Chapter 5
Cognition: Memory, Attention, and Learning

Abstract Memory, attention, learning are intertwined in the user's cognitive processing. These are the basic mechanisms of the user's cognitive architecture and thus provide the basis for cognition. Users have several types of memory that are important for computer use. Attention can be seen as the set of items being processed at the same time and how they are being processed. If there are more items stored in memory or the items in memory are better organized these effects will improve performance and provide the appearance of more attention. Users also learn constantly. The effects of learning lead to more items being stored in memory and allow the user to attend to more aspects of a task.

5.1 Introduction

Memory and attention both play an important role in interaction. Complementing these two facilities is the user's ability to learn things in a variety of ways. Together, these three concepts form the basics of the information processing mechanisms of a user's cognitive architecture. We consider the three concepts together here because of their interdependencies, focusing on the most important aspects with respect to computer users, rather than covering everything that is known about them.

It is worth noting at this point that the term *memory* is used in three different ways. The first refers to the mental function of retaining information about things—stimuli, events, images, ideas, and so on—when those things are no longer present. The second refers to the hypothesized storage system in the brain where this information is stored. The third refers to the information that is stored itself. In order to avoid ambiguities we refer to storage of items in and retrieval (or recall) of items from memory when we want the first meaning; we refer to memory when we want the second meaning; and we refer to items (or information) in memory when we want the third meaning.

F. E. Ritter et al., *Foundations for Designing User-Centered Systems*,
DOI: 10.1007/978-1-4471-5134-0_5,

Users' initial perceptions of an interface will be influenced by how they store and process information in short-term memory, and by how the information in their long-term memory helps them interpret that interface. The way people use a system will be greatly influenced by how well they can retrieve commands and locations of objects from memory. Similarly, their feelings of success with a system will be influenced by their biases in retrieving information about past successes and failures with the system.

Attention refers to the selective aspects of perception which function so that at any instant a user focuses on particular features of the environment to the relative exclusion of others. It plays a central role in interaction, where it often is not possible to interact with all aspects of the interface at the same time. Some interfaces require less attention from the user and this can be a good thing if it allows them to perform more than one task at a time efficiently.

User performance improves through learning. Learning is the most important process for adapting the user to the machine. There are several ways of describing learning, but for users the most important aspects of learning are probably learning facts and learning skills (or procedures) to perform tasks, but there is also learning to recognize images and perceptual-motor behavior.

5.2 Memory

Memory is one of the most studied areas in psychology. Understanding memory will help you as a designer to make it easier for users to memorize and later remember what they want or need to know. We begin by dealing with the structure of memory, which should enable you to follow the rest of the chapter more easily (this approach of providing a way to organize what you will learn is itself a result of memory research).

5.2.1 Types of Memory

Memory can be categorized in several ways. We will first look at memory based on where it is stored. Then we will examine memories by their content, including memories about a particular time and place (episodic), about object types (semantic), about facts (declarative), and about how to do a task (procedural). While there are other ways to conceptually organize memory, for system design, the set of categories we present here will give you a broad overview of the issues.

5.2.1.1 Iconic Memory

We can start our discussion of memory with perceptual-based information, specifically images. Perception, while fleeting, is not completely temporary. There is

an image left when you close your eyes or after an image you have been looking at has disappeared. This is called iconic memory.

Visual iconic memory holds only a few items. Some suggest a limit of two or three items (Zhang and Simon 1985); others suggest a few more. These items also decay (disappear) at a fairly fast rate. Sperling (1961) had subjects view a screen of numbers and then asked them to retrieve items after the items were no longer displayed. He determined that items exist in a temporary memory store for about 500 ms, with an exponential decay rate. If items are not processed, about half of the records of these items disappear in each half second.

Items can be put into short-term or long-term memory by processing them. However, as this takes time, the other items can decay and be lost from iconic memory. Thus items that appear on an interface for a short time have to be noticed and processed to be remembered.

5.2.1.2 Short-Term Memory

Short-term memory (STM) is a temporary memory store. Work by Atkinson and Shiffrin (1968) helped establish STM as a common concept in memory research. It can be considered analogous to the registers in a computer. Cognition writes information into short-term memory but, unlike the computer, the contents decay with time. You might start out across the room knowing the phone number you want to dial, or start to bring up a browser to type in a URL, but by the time you get there, physically or metaphorically, you may have forgotten the number or URL. This type of loss is one of the first ways that people get introduced to short-term memory.

George Miller, in a famous study (Miller 1956) found that, for unrelated objects, users could remember around seven meaningful items (plus or minus two). The estimate of the rate of loss of these items varies somewhat based on who is studied and what they are trying to remember. Some authors have found that half the information disappears in about 5 s if it is not rehearsed (practiced).

Short-term memory is often used to store lists or sets of items to work with. There are several interesting and immediate effects of memory of lists that are worth knowing about. The first effect, called primacy, is that items that appear at the start of a list are more easily retrieved from memory.

The second is that distinctive items in a list are better retrieved (the Von Restorff effect). For example, if they are printed in red ink or a bell goes off when you read them, or they are in some other way distinct or important, they will be better retrieved. The improvement in memorability requires distinctiveness—highlighting a whole text and putting a box around it does not help because nothing stands out. Similarly, writing everything in red does not help, but writing only one word in ten in red will help. This effect is also illustrated later in this chapter by discussing Nepal and encoding.

The third is that items in a list that make more sense, IBM, PDQ and XYZ, are better retrieved than items that do not have associations for everybody, such as

Table 5.1 Abbreviations that are more memorable in one culture than another. (Note that German capitalizes differently compared to English)

For German	For English
MfG = Mit freundlichen Grüßen with kind regards	ABC = American or Australian Broadcasting Company
CDU = Christliche demokratische Union a large German political party	PDQ = Pretty darn quick
SPD = Sozial demokratische Partei Deutschlands. Another German political party	ASAP = As soon as possible
DRK = Deutsches Rotes Kreuz. rescue service/ambulance/Red Cross	PIA = Peoria International Airport
ZDF = Zweites Deutsches Fernsehen. TV Station	TLAs = Three Letter Abbreviations or Acronyms
GmbH = Gesellschaft mit beschraenkter Haftung (Society with limited liability, i.e., a limited company)	CMU = Concrete Masonry Unit
For neither culture: http://us.support.tomtom.com/cgi-bin/tomtom_us.cfg/php/enduser/std_adp.php?p_faqid=2053&p_created=1092921663&p_sid=lXplZnAj&prod_lvl1=141&prod_lvl2=2030&cat_lvl1=2356&p_accessibility=&p_redirect=&p_lva=&p_sp=cF9zcmNoPSZwX3NvcnRfYnk9JnBfZ3JpZHNvcnQ9JnBfcm93X2NudD01NCw1NCZwX3Byb2RzPTE0MSwxODMsMTk1NCwyMDMwJnBfY2F0cz0yMzU2JnBfcHY9MS4xNDE7Mi4xODM7Mi4xOTU0OzIuMjAzMCZwX2N2PTEuMjM1NiZwX3NlYXJjaF90eXBlPWFuc3dlcnMuc2VhcmNoX2ZubCZwX3BhZ2U9MQ**&p_li=&p_topview=1	

SCE, ORD and PIA. If the user can group items into a meaningful item, to chunk it, the subitems are easier to recover because just the chunk has to be retrieved. These associations or chunks vary across people. We include a few examples in Table 5.1, including an item that would only make sense to a computing system.

Finally, the last items presented in a list are better retrieved as well (the recency effect). The primacy effect and the recency effect can be combined to create a serial position curve, as shown in Fig. 5.1.

These regularities suggest that you cannot increase your short-term memory by trying harder, but you can by presenting items in a particular order. Knowing more can also help. In air travel, for example, if you know that ORD is the airport code for O'Hare Airport, FRA is Frankfurt Airport (am der Main), and PIA is Peoria International Airport, then you only have to retrieve three codes rather than nine letters if you are presented with FRAORDPIA. This process was used in early phone systems in the US where regions were given names from the associated numbers, for example, as immortalized in the song "Pennsylvania 6-5000", which would have been 726-5000 (and would not have been as catchy).

Later theories, such as that embodied in the ACT-R theory (Anderson 1993, 2007), propose that short-term memory is just activated long-term memory. As processing occurs, objects are moved directly from perception into long-term memory. The observed effects for items in short-term memory are just the effects

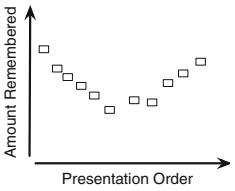

Fig. 5.1 The Serial Position curve. The primacy effect is that earlier items are better remembered and the recency effect is that items more recently encountered are better remembered

you see with less strong long-term memories. The number of items held in long-term memory that can be worked with at once is working memory, which we cover next.

5.2.1.3 Working Memory

Working memory is considered a more dynamic concept than STM. It is hypothesized as a temporary memory store (an audio or semantic scratchpad) with associated mechanisms for rehearsing, refreshing, and using the stored information. It also includes a mechanism of central or executive attention that regulates the contents of that memory store based on performing a task. This view is based on the models and definitions of Baddeley (1976, 1986). Working memory is seen less as a scratch pad than short-term memory, but it is viewed more within the context of the processing that will use it, and how the scratch pad and processing interact. This view of working memory suggests that increases in working memory can have numerous effects, ranging from more directed attention to better performance in general (Engle 2002).

These items in working memory are often rehearsed in what is called a phonological loop, where the material to be stored is repeated rapidly to oneself. This loop can hold about 2 s of verbal information. The direct implications are that items that are faster to pronounce take up less space. This has been found for numbers in different languages—languages with long names (in syllables) for numbers lead to fewer objects that can be retrieved. "Seven," which is two syllables long, for example, will take up more space than "one." Running through this loop only holds information; it does not alone increase memory for the items.

Working memory for a task is influenced by several factors. Focusing on a single task directly improves the amount of working memory available for that task. There also appear to be individual differences in working memory, with some people having more working memory than others (Daneman and Carpenter 1980; Lovett et al. 2000). Talking while doing a task provides some additional memory (not more traditional working memory, but more memory through the acoustical loop, which is repeating information verbally to yourself and hearing it, temporarily increasing your effective working memory), and in addition to slowing down performance can lead to more insights (Ericsson and Simon 1993). Further work suggests that extreme amounts of practice can lead to a type of increase in working memory (Ericsson and Kintsch 1995).

5.2.1.4 Long-Term Memory

Long-term memory (LTM) contains items that that you have permanently encoded. If items are processed enough they are put into long-term memory. Examples of different types of items that are typically held in long term memory include your name, the name of your dog, and how to start up your computer.

Items can be put into long-term memory from short-term memory. These items can be objects, associations, or procedures for doing a task. Studies in this area make various suggestions about the amount of time required to move items from short-term memory into long-term memory.

Encoding is the storing of items in memory so that they can later be retrieved. Items are put into long-term memory by processing them. More meaningful processing at encoding, such as using rehearsal, seems to make it possible to retrieve items more reliably over a longer period of time (Tulving 2002).

Declarative knowledge, like knowing the capital of Nepal, requires attention to put the items into long-term memory. This task is easier if the items are already known. Being able to retrieve the name of the capital of Nepal from memory would be harder if you did not know that Nepal was a country in South America. This fact may help you to be able to recall the role of encoding better.[1]

Procedural skills, such as typing, can get put into long-term memory without attention to this process (through implicit learning), but are usually best done with attention and deliberate practice. Procedural skills initially will require attention to perform; with practice they can require less attention, like driving or typing. The amount of processing you have to do can also influence how well these memories get created, which we will cover shortly.

Retention is the interval between encoding and retrieval. Activities during retention can cause forgetting, such as processing similar items; interruptions; and so on. Length of retention interval is important, too, which is covered in the section on learning.

Retrieval depends upon the information available to cue recall. What is retrieved depends on having information on hand—a request—to start a retrieval. Retrieval is usually better (within about 10% in accuracy or response time) the more similar recall circumstances are to the internal and external encoding circumstances (even down to temperature, lighting, mood, etc.).

We can remember much more information when it is meaningful and when its meaning is processed at encoding time. Later discovery of its meaning does not especially help when trying to retrieve it. Failure to process meaningful information also makes it harder to later retrieve that information. Ability to recall is thus affected by success at each step—encoding, retention, and retrieval.

[1] This perverse statement is explained in a later section. Most students remember this part of the book.

There is some debate about how long items in long-term memory are stored. Some theories propose that items decay to the point that they are no longer retrievable, and other theories propose that the items are still there, but no longer retrievable because they are no longer unique enough to be retrievable. The implications of these two different approaches may be basically the same for system design.

5.2.1.5 Declarative Versus Procedural Memory

Descriptions of memory often divide memory into two types to illustrate different effects. Perhaps the most common is the difference between declarative and procedural memory, which is used to categorize their contents based on how they are used.

The contents of declarative memory are facts or statements about the world, such as "The hippy is in the park," "the star is above the square," and "Colleen's hair is auburn." Retrieval of items from declarative memory is improved by practice, and the items are intentionally generated for sharing and verbalizing. One study estimated that it takes about 6 s of processing to encode an item for subsequent retrieval (Simon 1974), although the time required is context dependent: more complex items may take longer.

Declarative memory is used to store and retrieve information such as simple user instructions, user passwords, and to understand materials in interfaces.

The contents of procedural memory are acts, or sequences of steps that describe how to do particular tasks. These items can be viewed as a type of programming language for cognition. Examples of items that would be stored in procedural memory include how to type, how to ride a bicycle, and many aspects of how to program or use an interface.

Items in procedural memory are generally more robust against decay, and retrieval is often less context sensitive than items in declarative memory (Jensen and Healy 1998). Like declarative memory, retrieval and application gets faster with practice.

The ACT-R theory (a unified theory of cognition, or "UTC," realized as a computer program, and explained earlier in the introduction and used in the concluding chapter) uses these two types of memory explicitly, both procedural (rules) and declarative (chunks). The Soar theory (another UTC) represents declarative information as the result of a procedure to retrieve information from memory, and is thus a type of procedural memory.

5.2.1.6 Implicit Versus Explicit Memory

Memories can also be categorized as explicit or implicit: items stored in explicit memory are reportable, whereas items in implicit memory are not. Most declarative information is explicit in that it can be reported, whereas most procedural information is implicit in that the precise details are not reportable.

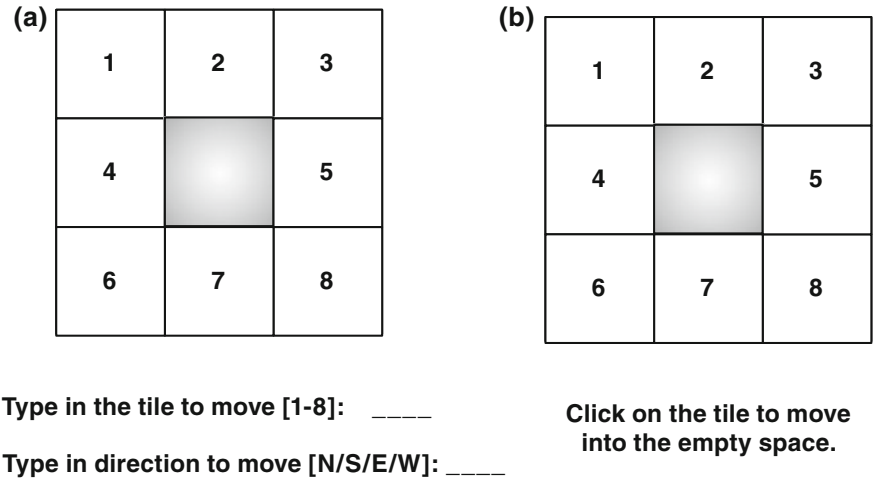

Fig. 5.2 Two types of interfaces for the 8 puzzle that lead to different types of learning. **a** Interface leading to more explicit representation. **b** Interface leading to more implicit representation

Some procedural information, such as how to use the Emacs text editor, a keystroke driven editor, starts out fairly explicit in that the user can describe what they are doing and why. Over time procedural information, or skills learned through trial and error without explicit, declarative reflection, can become implicit. In this case, the user can recognize objects, can have a way of performing a task but without being able to note why or how, and in some cases cannot even be able to recognize that they can do the task well. If the information remains in explicit memory users can perform tasks more robustly and, because they can describe how to do the tasks, they can help others more readily.

Users can be encouraged to store information in explicit memory by helping them develop a mental model of a task, and by providing them with time to reflect on their learning. Information gets put into implicit memory when the user works without a domain theory and learns through trial and error.

Work with the 8-puzzle (see Fig. 5.2) has illustrated this effect in interfaces. In this puzzle there are eight tiles and nine spaces. The task is to arrange the tiles in numerical order (left to right, top to bottom). The interface in Fig. 5.2a requires users to note the tiles that had to be moved and where to move them to, as a way of encouraging users to plan and think ahead. It also provided them with an explicit representation of steps for their reflection. Users that saw the interface that required only clicking on the tile to move it into the adjacent blank space (Fig. 5.2b) needed less explicit input and appeared to encourage behavior that led to less information being stored in memory.

In the first interface subjects reflected on their moves and developed an explicit representation of the problem. They took fewer moves to perform the task than those who used the second, direct manipulation, interface (although they took more time).

The subjects in the second interface were less able to describe how to solve the puzzle, but were slightly faster at solving it (Golightly et al. 1999; O'Hara and Payne 1998). This effect, of increased response time leading to fewer, more thoughtful, commands has been seen before (in a study by Forgie cited in Nickerson 1969, p. 171).

These results raise interesting questions about interface design. If learning is an important feature of the system, then interfaces that encourage reflection and learning about the domain may be more suitable, even though it takes more time to do the task. Tutoring systems that support the development of more explicit representations, for example, will help learners be able to explain and later teach how to do a task.

5.2.1.7 Prospective Memory

Prospective memory is also important for users. It is a form of memory that involves remembering to do something at the appropriate time based on either events or absolute times. The storage of information for future activities (both in the short- and long-term) is prone to failure and appears limited. There have been tools for centuries to help with retrieving these items from memory (strings tied around fingers), but there are also now computational-based tools to support prospective memory, such as time schedulers, calendars, and To Do lists, particularly on smartphones.

5.2.2 Mnemonics and Aids to Memory

There are several ways of improving memory performance—both storage and retrieval—which exploit the way that memory is arranged and operates. The use of mnemonics, for example, is a technique that helps to increase the amount or quality of information, or the speed at which it is retrieved.

One of the first mnemonics is the method of loci (in Latin, 'places'). In this mnemonic, a speaker (typically) would store in memory something familiar to them, like a set of locations in their house. They would then associate the items that they later wanted to with these locations. Retrieving a set of items in this way is much more robust than trying to recall each item individually without any associated, structured context. Users are not particularly likely to use this method, but you might when giving a talk, by placing the major points you want to make one per room in a familiar house. Expert memory demonstrations often use either this technique, or a similar one based on making up a story using the objects. One notable subject increased their memory span to 80 digits over a course of 230 h of practice by using athletics running times to group the digits (Ericsson et al. 1980).

Another popular mnemonic is to have a phrase to cue the retrieval of a set of things. For example, "Active Penguins Seek the Nearest Deep Pool" is a mnemonic for the seven layers of the Open Systems Interconnection (OSI) model of

networks, and "Richard Of York Gave Battle In Vain" is a mnemonic for the colors of the rainbow in sequence.

Probably the most useful aid to recalling items for users is recognition. Recognition memory is more robust than recall memory. It is easier to recognize something that you have previously seen than it is to recall what it was that you saw. Many interfaces take advantage of recognition memory by putting objects or actions in a place where they can be recognized instead of requiring the user to recall them. Dialogue boxes and explicit links in web pages are the most overt form of this. Menus hide the cues one level or more, but the same process is at work.

The trade-off here is that for experts the recognition process and its application in an interface is often much slower than the recall process. For example, looking for and then recognizing objects on a menu to perform a file manipulation task is typically slower than recalling and using keystroke commands to do the same task. More expert users, or those doing a task quite often, will be able to use recall memory, and most likely will want to for greater efficiency. The use of recognition memory appears to require offering the user multiple, related items from which the user has to recognize (and select) the one they want. This represents a design trade-off, which will be best addressed if you know your user's tasks as well as the available technologies. A simple solution is to provide shortcuts of some kind, and provide the user with access to them as they use the interface, almost turning the interface into a tutor of itself.

Anomalous or interesting things are better retrieved from memory. As noted earlier, the learning of declarative information is influenced by the content. For example, Nepal is not in South America, so if you knew this when you were reading a previous section of this chapter, you are likely to recall that section better because it stood out. This von Restorff effect was originally seen when looking at learning lists of words. If a bell was rung when a word was presented, that word was subsequently better recalled. This effect applies in general to things that are distinctive. For example, a word in red will be easier to recall if it appears in a list of words in black. (Putting all the words in red will not work nor will highlighting all of a section to memorize; the point is for objects to stand out from similar items, so highlighting the entire book does not work!)

Finally, practice and repetition of the target task helps. This is another example of the Von Restorff effect. You should now be able to recall it better because you have seen it and, more importantly, processed it more often. Practice and repetition help to increase the amount of information stored for later retrieval from memory, but only if attention is paid to the stimuli and the stimuli are elaborated. A basic and early theory in memory research was about levels of processing. Greater processing of stimuli leads to better subsequent retrieval of those stimuli. Counting the number of vowels in a piece of text does not lead to very good retention of that text. Reading the text leads to better retention, and arguing with the text or another reader helps even more; rewriting in your own words is even better. This is one reason that teachers ask you about what you have read and encourage you to process it actively.

These aids to improving memory performance are presented more formally in study guides. Another method, PQ4R, is described below.

5.2.3 PQ4R: A Way to Improve Reading Comprehension

A common question about memory is how to apply what we know about it to learning. One approach to integrate what we know about memory and learning into a study method is the PQ4R method (preview, question, read, reflect, recite, review). In this method, the learner first previews the material. This helps with distributing the learning, and it also starts to form a structure for learning, a type of elaboration (Thomas and Robinson 1972).

The next step is to generate questions that the reading should answer. These can come from the preview, and from your goals and previous knowledge.

The four Rs come quickly then. The reading is done with the preview and questions in mind. This should be a more situated experience in that the point of the reading and scope of the reading are clearer.

After reading, the material is reflected upon. This is canonically done by writing up a short summary and explicitly writing out answers to the questions from the second set.

Then the material is recited, that is, spoken out loud. This helps form different memories (verbal ones), and also adds to the distributed learning approach that has been set up.

Finally, all the materials are later reviewed, or studied. This allows another pass and also allows any questions arising during the break to be answered.

This book is designed to help support the use of PQ4R and similar learning methods that emphasize multiple passes and reflection during learning. On the book level, it does this by providing a detailed table of contents that can help with previews, introductory chapters that note why you may be interested in this material, and chapters providing structures (the ABCS, cognitive architectures) that can be used to organize the material. On the chapter level, it provides abstracts for each chapter that can serve as previews, and questions at the end of each chapter than can be used to guide learning.

You may see this sort of behavior exhibited by users when they get multiple chances to learn in the same interface using multimodal output, or when the instructions on how to use the material provide help with these stages by providing an overview, for example.

5.2.4 Memory Biases

There are other aspects of memory that need to be considered when working with and designing for users. Poor decision making, for example, is often influenced as

much by the way memory retrieval supports decision making as by how the choices themselves are made. The inherent biases associated with memory can often affect how well it operates.

5.2.4.1 Interference

The retrieval of items from memory can either be hindered or helped by other items in memory. If two items are very similar they can be confused for one other. For example, if you spend a long time knowing someone's name incorrectly, or you type in a command incorrectly a few times, it can take much longer to correct this error. This type of interference has been proposed as one of the primary aspects that makes learning arithmetic difficult (Siegler 1988). The intelligent tutoring systems based on ACT-R, therefore, do not allow the user to practice incorrect knowledge (Anderson et al. 1989). This is likely to be one of the ways that they can help students learn the same material in one-third of the typical time (Corbett and Anderson 1990). Interface designers should thus be mindful of errors and how interfaces help users to not remember them!

5.2.4.2 Retrieval Biases

Items presented towards the beginning of a sequence (primacy) or towards the end of a sequence (recency) are more successfully retrieved from memory. When it comes to reasoning about a situation or a set of activities, the items retrieved to support reasoning will be biased in those two directions. The use of an external memory aid and formal analysis can help.

The von Restorff effect will also apply. Information that is stored in a context that is distinctive will be easier to retrieve. The relative amounts or numbers of items retrieved will thus not support appropriate reasoning, as the items will not be retrieved in proportion to their occurrence in the world.

5.2.4.3 Encoding Effects

The content of an item that is to be stored in memory can be influenced by its encoding. The location of the item, the location of the user, sights, smells, sounds, and mental state can all be included in the way the item is encoded for storage. Examples of this include certain smells bringing back childhood memories, and certain people only recognized well in certain contexts. For example, we find it much harder to recognize students outside the classroom environment in which we met them.

Users will have these same effects. The information that they can retrieve for computer systems may in some cases be tied to aspects of the interface that you did not intend or may not even have control over. The particular terminal or computer

they use, their home page, or where they put a manual on a bookshelf may all become part of their knowledge about how to use a system because these aspects can influence the retrieval of items from memory. This also suggests that, for more robust recall of items from memory, users should be supported in building up a variety of retrieval cues and have a chance to practice (i.e., rehearse) these items in a variety of situations before they have to retrieve them in earnest.

5.2.4.4 Priming

Priming refers to how presenting objects before their use, sometimes quite briefly, facilitates their use and the use of items related to them. Typically, primes are presented for short periods of time, under a second, but it applies to all time periods. For example, anything about Nepal or South America would be more recognized by readers of this text because they have been used previously.

This approach can be used to facilitate the retrieval of items from memory. When presented with a list of words to remember, such as "bed rest awake tired dream wake snooze blanket doze slumber snore nap peace yawn drowsy," many subjects will report that "sleep" was included as well. This effect can be explained by noting that many, if not all, of these words prime the word "sleep." When it is time to recall the words, the word "sleep" is also very active, and this is (falsely) used as a cue that it appeared in the list as well. The book by Intons-Peterson and Best (1998) explores this and related topics of distortions in memory.

Users will find it easier to retrieve items from memory that have been recently used. This is a possible mechanism to explain why consistency within a system or between a manual and a system is important. Names of items in the manual will be more quickly identified in the interface if the same name is used. Related concepts will also be helped, but to a lesser extent.

This effect also can lead to retrieving information to support previously held beliefs, rather than generating new ones. It is important then to provide users with external memory aids to help in the retention and analyses of situations.

5.2.4.5 The Loftus Effect

The Loftus effect (e.g., Loftus 1975), also known as the misinformation effect, describes how people take on knowledge implicitly from questions. Asking someone a question often leads them to assume and later learn the facts implicit in the question. Asking someone "What color hat was the suspect wearing?" will lead many people to infer and later recall (without understanding that they just inferred and did not see) that the suspect was indeed wearing a hat. In one of their original studies Loftus and her colleagues asked subjects if they recalled meeting Bugs Bunny at Disney World as a child (Loftus 2003). Depending on the condition, about a third could erroneously recall this (Bugs is with Warner Brothers, not Disney!). Instructions and manuals can suggest capabilities not generally

available, or could help support users by noting the interconnected nature of most systems. Multiple choice exams can, with their questions, remind you of components of the answer. For example, a question that asked you "Which of the following is a type of memory?" implies that there are several types. This type of test could help create false memories.

If used directly, this effect can help users. Queries to users that provide context to the question or even a default action can help users.

5.2.5 Implications for System Design

What we already know about human memory provides numerous implications for improving the lot of the user. Users will be able to learn interfaces that are similar to other interfaces or other knowledge structures more easily than completely new interfaces. Users are limited in the amount of new information that they can perceive, comprehend, and learn. A fairly direct implication is to use words that users know, and use the words consistently to strengthen the chances of later successfully retrieving these words from memory.

It has long been known that retrieving names from memory is faster than naming objects. This suggests that, instead of displaying icons, we might be better served by displaying words (Chilton 1996). The approach, of using names instead of icons would also help the visually impaired, who rely on software to translate computer interfaces into verbal representations. Raskin (2000, Sect. 6.3), provides further interesting examples and arguments encouraging the use of words instead of icons. Figure 5.3 shows icons used for translation during international travel, as well as some icons from interfaces that are more difficult to interpret.

Our understanding of memory can and should influence password choice. This knowledge can inform more than "experience and common sense," which is sometimes used. Users will want to choose passwords that are hard to guess, and systems may enforce this. However, many passwords that are hard to guess are arbitrary (that is, not meaningful to users) strings of letters, digits, and punctuation, such as "ouibou94!3"). Thus, some strong passwords are hard to recall when users need them. There have been surveys on what passwords people use when they are not given any guidance (the results are pretty scary from a security standpoint). What needs to happen more is helping users generate mnemonics for remembering the passwords, rather than writing them down on paper or installing them in a computer-based passport (unless you trust the computer company selling the passport, which many state attorney generals do not). The issue of recallable yet strong passwords is one of the topics routinely dealt with under the auspices of the topic of usability of security. Some analysts have argued for word-based passwords, such as 'islandcarekeyboard', which is long and thus hard to guess, but easier to recall as each chunk is several letters rather than a single letter.

There are several poorly understood but important questions relating to the use of memory with respect to system design:

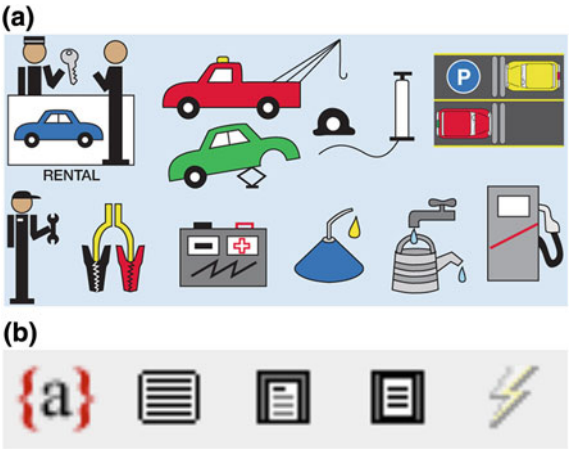

Fig. 5.3 The *top* figure is taken from a Kwikpoint travel guide designed to help communicate in a foreign language (you point at what you mean), used with permission. Icons on the *bottom* are taken from Word 2011 (The icons in Fig. 5.3b mean, from right to left: view field codes, zoom 100%, online layout, page layout, and automatic change. Word is improving in that you can now mouse over the icons to learn their names.)

- The impact of interruptions. Many work environments include numerous interruptions. The effect of these on the way that items are stored in, and retrieved from, memory is just starting to be studied. What work exists suggests to us that the length of interruption is less important than the similarity of material processed.
- Other memory tasks. Most of the memory literature concerns simply learning a list and then recalling it sometime later. Many work-related memorial tasks involve retaining some piece of information for a short time, then replacing it with some similar piece of information ('keeping track' tasks). These have not been studied very often.
- Support for memory. Most of the literature is studies of processes and architectures of memory—but our interest is in preventing the need for memory and providing support for tasks requiring the use of memory. This is an active area for design.

5.3 Attention

> Everyone knows what attention is. It is the taking possession by the mind in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought…It implies withdrawal from some things in order to deal effectively with others, and is a condition which has a real opposite in the confused, dazed, scatterbrained state.
>
> William James

Fig. 5.4 This picture shows someone (*circled*) doing more tasks than their system (both vehicle and head) were designed for—driving in a crowded urban environment and talking on a hand-held cell phone. The phone in this position takes both verbal and motor attention, and it causes an additional planning load to operate the car one-handed, and leads to poor situation awareness (i.e., they did not see the person on the street who took this picture). Also note the Gestalt effect of the can and tree!

Designing systems to attract, manage, and maintain attention is important. Figure 5.4 shows a user with a task that requires attention (driving), who is also attempting to perform an additional task (talking on a cell phone). If you would like a short demonstration of attention, go to http://www.youtube.com/watch?v=-AffEV6QlyY, which takes advantage of your ability to pay attention. It takes some time to understand, and you should not be too easily shocked.

Attention refers to the selective aspects of perception which function so that at any instant a user focuses on particular features of the environment to the relative (but not complete) exclusion of others. There are several useful metaphors for describing attention. It can be seen as a set of buffers that hold information for processing. It is directly related to that processing as well.

You might imagine attention as the area and process of central cognition, such as work being done on a table. Tasks that take less attention need less of the table or less of the time of the person working at the table. Many processes in human cognition are thus closely tied to the concept of attention as a space for cognition, and various parts of a processor can support attention.

Figure 5.5 shows two common and important uses of attention. Both situations involve aspects of motor control and cognition, and require monitoring the situation and taking corrective action. In both cases performance suffers when the tasks are not given sufficient attention.

There are also other aspects of human behavior that are not based on focal or conscious attention. We can still do some information processing without consciously attending to information—how else would we know to shift our attention

Fig. 5.5 A car dashboard at night (*left*) and a plane's cockpit during the day (*right*). Note similarities and differences, including the use of luminance, color, and some similar interaction devices

to some new information? An example of this is the 'Cocktail-party effect', where you can hear your name in a different conversation even though you are not paying attention to that conversation in a crowded room.

Many researchers believe that our ability to process information without focal attention is limited to surface features, syntactic properties, or similar shallow features. In this view we cannot process the meaning of something without conscious attention. So we cannot remember something in the long-term without paying conscious attention to it. This is thus the difference between hearing and understanding, for example. Some evidence suggests that some meaning can be processed without attention, but not very much, and that any long-term memory that results will almost exclusively be of surface features.

The study of skilled, procedural performance reveals that we can, after extensive practice, perform many things without paying much conscious attention to them. Some of these skills (e.g., driving) are very sophisticated. With practice we do not have to pay attention to as many subparts of the task. This allows us to do another task at the same time (such as talk while drive). It also means that less is actively processed, decreasing our memory for those aspects of the task. This perhaps explains why it is hard to give directions for familiar routes, as the features are no longer processed by attention.

5.3.1 Wickens' Theory of Attentional Resources

There are some more elaborate theories of attention. Perhaps the best known of these is the theory of attentional resources developed by Wickens (e.g., Wickens and Hollands 2000), illustrated in Fig. 5.6. In this theory, Wickens proposes the idea that users have multiple types of resources as a way of explaining how people time-share across tasks and variations across people. Resources affect which part of perception

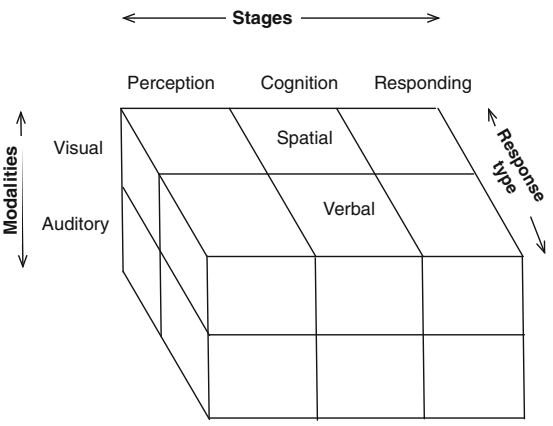

Fig. 5.6 The conceptual components of Wicken's model of the user's processing, modality, and places for attention

is used (visual or auditory), response type (choice of response type, and execution of that response, spatial or verbal), and the stages of processing (perception, cognition, and responding). These define resources for processing and holding information, with some tasks using more of one type of resource than another.

Most of the evidence to support Wickens' model of attentional resources comes from studies of dual task performance. Evidence for two different types of resources, for example, can be found by varying the difficulty of responding to one task and looking at performance on a concurrent (more) perceptual task. The performance on the perceptual task is (more or less) constant, even though more resources are needed for the responding task.

5.3.2 An Information Processing Model of Attention

The ACT-R theory (Anderson and Lebiere 1998) incorporates a model of attention that summarizes one of the common theories in this area, that of attention as a spotlight in the mental world which is directed at what is being thought about. Attention in ACT-R is represented as activation of concepts in its declarative memories. People with more attention have more activation available (Lovett et al. 2000). This activation of memory creates the spotlight that focuses on the mental objects that are being used in processing. These objects may include semantic memories and episodic memories, as well as current goals and processes.

If more activation is available, this allows more objects to be manipulated at the same time. This saves time re-retrieving them from memory and allows larger, more complex objects to be created and used. Each of the objects has an associated strength. Objects that are more familiar have a higher associated strength and, as a result, need less activation to be matched against procedural knowledge than less familiar objects.

Objects in the various perceptual input buffers, such as vision and hearing, need to be brought into memory in central cognition by a transfer process. Moving objects into central cognition does not happen automatically, but has to done intentionally. The amount of time the new memories are processed and how they are processed will determine how strong the resulting memories are.

5.3.3 Divided Attention

Users who are attempting to do two tasks at the same time will have to move more information around. They will start doing one task, and during processing on that task they will have to notice when it is time to swap tasks and work on the second task. In other words, they will need to divide their focus of attention between the tasks they are performing.

How the two tasks interact will depend on several things. Users will perform better if they can spend larger blocks of time on each task. Users that are more practiced with each task will be able to switch more smoothly and will be able to activate the necessary memories to assist in doing the new task faster. Practicing doing the tasks together will also lead to better overall performance. In this case, the user might be learning how to interleave subtasks.

Attention is influenced by both the frequency of access and type of use of items, so dual tasks that use different perceptual buffers will interfere less with each other. People can learn to drive and talk at the same time in normal weather conditions, because driving does not use a lot of audio cues. That leads to the two tasks not using the same perceptual buffers very much. At least one theory in this area, EPIC, proposes that the only bottlenecks in performance are perception and action (Kieras et al. 1997; Meyer and Kieras 1997).

Pairs of dual tasks, such as reading email and web browsing, based on a computer interface will almost certainly interfere with each other to some extent. The user will be using the same resources, including perception, aspects of cognition, and output, typically vision, memory, and motor output, for each task. The perceptual buffer will have to be refocused or directed to a different part of the screen. These factors will make it harder to do each of the tasks when the other is present.

5.3.4 Slips of Action

Attention and skilled behavior have a critical role in the occurrence of slips of action, where people have the right intention but perform the wrong action (Norman 1981; Reason 1990). People essentially work in one of two modes:

1. Using open loop control: behavior is based on anticipation and feedforward rather than feedback, so there is little or no need for them to monitor the result of their actions. Other activities can be performed at the same time—automatized driving and game play and typing are examples.

2. Using closed loop control: performance is mediated using feedback on any actions carried out, so conscious monitoring of behavior is required. Only one activity can be carried out at one time. Learning how to drive, deliberate walking on a rock field, editing a manuscript are examples, or learning how to play a game by mapping commands to a controller.

Very few tasks can be performed completely using open loop control. Slips of action typically occur when open loop control is being used instead of closed loop control, such as:

- When users overlook information that affects behavior. For example, users that do not note the mode of a word-processor or the location of the insertion point and input and modify text in ways that they did not intend.
- When users continue performing a familiar activity even though they intended to do something different. For example, clicking on the "Yes" button in response to the prompt "Do you really want to delete this file?"
- When users fail to correctly discriminate between relevant objects in the world—performing an action on unintended objects. For example, trying to click on an image icon on a web-based form.

Although it is not clear that we can design to always pre-empt these slips, we can predict the kinds of circumstances when they occur. We can also recognize the power of open loop behavior that can foil "Do you really want to…?" dialogues. We will return to discuss slips of action and other types of erroneous behavior in Chap. 10.

5.3.5 Interruptions

Interruptions are becoming increasingly common in interfaces as systems become more advanced and can be given tasks to perform asynchronously. There are also other causes of interruptions, including colleagues, phones, and email systems that beep when you have new messages.

In many ways interruptions can be seen as a secondary task, and hence the work in this area is effectively a subset of the work on dual task performance, where the user is trying to do two tasks at once. Interruptions are effectively the secondary task, which is much less important, generally not regarded as part of the user's main task, and is not under control of the user.

Interruptions do appear to cause real decrements in performance (Bailey and Konstan 2001; McFarlane 1999) and users do not like them because they lead to changes in affect (Bailey and Konstan 2001). McFarlane (1997) has attempted to create a taxonomy of interruption types and their implications for design. His results indicate that the choice of how to deal with interruptions will depend on the importance and types of the main task and of the interruption. Dealing with

interruptions immediately disrupts the main task more than taking the interruption between subtasks. Dealing with interruptions at scheduled times leads to better performance on the main task, but poorer performance on the interruptions. Intermediate strategies offered different trade-offs (McFarlane 1999).

Sometimes interruptions are useful. It has been hypothesized that interruptions are useful for solving hard problems (where there is a so-called incubation effect). This effect has been noticed by programmers who get stuck on a problem—coming back to a hard problem later sometimes makes the problem easier. Being interrupted for minutes, hours, or days in these tasks allow the user to forget their mistakes and mis-starts, or to receive suggestions from the environment. Kaplan (1989) found that at least part of the incubation effect came from cues in the environment. One problem he gave subjects was "What goes up a chimney down but not down a chimney up?" Subjects who were called and asked by Kaplan if he had left an umbrella in their office were much more likely to solve the problem, but did not attribute the cause to his call.

As you consider the larger context of your users' tasks, you should keep in mind the possible effects of interruptions. Some interfaces will be less sensitive to the effects of interruptions. We are just starting to be able to predict and measure these effects.

5.3.6 Automation Deficit: Keeping the Human in the Loop

If you want to get funny looks, while you are driving a car, ask your passenger "Do you ever wake up and find out you are driving and wonder where you are going and where the other cars are?" Some find this humorous, and others, if they do not know you well, may be startled. This is similar to what happens when someone has to resume or take over a task at short notice when the system relinquishes control because it does not know what to do. This happens to people such as aircraft pilots and power plant operators who find themselves *out of the loop*, i.e., not being kept up to date by the systems about what it is doing. The user will then allocate their attention to dealing with the task, but they have to spend time and effort trying to understand the current situation before they can diagnose the problem and resolve it.

Generally, the way to avoid this problem is to include more status information to keep the user continually informed before the situation requires their input. This allows one to develop and maintain a mental model of that situation and how it is unfolding, so that users can decide when they may need to take action.

This whole argument about the need to keep the human involved in the control loop was first summarized by Bainbridge (1983), and it remains as an issue today (Baxter et al. 2012). Woods and Patterson (2001) explain some of the effects on users if the technology does not keep them kept informed about what it is doing.

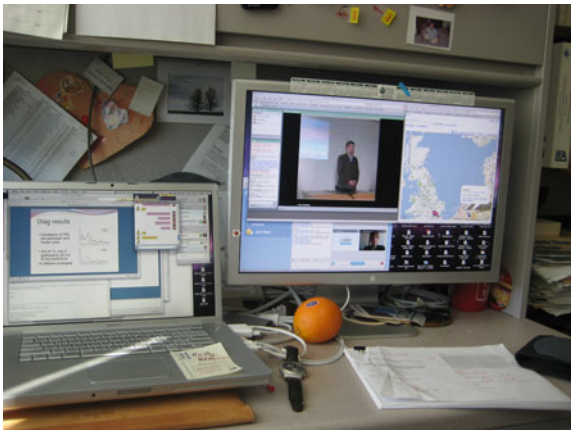

Fig. 5.7 This system shown here included an instant messenger (iChat) text and video chat, a proprietary video conferencing tool, slides of the talk online and printed, as well as paper-based reminders and notes. In this task, however, the subtasks were integrated: to watch the man in the suit talk and ask him questions

5.3.7 Implications for System Design

Interfaces can help users do more than one task. Figure 5.7 shows a workplace where the user is doing multiple subtasks related to listening remotely to a talk (watching the talk, reading the talk slides, looking at the map of where the talk is, timing the talk, reading a paper about the talk, waiting for a snack). Typically, this is by helping them do each of the tasks more easily and by requiring less attention to do each of the tasks. More recent work is attempting to find the times and ways to interrupt the user at more appropriate moments.

Knowing how attention and perception work can help create better indicators of the arrival of another task, such as a beep or flash, or a change in its priority. Knowing about attention also helps with design because you know that attention is not limitless, and is sometimes missing. How and when to best interrupt a task and how to best support various sets of dual tasks remain open problems.

It is possible to increase the user's attention by including multiple modes of input and output. For example, when users need more attention, adding voice input and voice output as interaction modalities to supplant a visual display can provide users with the ability to watch their performance in a flight simulator. That way, they are receiving feedback on it at the same time (Ritter and Feurzeig 1988).

In applied settings, where attention is paid to multiple processes, some that are time critical, it can be important to pay attention to a process before it becomes necessary to control the process. The section on automation deficit explained this more fully.

5.4 Learning and Skilled Behavior

Learning is where performance changes with practice, typically getting faster, becoming less demanding, and generating fewer errors. Learning is important for users. They learn in several ways, including learning new information and new

skills. Learning is the primary way for them to improve the strength of their memories. As they retrieve something more often, their memory strengths improve as well. Thus, retrieving a password is faster and more robust after 2 trials than 1, and after 100 trials than 99 (all else being equal), but with ever decreasing improvements with practice.

Users also use learning to help with attention. As they learn more, they can recognize situations and generate responses faster. This gives them more time to pay attention to a task or to a secondary distracting task.

Users can also learn to adjust their behavior to fit to the interface. If the interface does not describe an action the way the user would, the user typically has to learn the new name, which they can and will do. If the interface has steps or substeps to an action, the user has to learn these as well. If objects are not where the user expects, the user has to learn their location. The good news is that learning can often happen without much distress to the user, but woe to the interface that requires the user to learn everything!

We will present here a theory of how users learn. This theory has many implications, including how users' behavior changes with learning, what extreme practice looks like, and what this can mean for interface design. For more information on learning, see Anderson (1982, 1995, 2004) and Lindsay and Norman (1977).

5.4.1 The Process of Learning

A description of the general process of learning (and types of task performance) has arisen in several areas, including behavioral psychology (Fitts 1964), cognitive ergonomics (Rasmussen 1983), and cognitive psychology (Anderson 1982). These theories all start with the learner acquiring declarative information about the domain. At this stage, problem solving is very difficult. The concepts are not always connected to each other. The learning might be seen as learning how to learn more about the task. What has been learned at this point might not be enough to be able to do the task yet. Basic declarative knowledge of the task is learned at this stage, such as the initial problem-solving state and the actions available for problem solving. Problem solving, when it is possible, requires considerable effort, and is not always correct. Problem solvers might not have much confidence in their performance. Anderson calls this the cognitive stage. Rasmussen calls this type of behavior knowledge based, as shown at the top of Fig. 5.8.

When learners are at this first stage, behavior occurs at the most fundamental level, when there are no direct rules to inform the user what to do. Expert users may have to resort to this stage in emergencies or novel situations, such as when birds get ingested into an aircraft engine, or when trying to change Unix permissions on shared files. These conditions require deliberate thought and reasoning about the state of the situation or system based on the user's knowledge (i.e., mental model) of the system, and then about what action to perform next. As you might expect, work proceeds slowly because each step has to be deliberated over from first principles.

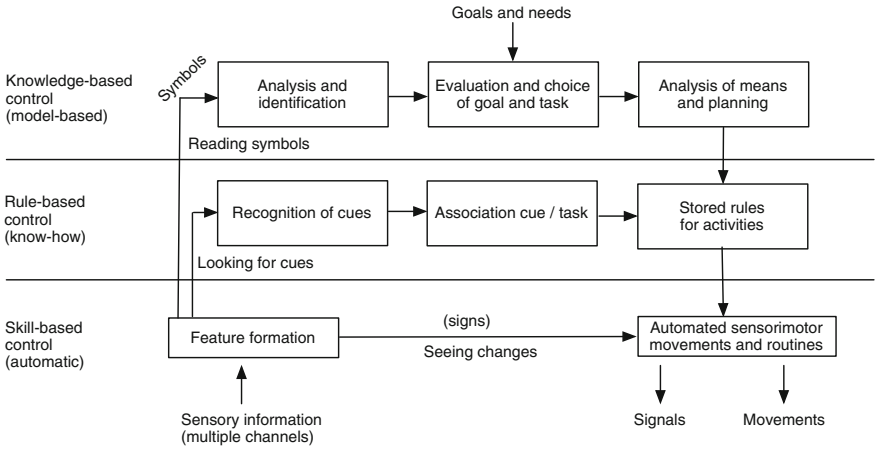

Fig. 5.8 A schematic of Rasmussen's theory of performance. (Adapted from Rasmussen 1983)

With learning and practice, the user progresses to the associative stage, or rule-based stage (as shown in the middle of Fig. 5.8). The learner can solve problems more routinely and with less effort. The declarative knowledge has been compiled into procedures relevant to the task domain that can be performed directly. At this stage users may often be able to just recognize what needs to be done. For users of complex systems, behavior becomes a conscious activity and is based on familiar rules, either dictated or acquired, such as changing lanes when driving a car, or formatting a paragraph in a Word document using styles.

The final stage, called skills or tuning, tunes the knowledge that is applied. At this point the user still gets faster at a task, but the improvements are much smaller, as they are smaller adjustments. New declarative information is rarely learned, but small adjustments to rule priorities happen. At this point users consider themselves, if not experts, to be real users. Rasmussen calls this level of performance skill-based. Anderson calls it the autonomous stage.

At the skill-based level, performance is much more automatic, as shown at the bottom of Fig. 5.8. Skill-based performance usually occurs when users are performing routine functions in their normal operating environment. Much of their behavior is no longer available to conscious thought, or available for verbalization etc. Users not only perform the task faster, they may also appear to have more attention to provide to other tasks. Examples include changing gear in a car with a manual shift gearbox, and cutting and pasting text.

These stages of learning have been noticed in several areas of formal reasoning. Formal reasoning involves problems with known goals (like solve for x), and equations or rules that can transform representations (such as adding 2 to each side of an equation). Formal reasoning is used in areas such as physics problem solving (Larkin 1981; Larkin et al. 1980a, b), geometrical proofs and solving algebraic problems.

In these types of problems the problem solvers (and computer models) start with domain knowledge and the goals for which they are trying to find answers. Novices work backward from the goal. In the case of physics problem solving, the task is to derive a value of a variable (like final speed) given some known variables (like mass, initial speed, and force) and some equations (like Force = Mass × Acceleration). Novices tend to work back from what they are trying to solve for final speed, chaining inference rules together until they find known variables. If they need to compute the final speed then they look for an equation that computes speed. Then they look for more equations to find the variables in the first equation, until they bottom out with known variables. This is known as backward chaining or bottom-up reasoning.

As the reasoning terminates in the answer, how to apply the rules in a forward sense, without search, is learned. More expert behavior is a mix of these approaches. With even more practice experts can reason in a forward sense. "If I have speed and time then I have acceleration, then if I have acceleration I have….." The answer in this case is found more quickly and with less effort. This is known as forward chaining or top-down reasoning.

This type of reasoning difference is seen in computer repair and troubleshooting, software usage, and other domains where formal reasoning can be applied. Better interfaces will support the novice by providing the appropriate domain knowledge needed to learn the inferences, and support the novice and expert by providing the state information to reason from.

Learning can also occur in the opposite direction, from (implicit) skills back to knowledge about these skills, but this learning is less well understood. Implicit learning can occur when the user is working at the skill level, with knowledge eventually being derived on the cognitive level through the observation of one's own behavior. This type of learning, which is an active area of research in psychology, often leads to or arises from strategy changes.

Whichever way the user learns, some key phenomena survive:

- The ability to recognize correct/incorrect items comes before the ability to generate correct items.
- Knowledge is not acquired in an all-or-nothing way. Novices go through a stage of fragile knowledge, trusting that what has been acquired is correct, whereas sometimes it may be incorrect.
- Experts acquire a rich repertoire of representations of their knowledge. It is not only that experts know more than novices; what they know is much better organized, understood to a greater depth, and more readily available.

5.4.2 Improvements from Learning

Perhaps the biggest regularity of human behavior in general and users in particular is that the changes due to learning leads to them getting faster at performing a task the more they do it. Similar curves showing decreases in performance times have

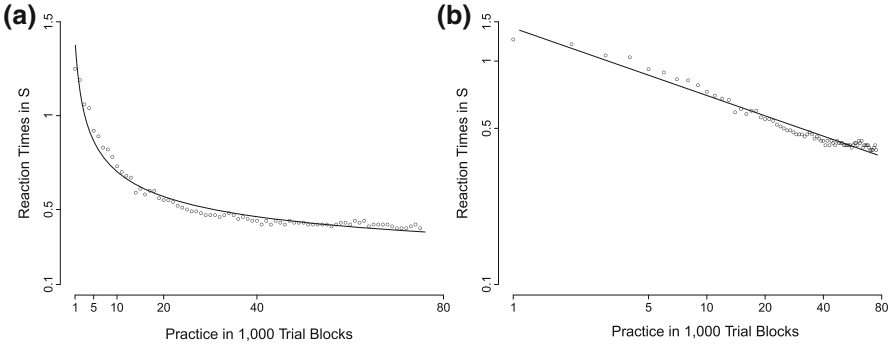

Fig. 5.9 Time to perform a simple task (pushing a combination of buttons) on a linear plot (**a**) and log–log plot (**b**) as well as a power law fit to the data shown as the *solid line* on each plot (adapted from Seibel 1963, and previously used in Ritter and Schooler 2001, reproduced here with permission)

been seen in tasks ranging from pushing buttons, reading unusual fonts or inverted text, doing arithmetic, typing, using a computer, generating factory schedules, all the way up to writing books (Ohlsson 1992). These improvement curves are also found when large groups work together, for example, building cars.

There are huge improvements initially, although users rarely report satisfaction with these improvements. The improvements decrease with time, however, following a monotonically decreasing curve. This is shown in Fig. 5.9 for the Seibel task, a task where you are presented with a pattern of ten lights and you push the buttons for the lights that are on. Each point represents the average for doing 1,000 patterns. Notice that with extended practice, performance continues to improve, but by smaller and smaller increments. Over a wide variety of tasks and over more than seven orders of magnitude (hundreds of thousands of trials), people get faster at tasks.

With data with changes this wide and with small changes becoming important later in the curve it becomes difficult to see the improvements. This is why the data are plotted using logarithmic axes: the difference between two points is based on their logarithms, as shown in Fig. 5.9. Typically, on these log–log plots, learning curves follow pretty much a straight line.

The mathematical representation of this curve is currently somewhat disputed. Some believe that the learning curve is an exponential curve, and others think it is a power equation (thus, called the power law of learning) of the form shown in Eq. (5.1):

$$\text{Time of a trial} = \text{Constant1}\,(\text{Number of trial} + \text{PP})^{-\alpha} + \text{Constant2} \qquad (5.1)$$

where Constant1 is the base time that decreases with practice, PP is previous practice on the task, α (alpha) is a small number typically from 0.1 to 0.5, and Constant2 is the limitation of the machinery or external environment (reviewed in Newell 1990, Chap. 1; see also Anderson 1995, Chap. 6).

PP is either estimated, measured directly, or ignored. In most cases it is ignored because the task is unique enough to the learner. In other cases, such as taking up a familiar task, the equation does not fit as well.

Constant2, the minimum time due to the limitations of the environment, is computed from equations describing the world or using physical equipment. For example, you might measure how long it takes a ball to fall to the ground with a camera if the task involves catching a falling ball, or you might record how fast an interface can accept keystrokes when driven by a computer program.

α is thus found typically by plotting the data on a log–log plot, and fitting a straight line. This fitting is typically done by fitting a linear equation (which has a known closed form solution) to the log of the trials and the log of the time. The more accurate but more computationally expensive approach is to fit the power law equation including the constants that are appropriate using an iterative algorithm.

Having an equation to predict learning is important for several reasons. For science, it helps summarize learning, and comparison of the constants is a useful way to characterize tasks. It also has practical applications. For engineering, design, and manufacturing, it predicts how fast users will become with practice. These equations are used in manufacturing to predict factory output and profitability.

Others believe that the match to a power law is an artifact of averaging the data from multiple people and multiple series, and that the curve is best described as an exponential when the data is examined in its purest form (Heathcote et al. 2000). Both cases have basically the same implications for users with limited lifetimes and physical equipment, but have different implications for the details of how the underlying cognitive architecture is implemented.

The improvement in performance time itself does not appear to delineate the stages of learning noted earlier. This may be because the first stage of learning, where performance might be quite difficult, has to reach a nearly complete level before the task can even be performed. There are hints of this in Fig. 5.9b, where the initial slope is fairly shallow in the log–log plot, perhaps more shallow than would be expected. In complex tasks, the transition of rule learning and rule tuning within task might lead to steady improvement, and be masked by the large number of rules and sub-tasks. Looking at individual users working on well-measured tasks may allow these stages to be seen. When this has been done, strategy changes can be seen (Delaney et al. 1998).

In addition to time reducing with practice, several other aspects of performance improve as well (Rabbitt and Banerji 1989). Errors decrease with practice. The variance in the time to do a task also decreases. That is, with practice the range of expected times decreases. Some think that this decrease in variability is what leads most to the speedup because the minimum time to perform a task generally does not decrease with practice (although there are clearly notable exceptions to this, for example, for tasks that cannot be completed by novices).

There appear to be two places where learning does not get faster. Users cannot get faster when the machinery they are working with cannot keep up with them.

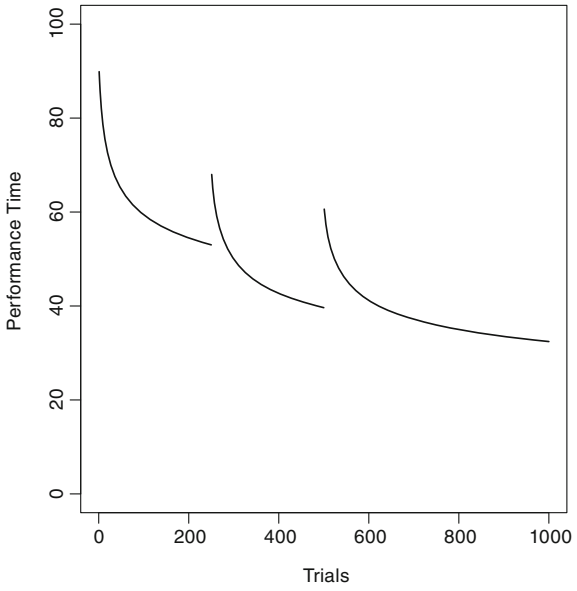

Fig. 5.10 The learning curve for a series of better strategies, showing the initial slow down and then savings due to switching strategies

This was first noted when cigar rollers improved up to a point and then stopped. It was found that the users were rolling faster than the machine could pass materials to them (Snoddy 1926).

The second place where users do not get faster is when they change strategies. As a user picks up a new strategy, they will often experience a slowdown, moving back on the practice curve for that strategy. However, with practice, performance on this curve with a lower intercept improves, usually to be much better than the previous strategy (Delaneyet al. 1998). Figure 5.10 illustrates this effect.

If your system supports multiple learning strategies, you should consider helping the user transition between them. In text editors, for example, there are several ways to find a particular text string, including scrolling and searching line-by-line, scrolling and searching by paragraph, and using the inbuilt search function. In one survey (Card et al. 1983), most users were not using the most efficient strategy (searching), but moving line-by-line. Experts use searching, which can be 100 times faster than scrolling.

5.4.3 Types of Learning

Learning can be described in several ways. It can be organized by the types of memories that are created or practiced. One common way to distinguish types of learning is as declarative and procedural. Another common way is as implicit and explicit. The distinctions between these classifications are still being argued over,

but they represent useful and interesting differences about users that help with interface design.

Declarative learning is learning facts (declarations). The "power button is on the keyboard" and "the computer manager's office is in 004A" are two examples. Declarative learning can be separated into two subtypes of recognition and recall. Recognition memories are easier to build than recall memories. That's why multiple choice tests seem easier—you just have to recognize the answer. This corresponds to the first stage of declarative learning.

Procedural learning is learning how to do procedures. Using an interface, and playing a computer game are examples of this. Procedural memories are probably more complex than declarative memories in that they generally support the skill being performed in a wide variety of environments and slightly different situations, which thus represent more knowledge. These memories have to come after the declarative representations are available to create them.

These two types of learning have different regularities associated with them. Declarative learning can, by definition, be described and reported. Procedural memories cannot be directly reported (Ericsson and Simon 1993). You cannot directly describe the knowledge you use to ride a bike. You can, however, accurately report the declarative knowledge that you use to generate your procedures (like keep your weight balanced) and what is in your working memory as you are doing the task (there is a parked car ahead). You can also watch yourself do a task and attempt to describe much of what you were paying attention to as you did it. What you think you were paying attention to when doing the task will, however, be dependent on your mental model of the task and how demanding the task is. This approach is called introspection.

There are fundamental problems with introspecting like this. While introspection can lead to useful and helpful insights, it does not lead to complete and valid theories of human thinking (Ericsson and Simon 1993). Just as you can't program a computer to write out the instructions as it does them (the instruction to write out replaces the instruction it copies), you can't think about thinking very accurately while thinking. Mainstream psychology has rejected introspection as a true representation of how people think, but you may find it useful for inspiration for ideas that can be later validated by other approaches. In our experience it is sometimes useful inspiration, but it is sometimes just way off because of our biases about how we would like to think we think.

As an example of this, users think that they cannot learn new key bindings between keys and commands (e.g., <Ctrl-s> to search vs. <Ctrl-f> to search). When key bindings in an editor were changed on users on the second day of a study about learning to use a text editor, for the first hour or two the users felt much, much slower. They were slower than they were at the end of the first day, but faster than when they started. They regained their skill level by the end of the second day (Singley and Anderson 1989). This study illustrates three important things about users. The first is that they are always learning. The second is that introspection often leads to incorrect conclusions. They rather disliked the new interface, but they adapted more quickly than they thought they did. The third is that the users were able to transfer much of

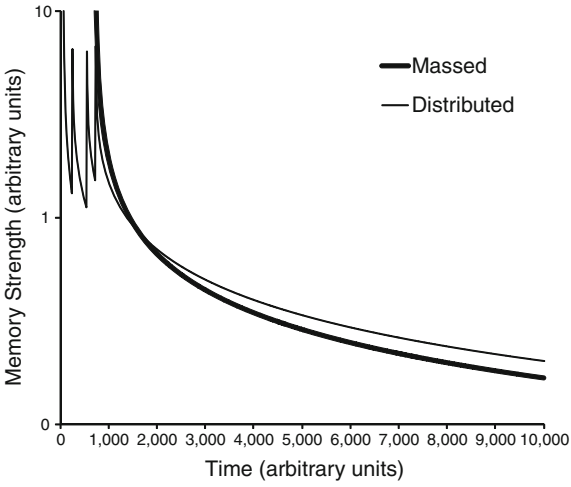

Fig. 5.11 Massed versus distributed practice in relation to time to learn a list. Given exponential decay, which is often assumed for memory, the *solid line* shows a given amount of practice as one block, and the *dashed line* shows the same amount spread out over a longer period. The distributed practice has higher activation after the second practice, and will for the remainder of the curve

what they had learned from the old interface to the new interface. Only the key bindings changed; the underlying approach and the command structures did not, so the users were able to apply most of what they had learned. Another similar example is that users seem to prefer mice over light pens, even though the light pens were faster to use (Charness et al. 2004).

Another way to represent learning is with the implicit/explicit distinction. Implicit learning seems to be automatic, is based on practice, is not improved by reflection, and produces knowledge that cannot be verbalized. This might be roughly equivalent to the rule tuning stage. If the rules are created based on a simple domain theory, but a more complex domain exists, then additional learning can occur.

Explicit learning proceeds with full consciousness in a hypothesis testing way; it produces knowledge that can be verbalized. This is examined in more detail in a later section on problem solving.

These distinctions become important when teaching users how to use an interface. Some information is reported to them as declarative knowledge to be learned (where things are, who other users are, and what are the objects), and some information consists of procedural skills such as how to do a task.

Learning can be massed or distributed. Massed refers to learning that occurs at a single time, for example, cramming for a test. Distributed learning occurs with breaks in time between the learning episodes. Figure 5.11 shows how much better distributed learning can be. Distributed learning takes less total time (sometimes one-third of the time), and the retention is better, sometimes 50% better. Anything you can do to assist your users to learn in a distributed fashion will help their learning. Some interfaces now put up hints, which appears to be a way to support distributed learning.

Finally, we should note again that learning is rarely complete. It is easy to think that if you have learned to operate a system your learning about that system is complete. This needn't be so; many users can sometimes perform very competently with very little knowledge, and other systems, such as UNIX and the Emacs editor are complex enough that once competent, users can continue to learn new ways to use them and new components for several years.

5.4.4 Skilled Behavior, Users in Complex Environments

Real human skills are a complex mixture of these levels of learned behavior. In most cases, routine users will work with an open-loop behavior at the skilled level. That is, they will be able to perform most tasks in a routine way using existing, well-used knowledge and not check all of their steps. Their behavior will be open-loop, that is, they will not check all of their work. They will not close the loop by checking that things worked correctly because in most cases they no longer need to. If they do make mistakes, they will be able to recognize them quickly. They will continue to get faster with practice, but the improvement will be minor. There will also be some closed-loop behaviors for tasks that are less well practiced, and in these behaviors users will be more careful and check their work.

For example, airplane pilots often operate at all three levels. Some aspects of their behavior are automatic; for others they refer to rules and procedures, whilst for others, particularly in non-routine emergencies, they reason on the basis of their knowledge about the plane and learn. (In routine emergencies they will use checklists.)

Rasmussen's (1983) argument is that good design needs to support all three levels of operation, not just one. We can also note a social human factor here. If there are multiple operators, they may operate at different levels at different times because at the knowledge level they may have different knowledge, which may give rise to conflict or to strength, depending on how these differences in approach are resolved.

The knowledge level implies a certain amount of planning activity, but you will find there are those who believe that people do not engage in planning—arguing that behavior is situated in a context (e.g., Suchman 1983). In other words, they argue that we perceive the situation and decide what to do then, not on the basis of some pre-formed plan.

Two responses can be made to this.

- One can do both—have a plan and allow the situation to change it (e.g., performing actions in a different order from that planned).
- Planning seems best related to Rasmussen's knowledge-level, whereas situated action is 'rule-level' behavior. Again, these are not exclusive options—some people in some skills may function exclusively at one level, whilst others may switch between levels at high speed.

Fig. 5.12 A scene from a fishing port in Massachusetts where knowledge can influence perception (There is a boat that has sunk next to the dock. Only its mast is visible next to the boat on the right. The book's web site provides a more detailed picture showing the missing boat.)

Learning will also influence perception. It was not immediately apparent to the photographer from the photograph in Fig. 5.12 what was wrong with this scene (although it looked 'odd'), but it would be apparent fairly quickly to a boat captain. This is a simple example of where knowledge or learning, can influence other processes, such as vision.

5.4.5 Expertise

With extended practice, users become experts at their task. Generally, to become world class, it takes about 10 years of practice as well as some deliberate reflection and declarative learning, either from extensive self-tutoring or from a coach (Ericsson 1996; Hayes 1981; Simon and Chase 1973; Simonton 1996). There are exceptions, but these researchers argue that exceptions are truly rare. Less time is required to attain local or national prominence in a particular field, or where a task or technology is new. While practice is a usual and necessary condition, simple practice is not enough to guarantee an expert level of performance: coaching and deliberate practice are needed (Ericsson 1996).

This level of performance is interesting to people because it shows how good performance can be. When the practice is on a socially important task, such as flying planes or programming, it can also be quite rewarding to the practitioners.

Some theories suggest that with practice the user gains more than just speed in the task. In some instances they appear to have greater memory for, and can pay more attention to the task at hand (Ericsson and Kintsch 1995). In all cases, they have more knowledge and better anticipation of what will happen in the task, and in nearly all cases they have more accurate perception for and of the task details.

5.4.6 Transfer

Transfer of learning is important. After a skill has been learned, the goal is often to reuse or apply the knowledge to a new situation. If no transfer occurred, then every situation would be a new situation. If transfer was perfect, then few situations would be novel. Studies have shown that transfer is usually far from perfect, that it can be underestimated, and that it is possible to predict transfer effects.

Perhaps the earliest study on transfer had subjects read a story about how a king wished to invade a town, but his army was too big to come through a single gate in the town. So he split his troops up into smaller parties. Subjects then read about the use of lasers to attack cancer, but the problem was that the light was too intense to directly fire through the tissue at the cancer. What to do?

A surprising number did not think to use multiple lasers, which is what the transfer of knowledge would suggest. This lack of obvious (to the experimenter) transfer effect has been repeated many times. Problem solvers have trouble transferring knowledge or strategies where there are structural but not surface similarities. Thus, users do not always transfer useful information.

The second study to keep in mind is that of perverse Emacs. Singley and Anderson (1989) trained users for several hours with a powerful text editor called Emacs. Then, on day 2, some subjects were trained on Perverse Emacs, which was just like Emacs, but the keystroke commands were different. Subjects, we can imagine, did not like this at all, but at the end of the day, their performance was indistinguishable from the people who had used Emacs in day 1 and day 2. Thus, transfer can occur, but users do not always see it, and they do not like it when transfer leads to small mistakes (as would have often happened in the first hour for the users of Perverse Emacs). Thus, changing key bindings is likely to lead to frustration, but might not hurt users in the long run.

Finally, there are some tools to measure potential transfer. Kieras and Polson (1985) created a simple language to express how to do a task, and many task analysis techniques can be used in a similar way. Each instruction in the language, such as how to push a button, takes time to learn. Thus, the number of differences between instruction sets indicate how much knowledge can be transferred and how much must be learned. Analyses of popular operating systems suggest that there are real differences between them, with one being a subset of the other (and one thus being easier to learn).

5.4.7 Implications for System Design

All users will learn and get faster at doing a task as they repeatedly do it. This is an important way that users fit themselves to a task, covering up errors in design or compensating for trade-offs made in interface design that do not favor the user.

Interfaces that are initially too slow will become faster, but will require training (to learn the task) and practice (to get faster), and may not end up fast enough. Time to perform a task can be roughly predicted by comparison with other tasks or after several initial trials to get the slope of the learning curve (this approach is used in industry to predict manufacturing times). There are theories that should provide this information in the future when the theories become more tractable (Anderson 2007; Christou et al. 2009).

Theories also note what should be provided to users at different stages of their learning (Kim et al. 2013; Ohlsson 2008). Users must have some declarative information about a task before they can perform it. This means that interfaces and their associated system, including manuals, online help, web sites, and other users, should provide the new user with enough information to perform the task or at least to start to explore the environment. More generally, you should aim to support user task performance and learning at all levels, from knowledge-based to skill-based.

How procedural and declarative memory are used will depend on how they are represented. If the user has a more general representation and declarative knowledge associated with it, through practice the user can create procedural memories (Kim et al. 2013). The procedural knowledge can transfer to new problems more easily. Interfaces and systems (including instructional materials) that support more general representations and more task-oriented representations (vs. tool specific representations) will allow users to transfer their knowledge both in and out of the interface more readily.

The learning curve that can be created by observing learners can provide insights to system design directly. The start of the curve shows how difficult the interface is for novices. The amount of time to initially perform the task might be seen to be too high or at an acceptable level. Figure 5.13a shows a relatively shallow curve where novices are not so different from experts in performance. Figure 5.13b shows a curve where novices would be less happy, but which might be appropriate for a game. The slope of the curve can also provide insights. If the curve is very steep, and the system will be used often, the initial task times might not be an issue. If the interface is to be used only a few times, then the slope needs to get the user's task time to an acceptable level within that time frame. It can be the case that 150 s is an acceptable time, or it can be that 50 s is not and even the top curve is not fast enough—it will depend on the context.

Some users like to learn. This is probably most often seen in games, which have their explicit goal to teach the user some procedural skill (like driving a racing car), or some declarative knowledge (like the way round a dungeon). Most other users prefer not to learn, but will if the task is important enough.

If the users are likely to encounter situations where they may not have enough expertise to deal with the situation easily (e.g., plane malfunctions), you should consider providing them with instructions at that point (e.g., how to land when out of fuel). These instructions will not always help, but they often can. In computer systems these problems are often now being found on the Internet (e.g., how to repair a bad disk, how to clear a stuck keyboard).

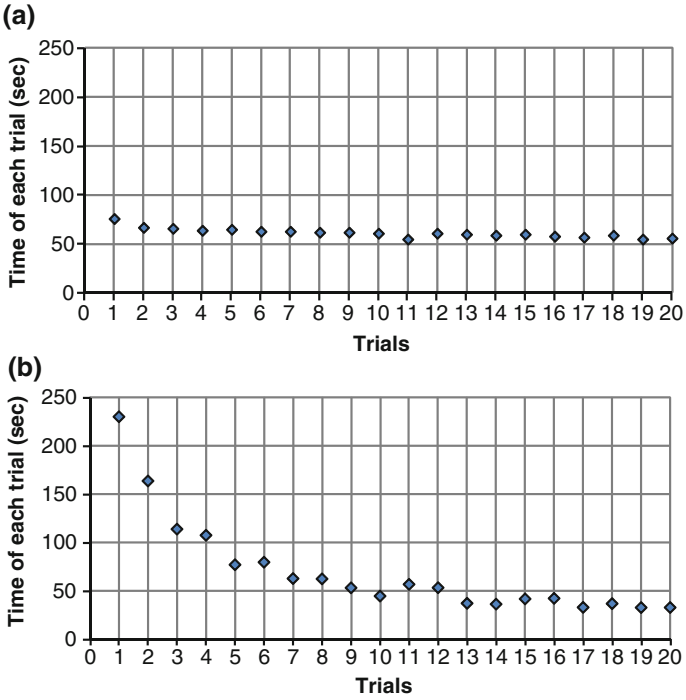

Fig. 5.13 Two learning curves with approximately the same final speed. **a** A relatively shallow learning curve. **b** A relatively steep learning curve

Systems that allow users to recognize actions they want to do will be easier initially to use than those that require users to recall commands. There is a trade-off, however, when the novices become experts. The experts will be able to recall the keystrokes or command names and will not wish to wade through the choices. For example, Linux experts like to use keystrokes in the command shell, while novices on Windows prefer a graphical user interface.

Systems that encourage users to reflect on how to perform the task may lead to different types of learning. Where the learning is incidental and not tied to educational intent or systems, this may not matter. On the other hand, tutoring systems should be careful that what users learn is what is intended. For example, the interface should not encourage users to just try key presses or button clicks to see what happens, which can lead to implicit learning that is tied to the representation of the interface and not to the domain.

Learning in such a situation has been seen in solving the 8-puzzle. Figure 5.2 shows such a puzzle. When users could directly click on the interface to indicate the tile to move, they took a larger number of moves and could verbalize their knowledge less well than users that had to work harder to interact with the interface, such as clicking on the tile and the direction to move, or using a speech interface.

The experimenters hypothesized that the increased cost of typing in the direction to move the tile lead users to think more and learn at a higher level. When users could just click, they created simple rules and learned implicitly (Golightly et al. 1999).

5.5 Summary

Memory, attention, and learning make up some of the core aspects of cognition of users. These are well studied and well understood areas in psychology, and their results are increasingly able to be packaged and applied to interface design.

There remain interesting areas for basic and applied research in these areas as technology creates new opportunities. These areas include multi-tasking, prospective memory aids, how to help users learn from systems, how to avoid biases in memory, how to help users manage things they need to remember (like passwords), and how to direct the user's attention appropriately across both short time spans (e.g., an hour) and long time spans (e.g., years of learning and working with a system like Unix). Knowledge of the details of these aspects of human behavior allow better interfaces to be built.

5.6 Other Resources

If you would like to learn more about the areas covered in this chapter, it is well worth reading the book by Alan Baddeley from 2004 on book on human memory. Baddeley is one of the pioneers of human memory research as we understand it today. This text renders his portfolio of scientific research accessible to a general audience, offering insights that will change the way you think about your own memory and attention:

Baddeley, A. (2004). *Your memory*: *A user's guide*. Buffalo, NY: Firefly Books.

You may well have heard of the magic number 7 about human memory before. If you'd like to read a paper that shows how this number was demonstrated to apply to human memory, it is worth seeing if you can get a copy of Miller's classic paper, published in a key psychology journal, *Psychological Review*. We note, however, that while this finding is still cited in the popular press as a well-known feature of human memory, subsequent studies have added more nuance to the observation–familiarity with the domain, modality (visual, auditory), and memory enhancement tactics can contribute to how many items are remembered:

Miller, G. A. (1956). The magic number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*, 81–97.

John Anderson's texts on cognitive psychology provide an excellent introduction also. These are more directed at beginning scholars wishing to dive deeper into this area:

Anderson, J. R. (1999). *Learning and memory* (2nd ed.). New York, NY: John Wiley and Sons.

Anderson, J. R. (2009). *Cognitive psychology and its implications* (7th ed.). New York, NY: Worth Publishers.

An excellent introduction to cognitive modeling is Allen Newell's Unified Theories of Cognition. Although written a while ago, the text offers a nice introduction to the thinking that launched this field of research, and to a particular cognitive modeling environment that we mention in passing, Soar:

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Elizabeth Styles' text on attention, perception, and memory offers a very good starting point and highlights how they are interdependent and interact:

Styles, E. H. (2005). *Attention, perception and memory: An integrated introduction*. Hove, UK: Psychology Press.

For an accessible book on how reflection and memory affect performance in a business setting, read Tom Demarco's *Slack*:

Demarco, T. (2001). *Slack: Getting past burnout, busywork, and the myth of total efficiency*. New York, NY: Broadway Books.

There are also many online demonstrations and resources in this area on sites such as The Exploratorium (http://www.exploratorium.edu/memory) and the EPsych site (http://epsych.msstate.edu).

5.7 Exercises

5.1 Find a user of a smartphone. It may have to be someone from outside your class. Ask them to note as many of the menu items on the smartphone as they can, and the structure of these menus. Have them do it without access to their phone.

Compare what they remember with the phone itself. It will be useful to do this for several people and compare the results. What does this tell you about phone usage and users' memory?

5.2 Find an online tutorial, perhaps from this class or another class, or an online tutorial to teach you Lisp or Java. Using an informal representation, break down what you learn from a sample lesson into these concepts: math, theory

of programming, programming constructs, language syntax, programming interface, online tutorial interface.

What are the results, and what do they suggest for the design of online tutorials?

5.3 With respect to a web site for a university department (or your company's), create a list of tasks and information that a user of a university department's web site would be able to retrieve from memory and a similar list that users might not initially be able to recall, but would recognize. An example of the first could include the main phone number, and an example of the second could include special resources or accomplishments of the department.

5.4 Recall of memories is more difficult than recognition. As an example of this, write down the 50 US states. You can adapt this task to your own country, for example, listing the UK counties, the Bundesländer of Germany, or the number of member states in the European Union.

This effect also works for computer interfaces. You could attempt to name the menu bar items for a menu driven interface that you use often.

5.5 Usability of security remains an ongoing active research area. Memory influences the use of computer security. There are several ways that this can happen. One way is in the choice and memorizing strategies for passwords. Read https://www.cs.cmu.edu/~help/security/choosing_passwords.html or find a similar study on how people choose passwords. If you have time, write down your previous passwords, or ask people you know to fill out a small survey on what type of password they have, or have had. What do the results say about human memory and about computer interface design for passwords?

5.6 The first step of this exercise is to prepare a task for users to perform. This task should take people about 2 min on their first attempt, and they should be successful nearly all of the time. Students have used such tasks as card sorting, shoe tying, word processing, paper airplane construction, typing a paragraph, and web navigation. You should prepare instructions, as you will be having someone else perform the task. If the task is longer than 2 min it gets hard to run the necessary number of trials. If it is less than 2 min it may get hard to time as the users get faster. Anywhere from 70 s to 210 s is likely to work easily, and greater or lesser amounts will work to a certain extent. Online tasks may be more appropriate for your course, but what's interesting is that it does not matter what task you choose or whether it is online or not.

To get a good look at the learning curve, you should run 2–5 users on your task, and the subjects should each perform the task at least 15 times. Because of the logarithmic scale, 100 times is twice as good as 10, a 1000 times is three times as good. You may choose to run 5 subjects on the task 15 times (at most, 5 subjects $\times$ 2 min $\times$ 15 times $=$ 150 min, so you may want to have help), or you might choose to run 2 users over a longer series of trials.

You should record the time taken to perform the task each time, and note whether the user made any minor errors, significant errors, or catastrophic

errors. Some of these suggest that something besides normal learning was going on, or that a trial was in some way unusual.

When you graph your results, you should see a power curve or exponential curve. Both linear and log–log plots are interesting ways to plot this data.

You should comment on how your results fit the predictions in this book, other resources, and what your results mean for your task and for related tasks.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*, 369–406.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1995). *Learning and memory*. New York, NY: Wiley.

Anderson, J. R. (2004). *Cognitive psychology and its implications* (5th ed.). New York, NY: Worth Publishers.

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?*. New York, NY: Oxford University Press.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13*(4), 467–505.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In K. W. Spence & J. T. Spence (Eds.), *The psychology of learning and motivation* (Vol. 2). New York, NY: Academic Press.

Baddeley, A. D. (1976). *The psychology of memory*. New York: Basic Books.

Baddeley, A. D. (1986). *Working memory*. Oxford, UK: Oxford University Press.

Bailey, B. P., & Konstan, J. A. (2001). The effects of interruptions on task performance: Annoyance, and anxiety in the user interface. In *Interact, 2001* (pp. 593–601).

Bainbridge, L. (1983). Ironies of automation. *Automatica, 19*(6), 770–775.

Baxter, G., Rooksby, J., Whang, Y., & Kahajeh-Hosseine, A. (2012). The ironies of automation… still going strong at 30? In *Proceedings of ECCE 2012 Conference* (pp. 65–71). Edinburgh, North Britain.

Card, S. K., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Charness, N., Holley, P., Feddon, J., & Jastrzembski, T. (2004). Light pen use and practice minimize age and hand performance differences in pointing tasks. *Human Factors, 46*(3), 373–384.

Chilton, E. (1996). What was the subject of Titchner's doctoral thesis? *SigCHI Bulletin, 28*(2), 96.

Christou, G., Ritter, F. E., & Jacob, R. J. K. (2009). Knowledge-based usability evaluation for reality-based interaction. In G. Christou, E. L.-C. Law, W. Green & K. Hornbaek (Eds.), *Challenges in the evaluation of usability and user experience in reality-based interaction (workshop proceedings). At CHI 2009 Conference on Human Factors in Computing Systems, Boston, MA, 2009. CHI 2009 Workshop: Challenges in Evaluating Usability and User Experience in Reality Based Interaction* (pp. 36–39). Toulouse, France: IRIT Press.

Corbett, A. T., & Anderson, J. R. (1990). The effect of feedback control on learning to program with the LISP tutor. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society* (pp. 796–803). Erlbaum: Hillsdale, NJ.

Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior, 19*, 450–466.

Delaney, P. F., Reder, L. M., Staszewski, J. J., & Ritter, F. E. (1998). The strategy specific nature of improvement: The power law applies by strategy within task. *Psychological Science, 9*(1), 1–8.

Engle, R. W. (2002). Working memory capacity as executive attention. *Current Directions in Psychological Science, 11*(1), 19–23.

Ericsson, K. A. (Ed.). (1996). *The road to excellence: The acquisition of expert performance in the arts and sciences*. Mahwah, NJ: Erlbaum.

Ericsson, K. A., Chase, W. G., & Faloon, S. (1980). Acquisition of a memory skill. *Science, 208*, 1181–1182.

Ericsson, K. A., & Kintsch, W. (1995). Long-term working memory. *Psychological Review, 102*, 211–245.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data* (2nd ed.). Cambridge, MA: MIT Press.

Fitts, P. M. (1964). Perceptual-motor skill learning. In A. W. Melton (Ed.), *Categories of human learning* (pp. 243–285). New York, NY: Academic Press.

Golightly, D., Hone, K. S., & Ritter, F. E. (1999). Speech interaction can support problem solving. In *Human-Computer Interaction—Interact '99* (pp. 149–155). IOS Press.

Hayes, J. R. (1981). *The complete problem solver*. Philadelphia: The Franklin Institute Press.

Heathcote, A., Brown, S., & Mewhort, D. J. K. (2000). The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review, 7*(2), 185–207.

Intons-Peterson, M. J., & Best, D. L. (1998). Introduction and brief history of memory distortions and their prevention. In M. J. Intons-Peterson & D. L. Best (Eds.), *Memory distortions and their prevention*. Mahwah, NJ: Erlbaum.

Jensen, M. B., & Healy, A. F. (1998). Retention of procedural and declarative information from the Colorado Driver's Manual. In M. J. Intons-Peterson & D. L. Best (Eds.), *Memory distortions and their prevention* (pp. 113–124). Mahwah, NJ: Erlbaum.

Kaplan, C. (1989). *Hatching a Theory of Incubation: Does putting a problem aside really help? If so, why?*, Carnegie-Mellon University, University Microfilms International, Catalog #9238813.

Kieras, D. E., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22*, 365–394.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230–275.

Kim, J. W., Ritter, F. E., & Koubek, R. J. (2013). An integrated theory for improved skill acquisition and retention in the three stages of learning. *Theoretical Issues in Ergonomics Science, 14*(1), 22–37.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 311–334). Hillsdale, NJ: Erlbaum.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980a). Expert and novice performance in solving physics problems. *Science, 208*, 1335–1342.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980b). Models of competence in solving physics problems. *Cognitive Science, 4*, 317–345.

Lindsay, P. H., & Norman, D. A. (1977). *Human information processing* (2nd). San Diego, CA: Harcourt Brace Jovanovich.

Loftus, E. F. (1975). Leading questions and the eyewitness report. *Cognitive Psychology, 7*, 560–572.

Loftus, E. F. (2003). Our changeable memories: Legal and practical implications. *Nature Reviews Neuroscience, 4*, 231–234.

Lovett, M. C., Daily, L. Z., & Reder, L. M. (2000). A source activation theory of working memory: Cross-task prediction of performance in ACT-R. *Journal of Cognitive Systems Research, 1*, 99–118.

McFarlane, D. C. (1997). *Interruption of people in human-computer interaction: A general unifying definition of human interruption and taxonomy* (Tech Report No. NRL/FR/5510–97-9870): Naval Research Laboratory. (also at infoweb.nrl.navy.mil/htbin/webcat).

McFarlane, D. C. (1999). Coordinating the interruption of people in human-computer interaction. In *INTERACT '99* (pp. 295–303). IOS Press.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104*(1), 3–65.

Miller, G. A. (1956). The magic number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*, 81–97.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Nickerson, R. (1969). *Man-Computer interaction: A challenge for human factors research. Ergonomics, 12* (pp. 501–517). (Reprinted in 1969 in *IEEE Transactions on Man-Machine Systems, 10*(4), 164–180).

Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88*, 1–15.

O'Hara, K., & Payne, S. J. (1998). Effects of operator implementation cost on performance of problem solving and learning. *Cognitive Psychology, 35*, 34–70.

Ohlsson, S. (1992). The learning curve for writing books: Evidence from Professor Asimov. *Psychological Science, 3*(6), 380–382.

Ohlsson, S. (2008). Computational models of skill acquisition. In R. Sun (Ed.), *The Cambridge handbook of computational psychology* (pp. 359–395). Cambridge: Cambridge University Press.

Rabbitt, P., & Banerji, N. (1989). How does very prolonged practice improve decision speed? *Journal of Experimental Psychology: General, 118*, 338–345.

Raskin, J. (2000). *The humane interface*. Boston, MA: Addison-Wesley.

Rasmussen, J. (1983). Skills, rules, knowledge: Signals, signs and symbols and other distinctions in human performance models. *IEEE Transactions: Systems, Man, & Cybernetics, SMC-13*, 257–267.

Reason, J. (1990). *Human error*. Cambridge, UK: Cambridge University Press.

Ritter, F. E., & Feurzeig, W. (1988). Teaching real-time tactical thinking. In J. Psotka, L. D. Massey, & S. A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned* (pp. 285–301). Hillsdale, NJ: Erlbaum.

Ritter, F. E., & Schooler, L. J. (2001). The learning curve. In W. Kintch, N. Smelser, & P. Baltes (Eds.), *International encyclopedia of the social and behavioral sciences* (Vol. 13, pp. 8602–8605). Amsterdam: Pergamon.

Seibel, R. (1963). Discrimination reaction time for a 1,023-alternative task. *Journal of Experimental Psychology*, *66*(3), 215–226.

Siegler, R. S. (1988). Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General, 117*(3), 258–275.

Simon, H. A. (1974). How big is a chunk? *Science, 183*, 482–488.

Simon, H. A., & Chase, W. G. (1973). Skill in chess. *American Scientist, 61*(393–403), FIND.

Simonton, D. K. (1996). Creative expertise: A life-span developmental perspective. In K. A. Ericsson (Ed.), *The road to excellence: The acquisition of expert performance in the arts and sciences* (pp. 227–253). Mahwah, NJ: Erlbaum.

Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.

Snoddy, G. S. (1926). Learning and stability. *Journal of Applied Psychology, 10*, 1–36.

Sperling, G. A. (1961). The information available in brief visual presentations. *Psychological Monographs, 74*(whole No. 498).

Suchman, L. (1983). Office procedures as practical action: Models of work and system design. *ACM Transactions on Office Information Systems, 1*, 320–328.

Thomas, E. L., & Robinson, H. A. (1972). *Improving reading in every class: A sourcebook for teachers*. Boston, MA: Allyn & Bacon.

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology, 53*, 1–25.

Wickens, C. D., & Hollands, J. G. (2000). *Engineering psychology and human performance* (3rd ed.). Prentice-Hall: Upper Saddle River, NJ.

Woods, D. D., & Patterson, E. E. (2001). How unexpected events produce an escalation of cognitive and coordinative demands. In P. A. Hancock & P. A. Desmond (Eds.), *Stress, workload, and fatigue* (pp. 290–302). Mahwah, NJ: Erlbaum.

Zhang, G., & Simon, H. A. (1985). STM capacity for Chinese words and idioms: Chunking and acoustical loop hypotheses. *Memory and Cognition, 13*(3), 193–201.