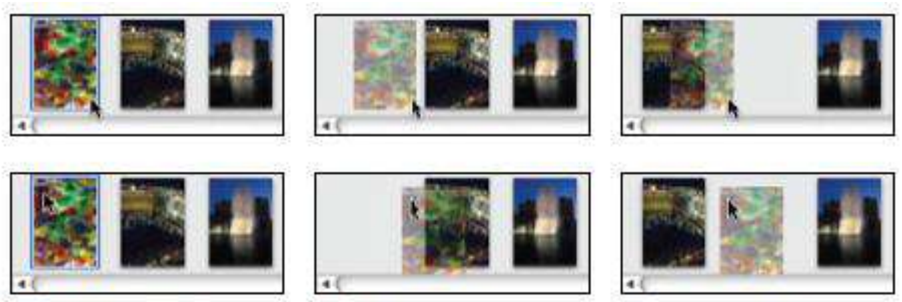
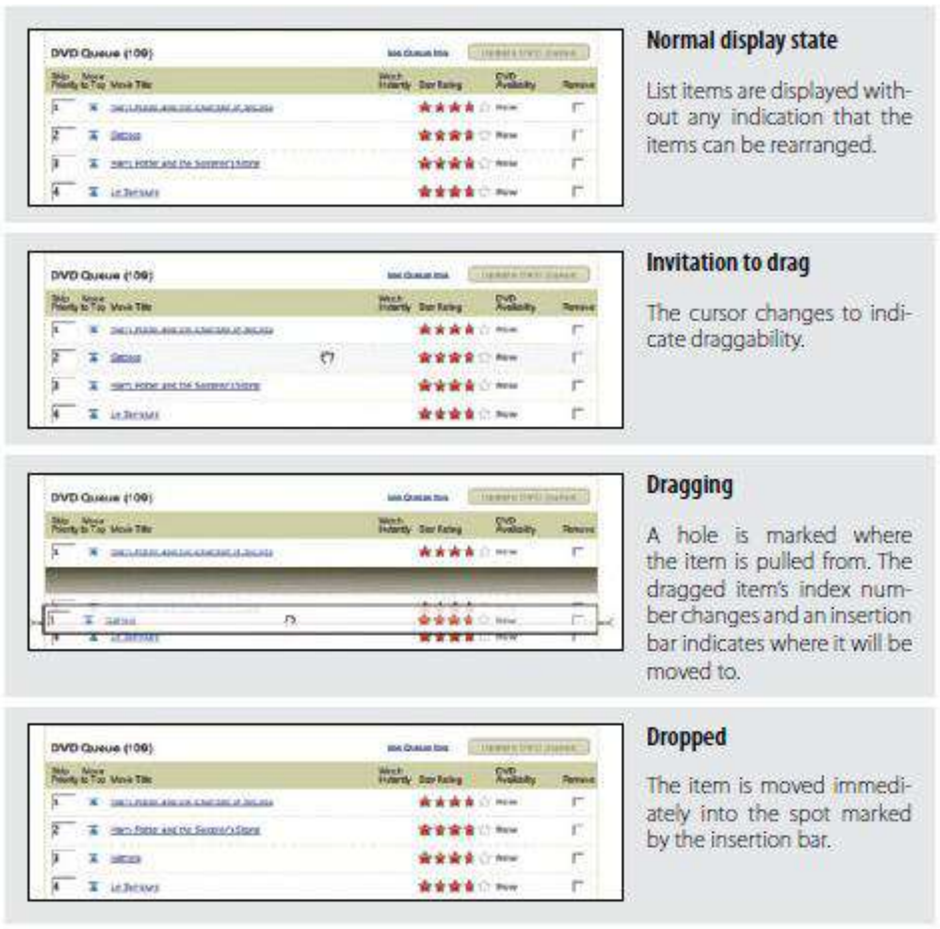


Figure 2-19. *iPhoto uses cursor position: when the cursor crosses a threshold (the edge of the next photo), a new position is opened up*

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 2-19, 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.

Insertion target
Just as with Drag and Drop Modules, placeholder targeting is not the only game in town. You 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 2-20).



Normal display state

List items are displayed without any indication that the items can be rearranged.

Invitation to drag

The cursor changes to indicate draggability.

Dragging

A hole is marked where the item is pulled from. The dragged item's index number changes and an insertion bar indicates where it will be moved to.

Dropped

The item is moved immediately into the spot marked by the insertion bar.

Figure 2-20. *A Netflix queue can be rearranged via drag and drop*

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 2-22).

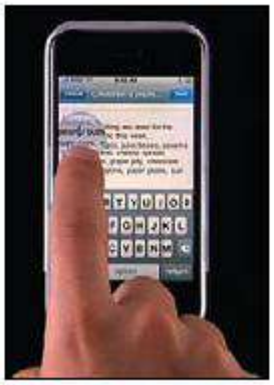
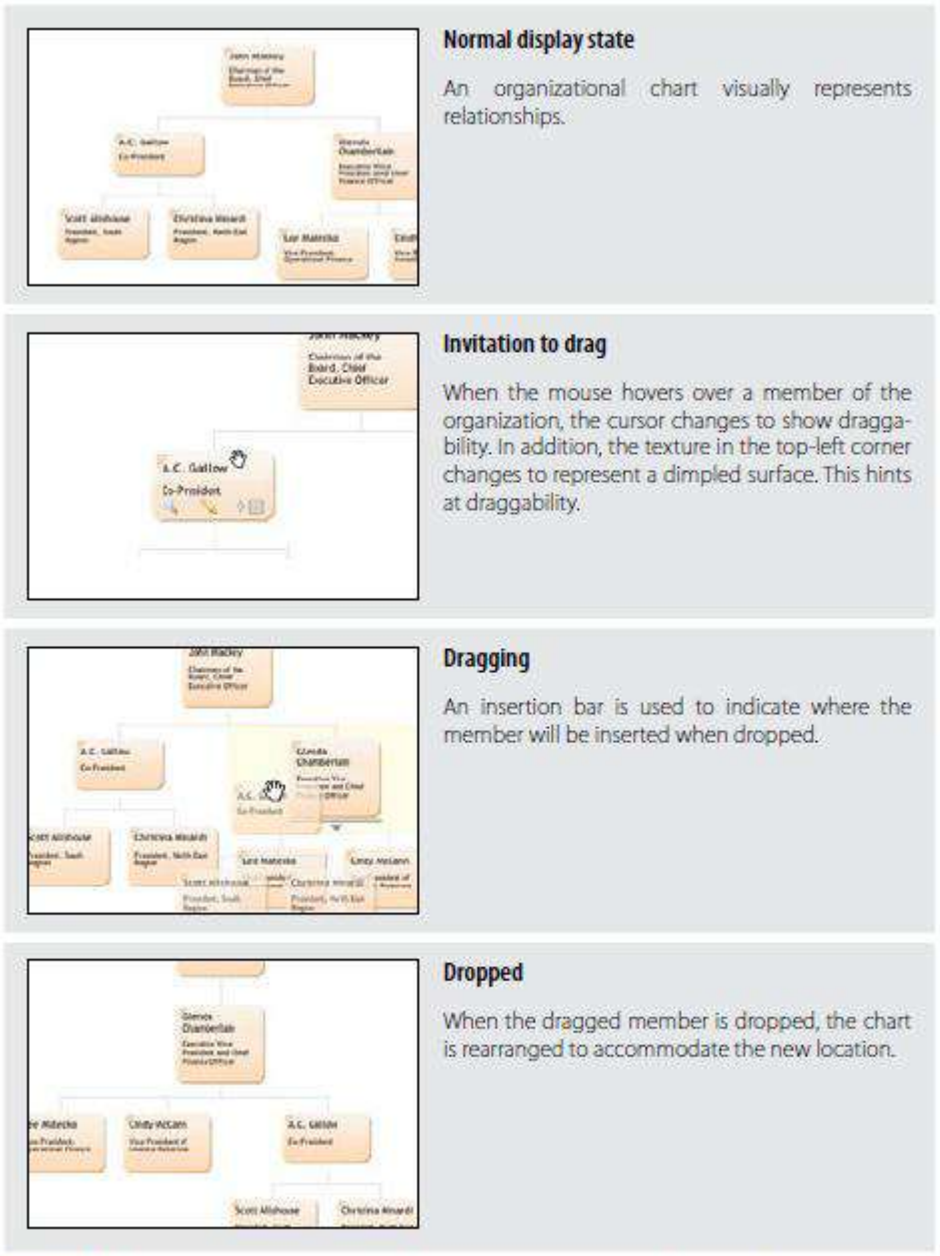


Figure 2-22. *The iPhone provides a drag magnifier lens that makes it easier to position the cursor*

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 (Figure 2-23).



Normal display state

An organizational chart visually represents relationships.

Invitation to drag

When the mouse hovers over a member of the organization, the cursor changes to show draggability. In addition, the texture in the top-left corner changes to represent a dimpled surface. This hints at draggability.

Dragging

An insertion bar is used to indicate where the member will be inserted when dropped.

Dropped

When the dragged member is dropped, the chart is rearranged to accommodate the new location.

Figure 2-23. *Cogmap allows organizational charts to be rearranged on the fly with drag and drop*

Drag feedback: Highlighting

Bubbl.us, an online mind-mapping tool, simply highlights the node that will be the new parent (Figure 2-24).

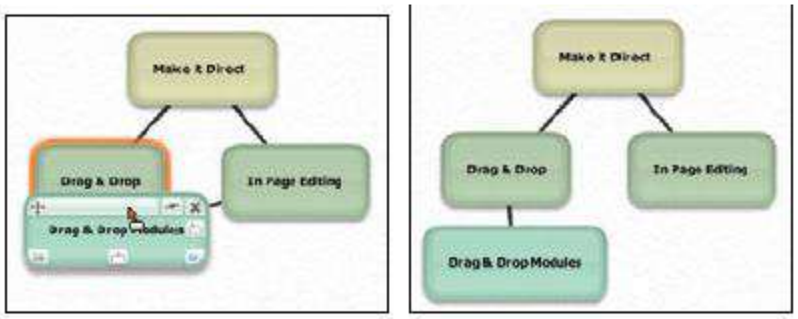


Figure 2-24. *Bubbl.us provides a visual indication of which node the dropped node will attach itself to*

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 2-25).

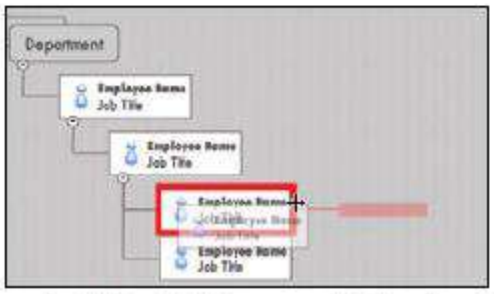
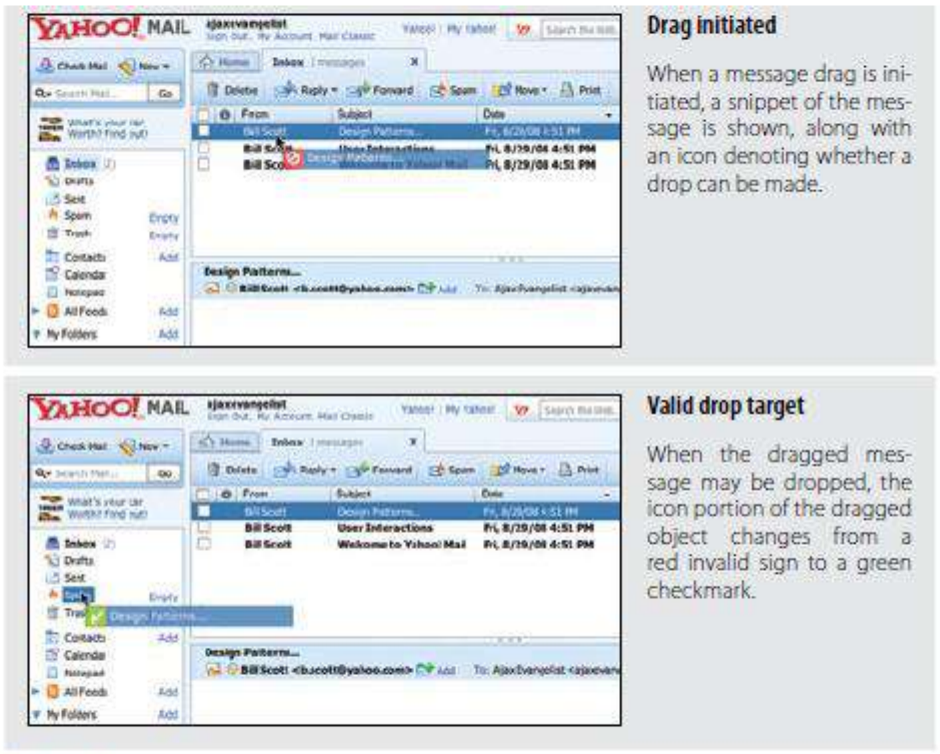


Figure 2-25. *Mind Manager is a desktop tool that uses a combination of insertion targeting plus a clear preview of the drop*

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 (Figure 2-26).



Drag initiated

When a message drag is initiated, a snippet of the message is shown, along with an icon denoting whether a drop can be made.

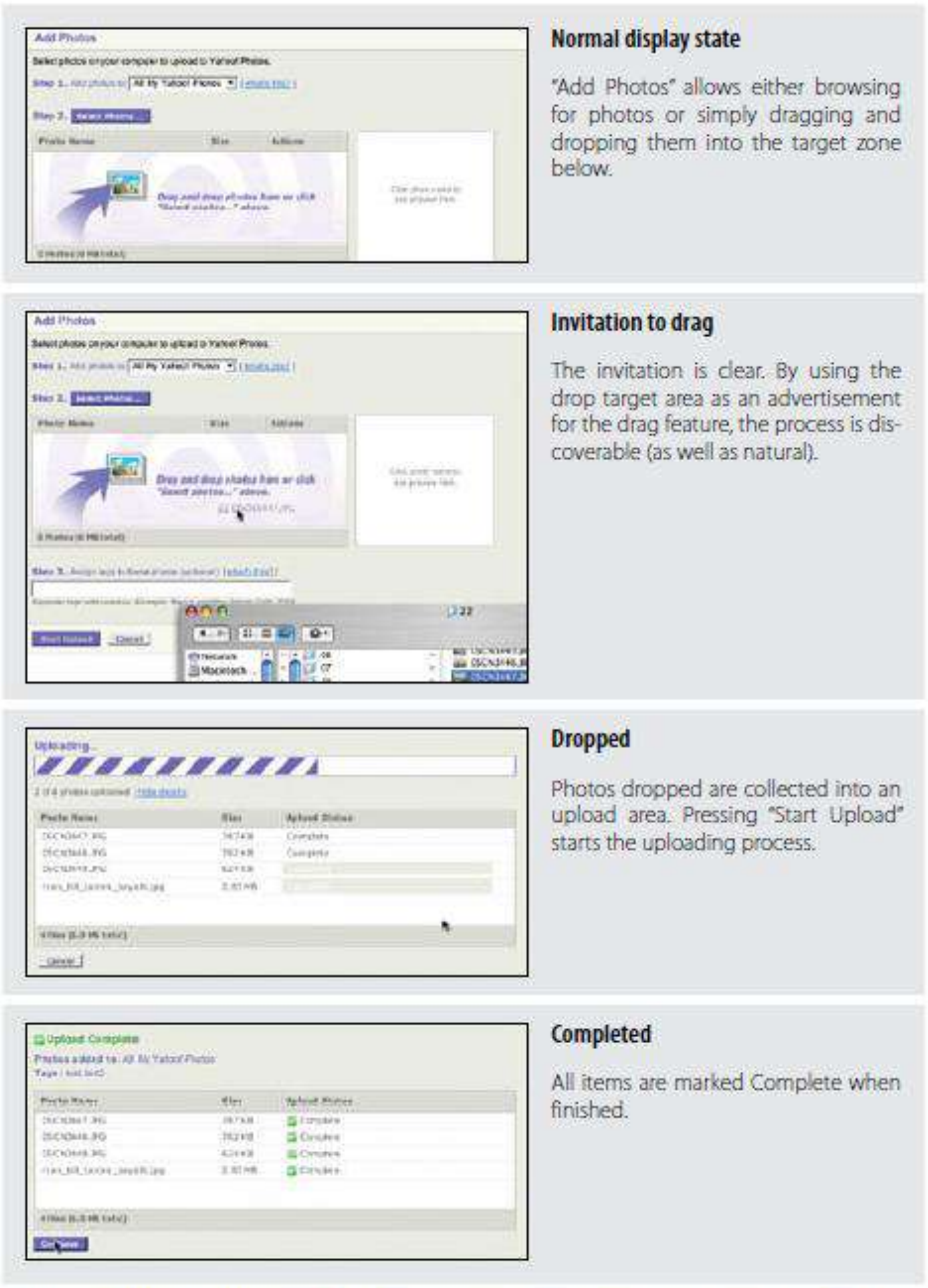
Valid drop target

When the dragged message may be dropped, the icon portion of the dragged object changes from a red invalid sign to a green checkmark.

Figure 2-26. *Yahoo! Mail allows messages to be dragged to folders*

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 and dropping them into the target zone below.

Invitation to drag

The invitation is clear. By using the drop target area as an advertisement for the drag feature, the process is discoverable (as well as natural).

Dropped

Photos dropped are collected into an upload area. Pressing "Start Upload" starts the uploading process.

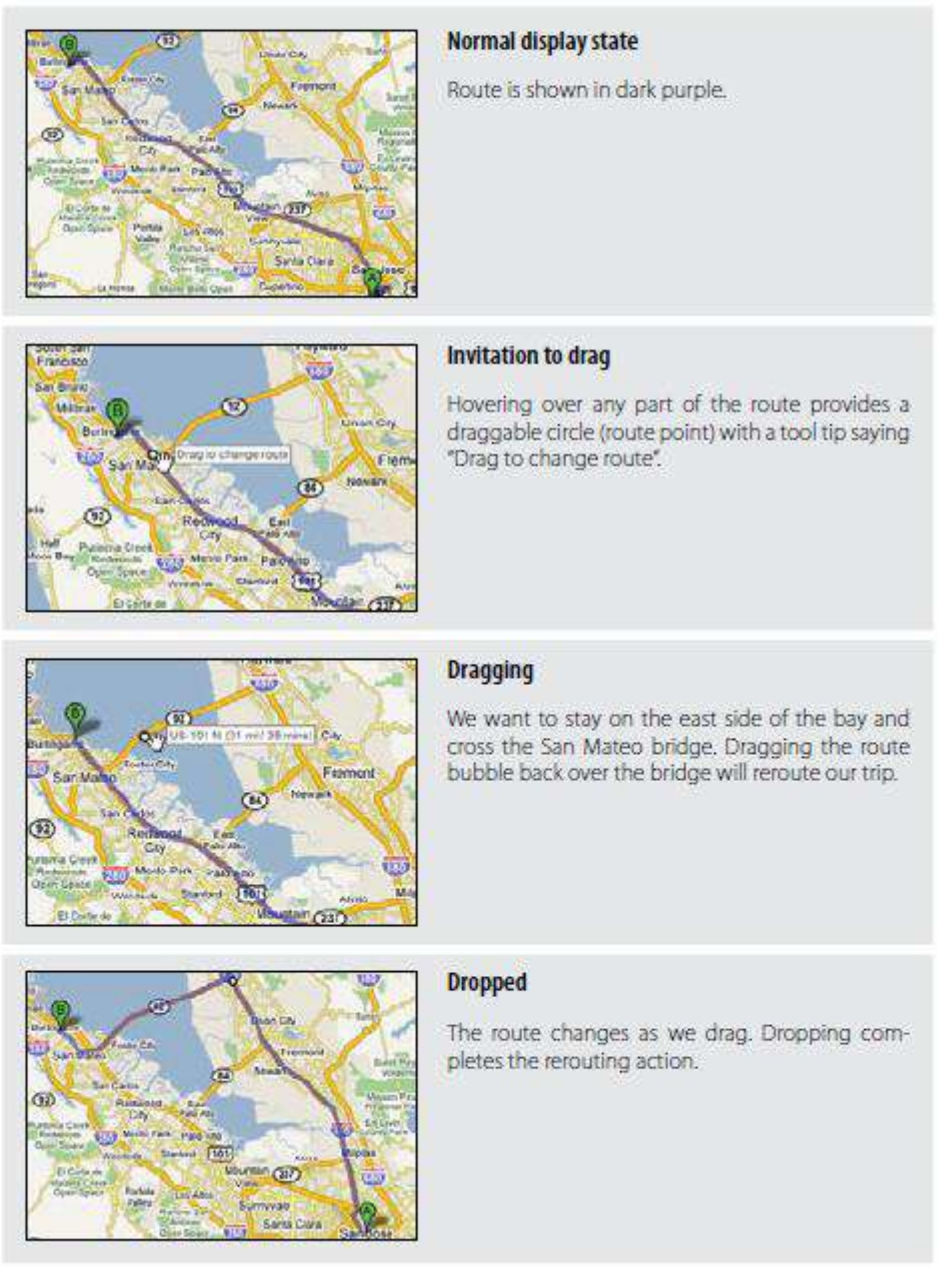
Completed

All items are marked Complete when finished.

Figure 2-30. *Yahoo! Photos provided a way to upload files directly to the site via drag and drop from the user's filesystem into the web page*

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 2-33).



Normal display state

Route is shown in dark purple.

Invitation to drag

Hovering over any part of the route provides a draggable circle (route point) with a tool tip saying "Drag to change route".

Dragging

We want to stay on the east side of the bay and cross the San Mateo bridge. Dragging the route bubble back over the bridge will reroute our trip.

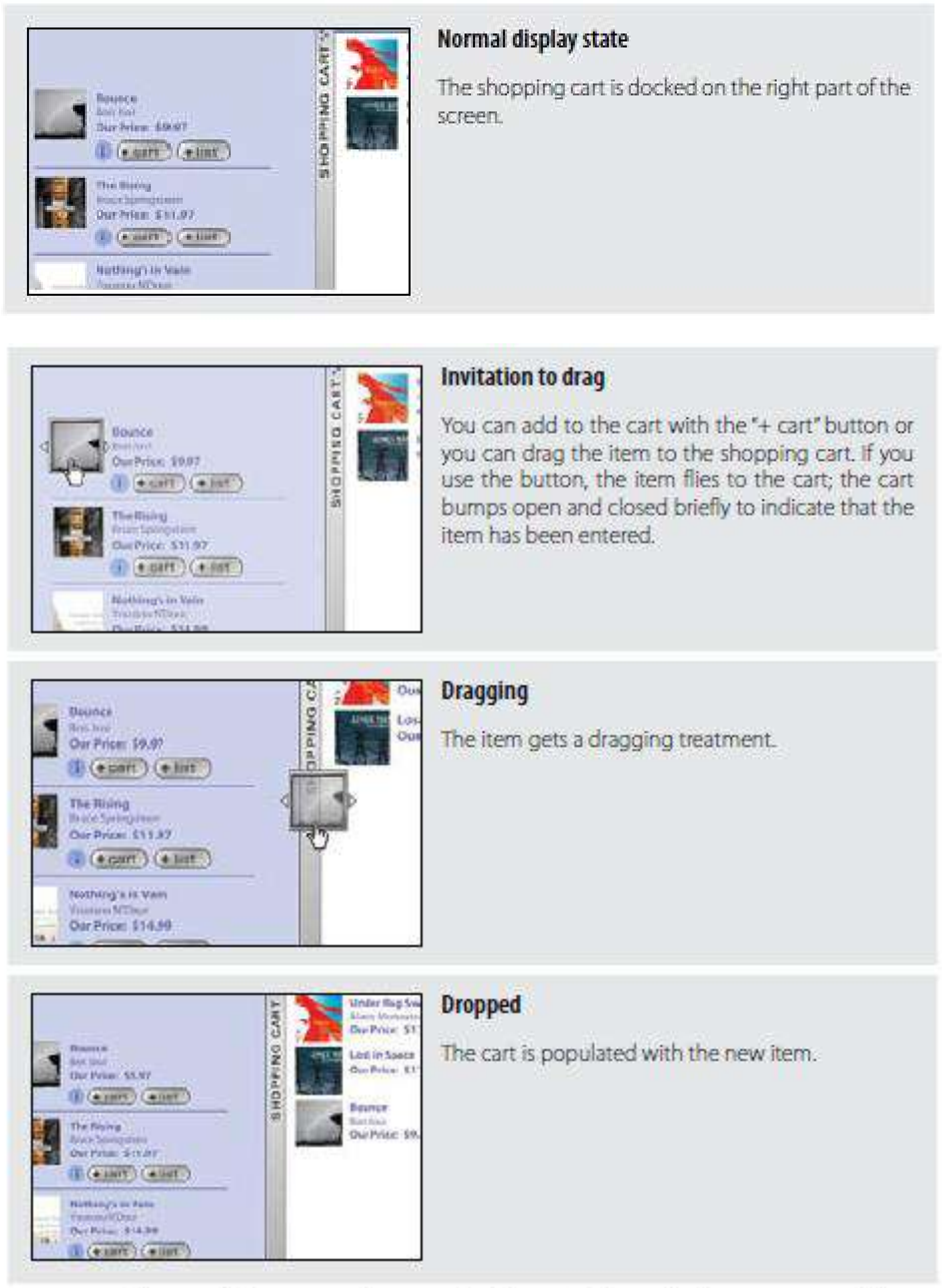
Dropped

The route changes as we drag. Dropping completes the rerouting action.

Figure 2-33. *Rerouting in Google Maps is as simple as drag and drop*

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 2-34).


Normal display state

The shopping cart is docked on the right part of the screen.

Invitation to drag

You can add to the cart with the "+ cart" button or you can drag the item to the shopping cart. If you use the button, the item flies to the cart; the cart bumps open and closed briefly to indicate that the item has been entered.

Dragging

The item gets a dragging treatment.

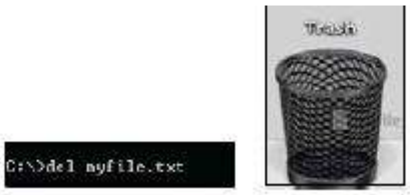
Dropped

The cart is populated with the new item.

Figure 2-34. *This Laszlo shopping cart demo uses both drag and drop and a button action to add items to its shopping cart*

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, you simply dragged a file to the trashcan (Figure 3-1).



```
C:\>del myfile.txt
```

Figure 3-1. *DOS command line for deleting a file versus dragging a file to the trash on the Macintosh*

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 (Figure 3-2).

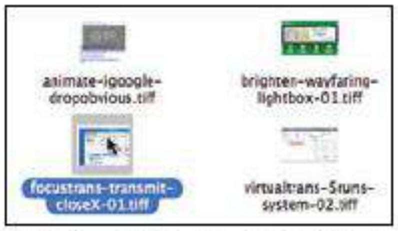
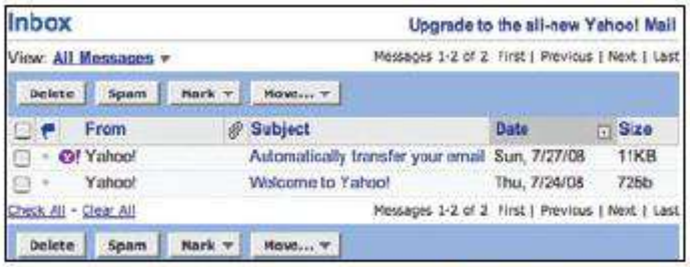

animate-igoogle-dropobvious.tiff
brighten-wayfaring-lightbox-01.tiff
focustrans-transmit-closeX-01.tiff
virtualtrans-5runs-system-02.tiff

Figure 3-2. *Files can be selected directly on the Macintosh; Object Selection is the most common pattern used in desktop applications*

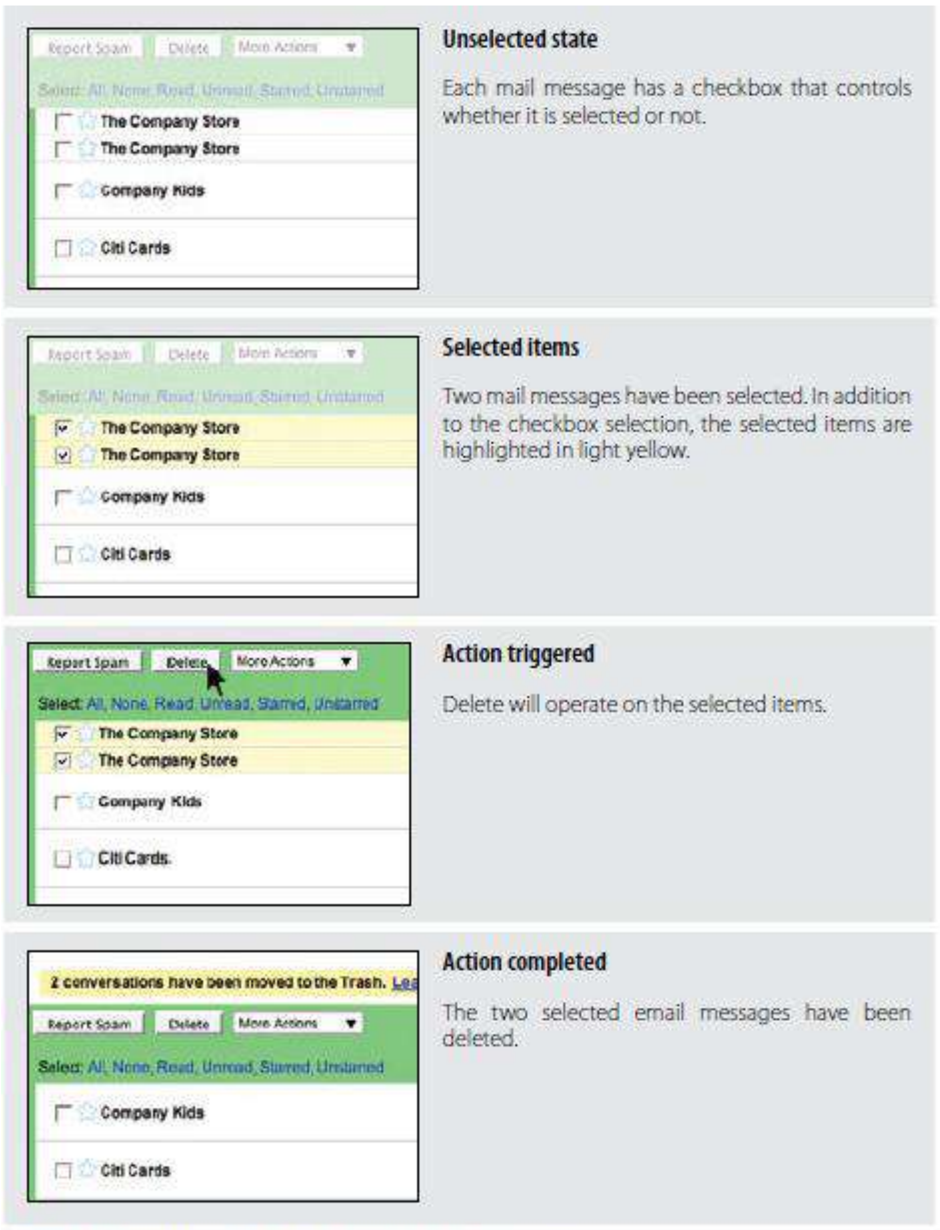
Toggle Selection

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 of this can be seen in Figure 3-3 with Yahoo! Mail Classic.


Inbox
Upgrade to the all-new Yahoo! Mail
View: All Messages
Messages 1-2 of 2 First | Previous | Next | Last
Delete
Spam
Mark
Move...
Check All - Clear All

		From	Subject	Date	Size
☐		Yahoo!	Automatically transfer your email	Sun, 7/27/08	11KB
☐		Yahoo!	Welcome to Yahoo!	Thu, 7/24/08	726b

Figure 3-3. *In Yahoo! Mail Classic a mail message can be selected by clicking on the corresponding row's checkbox*


Report Spam
Delete
More Actions
Select: All, None, Read, Unread, Starred, Unstarred
The Company Store
The Company Store
Company Kids
Citi Cards
2 conversations have been moved to the Trash.

Unselected state

Each mail message has a checkbox that controls whether it is selected or not.

Selected items

Two mail messages have been selected. In addition to the checkbox selection, the selected items are highlighted in light yellow.

Action triggered

Delete will operate on the selected items.

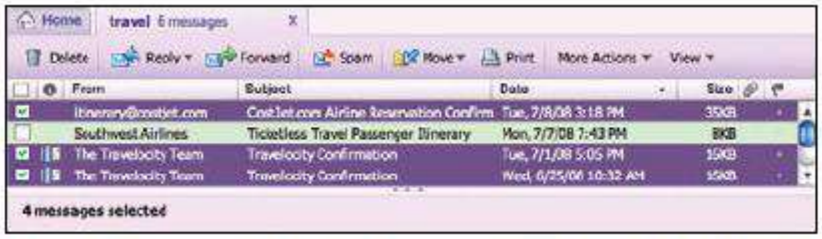
Action completed

The two selected email messages have been deleted.

Figure 3-4. *Gmail uses checkbox selection to operate on messages*

Scrolling versus paging

The previous examples were with paged lists. But what about a scrolled list? Yahoo! Mail uses a scrolled list to show all of its mail messages (Figure 3-5). 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.

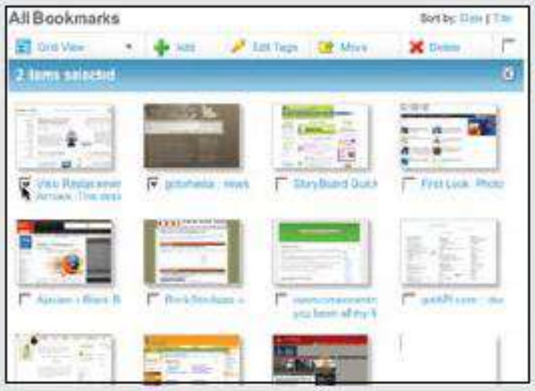


	From	Subject	Date	Size
✓	itinerary@costjet.com	CostJet.com Airline Reservation Confirm	Tue, 7/8/08 3:18 PM	35KB
☐	Southwest Airlines	Ticketless Travel Passenger Itinerary	Mon, 7/7/08 7:43 PM	8KB
✓	The Travelocity Team	Travelocity Confirmation	Tue, 7/1/08 5:05 PM	15KB
✓	The Travelocity Team	Travelocity Confirmation	Wed, 6/25/08 10:32 AM	15KB

4 messages selected

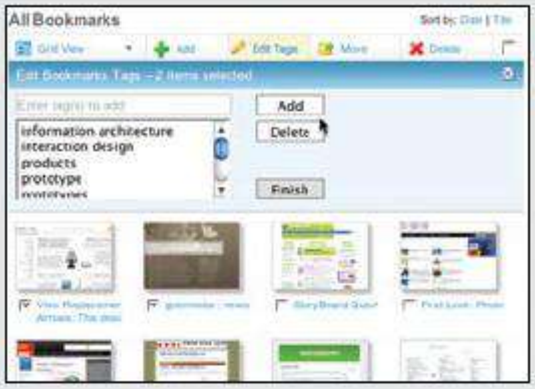
Figure 3-5. *Yahoo! Mail uses a scrolled list for its messages; selection includes what is in the visible part of the list as well as what is scrolled out of view*

Making selection explicit
With Yahoo! Bookmarks you can manage your bookmarks by selecting bookmarked pages and then acting on them. The selection model is visually explicit (Figure 3-6).



Selected items

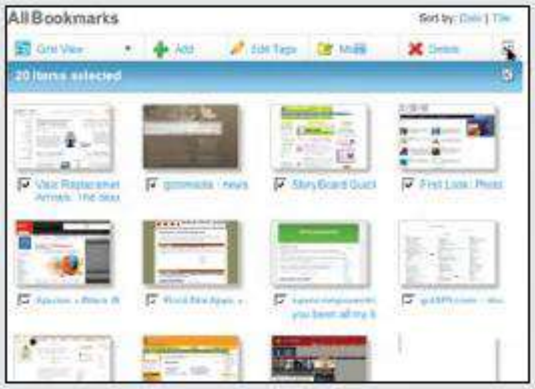
When items get selected, a status bar appears that keeps a tally of the number of items selected. The close button (x) is an alternate way to de-select the selected items.



Tools act on selection

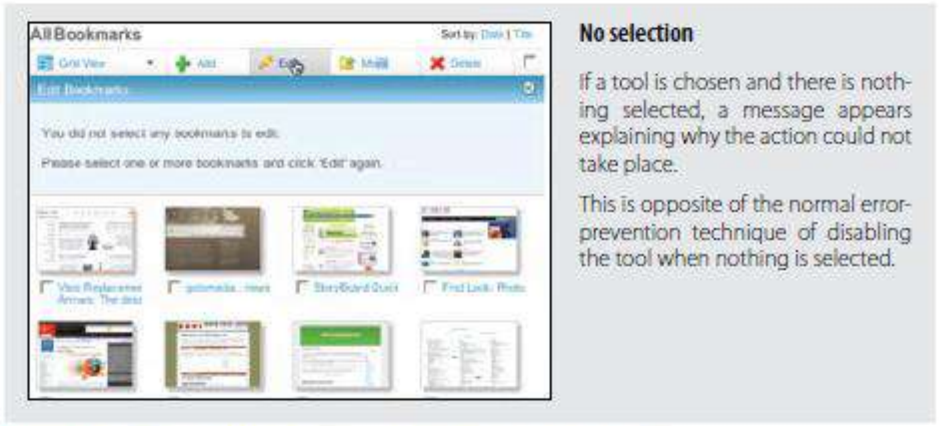
When a tool is activated ("Edit Tags") a command area slides into place.

The status area becomes the title bar for the command area.



Select all

A "select all" checkbox selects all items on the page. The selection status then shows the current number of items selected.



You did not select any bookmarks to edit.

Please select one or more bookmarks and click 'Edit' again.

No selection

If a tool is chosen and there is nothing selected, a message appears explaining why the action could not take place.

This is opposite of the normal error-prevention technique of disabling the tool when nothing is selected.

Figure 3-6. *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.

Collected Selection

Toggle Selection is great for showing a list of items on a single page. But what happens if you 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, you can select items as you move from page to page. The selections are remembered for each page. If you 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.

Keeping the selection visible

The real challenge for multi-page selection is finding a way to show selections gathered across multiple pages. You 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 3-9).

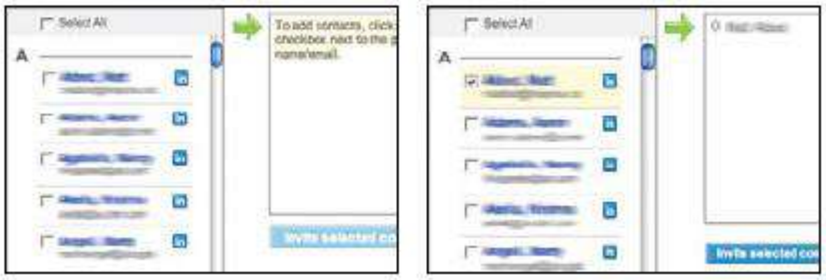
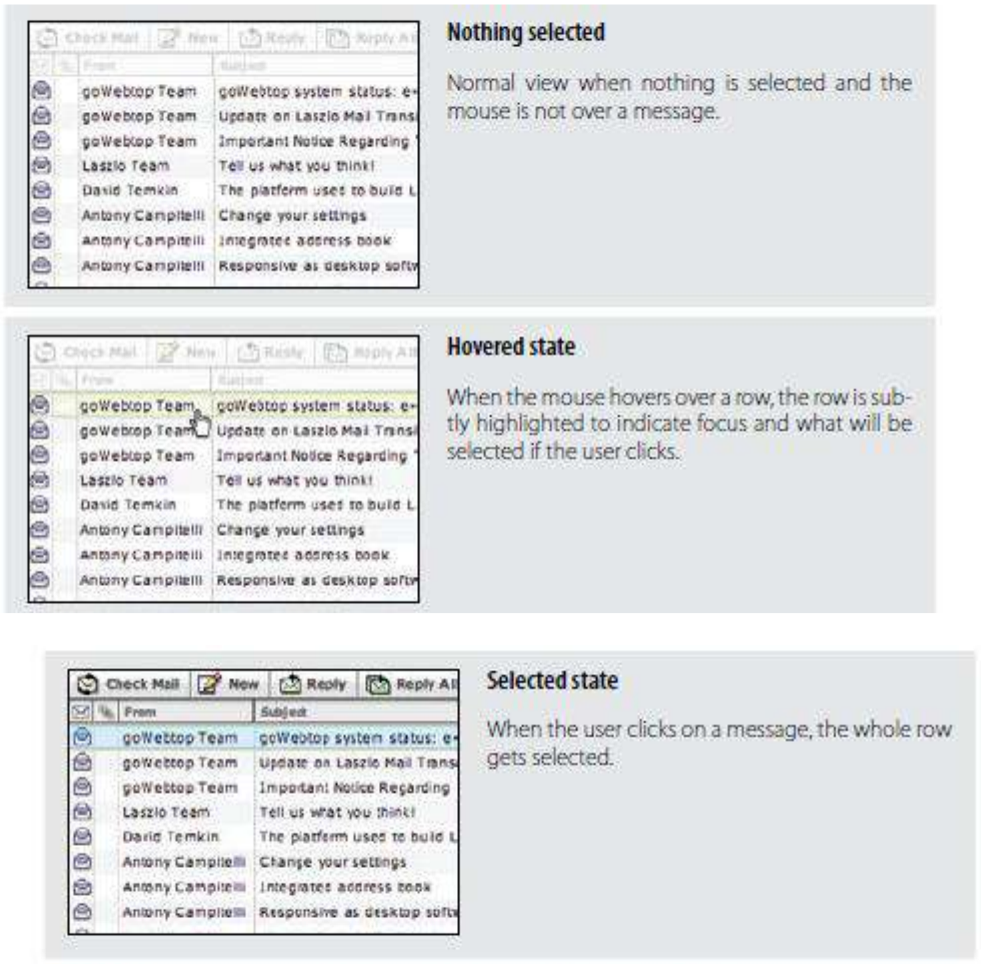


Figure 3-9. *LinkedIn provides a holding place for saving selections across multiple pages*

Object 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 3-11).



Nothing selected

Normal view when nothing is selected and the mouse is not over a message.

From	Subject
goWebtop Team	goWebtop system status: e-
goWebtop Team	Update on Laszlo Mail Trans
goWebtop Team	Important Notice Regarding
Laszlo Team	Tell us what you think!
David Temkin	The platform used to build L
Antony Campitelli	Change your settings
Antony Campitelli	Integrated address book
Antony Campitelli	Responsive as desktop softw

Hovered state

When the mouse hovers over a row, the row is subtly highlighted to indicate focus and what will be selected if the user clicks.

From	Subject
goWebtop Team	goWebtop system status: e-
goWebtop Team	Update on Laszlo Mail Trans
goWebtop Team	Important Notice Regarding
Laszlo Team	Tell us what you think!
David Temkin	The platform used to build L
Antony Campitelli	Change your settings
Antony Campitelli	Integrated address book
Antony Campitelli	Responsive as desktop softw

Selected state

When the user clicks on a message, the whole row gets selected.

From	Subject
goWebtop Team	goWebtop system status: e-
goWebtop Team	Update on Laszlo Mail Trans
goWebtop Team	Important Notice Regarding
Laszlo Team	Tell us what you think!
David Temkin	The platform used to build L
Antony Campitelli	Change your settings
Antony Campitelli	Integrated address book
Antony Campitelli	Responsive as desktop softw

Figure 3-11. *Laszlo WebTop Mail uses highlighting to indicate row selection*

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.

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 3-13). 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, you can drag a selection box around a group of items to add them to the selected set (in this case, photos).

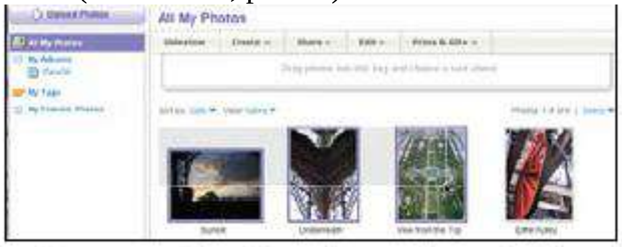


Figure 3-13. *Yahoo! Photos 3.0 created a rich drag selection mechanism for selecting photos*

Hybrid Selection

Mixing Toggle Selection and Object Selection in the same interface can lead to a confusing interface. Referring back to Yahoo! Bookmarks, you'll see an odd situation arise during drag and drop (Figure 3-14).

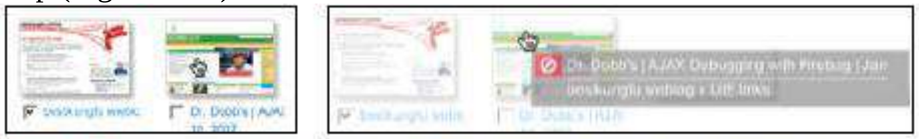


Figure 3-14. *In Yahoo! Bookmarks, one item is selected, but two items can be dragged by dragging on the unselected item*

Contextual Tools

Interaction in Context

Most 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.

However, this simplistic approach no longer exists in the current web application landscape.
As the Web has matured, a wide variety of application styles has emerged. On one end of the spectrum there are simple sites that need no more functionality than the hyperlink and a "Submit" button. On the other end of the spectrum there are full applications hosted as a website. Google Search and Yahoo! Mail are two typical applications that illustrate this variation (Figure 4-1).

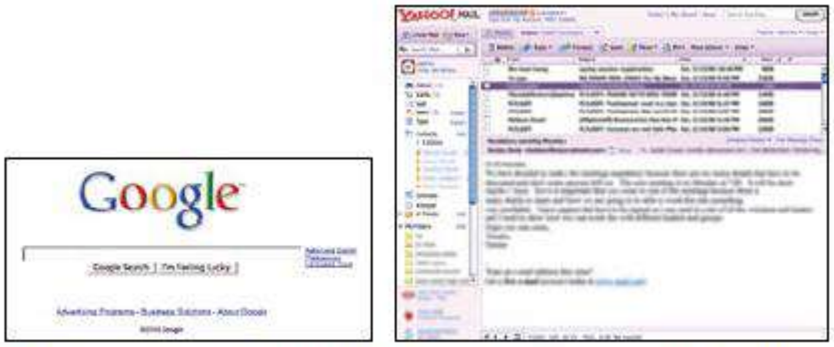


Figure 4-1. Google Search needs only hyperlinks and a search button; Yahoo! Mail, on the other hand, is a full-featured application with toolbars and menus

Figure 4-2. The Apple iPhone introduced touch-based interfaces to the consumer market

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.

Contextual Tools

We could simply isolate our functionality into islands of tools (toolbars and menus). But this would work against Fitts's Law by requiring more effort from the user. It would also add more visual weight to the page. Instead of interacting with the functionality separately, we can bring the functionality into the content with 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:

Always-Visible Tools

Place Contextual Tools directly in the content.

Hover-Reveal Tools

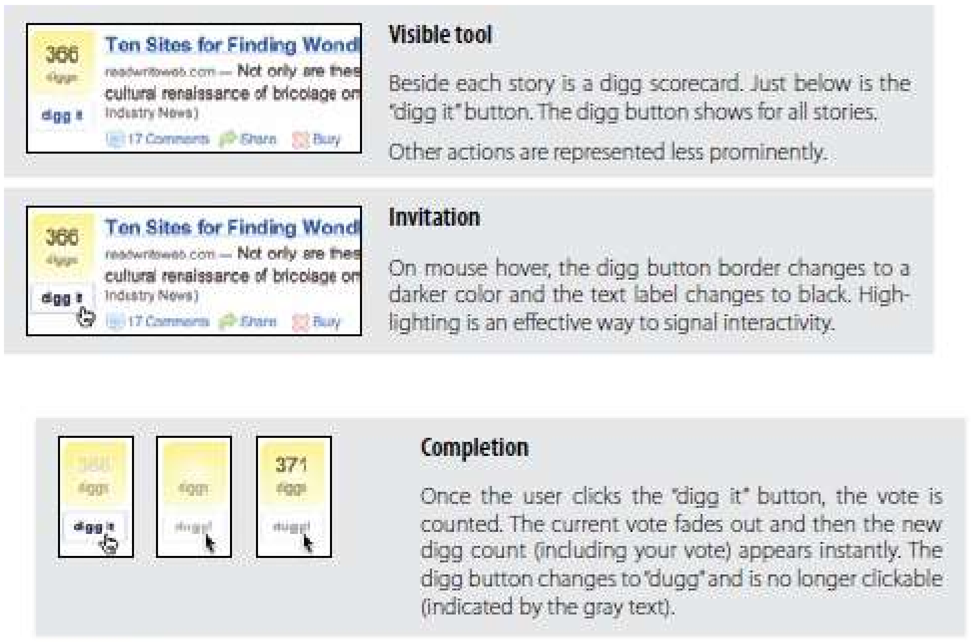
Show Contextual Tools on mouse hover.

Toggle-Reveal Tools

A master switch to toggle on/off Contextual Tools for the page.

Multi-Level Tools
Progressively reveal actions based on user interaction.
Secondary Menus
Show a secondary menu (usually by right-clicking on an object).

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 4-3).


366
diggs
digg it
Ten Sites for Finding Wond
readwriteweb.com — Not only are thes
cultural renaissance of bricolage on
Industry News)
17 Comments
Share
Bury
371
dugg!

Visible tool

Beside each story is a digg scorecard. Just below is the "digg it" button. The digg button shows for all stories.

Other actions are represented less prominently.

Invitation

On mouse hover, the digg button border changes to a darker color and the text label changes to black. Highlighting is an effective way to signal interactivity.

Completion

Once the user clicks the "digg it" button, the vote is counted. The current vote fades out and then the new digg count (including your vote) appears instantly. The digg button changes to "dugg" and is no longer clickable (indicated by the gray text).

Figure 4-3. *Digg's "digg it" button is a simple Contextual Tool that is always visible*

Relative importance
One way to clarify this process is to decide on the relative importance of each exposed action. Is the "digg it" action as important as the "bury it" action? In the case of Digg, the answer is no. 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 you 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 4-5).

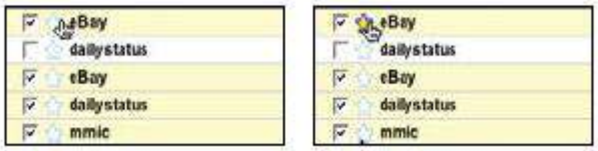
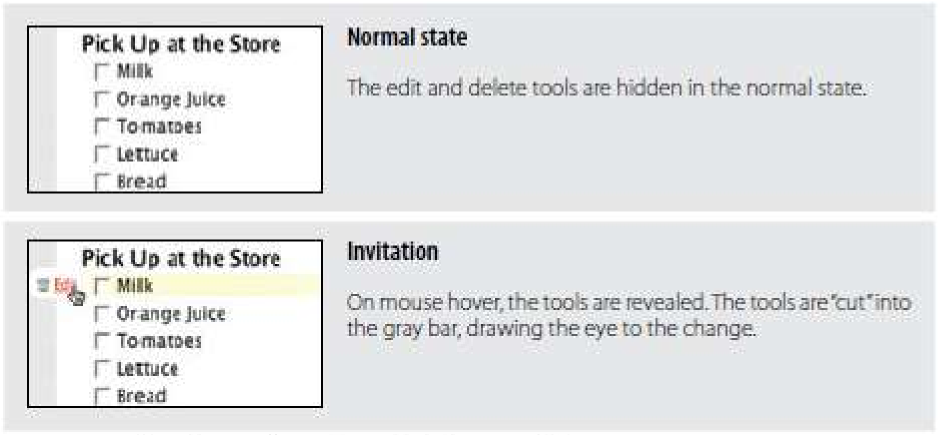


Figure 4-5. *Google Mail uses Contextual tools to flag favorites*

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 4-8). To-do items may be deleted or edited directly in the interface. The tools to accomplish this are revealed on mouse hover.



The edit and delete tools are hidden in the normal state.

On mouse hover, the tools are revealed. The tools are "cut" into the gray bar, drawing the eye to the change.

Figure 4-8. *Backpackit reveals its additional tools on mouse hover*

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. Figure 4-9 shows just how visually noisy that approach would have been.

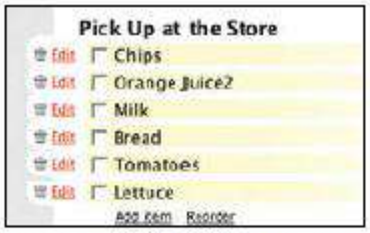
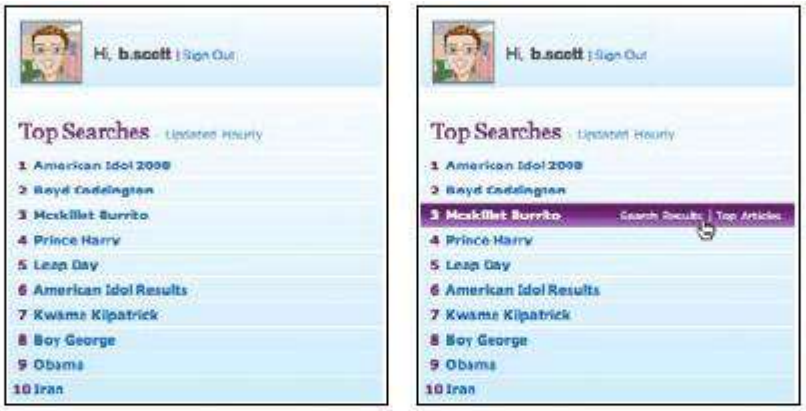


Figure 4-9. *What the Backpackit interface would have looked like if the Contextual Tools were always visible*

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.



Hi, b.scott | Sign Out

Top Searches Updated Hourly

1 American Idol 2008
2 Boyd Coddington
3 Mcskillet Burrito
4 Prince Harry
5 Leap Day
6 American Idol Results
7 Kwame Kilpatrick
8 Boy George
9 Obama
10 Iran

Hi, b.scott | Sign Out

Top Searches Updated Hourly

1 American Idol 2008
2 Boyd Coddington
3 Mcskillet Burrito Search Results | Top Articles
4 Prince Harry
5 Leap Day
6 American Idol Results
7 Kwame Kilpatrick
8 Boy George
9 Obama
10 Iran

Figure 4-10. *Yahoo! Buzz reveals additional tools for the top searches when the user hovers over each item*

Discoverability
A serious design consideration for Hover-Reveal Tools is just how discoverable the additional functionality will be. In the earlier Backpackit example (Figure 4-8), 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.
Flickr provides a set of tools for contacts. To avoid clutter, contact profile photos are shown without any tool adornment. When the mouse hovers over the contact's photo, a drop-down arrow is revealed (Figure 4-12). Clicking reveals a menu with a set of actions for the contact. This works because users often know to click on an image to get more information. Being drawn to the content is a good way to get the user to move the mouse over the area and discover the additional functionality.

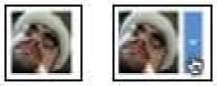


Figure 4-12. *Flickr reveals the drop-down menu on hover*

Anti-pattern: Hover and Cover
Figure 4-14 illustrates all four of these situations. In an early version of Yahoo! For Teachers,* hovering over a clipped item brought in three tools: copy, delete, and preview. However, when these tools were placed in an overlay, it covered the item to the right, making it hard to see that content and even navigate to it. In addition, since the overlay had some additional padding (as well as rounded corners), the image shown in the overlay was about two pixels off from the non-overlay version. This slight jiggle was distracting. To add insult to injury, the overlay was sluggish to bring into view.

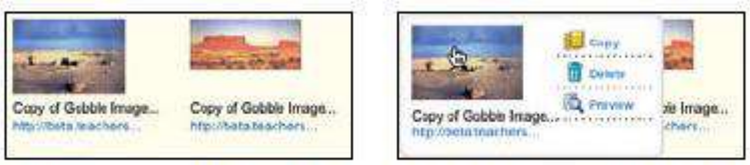


Figure 4-14. *An early version of the Yahoo! for Teachers beta revealed Contextual Tools in an overlay; the overlay covered more than half of the item to its right*

The final straw was if users wanted to delete several items, they would hover over the image, wait for the overlay, click Delete, then be forced to move out and back in again to activate the next image's Contextual Tools (Figure 4-15). Hover and Cover is a common anti-pattern that occurs when exposing an overlay on hover and hiding important context

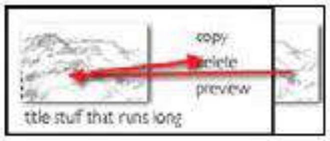


Figure 4-15. *Navigating required a zigzag approach to get around the tool overlay*

or further navigation.

Hover and Cover was resolved by no longer using an overlay. Instead, additional margin space was added to each image, and the Contextual Tools were hidden. On mouse hover, the tools were simply revealed, along with a border defining the image being acted on (Figure 4-16).

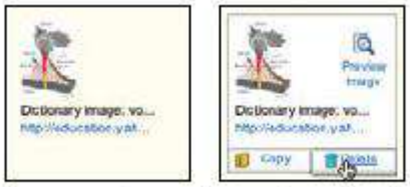
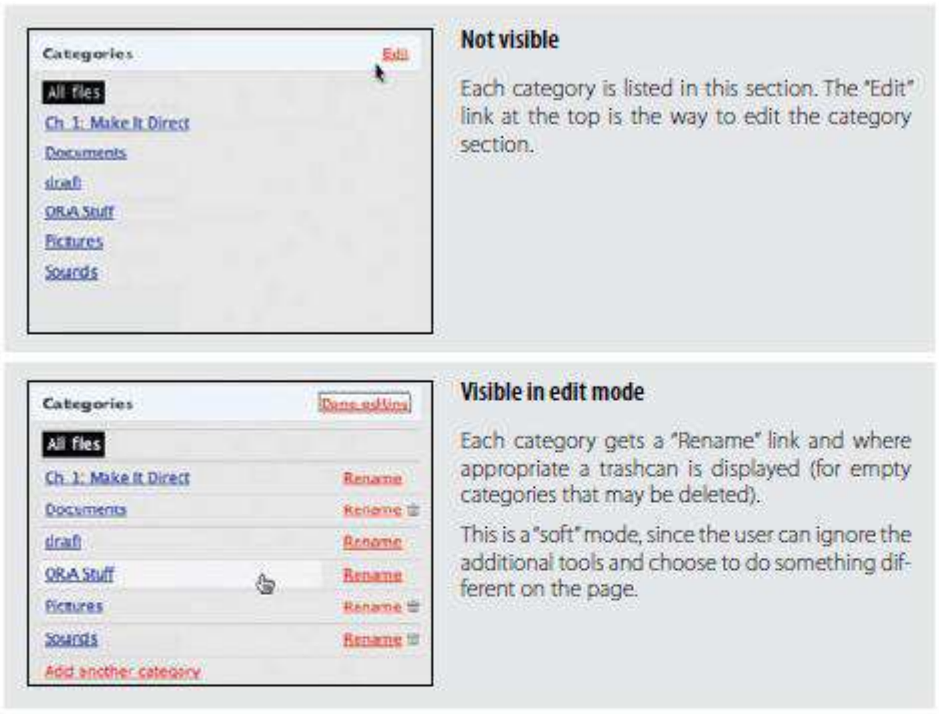


Figure 4-16. *In the redesigned version, tools were shown on hover directly surrounding the image instead of in an overlay*

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, which we discussed in Chapter 1 (Figure 4-19).



Not visible

Each category is listed in this section. The "Edit" link at the top is the way to edit the category section.

Visible in edit mode

Each category gets a "Rename" link and where appropriate a trashcan is displayed (for empty categories that may be deleted).

This is a "soft" mode, since the user can ignore the additional tools and choose to do something different on the page.

Figure 4-19. *Basecamp reveals category-editing tools only when the edit mode is turned on for the area*

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.

When would you use this technique? When the actions are not the main thing and you want to reduce visual noise. This fits the category example perfectly. Items are renamed or deleted occasionally. 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 4-20). This would allow the user to rename and unsubscribe without leaving the context of the reading pane.

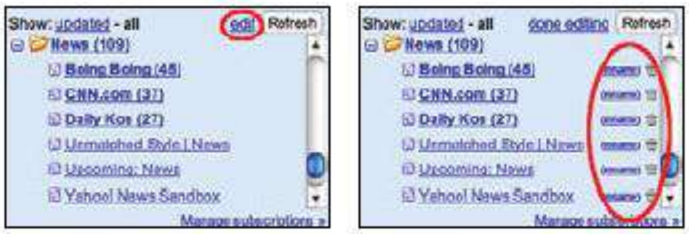
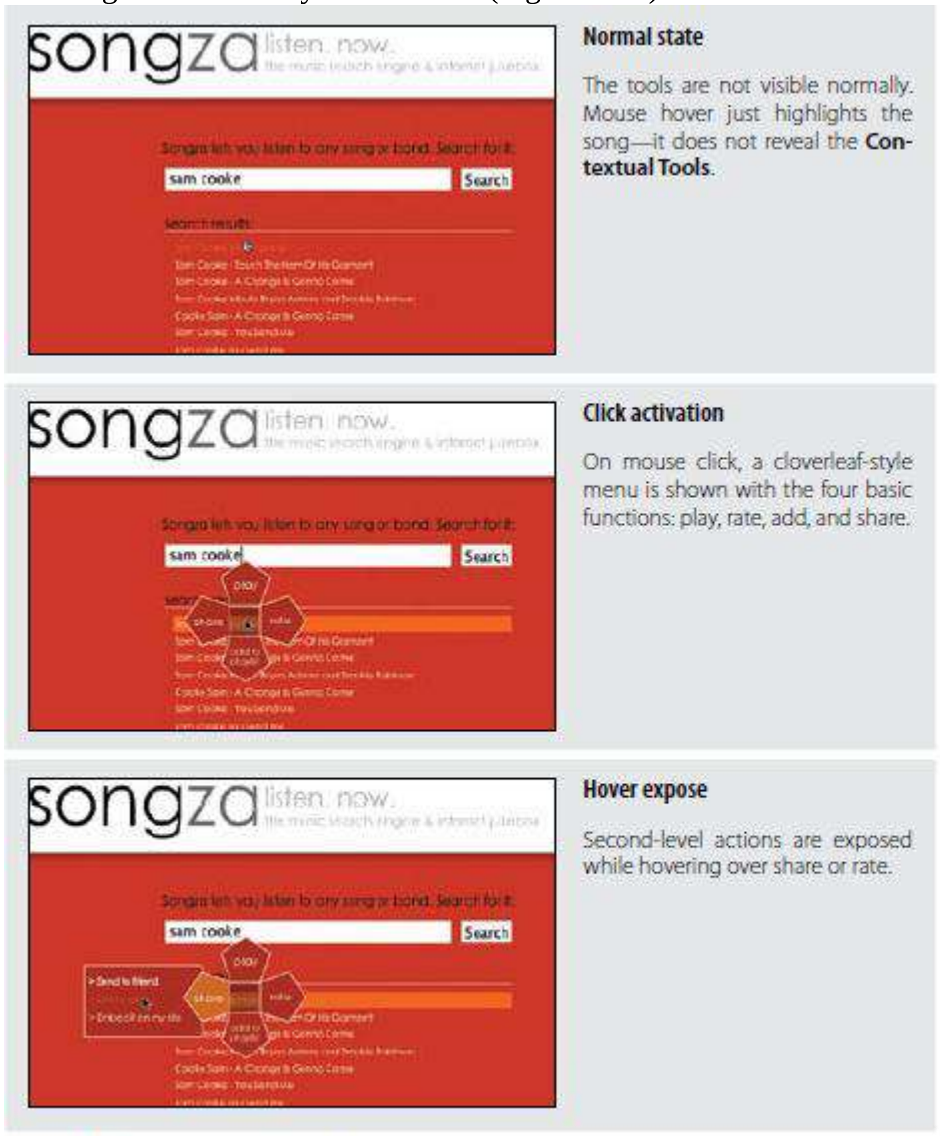


Figure 4-20. *Adding an "edit" link to Google Reader's feed list and toggling in common actions could potentially make it easier to manage subscriptions*

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 (Figure 4-21).



Normal state

The tools are not visible normally. Mouse hover just highlights the song—it does not reveal the **Contextual Tools**.

Click activation

On mouse click, a cloverleaf-style menu is shown with the four basic functions: play, rate, add, and share.

Hover expose

Second-level actions are exposed while hovering over share or rate.

Figure 4-21. *Songza uses a multi-level contextual tool 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 4-22).

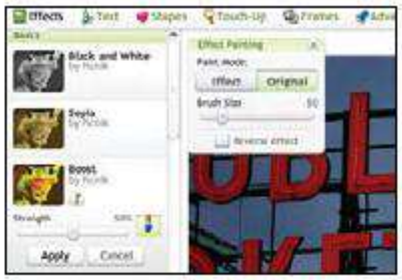
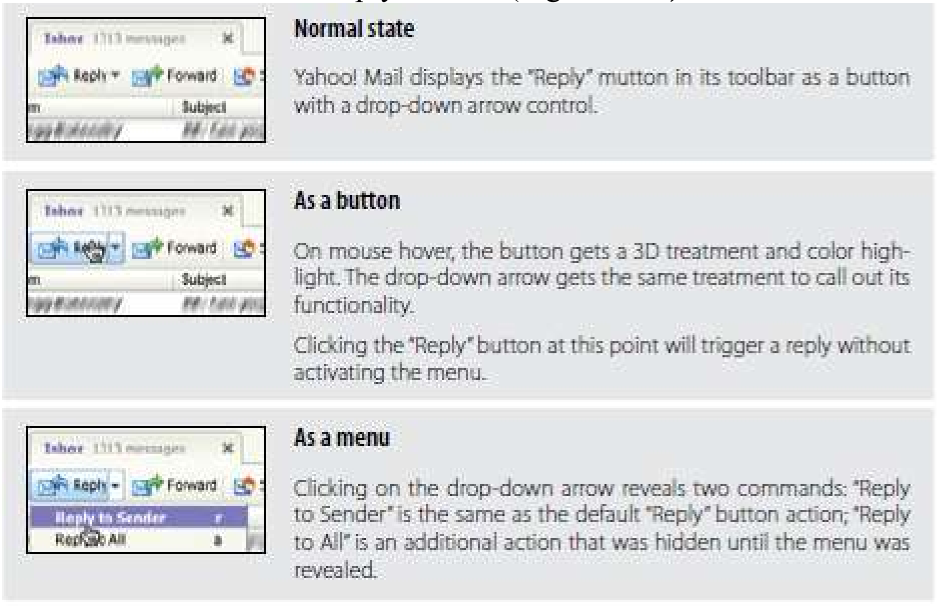


Figure 4-22. *Picnik wraps layers of Contextual Tools around the image being edited*

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 4-23).



Normal state

Yahoo! Mail displays the "Reply" mutton in its toolbar as a button with a drop-down arrow control.

As a button

On mouse hover, the button gets a 3D treatment and color highlight. The drop-down arrow gets the same treatment to call out its functionality.

Clicking the "Reply" button at this point will trigger a reply without activating the menu.

As a menu

Clicking on the drop-down arrow reveals two commands: "Reply to Sender" is the same as the default "Reply" button action; "Reply to All" is an additional action that was hidden until the menu was revealed.

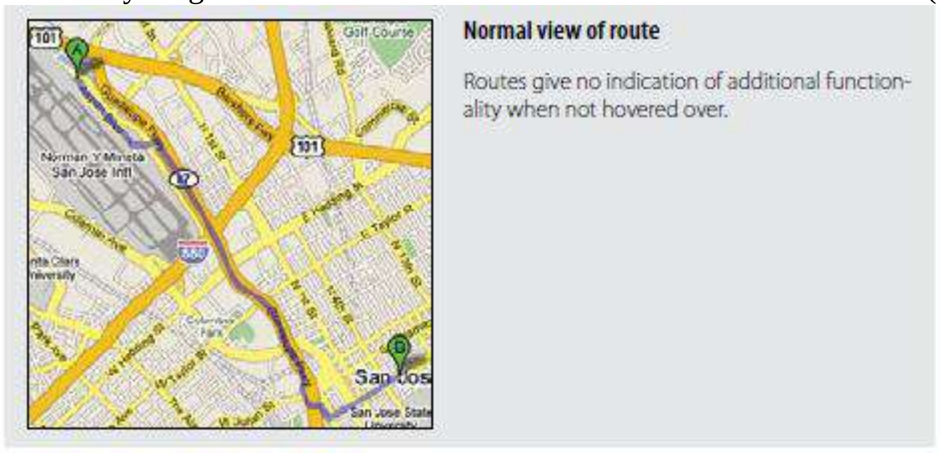
Figure 4-23. *Yahoo! Mail's "Reply" button looks like a drop-down when hovered over; clicking "Reply" replies to sender, and clicking the drop-down offers the default action as well as "Reply to All"*

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 defa • ult 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. Only after the mutton was made more accessible could he find the “Reply” command.

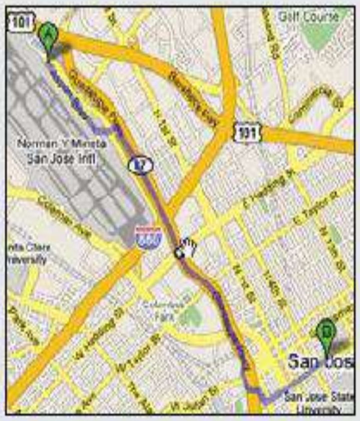
Secondary Menu

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 4-25).



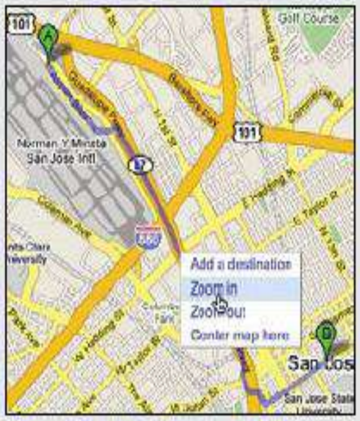
Normal view of route

Routes give no indication of additional functionality when not hovered over.



Invitation

When the mouse is over the route, potential stops are marked with a white circle.



Menu

Right-clicking on the item exposes four commands that act on the point selected: "Add a destination", "Zoom in", "Zoom out", and "Center map here".

Figure 4-25. *Google Maps uses a right-click menu to add new route stops or to adjust the map around the current point on the route*

Overlays

Overlays are really just lightweight pop ups. We use the term *lightweight* to make a clear distinction between it and the normal idea of a *browser pop up*. Browser pop ups are created as a new browser window (Figure 5-1). *Lightweight overlays* are shown within the browser page as an overlay (Figure 5-2). Older style browser pop ups are undesirable because:

Browser pop ups display a new browser window. As a result • these windows often take
time and a sizeable chunk of system resources to create.
• Browser pop ups often display browser interface controls (e.g., a URL bar). Due to security concerns, in Internet Explorer 7 the URL bar is a permanent fixture on any browser pop-up window.

Dialog Overlay

Dialog Overlays replace the old style browser pop ups. Netflix provides a clear example of a very simple Dialog Overlay. In the "previously viewed movies for sale" section, a user can click on a "Buy" button to purchase a DVD. Since the customer purchasing the DVD is a member of Netflix, all the pertinent shipping and purchasing information is already

on record. The complete checkout experience can be provided in a single overlay (Figure
5-
3).

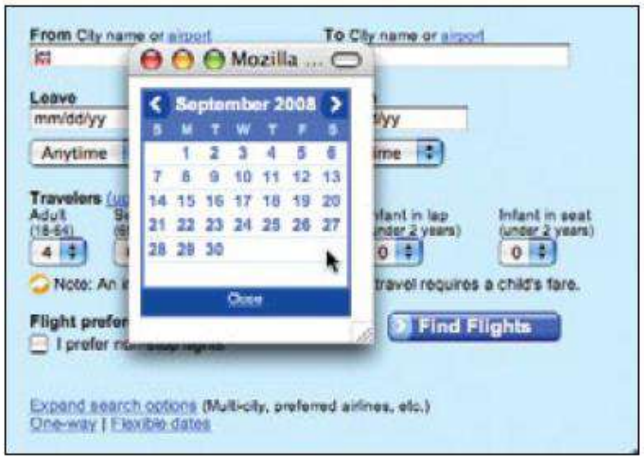


Figure 5-1. *If Orbitz used a browser pop-up window for its calendar chooser (it does not), this is how it might look*

Dialog Overlay

Dialog Overlays replace the old style browser pop ups. Netflix provides a clear example of a very simple Dialog Overlay. In the "previously viewed movies for sale" section, a user can click on a "Buy" button to purchase a DVD. Since the customer purchasing the DVD is a member of Netflix, all the pertinent

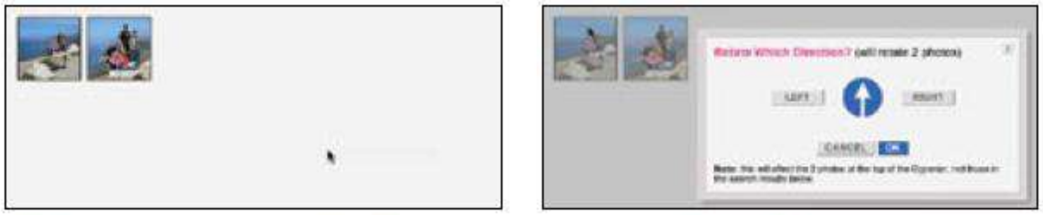
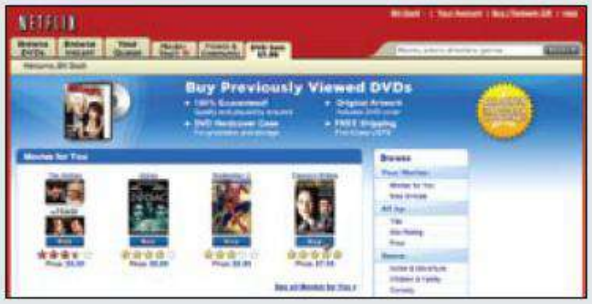


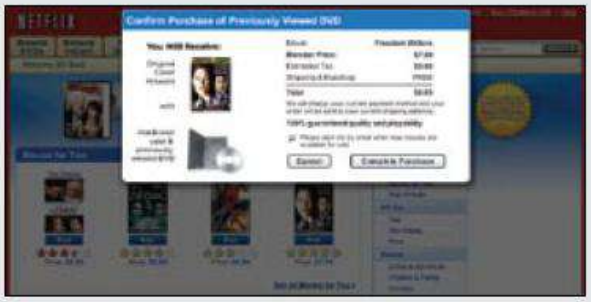
Figure 5-4. *Flickr also uses a Lightbox Effect to focus attention on the overlay*

shipping and purchasing information is already on record. The complete checkout experience can be provided in a single overlay (Figure 5-3).



Activation

Clicking the "Buy" button initiates the purchase process.



Overlay treatment

The confirmation dialog is shown in a lightweight overlay. Since the overlay is modal (interaction is only accepted in the overlay) the rest of the page is dimmed down. The user may also cancel the purchase.

Figure 5-3. *Netflix uses a lightweight pop up to confirm a previously viewed DVD purchase; in addition, it uses the Lightbox Effect to indicate modality*

The Lightbox Effect is useful when the Dialog Overlay contains important information that the user should not ignore. Both the Netflix Purchase dialog and the Flickr Rotate dialog are good candidates for the Lightbox Effect. If the overlay contains optional information, then the Lightbox Effect is overkill and should not be used.

Overlays can be modal* or non-modal. A modal overlay requires the user to interact with
it before she can return to the application. In both the Netflix example (Figure 5-3) and the Flickr example (Figure 5-4), the overlays are *modal*: users cannot interact with the main Netflix or Flickr page until they perform the action or cancel the overlay. In both cases, modality is reinforced with the Lightbox Effect. Dimming down the background cues the user that this portion of the interface cannot be interacted with.

Sometimes overlays are non-modal. An example of this can be found in the Netflix site. When a DVD is added to the user's shipping list (queue), a confirmation overlay is shown (Figure 5-5). While it may appear that the only way to dismiss the overlay is by clicking the "Close" box in the upper-right corner, in reality the user can click anywhere outside the overlay (in the dimmed area) and the overlay will dismiss. In this case the Lightbox Effect is used to focus the user's attention on the confirmation and recommendations available.

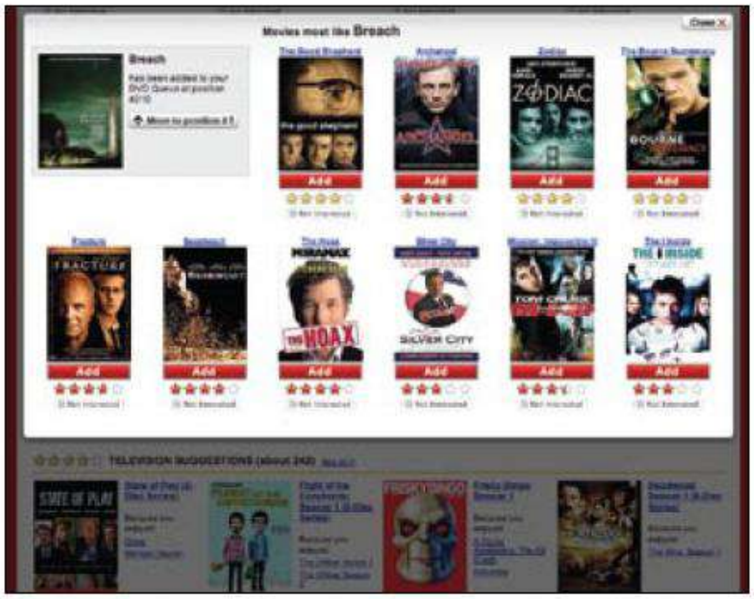
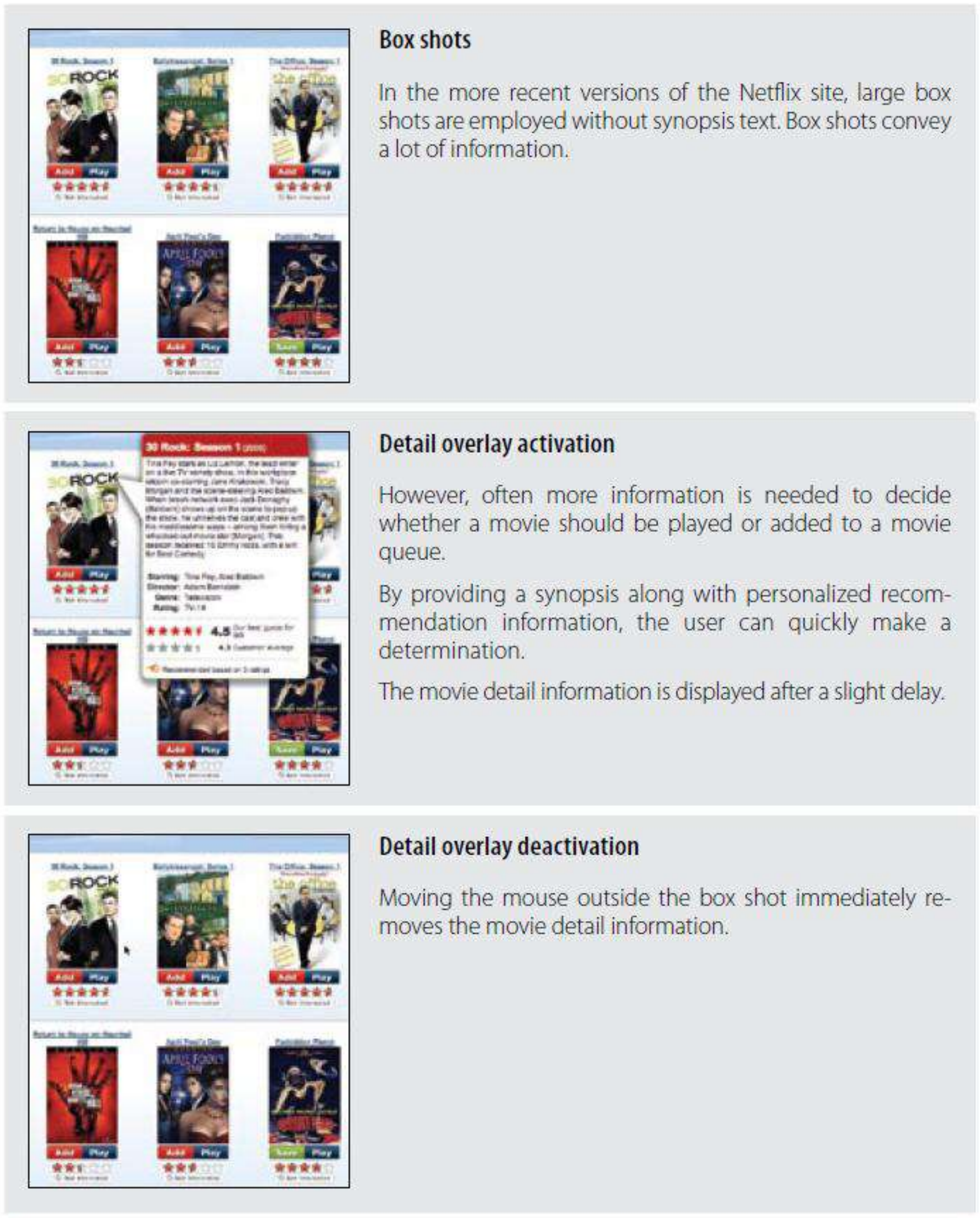


Figure 5-5. *Netflix uses a non-modal overlay with the Lightbox Effect to focus attention on the confirmation and recommendation*

The Lightbox Effect emphasizes that we are in a separate mode. As a consequence, it is not needed for most non-modal overlays. As an example, refer back to Figure 5-2, the Orbitz calendar pop up. Since the overlay is really more like an in-page widget, it would not be appropriate to make the chooser feel heavier by using a Lightbox Effect.

Detail Overlay

The second type of overlay is somewhat new to web applications. The Detail Overlay allows an overlay to present additional information when the user clicks or hovers over a link or section of content. Toolkits now make it easier to create overlays across different browsers and to request additional information from the server without refreshing the page. Taking another example from Netflix, information about a specific movie is displayed as the user hovers over the movie's box shot (Figure 5-8).



Box shots

In the more recent versions of the Netflix site, large box shots are employed without synopsis text. Box shots convey a lot of information.

Detail overlay activation

However, often more information is needed to decide whether a movie should be played or added to a movie queue.

By providing a synopsis along with personalized recommendation information, the user can quickly make a determination.

The movie detail information is displayed after a slight delay.

Detail overlay deactivation

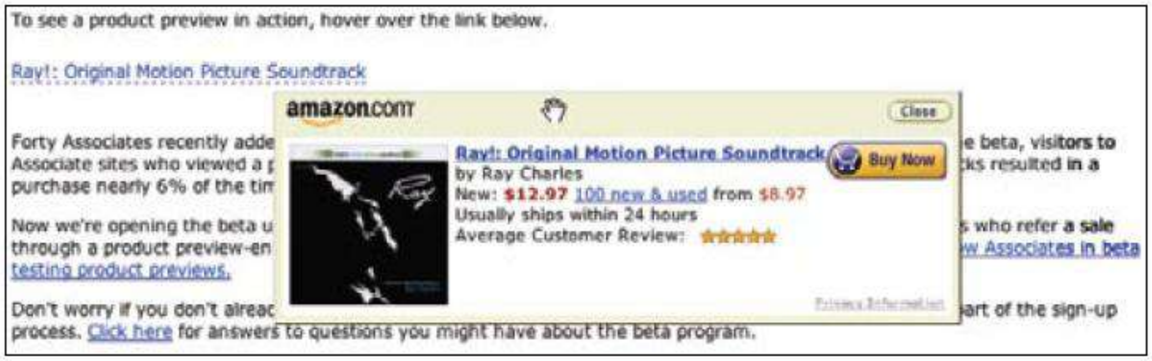
Moving the mouse outside the box shot immediately removes the movie detail information.

Figure 5-8. *Netflix shows "back of the box" information in an overlay as the user hovers over a movie's box shot*

Anti-pattern: Mouse Traps

It is important to avoid activating the Detail Overlay too easily. We have seen usability studies that removed the delay in activation, and users reported that the interface was "too noisy" 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". In Figure 5-10 the link "Ray! Original Motion Picture Soundtrack" activates an overlay providing information on the soundtrack and a purchase option. Presumably, this approach is intended to drive purchases—but it also presents an annoying experience.

Anti-pattern: Non-Symmetrical Activation/Deactivation
When the user moves her mouse over the link, the overlay springs up immediately. The only way she can remove the overlay is by clicking the small close button in the upper right. Using Non-Symmetrical Activation/Deactivation is also a general anti-pattern that should be avoided. It should take the same amount of effort to dismiss an overlay as it took to open it. Compare the Amazon approach to both the Netflix and Yahoo! News approaches. The activation is slightly harder (pause, slight delay) than the deactivation (immediate when mouse is moved away).


To see a product preview in action, hover over the link below.

Ray!: Original Motion Picture Soundtrack

Forty Associates recently adde … e beta, visitors to
Associate sites who viewed a p … cks resulted in a
purchase nearly 6% of the tim

Now we're opening the beta u … s who refer a sale
through a product preview-en … w Associates in beta
testing product previews.

Don't worry if you don't alreac … art of the sign-up
process. Click here for answers to questions you might have about the beta program.

Ray!: Original Motion Picture Soundtrack
by Ray Charles
New: $12.97 100 new & used from $8.97
Usually ships within 24 hours
Average Customer Review: ★★★★★

Figure 5-10. *Amazon shows a book-buying widget on simple hover—but requires clicking the "Close" box to dismiss it*

Another example of Non-Symmetrical Activation/Deactivation turns up in a previous version of Yahoo! Foods (Figure 5-11). To see all main ingredients for a recipe, the user clicked a red arrow. This activated an overlay with the ingredients. However, clicking on the arrow again did not collapse the overlay. Instead, the user had to click on the close button (red X).

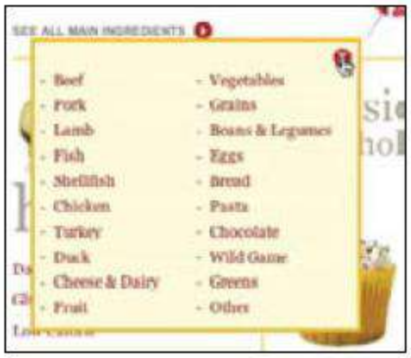


Figure 5-11. *Yahoo! Foods All Main Ingredients drop-down is activated by clicking on the arrow and can only be deactivated by clicking the close button (X)*

Anti-pattern: Needless Fanfare
One of the advantages of a lightweight overlay is the ability to pop it up quickly. After a slight delay in activation (recall the half-second delay used by Netflix), you would not want or need the overlay to come up slowly. But in the case of Borders online, this is precisely the approach taken (Figure 5-12). First the activation is immediate (no delay). This creates the noisy, mouse-trap interface just discussed. Second, there's a needless animation that zooms the box up into place and then back down when the mouse moves away from a book. Needless Fanfare is an anti-pattern to avoid.

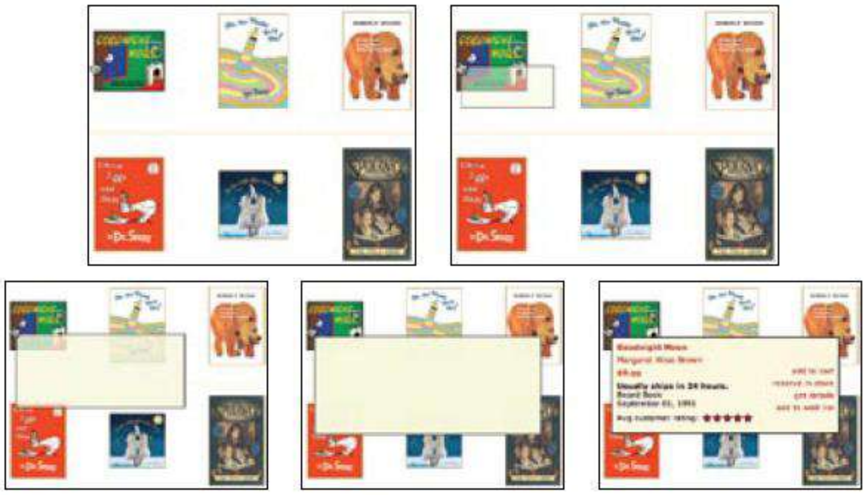
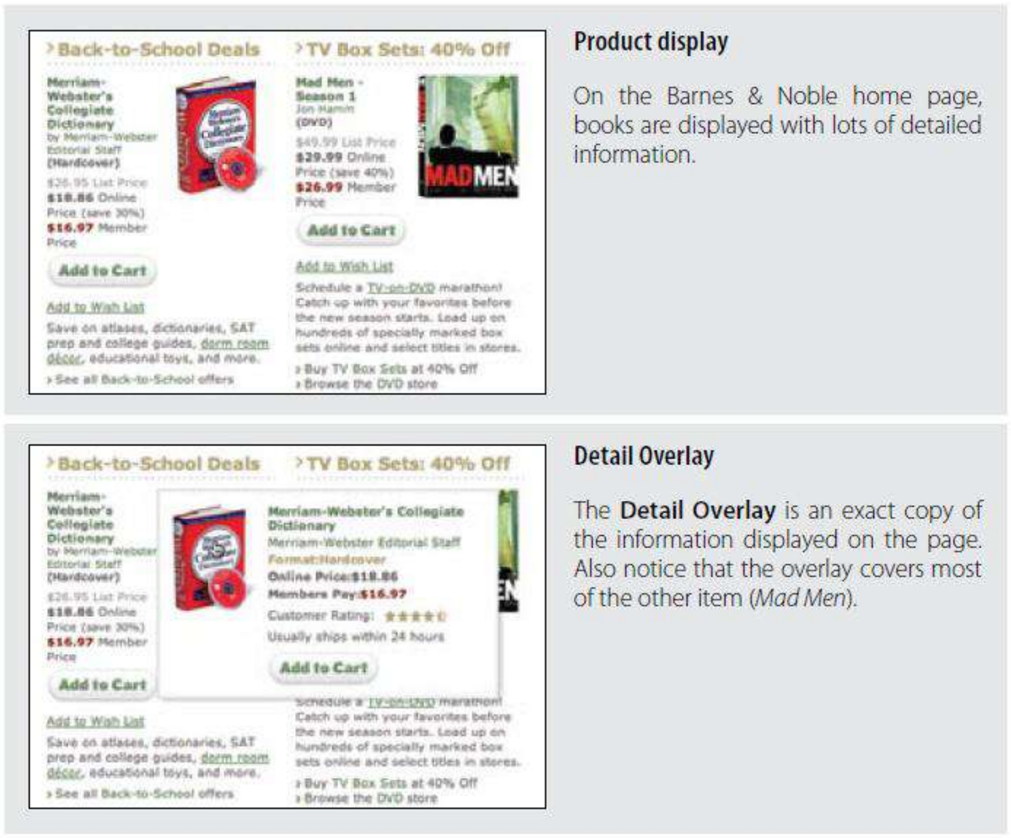


Figure 5-12. *Each Detail Overlay is preceded by the Needless Fanfare of a one-second animation that zooms the information into place*

Anti-pattern: Hover and Cover

We discussed the anti-pattern Hover and Cover in Chapter 2, and it's important to keep this anti-pattern in mind when providing a Detail Overlay. In the Netflix example (Figure 5-7), the verlay does not get in the way of moving to the next box shot. Even though it covers the neighboring box shot, moving the mouse outside the original one removes the overlay immediately, providing a clear path to get an overlay on the next box shot. Compare this to the Detail Overlay provided by barnesandnoble.com (Figure 5-13).



Back-to-School Deals

Merriam-Webster's Collegiate Dictionary by Merriam-Webster Editorial Staff (Hardcover)
$26.95 List Price
$18.86 Online Price (save 30%)
$16.97 Member Price
Add to Cart
Add to Wish List
Save on atlases, dictionaries, SAT prep and college guides, dorm room décor, educational toys, and more.
› See all Back-to-School offers

TV Box Sets: 40% Off

Mad Men - Season 1 Jon Hamm (DVD)
$49.99 List Price
$29.99 Online Price (save 40%)
$26.99 Member Price
Add to Cart
Add to Wish List
Schedule a TV-on-DVD marathon! Catch up with your favorites before the new season starts. Load up on hundreds of specially marked box sets online and select titles in stores.
› Buy TV Box Sets at 40% Off
› Browse the DVD store

Product display

On the Barnes & Noble home page, books are displayed with lots of detailed information.

Merriam-Webster's Collegiate Dictionary
Merriam-Webster Editorial Staff
Format:Hardcover
Online Price:$18.86
Members Pay:$16.97
Customer Rating: ★★★★☆
Usually ships within 24 hours
Add to Cart

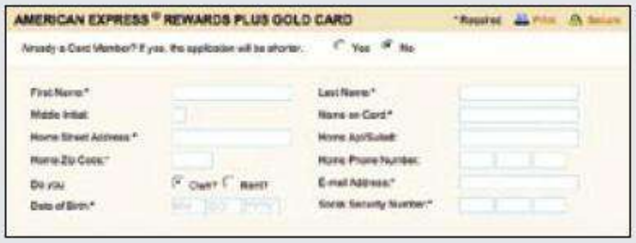
Detail Overlay

The **Detail Overlay** is an exact copy of the information displayed on the page. Also notice that the overlay covers most of the other item (*Mad Men*).

Figure 5-13. *Barnes & Noble does not need Detail Overlay since the information is exactly the same as displayed on the page*

Input Overlay

Input Overlay is a lightweight overlay that brings additional input information for each field tabbed into. American Express uses this technique in its registration for premium cards such as its gold card (Figure 5-15).

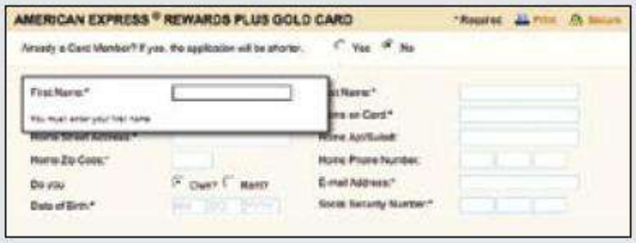

AMERICAN EXPRESS® REWARDS PLUS GOLD CARD *Required Print Secure

Already a Card Member? If yes, the application will be shorter. Yes [✓ No]

First Name* | Last Name*
Middle Initial | Name on Card*
Home Street Address* | Home Apt/Suite#
Home Zip Code* | Home Phone Number
Do you [✓ Own?] Rent? | E-mail Address*
Date of Birth* | Social Security Number*

Input form

The form is displayed with simple prompt/input for each field. No additional help information is shown statically.


AMERICAN EXPRESS® REWARDS PLUS GOLD CARD *Required Print Secure

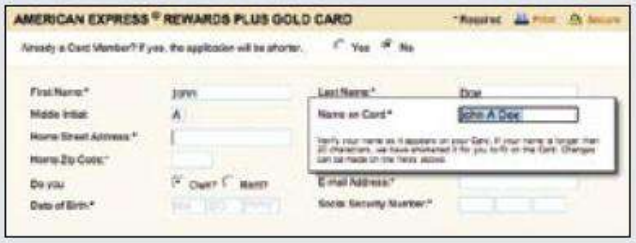
Already a Card Member? If yes, the application will be shorter. Yes [✓ No]

First Name*

You must enter your first name

Input overlay

Tabbing or clicking into any field wraps the field in an overlay. The overlay provides additional input information.


AMERICAN EXPRESS® REWARDS PLUS GOLD CARD *Required Print Secure

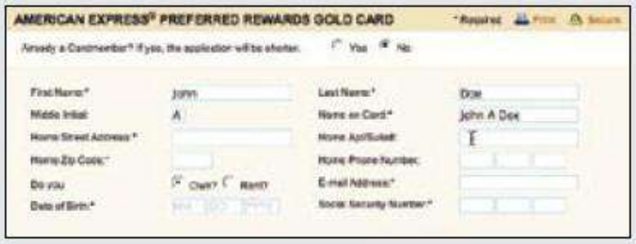
Already a Card Member? If yes, the application will be shorter. Yes [✓ No]

First Name* John | Last Name* Doe
Middle Initial A | Name on Card* John A Doe

Verify your name as it appears on your Card. If your name is longer than 20 characters, we have shortened it for you to fit on the Card. Changes can be made on the fields above.

Obscuring fields

The overlay does obscure fields just below it, but not to the left or right.


AMERICAN EXPRESS® PREFERRED REWARDS GOLD CARD *Required Print Secure

Already a Cardmember? If yes, the application will be shorter. Yes [✓ No]

First Name* John | Last Name* Doe
Middle Initial A | Name on Card* John A Doe

Deactivation

Clicking anywhere removes the overlay. This lets the user click through the field covered by the overlay.

Figure 5-15. *American Express provides Input Overlays to guide the user through the signup process*

Anti-pattern: Hover and Cover

But what about the anti-pattern, Hover and Cover? Doesn't this cause the same issues that we saw earlier? For example, in Figure 5-15 ("Obscuring fields"), the "Name on Card" overlay hides the "Home Apt/Suite#" and "Home Phone Number Fields" fields below it. There are several reasons that American Express was able to employ this overlay in forms and "get away" with covering some fields during input:

Field traversal

The field traversal is side-to-side. The overlay for the "First Name" field (Figure 5-15, "Input overlay") does not obscure the next field, "Last Name".

Tab navigation

Since tabbing is a primary navigation for forms, the mouse is not needed to navigate. This allows navigation to fields even if they were covered.

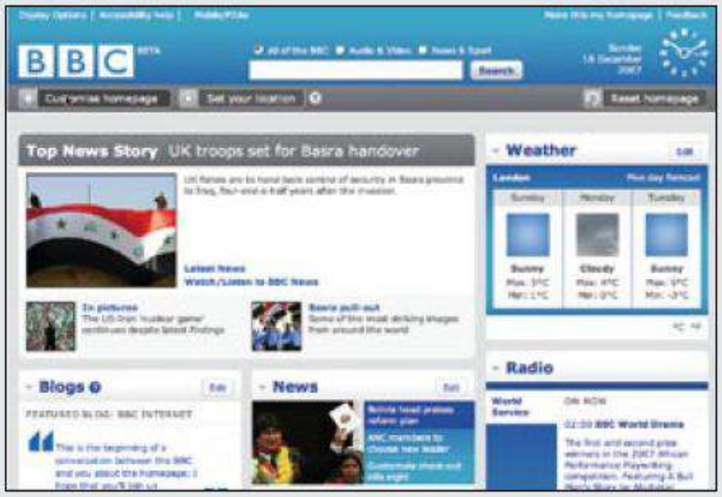
One-click deactivation

Clicking anywhere hides the overlay. That means that when the “Name on Card” overlay is shown (Figure 5-15, “Obscuring fields”), clicking anywhere in the “Home/ Apt Suite#” field will remove the overlay, allowing the user to click in the previously hidden field (Figure 5-15, “Deactivation”).

Inlays

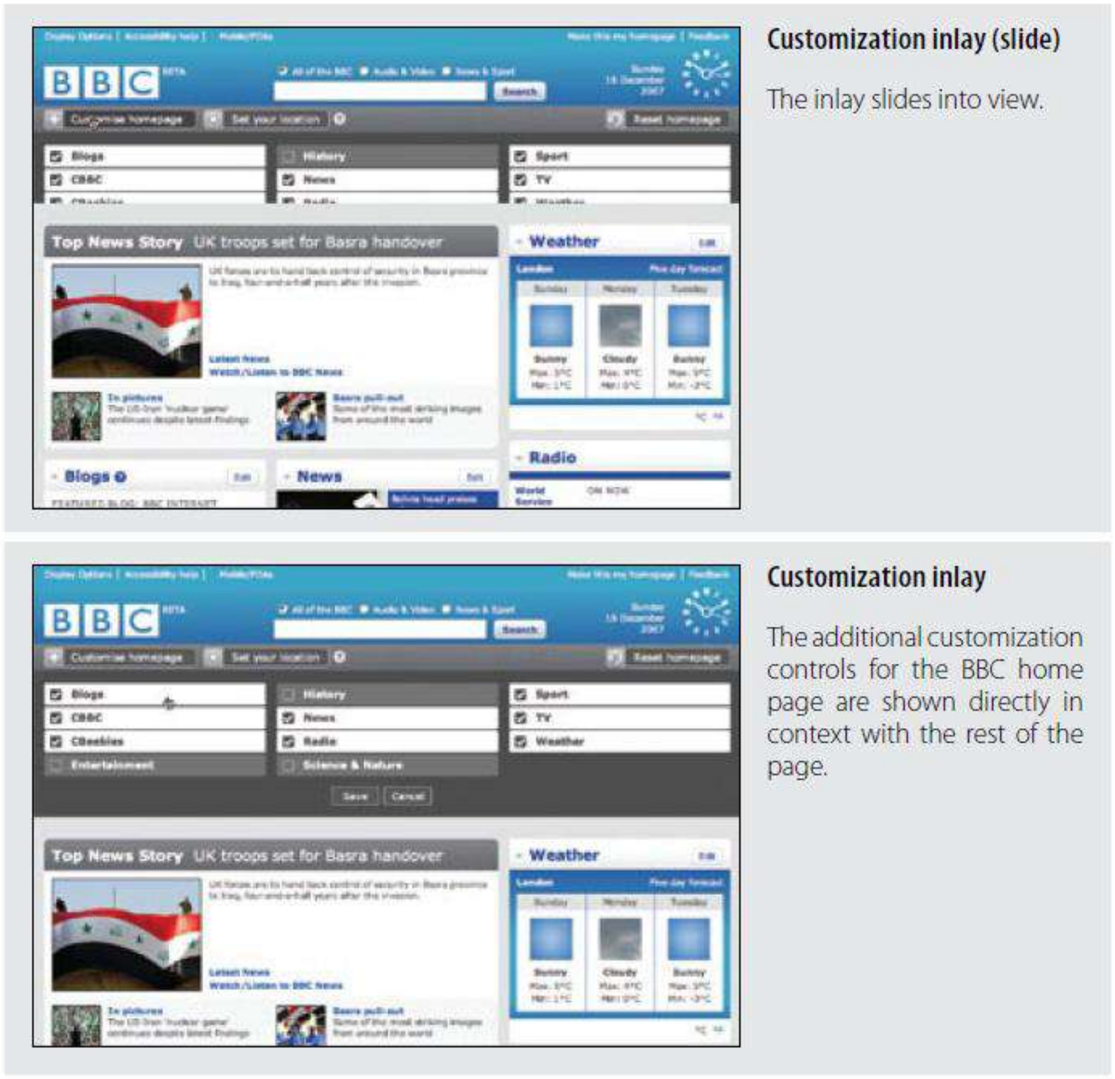
Dialog Inlay

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 customization controls for its home page (Figure 6-1).



Activation

The “Customize homepage” button activates the customization inlay.



Customization inlay (slide)

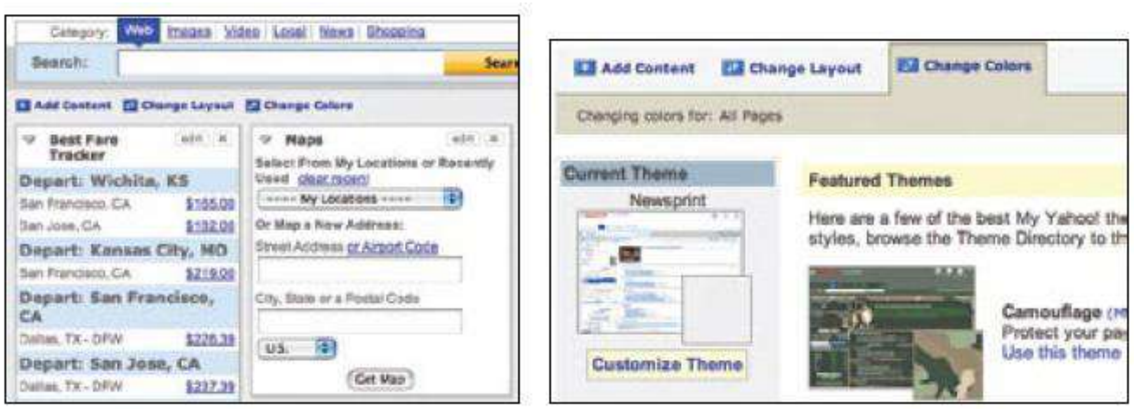
The inlay slides into view.

Customization inlay

The additional customization controls for the BBC home page are shown directly in context with the rest of the page.

Figure 6-1. *The BBC home page puts its customization tools in an inlay that slides out when activated*

In context
This Dialog Inlay is similar to a drawer opening with a tray of tools. Instead of being taken to a separate page to customize the home page appearance, the user can make changes and view the effects directly. The advantage is the ability to tweak the page while viewing the actual page.
My Yahoo! also provides a Dialog Inlay for revealing its home page customization tools.
The original version of My Yahoo! did not use this approach. Instead, customizations would take the user away to a separate page (Figure 6-2

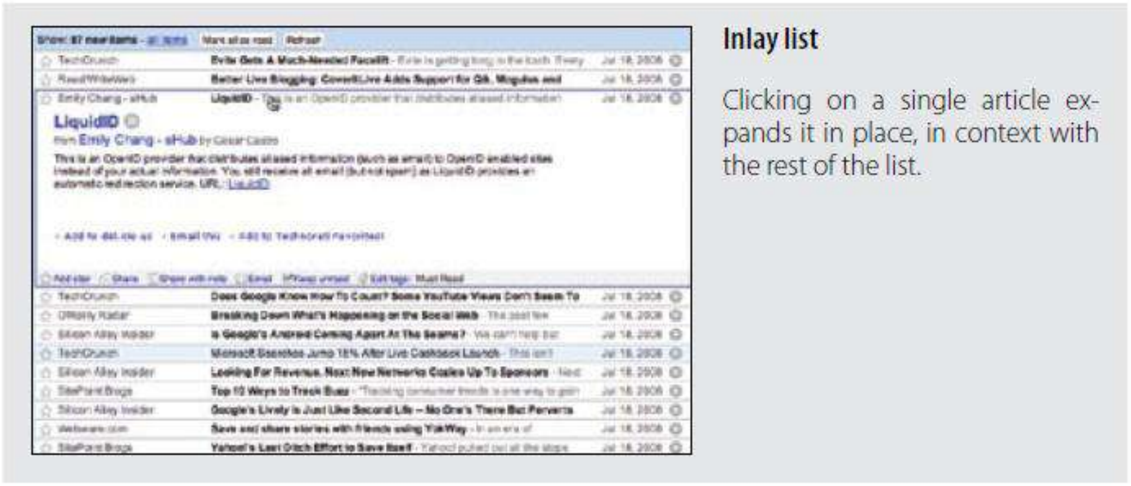


Here are a few of the best My Yahoo! the
styles, browse the Theme Directory to th

Figure 6-2. *In an original version of My Yahoo!, clicking on "Change Colors" takes you to a separate page to customize colors*

List Inlay

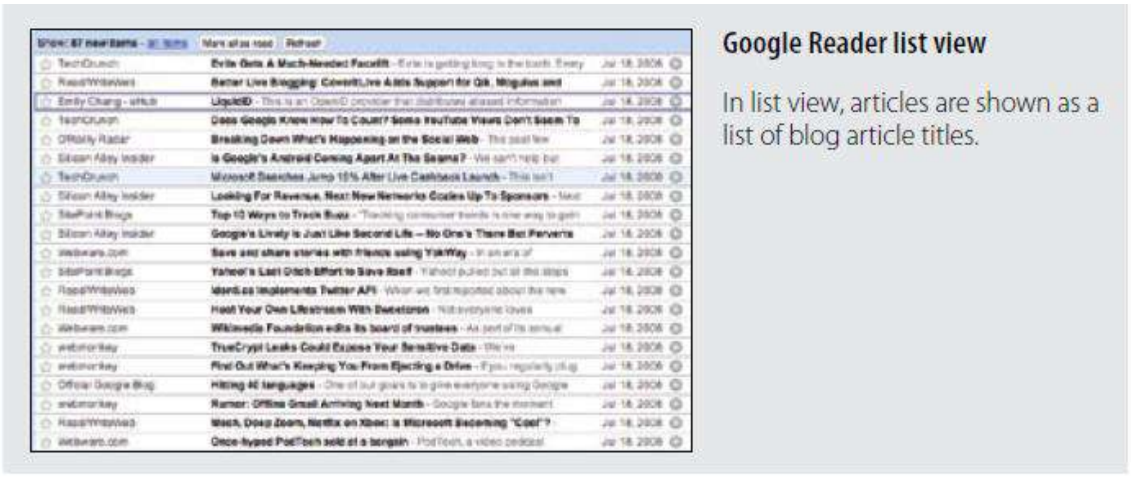
Lists are a great place to use Inlays. Instead of requiring the user to navigate to a new page for an item's detail or popping up the information in an Overlay, the information can be shown with a List Inlay in context. The List Inlay works as an effective way to hide detail until needed—while at the same time preserving space on the page for high-level overview information. Google Reader provides an expanded view and a list view for unread blog articles. In the list view, an individual article can be expanded in place as a List Inlay (Figure 6-5).



Inlay list

Clicking on a single article expands it in place, in context with the rest of the list.

Figure 6-5. *In list view, Google Reader shows all articles as a collapsed list—except for the one that is currently selected*



Google Reader list view

In list view, articles are shown as a list of blog article titles.

Parallel content

The Yahoo! Autos Car Finder tool (Figure 6-7) uses an accordion-style interaction for search filters that allows more than one pane to be open at a time. This choice makes sense because the decisions needed for one detail pane may be affected by the details of another pane. However, one problem with this specific implementation is the lack of information when a pane is closed. For example, no summary information is given for the “Price” tab. Looking at that button, it is not clear whether search criteria has been set or what it might be set to without first opening the pane.

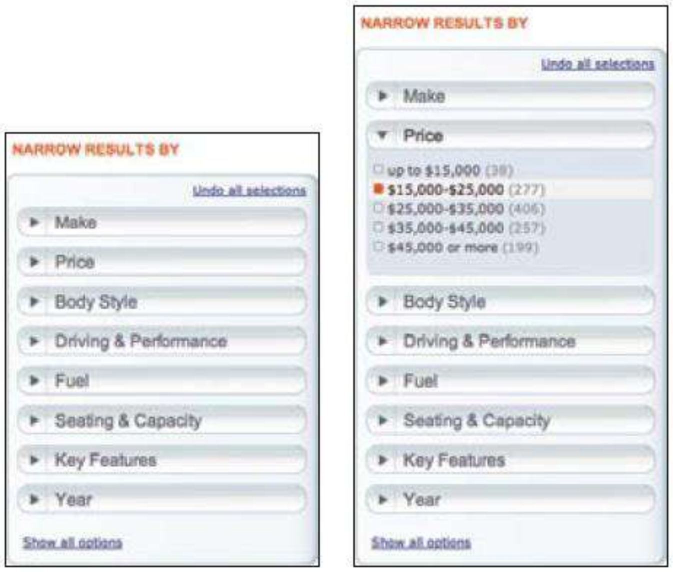
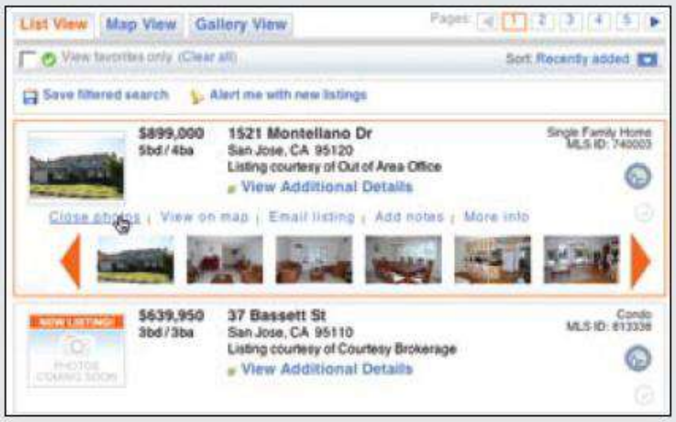


Figure 6-7. *Yahoo! Autos places filter criteria in Accordion panes; when panes are hidden, no summary information is provided*

Detail Inlay

A common idiom is to provide additional detail about items shown on a page. We saw this with the example of the Netflix movie detail pop up in Chapter 5 (Figure 5-8). Hovering over 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 for a real estate listing with a Detail Inlay (Figure 6-10).


List View
Map View
Gallery View
Pages: 1 2 3 4 5
View favorites only (Clear all)
Sort: Recently added
Save filtered search
Alert me with new listings
Close photos | View on map | Email listing | Add notes | More info
NEW LISTING!
PHOTOS COMING SOON

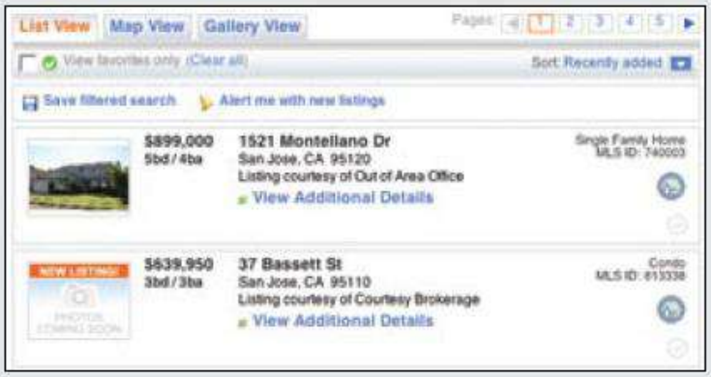
$899,000 5bd / 4ba 1521 Montellano Dr San Jose, CA 95120 Listing courtesy of Out of Area Office View Additional Details Single Family Home MLS ID: 740003

$639,950 3bd / 3ba 37 Bassett St San Jose, CA 95110 Listing courtesy of Courtesy Brokerage View Additional Details Condo MLS ID: 613338

Detail overlay

The **Detail Inlay** contains thumbnails of house photos. Clicking on an individual thumbnail pops up a **Detail Overlay** with a larger photo of the house.

Figure 6-10. *Roost provides both Detail Inlay and Detail Overlay patterns to show home photos*

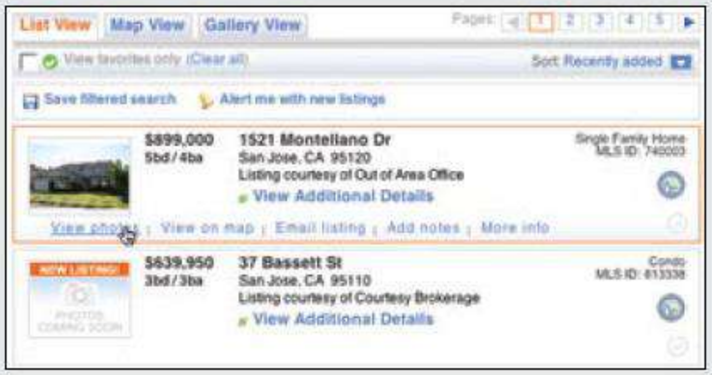


$899,000 5bd / 4ba 1521 Montellano Dr San Jose, CA 95120 Listing courtesy of Out of Area Office View Additional Details Single Family Home MLS ID: 740003

$639,950 3bd / 3ba 37 Bassett St San Jose, CA 95110 Listing courtesy of Courtesy Brokerage View Additional Details Condo MLS ID: 613338

In-context tools

Hovering over a real estate listing brings in a set of in-context tools, including the "View photos" tool.


List View
Map View
Gallery View
Pages: 1 2 3 4 5
View favorites only (Clear all)
Sort: Recently added
Save filtered search
Alert me with new listings
View photos | View on map | Email listing | Add notes | More info
NEW LISTING!
PHOTOS COMING SOON

$899,000 5bd / 4ba 1521 Montellano Dr San Jose, CA 95120 Listing courtesy of Out of Area Office View Additional Details Single Family Home MLS ID: 740003

$639,950 3bd / 3ba 37 Bassett St San Jose, CA 95110 Listing courtesy of Courtesy Brokerage View Additional Details Condo MLS ID: 613338

House photos inlay

Clicking on the "View photos" link expands the real estate item to include a carousel of house photos.

Virtual Pages

Overlays allow you to bring additional interactions or content in a layer above the current page. Inlays allow you to do this within the page itself. However, another powerful approach to keeping users engaged on the current page is to create a *virtual page*. That is to say, we create the illusion of a larger virtual page.

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 get to additional content. Of course, websites could preload data and allow the user to scroll through it. However, this process led to long delays in loading the page. So most sites kept it simple: go fetch 10 items and display them as a page and let the user request the next page of content. Each fetch resulted in a page refresh.

The classic example of this is Google Search. Each page shows 10 results. Moving through the content uses the now-famous Google pagination control (Figure 7-1).

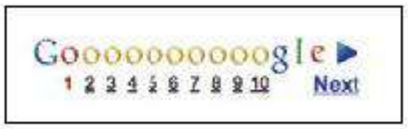
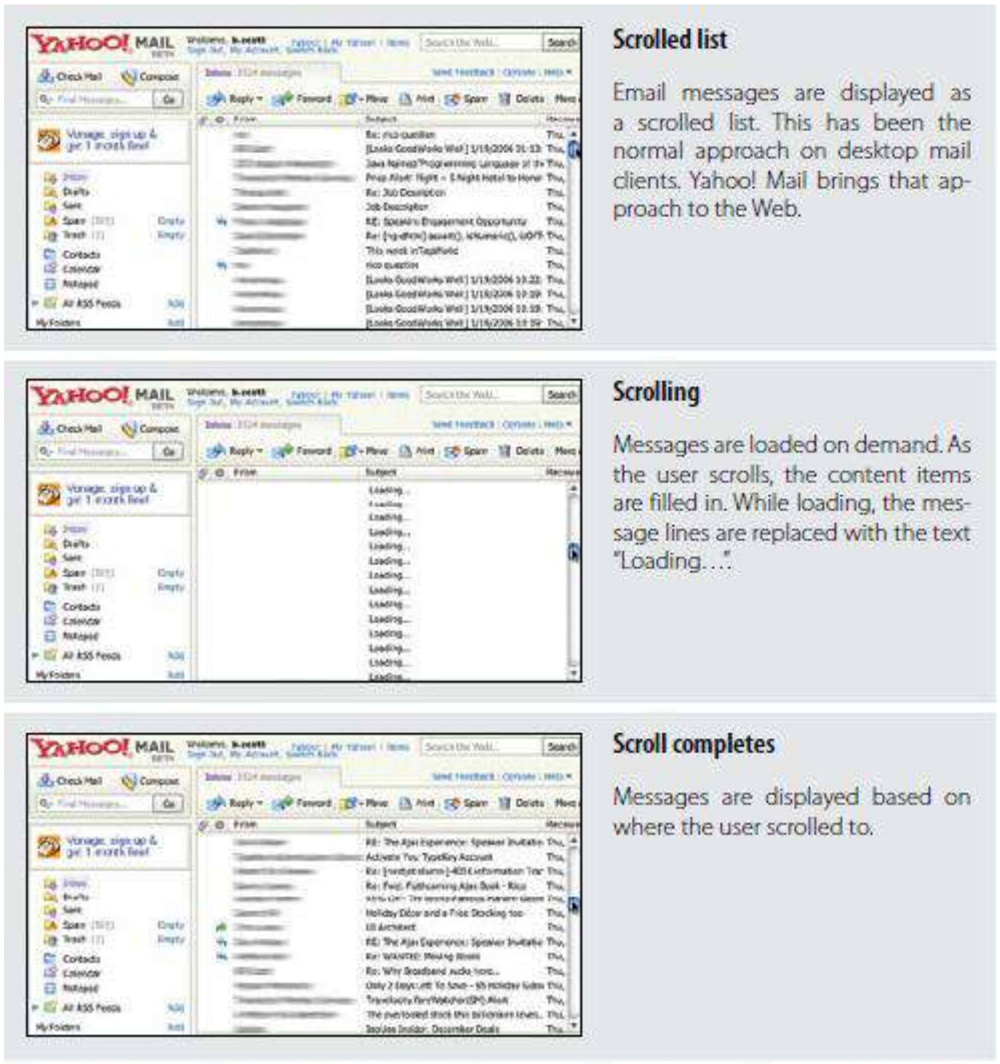


Figure 7-1. *The now-famous Google pagination control illustrates the most common way to move through data on the Web*



Scrolled list

Email messages are displayed as a scrolled list. This has been the normal approach on desktop mail clients. Yahoo! Mail brings that approach to the Web.

Scrolling

Messages are loaded on demand. As the user scrolls, the content items are filled in. While loading, the message lines are replaced with the text "Loading…".

Scroll completes

Messages are displayed based on where the user scrolled to.

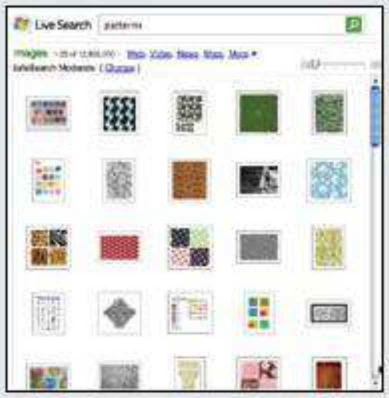
Figure 7-2. *Instead of showing just the messages that can be displayed on a single page, Yahoo! Mail dynamically loads messages as the user scrolls*

Loading status

There are a few downsides to the Yahoo! Mail version of Virtual Scrolling. First, if the loading is slow, it spoils the illusion that the data is continuous. Second, since the scrollbar does not give any indication of where users are located in the data, they have to guess how far down to scroll. A remedy would be to apply a constantly updating status while the user is scrolling.

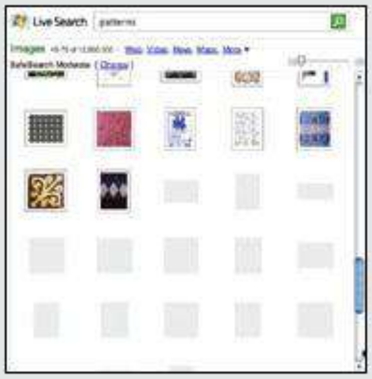
Progressive loading

Microsoft has applied Virtual Scrolling to its image search. However, it implements it in a different manner than Yahoo! Mail. Instead of all content being virtually loaded (and the scrollbar reflecting this), the scrollbar reflects what has been loaded. Scrolling to the bottom causes more content to load into the page (Figure 7-3).



Scrolled list

12,500,000 image results are represented as a scrolled list. Obviously there is no way to accurately represent that many items in a list with a scrollbar. Notice the scrollbar shows size relative to the amount of data that has been loaded.

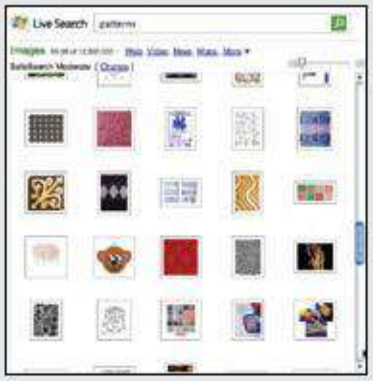


Scrolling

By scrolling into the area where results have not been loaded, images are initially represented as gray squares to indicate that they are currently not loaded.

As each image is loaded it replaces the gray squares.

At the top, the start and end range of the visible images is displayed ("Images 46–70 of 12,500,00").



Scroll completes

Image results are fully loaded, and the scrollbar is updated to reflect where this page is in relation to the previously loaded content.

Figure 7-3. *Microsoft Live Image Search uses Virtual Scrolling to fetch additional search results*

This type of Virtual Scrolling (where the scrollbar only reflects what has been directly loaded) works well for search results since relevance starts dropping off the further you move through the data. But with mail messages, this is not the case and would not be a good approach since users need to access messages beyond just those loaded at the top.

One more example illustrates an endless wall of pictures and uses a novel approach to a scrollbar control for Virtual Scrolling. PicLens is a Firefox add-on that allows viewing images from Google Search, Flickr, and other services to be displayed in the browser as a wall of photos that can be scrolled through (Figure 7-4).

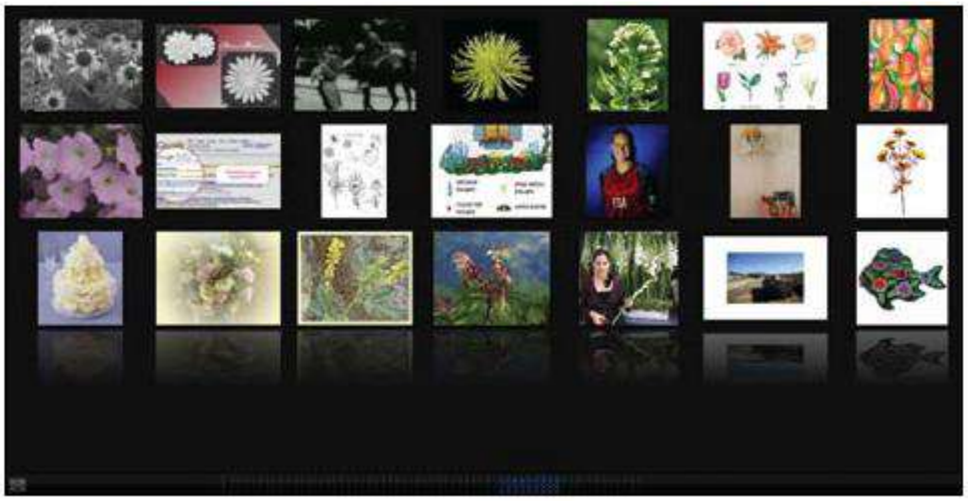
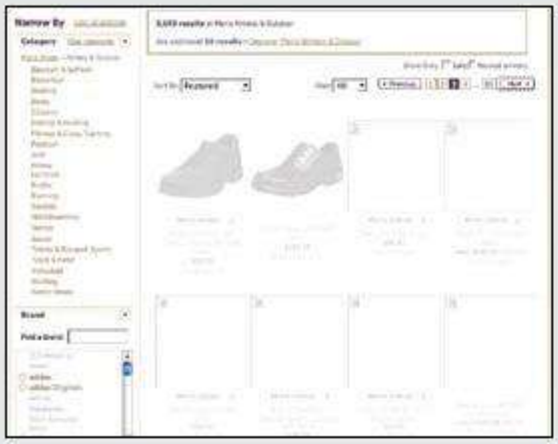


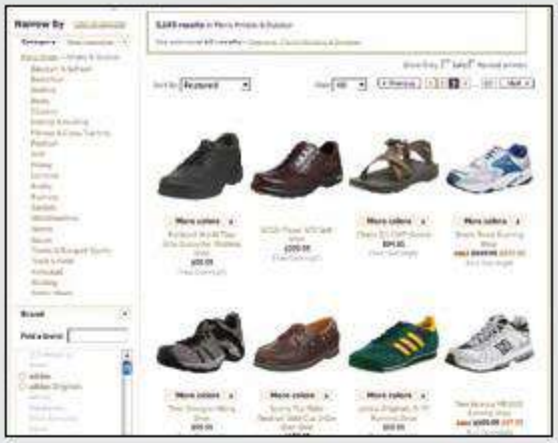
Figure 7-4. *PicLens provides an endless wall of photos; the scroller at the bottom continues to expand with more content*

Inline Paging
What if instead of scrolling through content we just wanted to make pagination feel less like a page switch? 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 does with its search results (Figure 7-6).



Inline Paging

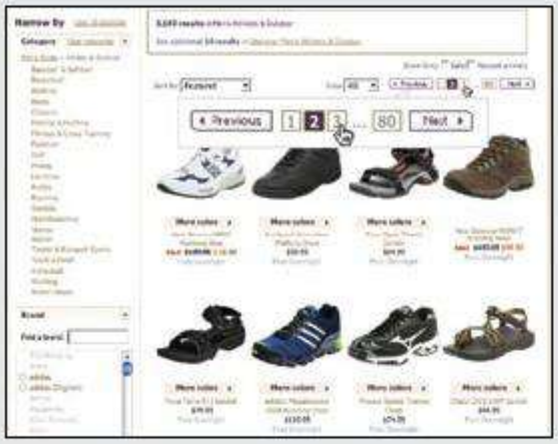
Clicking to "page 3" causes just the search results area to update with the third page of results.



Page remains stable

The rest of the page stays stable when the new "page" of results is brought into view.

Figure 7-6. *Endless uses Inline Paging to create a seamless experience moving through search results*

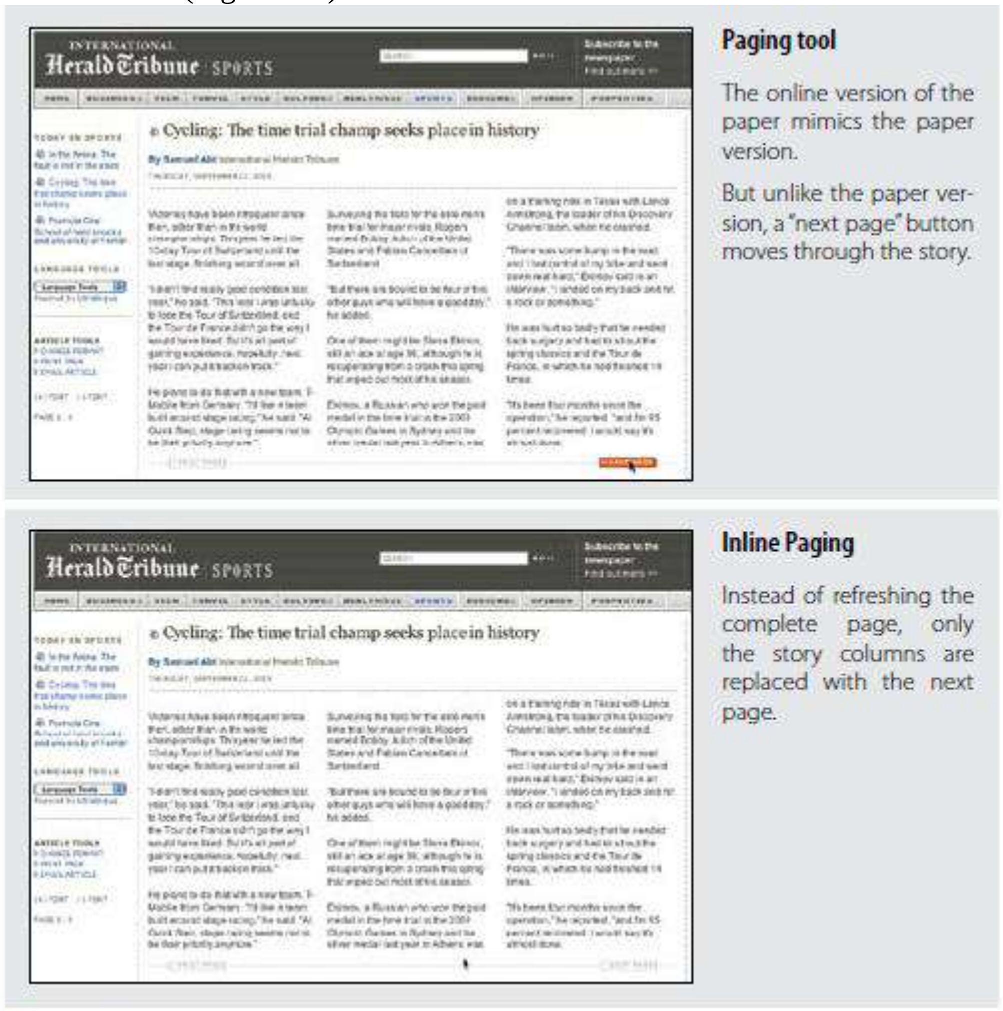


Paginated results

Searching for "Men's athletic shoes" displays a traditional-looking set of search results. The pagination controls are familiar (shown as an exploded callout).

Natural chunking

Inline Paging can also be useful when reading news content online. The *International Herald Tribune* applied this as a way to page through an article while keeping the surrounding context visible at all times (Figure 7-7).


INTERNATIONAL Herald Tribune SPORTS
Cycling: The time trial champ seeks place in history
By Samuel Abt International Herald Tribune
INTERNATIONAL Herald Tribune SPORTS
Cycling: The time trial champ seeks place in history
By Samuel Abt International Herald Tribune

Paging tool

The online version of the paper mimics the paper version.

But unlike the paper version, a "next page" button moves through the story.

Inline Paging

Instead of refreshing the complete page, only the story columns are replaced with the next page.

Figure 7-7. *The International Herald Tribune uses Inline Paging to seamlessly move through a story without losing context*

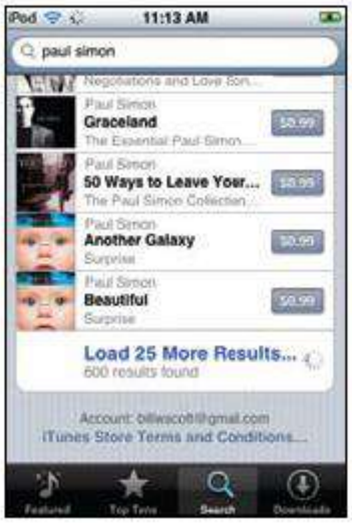


Figure 7-9. *The iPhone iTunes store displays 25 songs initially; the "Load 25 More Results..." button fetches 25 more songs*

Each tap of the "Load 25 More Results…" button loads 25 more songs. The number of songs loaded is cumulative. With the first tap there are now 50 songs; the third tap, 75 songs; and so on. Normally no scrollbar is shown. However, if the user places a finger on the list to move up or down, the scrollbar is displayed (Figure 7-10). Since the finger is the "scroller," the scrollbar becomes just an indicator of how many songs are loaded and where the user is in the list.

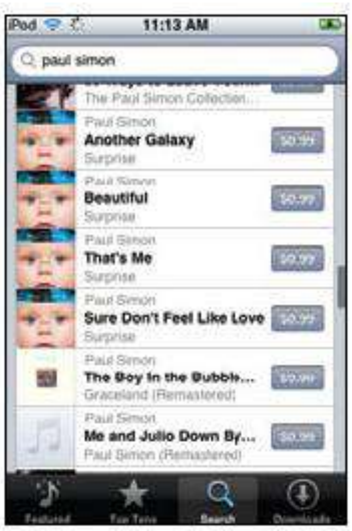


Figure 7-10. *The scrollbar displays when the user starts scrolling through the content; the scrollbar is just a feedback mechanism to indicate scrolling is happening, how much content is loaded, and where the user is in the scroll operation*

Figure 7-10. *The scrollbar displays when the user starts scrolling through the content; the scrollbar is just a feedback mechanism to indicate scrolling is happening, how much content is loaded, and where the user is in the scroll operation*

Scrolled Paging: Carousel

Besides Virtual Scrolling and Virtual Paging, there is another option. You can combine both scrolling and paging into Scrolled Paging. Paging is performed as normal. But instead the content is “scrolled” into view.

The Carousel pattern takes this approach. A Carousel provides a way to page-in more data 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. The additional 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 navigate back and forth through their photo collection (Figure 7-12).

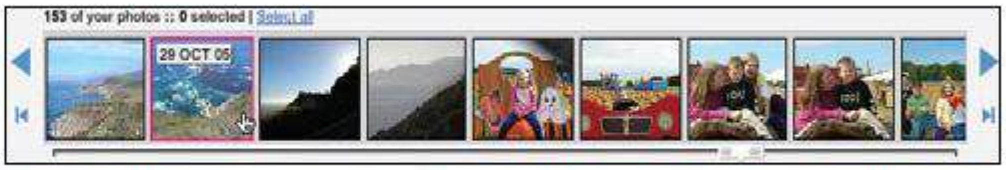
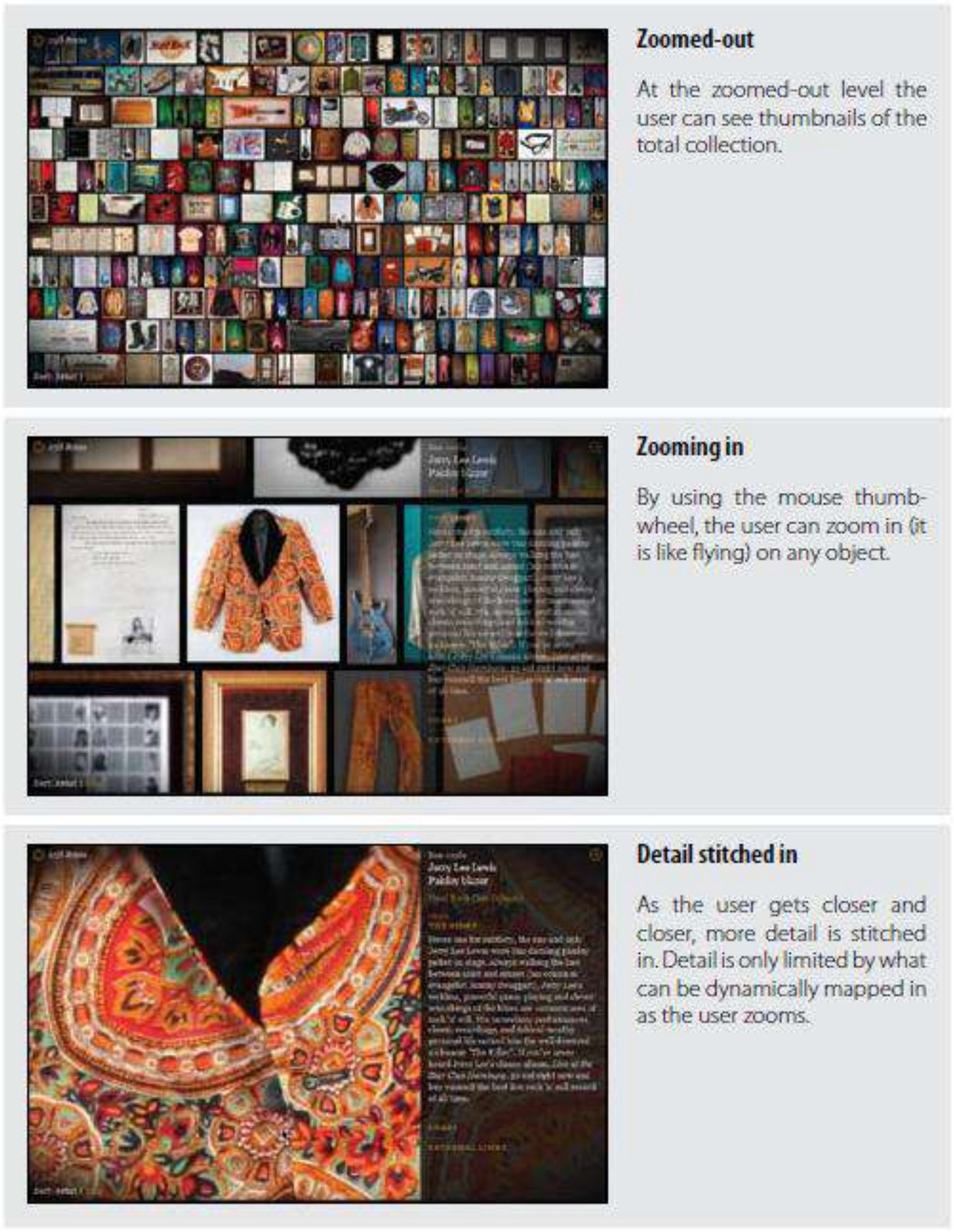


Figure 7-12. *Flickr uses a Carousel to allow the user to get photos in chronological order; this makes it possible to find photos without leaving the current page*

Zoomable User Interface
A Zoomable User Interface (ZUI) is another way to create a virtual canvas. Unlike panning or flicking through a flat, two-dimensional space, a ZUI allows the user to also zoom in to elements on the page. This freedom of motion in both 2D and 3D supports the concept of 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 advent of Silverlight, this type of interface is starting to emerge and may be commonplace in the not-too-distant future.



Zoomed-out

At the zoomed-out level the user can see thumbnails of the total collection.

Zooming in

By using the mouse thumb-wheel, the user can zoom in (it is like flying) on any object.

Detail stitched in

As the user gets closer and closer, more detail is stitched in. Detail is only limited by what can be dynamically mapped in as the user zooms.

Figure 7-15. *Hard Rock Café uses a zoomable user interface (ZUI) to allow its memorabilia collection to be viewed online*

Paging Versus Scrolling

Leading web designers and companies have taken different approaches to solving the same problems. Yahoo! Mail chose Virtual Scrolling. Gmail chose Inline Paging. How do you 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 transient but 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 relevant the further users go in the data, Inline Paging works well (as with the iPhone).
• For transient data, if you don't care about jumping around in the data to specific sections, consider using Virtual Scrolling (as in Live Image Search).
• If you are concerned about scalability and performance, paging is usually the best choice. Originally Microsoft's Live Web Search also provided a scrollbar. However, the scrollbar increased server-load considerably since users are more likely to scroll than page.
• If the content is really continuous, scrolling is more natural than paging.
• If you get your revenue by page impressions, scrolling may not be an option for your business model.
• If paging causes actions for the content to become cumbersome, move to a scrolling model. 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 continuous scrolled list is intuitive.

Process Flow

Sometimes tasks are unfamiliar or complicated and require leading the user step-by-step through a Process Flow. It has long been common practice on the Web to turn each step into a separate page. While this may be the simplest way break down the problem, it may not lead to the best solution. For some Process Flows it makes sense to keep the user on the same page throughout the process.
Google Blogger
The popular site Google Blogger generally makes it easy to create and publish blogs. One thing it does not make easy, though, is deleting comments that others may leave on your blog. This is especially difficult when you are the victim of hundreds of spam comments left by nefarious companies hoping to increase their search ranking.
Blogger forces you to delete these comments through a three-step process. Each step is an individual page, all punctuated with a page refresh (Figure 8-1).we will look at these 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 shoppers will 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 constrained by the available inventory. As the user makes decisions, the set of choices gets more and more limited.

This type of product selection is typically handled with a multi-page workflow. On one page, the user selects a shirt and its color and size. After submitting the choice, a new page is displayed. Only when the user arrives at this second page does he find out that the "true navy" shirt is not available in the medium size.

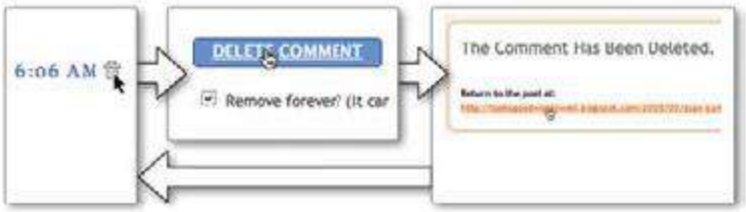


Figure 8-1. *Google Blogger forces you through a three-step process for each comment you delete, which is especially tiresome if you have dozens of spam comments to delete*

The Gap accomplishes this kind of product selection in a single page (Figure 8-3) using Interactive Single-Page Process. The purple shirt is available in all sizes from XS to XXXL. Hovering over the dark blue shirt immediately discloses that this color is only available in XS and S sizes.

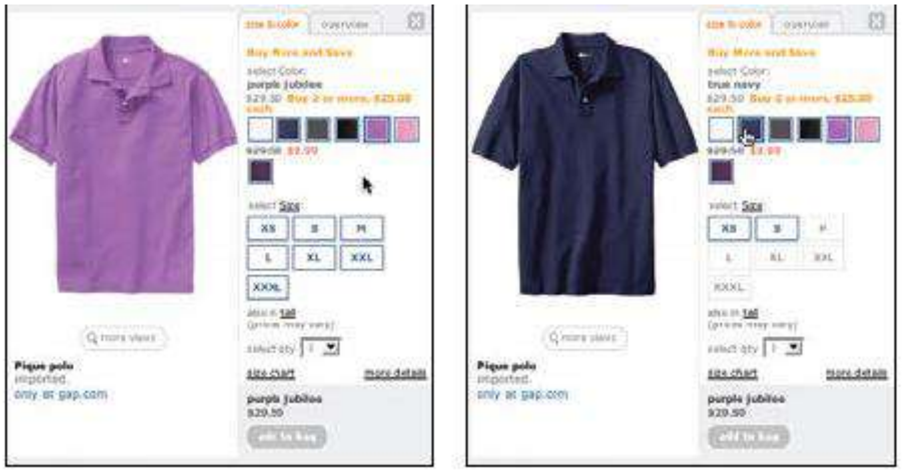
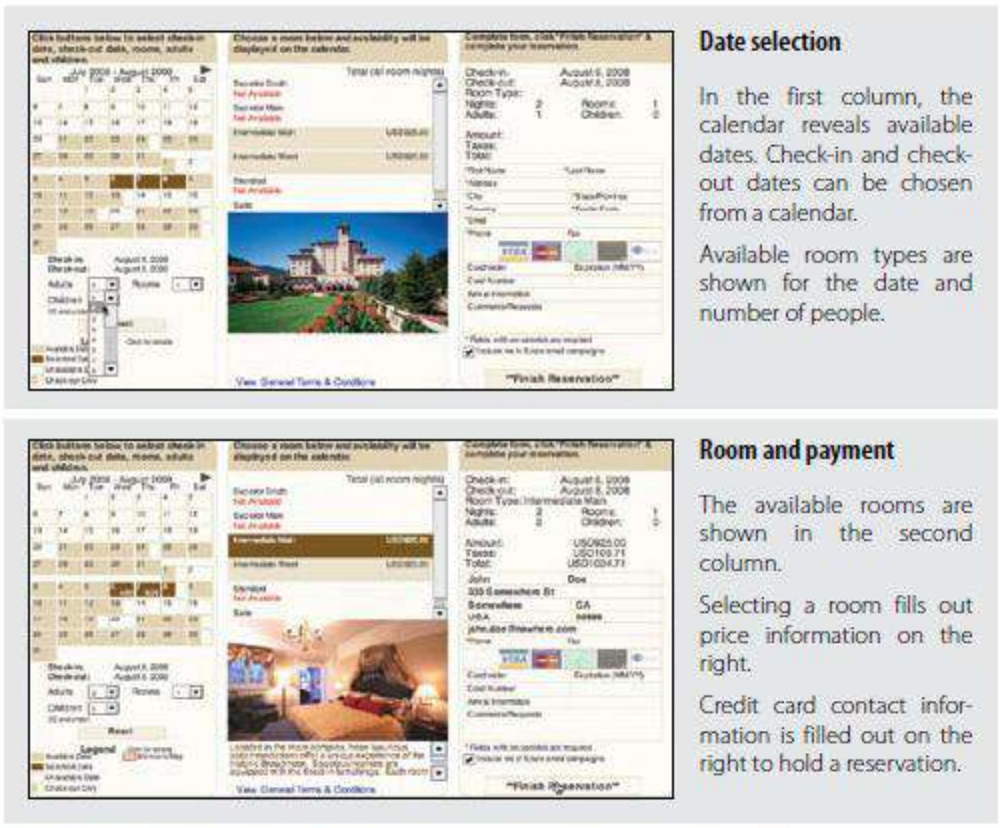


Figure 8-3. *The Gap uses Interactive Single-Page Process to reflect the sizes for each product color choice in real time*

Keeping users engaged
Broadmoor Hotel uses Interactive Single-Page Process for room reservations (Figure 8-5).



Date selection

In the first column, the calendar reveals available dates. Check-in and check-out dates can be chosen from a calendar.

Available room types are shown for the date and number of people.

Room and payment

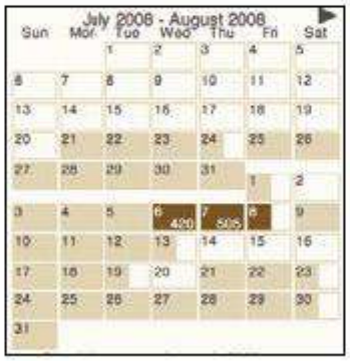
The available rooms are shown in the second column.

Selecting a room fills out price information on the right.

Credit card contact information is filled out on the right to hold a reservation.

Figure 8-5. *Broadmoor Hotels provides a one-page interactive reservation system*

Each column represents what would normally be presented on a separate page. In the first column, a calendar discloses availability up front. This prevents scheduling errors. Selecting the room from the second column updates both the room picture and the pricing. The pricing is reflected back on the calendar days (Figure 8-6) as well as in the third column where credit card and contact information is entered.



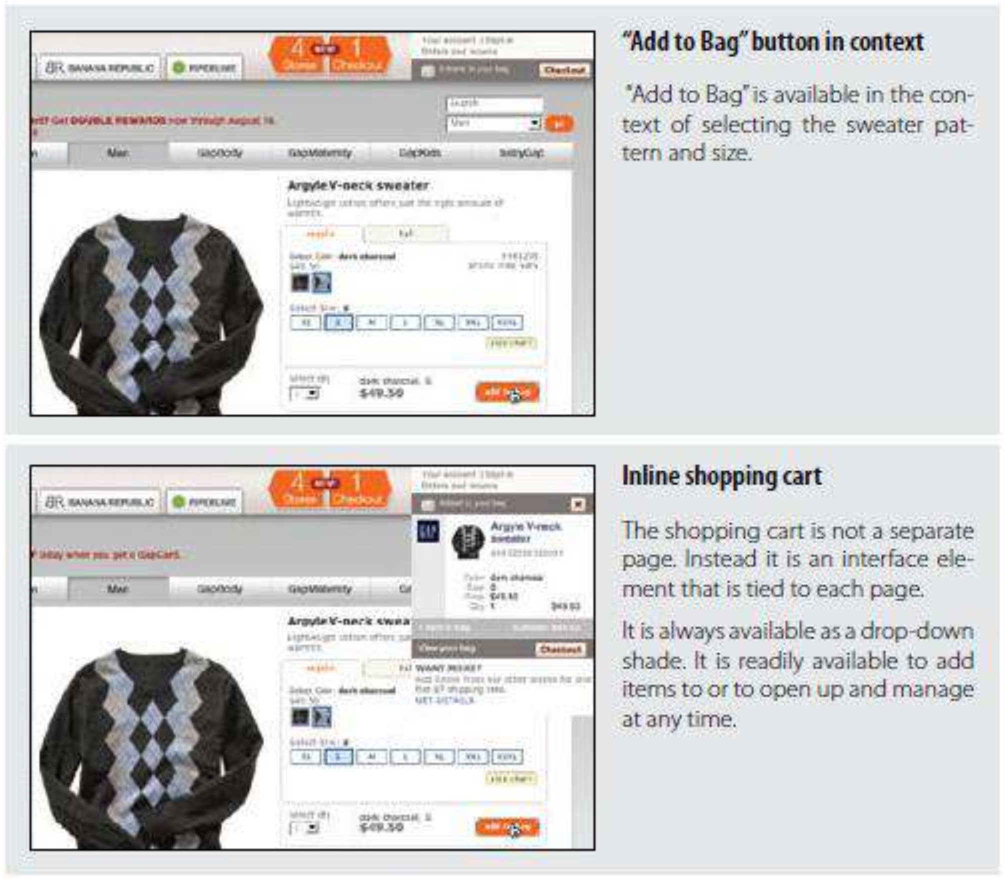
July 2008 - August 2008

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6 420	7 505	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Figure 8-6. *The Broadmoor calendar shows availability and pricing information*

Benefits
Adobe calls out the Broadmoor one-page reservation interface in its Adobe Showcase.* It states 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 less than one.

Inline Assistant Process
Another common place where multiple pages are used to complete a process is when adding items to a shopping cart. As mentioned earlier, Amazon provides the typical experience (Figure 8-2). So what magic can we apply to move this from a multi-page experience to a single-page experience? Instead of thinking about the cart as a process, we can think about it as a real-world object. Given this mindset, the cart can be realized in the interface as an object and be made available on the page. The Gap employed an Inline Assistant Process pattern for its shopping cart when it re-launched its site a few years back (Figure 8-7).



"Add to Bag" button in context

"Add to Bag" is available in the context of selecting the sweater pattern and size.

Inline shopping cart

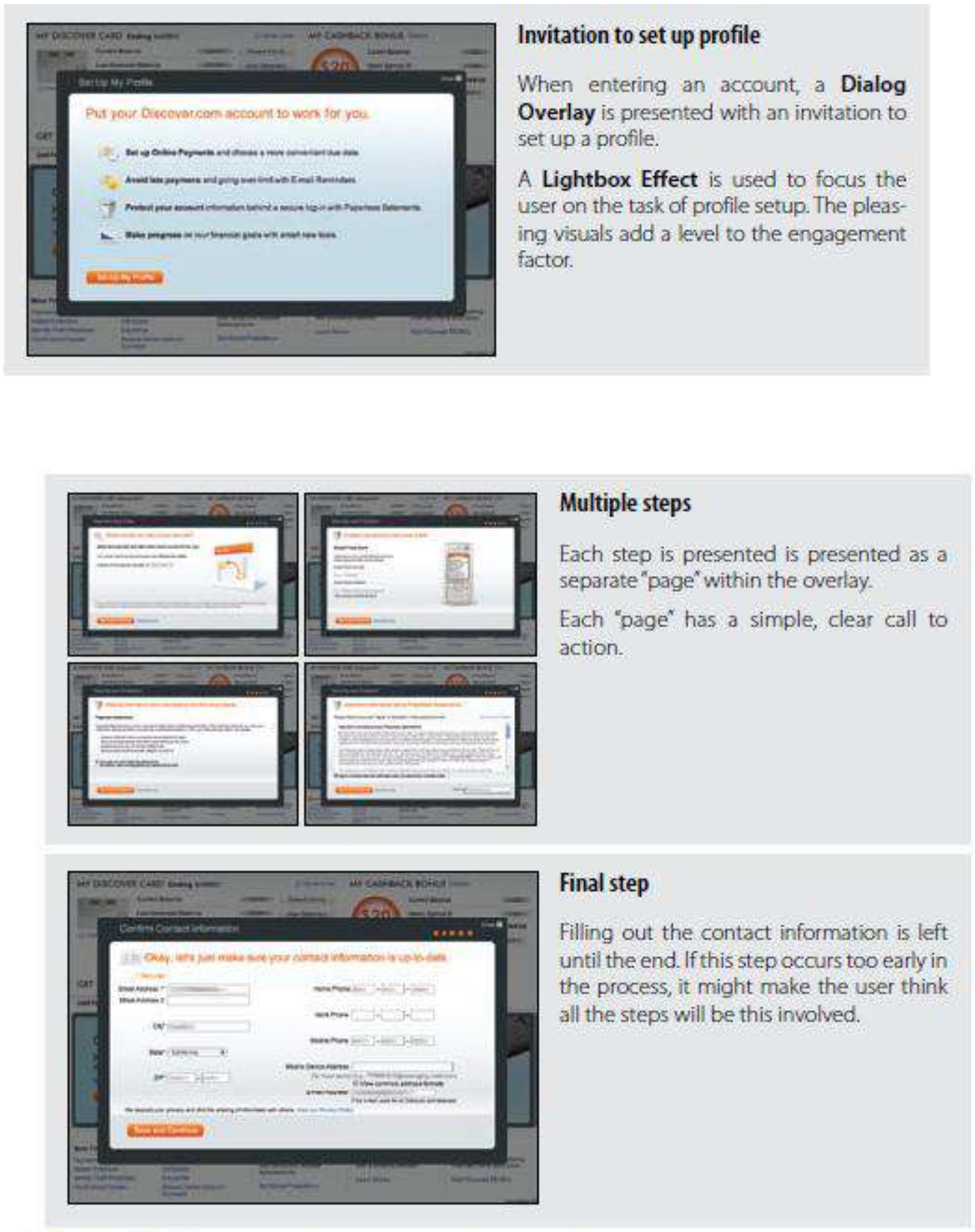
The shopping cart is not a separate page. Instead it is an interface element that is tied to each page.

It is always available as a drop-down shade. It is readily available to add items to or to open up and manage at any time.

Figure 8-7. *The Gap's single-page "Add to Bag" process*

Dialog Overlay Process

As mentioned before, any page switch is an interruption to the user's mental flow. In addition, any context switch is a chance for a user to leave the site. We seek an experience that has as little mental friction as possible. But sometimes the step-by-step flow is necessary. Overlays allow us to keep the context of the page yet present a virtual space to conduct a conversation with the user. Discover.com recently expanded its account section with a more detailed profile. The profile captures things like your payment date, mobile fraud alerts, paperless statements, and general contact information (Figure 8-11). The overlay pops up when you first enter your account.


Set Up My Profile
Put your Discover.com account to work for you.
Confirm Contact Information
Okay, let's just make sure your contact information is up-to-date.

Invitation to set up profile

When entering an account, a **Dialog Overlay** is presented with an invitation to set up a profile.

A **Lightbox Effect** is used to focus the user on the task of profile setup. The pleasing visuals add a level to the engagement factor.

Multiple steps

Each step is presented is presented as a separate "page" within the overlay.

Each "page" has a simple, clear call to action.

Final step

Filling out the contact information is left until the end. If this step occurs too early in the process, it might make the user think all the steps will be this involved.

Figure 8-11. *Discover encapsulates its "Create Your Profile" flow in a Dialog Overlay*

Clear status

The other interesting touch in this example is providing an indication of the number of steps (Figure 8-13). There are any number of ways to indicate this information; the important thing is to give some indication of what the users are dealing with when they start. Usually three steps are ideal. In this case, there are five steps. But as we mentioned, the early steps are single actions.

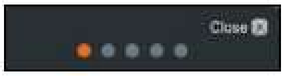


Figure 8-13. *Discover uses the orange dot to indicate how far the users are in the process and gray dots to indicate what they have left*

Configurator Process
Sometimes a Process Flow is meant to invoke delight. In these cases, it is the engagement factor 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 (Figure 8-15).

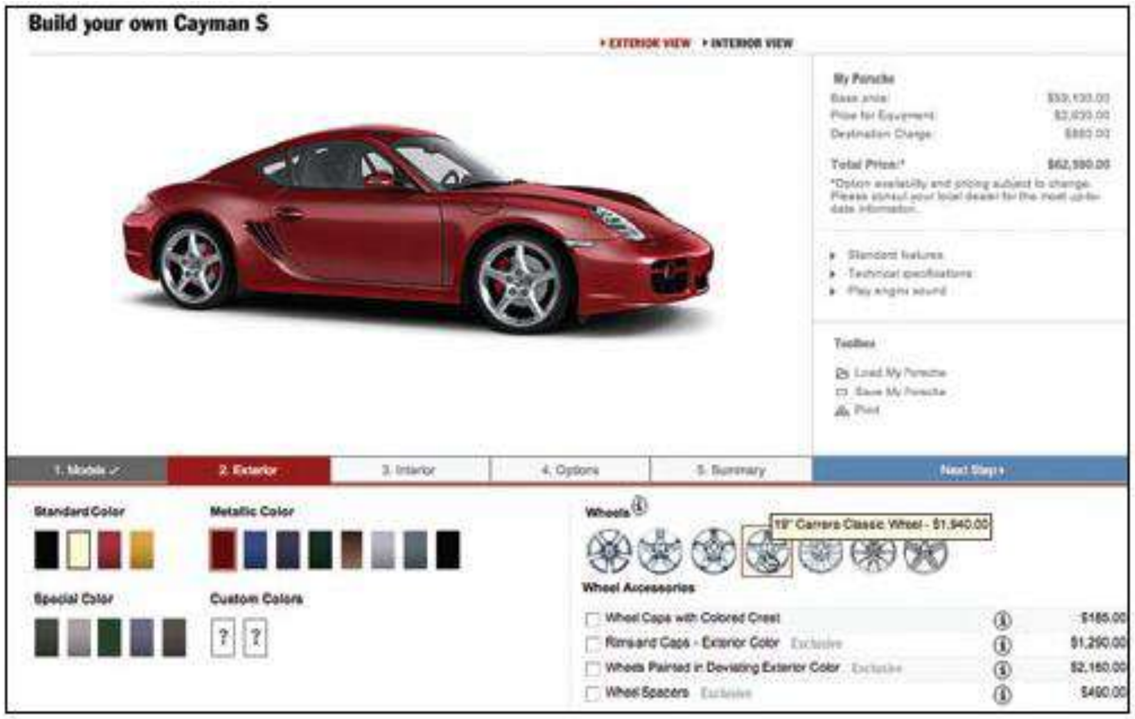


Figure 8-15. *Porsche's car configurator provides an engaging way to customize a car*

Static Single-Page Process
The Apple example illustrates another way to get rid of multiple pages in a Process Flow. Just put the complete flow on one page in a Static Single-Page Process. The user sees all the 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.

The other flow provided by eBay is a simplified **Static Single-Page Process** (Figure 8-19).

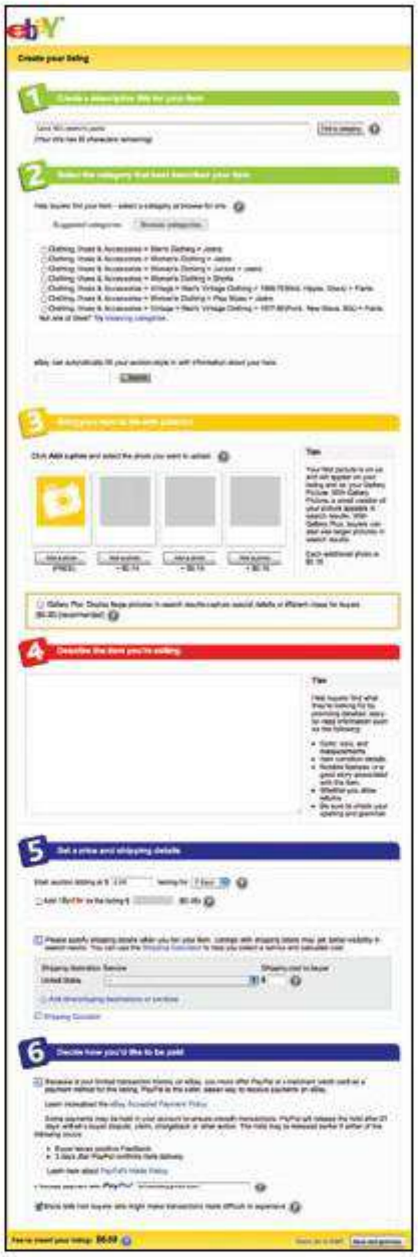

Create your listing
1
2
3
4
Describe the item you're selling
Tips
5
Set a price and shipping details
6
Decide how you'd like to be paid

Figure 8-19. *eBay displays a simplified "Sell Your Item" as a Static Single-Page Process*