Chapter 6
Cognition: Mental Representations, Problem Solving, and Decision Making

Abstract There are several higher level structures built upon the basic structures of memory, attention, and learning in the user's cognitive architecture. These representations and behaviors include mental models, problem solving, and decision making. These structures and processes form the basics of higher level cognition when interacting with technology, and describe some of the ways that users represent systems and interfaces, and how users interact with and use systems. Mental models are used to understand systems and to interact with systems. When the user's mental models are inaccurate, systems are hard to use. Problem solving is used when it is not clear what to do next. Problem solving uses mental models, forms a basis for learning, and can be supported in a variety of ways. Decision making is a more punctuated form of problem solving, made about and with systems. It is not always as clear or accurate as one would like (or expect), and there are ways to support and improve it. There are some surprises in each of these areas where folk psychology concepts and theories are inaccurate.

6.1 Introduction

When people interact with artifacts, even for the first time, they usually have some idea of what to do. If it is a piece of electrical equipment, they may first look for the on/off switch, for example, and then press that. The way that people interact with artifacts is governed by their mental representation of that artifact. These mental models develop over time: as people use the artifact more and more, they learn more about how it works and refine their mental model accordingly. The mental models can be either structural (based on the structure of the artifact in terms of its components) or functional (based on the behavior of the artifact).

As the complexity of artifacts increases, it becomes harder to develop a complete, accurate structural mental model of them. If you are someone who drives a car but takes it to someone else for maintenance, for example, the chances are that you have a relatively simple functional mental model of how the car

F. E. Ritter et al., *Foundations for Designing User-Centered Systems*,
DOI: 10.1007/978-1-4471-5134-0_6,

operates—when you turn the key, you expect the engine to start, and so on. This model will allow you to carry out limited troubleshooting if the car does not behave as you expect it to. If you are a car mechanic, however, you will have a much more developed structural model of how the car works, which will allow you to diagnose and remedy any problems.

Designers have a functional mental model of how users behave. These models may be naïve and contain several inaccuracies because they have created and refined these models based on their own experiences of how they themselves behave, rather than based on their experiences of how their users really behave. One of the main purposes of this book is to help designers to develop a more sophisticated and more accurate model of users. You should, at this point in the book, know that, while not all users are the same, they are similar in many ways. Users differ in terms of the knowledge they possess, their capabilities, and their attributes. They also share common structures in how they receive and process information, and interact with the world. Understanding how these differences and similarities influence their behaviors will help you to design systems that are better suited to a wider range of users.

You should have learned by this point in the book that users behave unusually when they find themselves in novel or rarely encountered situations. This is because their mental models are not fully developed for those situations. In these situations, users often have to go back to basics, and use their mental model to carry out some problem solving. The way that they problem solve has associated biases, weaknesses, and strengths that you may be able to deal with in your design.

As part of the problem solving process, users will often make decisions about what actions to take next. Decision making, like problem solving, has associated biases, weaknesses, and strengths that can be used to inform system design.

Problem solving and decision making are probably the most widely observable and, arguably, the most important behaviors of users. Many fields of research study these topics, including psychology, economics, and statistics. Here we focus on how people make judgments and decisions, and how to support these processes in system design.

6.2 Mental Representations

Users can make use of several types of representations when they are problem solving or using an interface to perform a task. For example, users can understand an electronic circuit as a series of propositions (the secondary accumulator is connected to the laser bank), as a diagram (shown in Fig. 6.1), or implicitly as a series of steps to make the circuit work, such as “First flip switch 1, and if that does not work…” (Bibby and Payne 1996). Each of the representations has different strengths and weaknesses. The propositions may be easier to learn, the schematic allows some types of problem solving, and the rules require less knowledge to be learned or stored for later use.

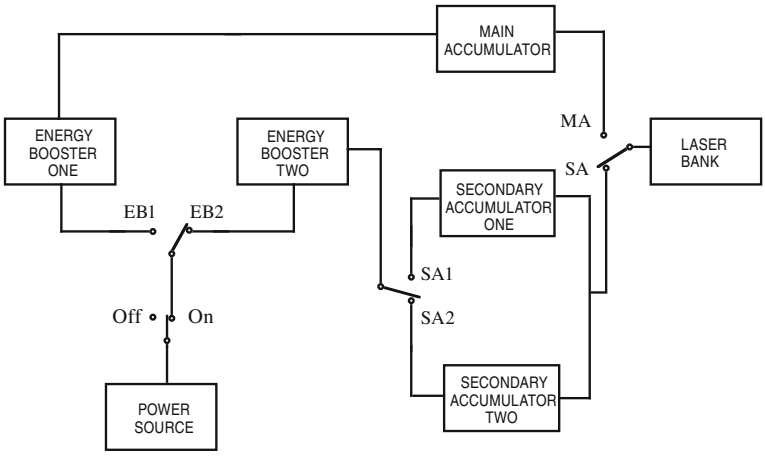

Fig. 6.1 A type of circuit schematic (based on Ritter and Bibby 2008)

6.2.1 Simple Representations

Representations can be classified in many ways. Two types of simple representations that occur repeatedly include semantic/propositional representations and visual/spatial representations.

Consider the information in Table 6.1, and attempt to memorize it. Then consider the information in Fig. 6.2. The information represented is the same in both places, but the representations give rise to different behavior and support different types of tasks.

The information about the sets of letters in Table 6.1 and Fig. 6.2 is probably better represented as a series rather than as a set of propositions—it is more succinct, easier to learn, and for most tasks it is easier to manipulate. The canonical series ABCD is particularly easy to memorize: it probably already exists as a series in your head. The other series is more difficult, but is probably easier to remember and use as a series than as a set of propositions.

The best representation for the workshop expenses will vary. For most tasks, users will be better off with the numbers themselves. The graphical representation on the left in Fig. 6.2 (created using the default settings of a common spreadsheet program) is less clear than the bespoke graph on the right in the same figure. The relative values can be quickly judged, but the absolute numbers will be more difficult to obtain. Both graphs are less useful than a table of numbers for many tasks.

Sometimes users will use propositional representations. Sometimes they will use a more visual representation. Interfaces should support the representation that users have, as well as be mindful of supporting the representation that will be most helpful for a task. It is the job of the designer to find the representations that best support the task, and those that users already have, and to balance the design to support these representations.

Table 6.1 Some propositional representations

Letters example—canonical
A is to the left of C
C is to the left of D
B is to the immediate right of A
Letters example—non-canonical
M is to the left of Y
Y is to the left of S
R is to the immediate right of M
Workshop expenses
Gross income was $3,500. Expenses for coffee breaks were $1,218; copying was $400, and AV rental was $1,200. Net profit was $882
Puzzle sequence
A series of numbers is 8 5 4 9 1 3 2 7 6
What comes next?

Letters example—canonical: A B C D
Letters example—non-canonical: M R Y S

Workshop expenses:

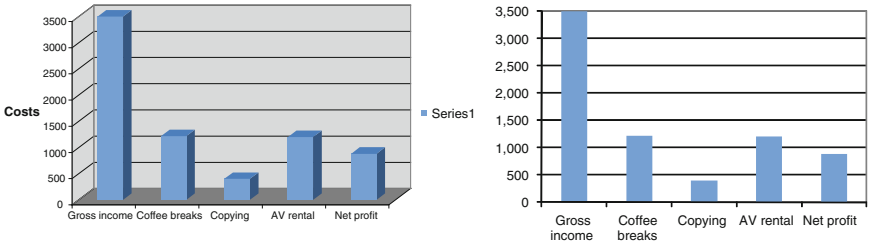

Fig. 6.2 Some graphical representations of the material in Table 6.1

6.2.2 User's Mental Models

The mental models that people use in performing tasks affect how they perceive and interact with the world. There are several different meanings of the term *mental model*, however, and the different ways in which it is used can be confusing (e.g., Gentner and Stevens 1983; Johnson-Laird 1983; Moray 1996, 1999; Norman 1983; Wilson and Rutherford 1989). For our purposes we think of mental models as a representation of some part of the world that can include the structures of the world (the ontology of the relevant objects), how they interact, and how the user can interact with them.

As an example, work out how many windows there are in the place where you live. The way that most people arrive at an answer is by referring to a structural mental model of their home. They then imagine walking around their model of the

home and looking around the rooms at the walls to see where the windows are located.

If you had to come up with a mental model of a printer, though, it may be more of a functional model. So it would include the notion that you feed paper into it, that the paper gets images and text written onto it, and that the paper comes out of the printer into some sort of hopper. The mental model might not specify what the something is that produces the images and text, but could include further knowledge about the paper path. Figure 6.1 is another example of possible mental model, this time of a Klingon laser bank.

If the user's mental model accurately matches the device, the user can better use the mental model to perform their tasks, to troubleshoot the device, and to teach others about the task and device. If their mental model is inaccurate, however, the user will make poorer choices about what actions to take, may not be as happy, and may be less productive.

The complexity of the mental model will vary by user, by system, and by context. If the structure of the system is relatively simple, we can think of the model as an isometric model, in which every feature of the artifact is represented by a corresponding feature in the model. As the structure increases, however, people try to manage the complexity by grouping together related features from the real artifact and representing them using a single point in the model. This is what is called a homomorphic model. For more complex systems, and systems of systems (e.g., the Internet, the Space Shuttle, power grids, and so on), these mental models are also more likely to be functional, based on the way that the system behaves in particular circumstances, rather than structural.

Understanding users' mental models is important for system designers (Carroll and Olson 1987; Revell and Stanton 2012). The model the user brings to the task will influence how they use the system, what strategies they will most likely employ, and what errors they are likely to make. It is, therefore, important to design the system in such a way that the user can develop an accurate mental model of it.

While this is not a computer example, the design of Philadelphia airport's Concourse A violates most people's mental model and illustrates the role of mental models in using a system. A diagram of it is shown in Fig. 6.3. Users coming in on the left expect to see Gate A2 between A1 and A3. They might know that gates are sometimes numbered odd/even by side of concourse, but, in this case, the nearest gate on the other side is not even visible from most areas around A1. There are numerous informal signs taped up at A1 noting where A2 is that have been created by the airline ground staff, who clearly get asked where A2 is quite often. There are several ways to fix this. The numbers could run A1, A2 and wrap from A6 on the top to A7 and A8, or A2 and A4 could be A12 and A14.

Figure 6.4 shows another example of how mental models can lead to ambiguous and even misleading interfaces. The interface designer has created a default of 'Best', but it is not clear whether this is the faster 8× speed or the more careful 1× speed. Which is best? (One of us always selects a slow speed.)

Fig. 6.3 Diagram of the Philadelphia airport concourse where the user's mental model does not support finding Gates A2 and A4

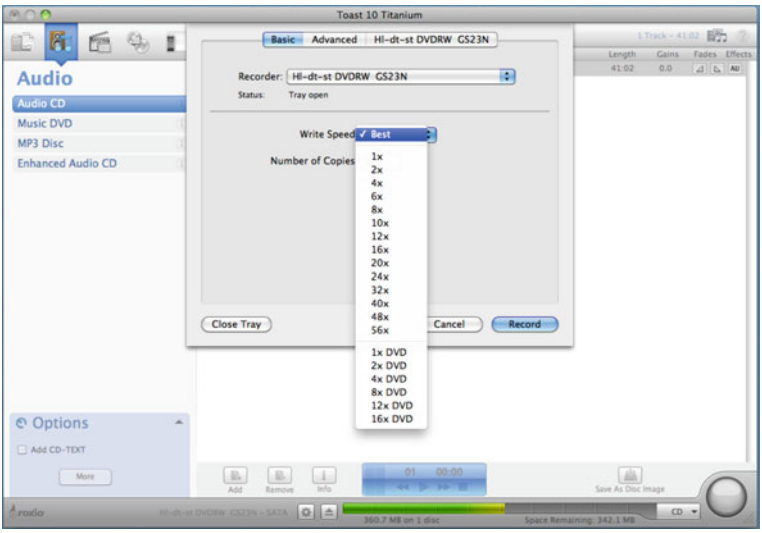

Fig. 6.4 Setting the speed for writing a CD. Which is best? Slow and careful, or faster but slightly more likely to throw an error? Taken from Toast 10 Titanium

Therefore, when you design a system you need to have an accurate mental model of how people will use it. This requires understanding how people will use it, the tasks they will perform using the system, and their normal working context. The developer's mental model of the system is often different from the user's mental model of the system (this book is intended to help designers build a better mental model of users in their own heads). Systems should describe things using conventions that are consonant with the users' mental models, for example, or be prepared to change either the users' mental models or the designer's. Making the system compliant with the user's mental model will almost certainly help reduce the time it takes to perform tasks, reduce learning time, and improve the acceptability of the system.

An important area of consideration for you and your users' mental model is processing speed and the complexity of the task. Computer programs and algorithms have a time to process an object that may be based on the number of objects (e.g., simply filing objects, or coloring them), or it may be based on their

relationship to other objects (e.g., sorting them, or looking for relationships). When the process is based on the number of objects, the algorithm is linear. When the process for each item is based on the number of other objects as well, the algorithm is nonlinear (e.g., based on the square of the number of objects which represents the number of possible comparisons). Sometimes there are algorithms that are not too bad with increasing numbers of objects, and other times the costs quickly expand beyond the capabilities of even modern computers. This area is addressed by algorithm complexity, and your users will not understand it well.

Further illustrative examples of problems in interfaces caused by a mismatch from the designer's view of the task and the user's mental model (such as doors that open the wrong way, and buttons that are hard to find) can be found in books such as Norman (1988, 2013) and on web sites of interface bloopers.

6.2.3 Feeling of Knowing and Confidence Judgments

Users will vary in how confident they are in their representations. So they will ask themselves questions about objects such as "Is this a clickable region?," about processes such as "Is this the right way to do it?," and results such as "Is this the right answer?."

The way that confidence judgments are analyzed varies across contexts and research domains. Computer science and artificial intelligence (AI), for example, have tried to provide algorithms for computing a confidence level based on the available evidence. Psychology has studied feeling of knowing, about word retrieval, strategy selection, and how close you are to the answer. Business has studied confidence in decision making. In each of these cases, the measures of confidence form another part of a mental representation.

Users will base their feeling of knowing on a wide range of information. This includes having successfully used the answer before, being familiar with the domain, social expectations from the person asking (do you know the way to….?), and social comparison (others would know the answer, so I probably should also know it). Good interfaces will help users develop appropriate levels of confidence in their representations and decisions. Often, this means providing information to support learning, including feedback on task performance, and also providing information to build a mental model. If users do not get feedback, their calibration about how well they are doing will be poor to non-existent; this applies across many professions and businesses (Dawes 1994).

6.2.4 Stimulus–Response Compatibility for Mental Models

One aspect of mental models that can be applied to interface design is the idea of stimulus–response (S–R or SR) compatibility. This aspect of behavior is that the

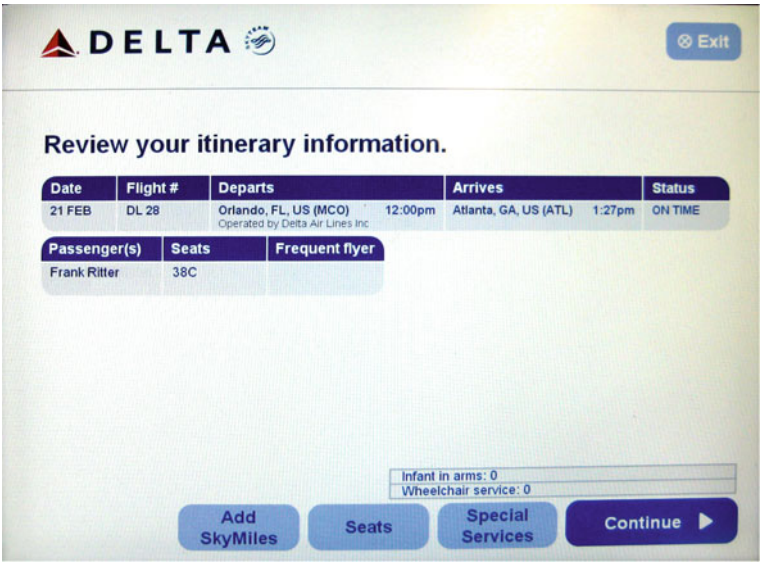

Fig. 6.5 An airport check-in kiosk. The phrase "Add SkyMiles," on the *bottom*, refers to their frequent flyer program account number which, if the interface is rarely used, can (and did) require the user to do extra work by asking an agent what SkyMiles is, and how to enter a frequent flyer number

stimulus and the response should be compatible. This is typically seen as having physical aspects of an interface (e.g., buttons) and displays (e.g., GUIs) match the world that they are representing. So the buttons to call an elevator to go up should be above the buttons that are used to call an elevator to go down. Systems where mappings like these are supported are faster to use and can lead to fewer errors.

Payne's (1995) work, reported in Chap. 1, provides several examples of how S-R can influence the usability of interfaces. The better designs (shown in Chap. 1 and explained further in the exercises) show that better S-R reduces errors by about 70% compared to a poor mapping, and that response times can be up to 30% higher with a poor mapping.

Another way to describe this aspect of behavior is making the task/action mappings appropriate. That is, the task the user is trying to perform should be easy to map to an action to perform. This is taken up in Chap. 12 as the Gulf of Execution.

The effect of S–R compatibility can also play out in computer interface menus and interfaces. If the user is searching for a widget, then an interface that explicitly labels something as a widget is a clearer interface than one that calls the widgets *thingamajigs*. Figure 6.5 shows an example where the phrase "Frequent flyer number" was replaced with "SkyMiles" (a type of frequent flyer number). Making

improvements like this to interfaces requires knowing the terms that the users will be using, which may require substantial work when the users are distant, and creating designs that can support multiple terms when different types of users search using different terms. In Fig. 6.5 changing "SkyMiles" to "SkyMiles frequent flyer number" would be appropriate.

A related example of using multiple names for the same object leading to confusion happens with air travel from Dulles Airport. Its name is Dulles International Airport, and its code is IAD. It is located in the Washington, DC area about 25 miles west of Washington, DC; however, it is actually in Sterling, Virginia, near the town of Dulles, VA. In other words, there are several ways that you could search to find the airport. Some users may use all of these terms, but some will not recognize some of them. This variety of naming conventions in the real world can make interface design complex. One way of simplifying things is to use multiple levels of naming. LON, for example, is the city code used to represent all the London airports. This helps users who are trying to fly to any London airport when searching for tickets, although they will not be able to fly into an airport with the code LON, and will have to select the specific destination airport, such as Heathrow (LHR), Gatwick (LGW), or Stansted (STN).

6.2.5 Implications for System Design

When designing systems, you need to think about the users (what types of users, the range of skill levels and attributes they may have, and so on), how they will carry out work using the system, and the context in which they will use the system. If you can talk with and observe users, this may provide you with a chance to learn how they represent themselves, their tasks, and the system.

Mental models will often provide you with guidance about how to improve the usability of the system by matching the mental models the users have about themselves, their tasks, and the system (Krug 2005). Knowing about the users and their mental models will also help identify where you need to support them to develop an accurate mental model of the system. The areas where support is needed may not be obvious to you as a designer if you already understand—have a mental model appropriate to your purposes—the system, but your users are likely to have a different mental model of the system (based on beliefs, knowledge, capabilities, and attributes). Their models will differ between users and over time, so your design needs to take this into account.

For example, photo manipulation software will have to teach the novice user about the concepts used to describe photos and manipulations to them. Cell phones may have to disambiguate cleanly what method is being used to make a call (phone or voice over IP), and what network the phone is connecting to (a phone company or a local wireless).

Fig. 6.6 A set of puzzles from a "Sunday box", designed for play (Copyright © 2006 James Dalgety, Hordern-Dalgety Collection. http://puzzlemuseum.com. Used with permission)

6.3 Problem Solving

Figure 6.6 shows a set of puzzles. Solving the problem of how to complete the puzzles can be great fun. This general notion of problem solving also applies when users try to perform tasks using systems in a particular context.

Problem solving essentially involves working out how to get from the current state of affairs to the goal that you are trying to achieve by taking appropriate actions. More formally, this can be described as applying operators to states to reach a goal. Problem solving is often studied by looking at how people solve puzzles, because the movement from one state to the next, and eventually to the goal, can be directly observed.

In the Tower of Hanoi, shown in Fig. 6.7, the goal is to move all the disks from the left hand post to the right hand post. There are several constraints. The disks can only be moved one at a time. The discs have to go on a post, and a disc cannot be placed on top of a smaller disk.

The first decision is therefore where to move the smallest disk? It can be moved to the middle or to the right post. The second decision is which disk to move, the smallest (probably not, but you could), and if you move the second smallest disk, there is only one post you can move it to. You continue in this way until all of the discs in the tower have been moved to another post. Although there are several ways to think about how to solve this puzzle, and several ways to solve it, there is only one correct minimal path.

The Tower of Hanoi has been used for a long time as a useful illustration of problem solving. Because it is a relatively simple, clear, but non-trivial problem, it has been used to study problem solving. There is a clear starting state and a clear goal state; the operations (operators) that can be applied are fairly evident and easy

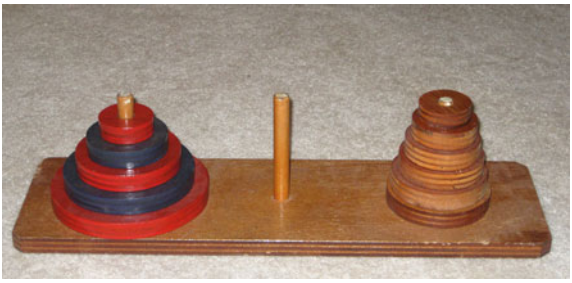

Fig. 6.7 The Tower of Hanoi puzzle. In this picture, two towers of disks are shown here, but only one tower is normally used. The goal is to move a tower, disk-by-disk, to the far peg. The tower on the *left* would be easier to work with because the disks are easier to tell apart than the disks on the *right*, and because there are fewer disks (five vs. six). An example tower to play with is on the book's web site

to perform, and the constraints are relatively clear. It is easy to see the state of the disks, and it is basically easy to judge how close you are to the goal state. It takes effort to solve the puzzle, but people can generally solve it, particularly for three or four disks.

6.3.1 The Importance of Problem Solving

Problem solving occurs when users do not know what to do next. This happens when they are learning. Novice users will learn a lot, and expert users will learn in novel and unusual situations. When an expert sees a new type of fault or a novel combination of faults they will normally have to resort to problem solving. They have to function at Rasmussen's (1983) knowledge level, using knowledge in a problem solving manner, using strategies such as trial and error, for example. In the Tower of Hanoi, the user does not know the full sequence of moves, so they will have to do some problem solving. Some users will reason about the problem in their head, and consider what moves are possible and where this will take them. Other users will start by moving disks; it is a little easier to study how these users problem solve because their behavior is more visible. In both cases, their initial behavior will be slow and effortful. As they learn to do the task, however, their performance becomes faster.

6.3.2 Examples of Problem Solving

When you are installing software, you might know the goal state, and you might even think you know the starting state, but get into trouble because you don't have

an accurate view of the actual starting state (you may have an old version installed) and you might not know the actual constraints on the installation process (e.g., you cannot install a new version correctly if the old version is still on the computer). Software can help with this, for example, noting the constraints for you.

In many hardware configurations for PCs there are hidden constraints: some disks require particular drives and graphics cards may require specific pieces of software as well. These constraints are often hidden, and sometimes can only be found through applying expensive operators, that is, configuring the hardware and then testing.

Problem solving is effortful and time consuming. In aviation they try to minimize the amount of problem solving that flight crews have to perform by providing a Quick Reference Handbook (QRH). This is essentially a set of procedures (checklists) that can be used to deal with known situations, both common and rare. In this way the pilots can operate at Rasmussen's rule-based level. In extreme situations, however, they may have to resort to problem solving. In the Sioux City air accident (NTSB 1990), for example, the aircraft lost all of its hydraulics systems whilst in the air, which basically meant that it could not be steered in the normal manner. The likelihood of such an event was so remote that there was no procedure to deal with it in the QRH, and even the aircraft manufacturer did not have a way to steer the aircraft. Fortunately there was another pilot on the aircraft who, together with the flight crew, worked out a way to steer the aircraft and get it down onto the ground.

6.3.3 Known Influences on Problem Solving

There are several known effects and influences on problem solving. These include ways that people like to problem solve and known ways that they are inefficient at problem solving. Knowing these effects can help build better interfaces and systems.

6.3.3.1 Based on Mental Models

Problem solving and decision making are thoroughly rooted in our world knowledge and world experiences. In trying to understand unfamiliar machines or unfamiliar behavior of common machines we try to construct a mental model (imagined world) in which we understand the device. This model enables us to operate the machine without having to recall what to do from memory. The correspondence between the imagined world and the real world is not important as long as the operations make sense in the imagined world, our mental model.

Mental models are thus used to perform Means-Ends Analysis (MEA) when problem solving. This approach to problem solving examines the current state of the device and notices differences between it and the solution. Systems that

support this type of analysis can help users. If a disk is full, the user will want to create a less full disk. What are the large files that can be deleted, or what are the files that are rarely used or that were temporary? Displays of this information provide the way for users to understand and apply operators to solve their problem.

Metaphors for design exploit this process by prompting an appropriate mental model of the device. However, one needs to be careful. This type of problem solving fails when the mental model is wrong. Novice writers, for example, look at papers and notice that the paper is not long enough, and simply attempt to add more pages (this is not how good papers or textbooks are written).

6.3.3.2 Avoids Apparent Backtracking

Early work on problem solving found that problem solvers do not like to move away from the goal. This is sometimes called backtracking. This was found with problems like the Missionaries and Cannibals problem and the Goat, Cabbage, and Wolf problem, but occurs in many problems, and can include, for example, installing software that has to be then uninstalled to move on. These problems are shown in Table 6.2. In these problems you have to make some moves that appear to take you away from the goal. In each of these problems this means carrying something that you have already taken across the river back to the starting side.

Problem solvers have particular difficulty finding and doing this move, presumably because it looks like going away from the goal. These changes can be differentiated from simple undoing progress from a dead end.

So, if you are building a system, you should be careful how you represent the goal and the state of the system. If an action is required that might look like it's going away from the goal, you might support the user by providing a display that shows that the move is in the right direction, or you might emphasize it in other ways or with other instructional materials. Or, if you are trying to make a game more difficult, you might require backtracking.

6.3.3.3 Functional Fixedness, Einstellung, and Insight Problems

Sometimes the ways we view objects and problems is biased by previous experience. This leads to viewing objects and how to view problems not being done as well as they could be. Functional fixedness and Einstellung are two examples of this type of effect.

Functional fixedness is where a person becomes fixated on a particular use of an object. Figure 6.8 provides an example. In this task, solving it requires not fixating on a common usage of the objects, and using them in a very reasonable but harder to think of way.

Einstellung is related to Functional Fixedness but refers to the situation where a person gets fixated on a strategy to solve a problem. The most famous example is solving a puzzle using water jugs (Luchins 1942). In this task, people were given a

Table 6.2 Puzzles that require backtracking to solve

Goat, cabbage, and wolf
You have a goat, a cabbage, and a wolf that you have to carry across a river. Your boat can only hold you and one other object. If you leave the goat with cabbage, it will eat the cabbage. If you leave the wolf with the goat, it will eat the goat. How do you ferry them across the river without anything getting eaten?
Missionaries and cannibals
There are three missionaries and three cannibals that have to cross a river. Their boat can only hold two people. If there are more cannibals on a side of a river than missionaries, the cannibals will eat the missionaries. How do they ferry themselves across the river without anyone getting eaten?

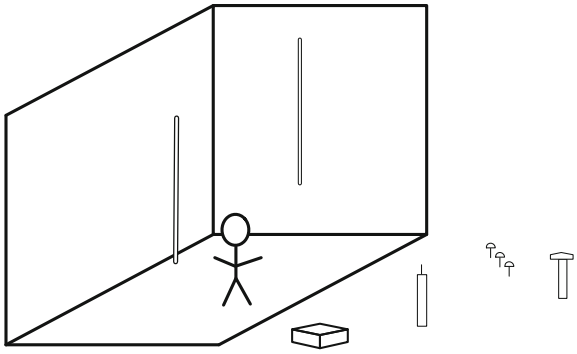

Fig. 6.8 You are in a room and have two strings to tie together (imagine you are MacGyver, a US TV personality who has to solve odd problems using odd things). You have a hammer, some tacks, a candle, and a box of matches. The two strings are too far apart for you to reach both of them when they are hanging down, but close enough to tie together if you have each in a hand. How do you tie them together?

series of problems using water jugs. In these problems, there were three jugs and a target amount of water to get. For example, one problem was to get 3 gallons using 1-, 4-, and 5-gallon buckets (fill the 4-gallon, empty into the 1-gallon, and be left with 3 gallons in the 4-gallon bucket). Over several trials they learned a strategy that worked (such as fill the first bucket, empty into the second bucket, fill the first bucket, empty into the third bucket, pour the second and third buckets together). When given a new problem that was solvable with a more efficient strategy (such as using 1-, 5-, and 7-gallon buckets, get 2 gallons by filling the 7-gallon, and emptying into the 5-gallon to get 2 gallons), they would continue to use the less efficient strategy (more pours), or if given a problem that required a different strategy, they would first try the old, unworkable strategy.

Functional fixedness and Einstellung both arise when people do not make full use of their knowledge, but are influenced too much by previous successes. As you design systems you should be mindful of what previous knowledge and mental

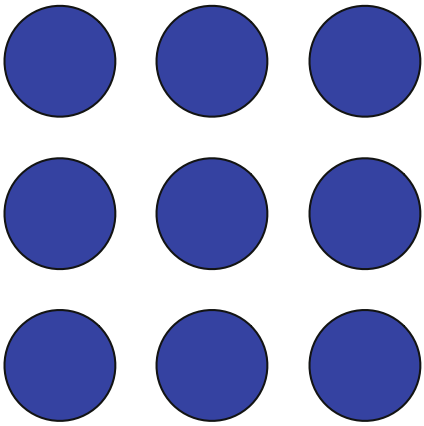

Fig. 6.9 Connect all the *dots* with four line segments without picking up your pencil

models they have encountered (functional fixedness) and what previous strategies they have used (Einstellung). Simply being aware that these effects can arise means that you can design systems that avoid them.

Insight problems are problems where novel behavior or understanding is required. Figure 6.9 shows a typical example. Most problem solvers have trouble solving this problem the first time. Solving it requires a singular insight, in this case, that you can and have to go outside the implied box the dots make up. Similarly, the problem in Fig. 6.8 is often presented as an insight problem because the way problem solving proceeds is that you are frustrated for a while until you 'see' the answer. While using Gimp to edit photos, for example, one of us has solved several insight problems.

Insight problem solving can be seen as having several stages (Sternberg and Davidson 1995):

Impasse: a point reached by an individual where they run out of ideas of operators or strategies to try that might solve the problem.

Fixation: an individual repeats the same type of solution again and again, even when they sees that it does not seem to lead to a solution; they become fixated on a solution.

Incubation: a pause or gap between attempts to solve a problem can sometimes appear to aid the finding of a solution, as if one is clearing one's mind of faulty ideas, or, as has been found, cues in the environment can help suggest solutions (Kaplan 1989).

The 'Aha' experience: the solutions to some insight problems can seem to appear from nowhere, like a Eureka moment.

Unless you are providing puzzles to your user as part of a game, you generally do not want to provide them with insight problems. Insight problems are hard to solve, are characterized by a relatively long solution time per number of steps, and

Fig. 6.10 An ATM machine that helps avoid the post-completion error because it does not retain the card. Another way is to return the card before giving money

unless you have an unusual set of users, they will not enjoy solving insight problems (although see the breakout box on "Need for cognition").

6.3.3.4 Post-completion Errors

The post-completion error (Byrne and Bovair 1997) is an interesting and important effect between memory and problem solving. This error arises when the goal for the task has been completed but the goals for the subtasks have not. For example, when you go to an old fashioned ATM machine, you will get your money before you get your card when making a withdrawal. You are there to get the money, not to get your card back. Once you have your money, you may walk away, leaving your card behind. Newer machines return your card before dispensing your money. Figure 6.10 shows another answer, a machine that does not retain the card.

Other examples of post-completion errors are available from email: when you write the email but do not send it, or when you mention an attachment but do not attach it. This error can also be seen in programming, where you allocate the memory for an array but do not release it when you have finished using it.

Post-completion errors provide a few suggestions for design. They suggest that the system should discourage the user from believing that they have completed the task until all the important subparts are done, and to put the most important goal last where technology and the situation permit.

6.3.3.5 Breakout Box: Need for Cognition

Many users do not want to do problem solving. They would prefer to perform the task directly. However, some users like to do problem solving; they like to think. These are people who do crosswords, solve Sudoku puzzles in meetings, like to build things, or debug systems. Cacioppo and Petty (1982), as well as previous researchers, describe this difference as a difference in the "need for cognition." Users with a great need for cognition agree with statements like:

> I really enjoy a task that involves coming up with new solutions to problems.
>
> I would prefer a task that is intellectual, difficult, and important to one that is somewhat important but does not require much thought.
>
> I would prefer complex to simple problems.

Thus, users with a higher need for cognition will more likely find poor interfaces a challenge rather than impossible. As you develop your system, you might consider how interested your users are in problem solving with it. While the literature on individual differences suggest that the need for cognition varies by individuals, it might also vary by task: for games, you can encourage more problem solving; for entertainment and informal use, there may be some problem solving involved; for high stake interfaces, such as health, transportation, or financial systems, most users will not want to solve problems.

The need for cognition has also been used to study how people are persuaded and form opinions. Those with a high need for cognition focus on the arguments put forward. Those with a low need for cognition focus more on peripheral cues, such as body language, social status of the speaker, and fluency of the speaker. Your users may also be the judges of your system in the same way; and you may wish to make sure your system appeals to both types of users.

6.3.4 Ill-Structured Problems

Some problems appear harder than others. An important group of problems are more difficult because they are ill-structured. That is, they are not clearly defined. Writing, for example, is ill-structured. When you are writing the goal state is not clear—it is not clear when you are done. It is also less clear when writing what all the operators are (unless you examine everything on a character by character basis—then the problem is completely intractable!). Many aspects of design can also be seen as ill-structured problems. For example, when is the new phone or new interface finished being designed? These problems are also called *ill-defined* or *messy* problems. They often occur in the real world, and a major step forward is to make them more defined.

Ill-structured problems can be ill-structured in several ways. They can have poorly defined operators or the operators can be unknown to the problem solver.

Fig. 6.11 The Chinese Ring puzzle. *Top*: classic version, provided by the Puzzle Museum (used with permission) of a classic version. *Bottom*: plastic version with the same search space, and slightly different state description and different operators (the knobs turn on a piece of plastic that slides in a track)

The Chinese Ring puzzles in Fig. 6.11 are like this. The puzzle on the top has the rings as actual rings, the puzzle on the bottom has a knob as a ring. The task is easy once you figure out what moves are possible and how close you are to the goal of removing the rings. A problem where you don't know or can't easily see the state of the system is also ill-defined. Most versions of the Chinese Ring puzzle also are difficult because you can't tell what the state is very easily; this is particularly true for the top puzzle in Fig. 6.11.

Ill-structured problems can be difficult because you do not know what the goal is or how close you are to the goal. Knowing how close you are to a solution is a problem for both of these Chinese puzzle versions. Tetris is pretty well structured; Rubik's cube is well structured, but users have trouble judging the state because it is so complex. Installing software can become very ill structured quickly if the error messages are not clear. Learning where you are in Rubik's cube or what software is not installed and is causing the error both improve problem solving.

Designers can help users with ill-structured problems in several ways. It is easier for users if the operators are clear. Thus, menus or lists of actions they can take help define the problem space. This list can also indicate what the constraints are on choosing operators. Having a clearly defined goal and good guidance towards the goal are also helpful. These distinctions can be modified or inverted for games and creative pursuits. Making problems less clear can make them more interesting and make the process of solving them a more creative experience.

Table 6.3 Problem solving requirements and several potential difficulties

Starting goal state	Can't tell if the system is at the starting state
Goal state	Can't tell directly what is a goal state
	Don't know the goal state
Intermediate states	Can't tell what state the system is in
	Can't tell distance to goal state
	Can't tell direction to goal state
Operators	Can't tell what are the operators
	Can't tell if operator had an effect
	Operators are difficult to perform (physically or mentally)
	Have to apply a lot of operators
	Can't tell which operators are safe/appropriate to use

6.3.5 Summary of Problem Solving with Implications for System Design

When you are creating systems, it is worth considering whether users will be problem solving and learning, or will they all be experts and be performing routine behavior all the time. In nearly every interface (although there are exceptions), some of the users some of the time will be problem solving. Small changes can often make it easier for them to problem solve.

Users will have to resort to problem solving with systems upon occasion. To help the users, you should support them in their problem solving (unless you are building games, when you may wish to make it more difficult deliberately) by making it obvious which operations can be performed in the current context, for example.

Making any aspect of problem solving more difficult will make an interface more difficult in several ways, as indicated by Table 6.3. You can make the user's life easier in several ways. These include making the state of the system visible, making the available operators known to the users (or enabling them to be learned easily), and having operators that can be applied physically or mentally. In some cases this may simply mean making the buttons bigger or the mouse-clickable regions easier to grab. The constraints on the application of operators should be consistent with the user's mental model, and the constraints should be visible or knowable. The results of applying the operators should be provided to the user.

6.4 Decision Making

Decision making is the result of problem solving. Where the amount of problem solving required is quite small, the problems solving and decision making may appear to blur together. Decisions can be small or quite large. They range from which button to push to which car to buy or where to land a failing airplane.

They can be quite simple and isolated, or they can occur within a string of decisions, perhaps while problem solving.

Decision making is studied by a wide range of disciplines, partly because there are many kinds of decisions, and partly because they affect all areas of human behavior. Interesting work on decisions are found in psychology (e.g., Gigerenzer et al. 1999; Kahneman et al. 1982; Evans 1990), business (e.g., Simon 1997), advertising (e.g., Ries and Trout 1986/2000), human factors (e.g., Cacciabue et al. 1992; Hoffman et al. 1998; Lehto 1997), HCI/marketing (e.g., Fogg 2003), and economics (e.g., Levitt and Dubner 2005). Here we focus on some of the generalities that are applicable to designing systems.

6.4.1 Decision Making is Often Not Rational

One of the fundamental truths in this area, and the first to note, is that often the most rational choices to an outside observer or experimenter are not chosen. People making decisions do not always take account of a lot of potentially relevant information when making decisions, and their decision making processes have general, knowable biases. That is, there are systematic ways that decisions are badly made.

Simon (1997) argues that, rather than trying to find the optimal answer, people typically satisfice. So, for example, if users were trying to find the cheapest price for something online they would choose a good enough price given the constraints of doing better which may involve looking at the price across 10 sites, rather than 500. In some sense, they are factoring in the price of further searching and the extra time needed to find a better price.

There are other problems with the way that people make decisions as compared to the way computers typically make decisions. These problems can often be related back to the cognitive processing capabilities introduced in earlier chapters. The rest of this chapter describes some of these systematic limitations, starting with simple decisions (Kahneman 2013).

6.4.2 Simple Decisions: Hicks Law and Speed–Accuracy Trade-Offs

As noted earlier, signal detection theory provides a simple way to categorize decisions in terms of their likelihood of being correct. It is also important, however, to know how long it takes to make a decision.

Pressing the corresponding button when a particular light illuminates involves a simple decision. It takes around 150–400 ms, depending on factors such as the strength of the light and the distance to the button. These numbers are useful in system design, because factors such as the time to press the brake pedal in a car

determines the safe following distance in driving, and the time to press a button determines useful speeds for agents in video games.

The time to push a particular button also depends on the complexity of the decision. Adding more lights and buttons, for example, will make the task more difficult and will thus increase the time to perform the task.

Hick's Law (or sometimes the Hick–Hyman Law) describes the time to make the decision in relation to the number of choices. When there are many choices the time to make a decision may be very long (e.g., deciding which car to purchase), but for simple light sets the relationship is as shown in Eq. (6.1), where the time to make a choice is related to the product of the number of available choices and a constant, *b*, which represents the time to make the choice and accommodates changes related to the task details. So, increasing the number of choices increases the response time, but in a logarithmic way, which means the rate of increase is slower than it would be if the relationship was linear.

$$\text{Time} = \text{b} \log_2(\text{n} + 1) \tag{6.1}$$

The equation can be extended to take account of non-equally likely choices, such as menus which include items that are rarely selected. If the choices get too large, however, or the user has to look through every item in the menu, they may have to scroll down the menu, and the relationship will become more linear with a larger constant cost per item. Where the user is a consumer, in an eCommerce situation, for example, they may not make a choice at all (Schwartz 2004).

It has been argued that the law arises from a series of choices that subdivide the list in two with each decision. Having algorithmic explanations is useful because it helps apply this theory to other situations and to situate it in larger tasks.

Hick's Law and signal detection theory (from Chap. 4) both suggest that errors and time can be traded off against each other: the so-called speed–accuracy trade-off. For any given task, the user's performance will often range between very careful and slow, to very fast but less accurate. Psychology researchers study both relatively accurate and relatively fast behavior, trying to find the bend in the curve shown in Fig. 6.12.

When creating systems you may thus want to let users know how fast they are and how many errors they are making to let them adjust their performance (and hence change their position along the curve). For the purposes of design, though, you may want to know the time required to perform the task and what the acceptable error rates are so that you can help the user to meet these criteria.

6.4.3 Stimulus–Response Compatibility for Decisions

Stimulus-response compatibility, introduced earlier in this chapter, notes that responses that match stimuli are faster and more reliable. This is also true for mapping between a task and an action, and these two concepts are closely related. The classic example, perhaps, is that elevator buttons for the higher floors in a

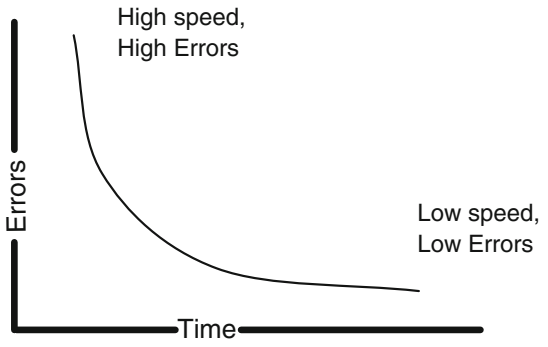

Fig. 6.12 The speed accuracy trade-off

Fig. 6.13 An elevator that has both good and bad SR features. The open button is closer to the doors (which are to the *left*, not shown), but the rooms do not map easily to the buttons, 100-level rooms are on floor 2, which does not match many of the most direct mental models, such as 100-numbered rooms being on floor 1, or floor 3 being above 2

building should be positioned above the buttons for the lower floors. Figure 6.13 shows another problem in an elevator control panel. It has good S-R compatibility for the door opening and closing buttons (the open is near to the door, the close is nearer the wall). The layout of the buttons for the floors are done in a British style that may also be familiar in Canada (ground floor is where you enter, and the first floor is the floor above that), but the rooms are numbered in such a way that labels have to be attached to the panel to explain to users where they are (e.g., 200s are on the 3rd floor). This might also be the result of a cultural mismatch between an American company and a Canadian hotel, but, given the numbering scheme, one can only wonder.

An old tale from Dick Pew (and documented in Norman's 1988 *POET* book) is that in a nuclear power plant the controls to raise and lower the rods were not clear, so the operators put beer taps on them to differentiate between them—with Loewenbrau on one (low) and Heineken on the other (high). This may be an urban myth, but it provides another clear illustration of what we mean by stimulus–response compatibility.

Probably the most powerful version of compatibility is the spatial compatibility that exists in the elevator example (and in Payne's example in Fig. 1.1). This effect can influence computer interfaces in many other places that use spatial and metaphorical content (e.g., that use relationships like up, down, left, right, higher, lower, more, less). Designs that violate this compatibility are likely to lead to a greater number of errors and take longer to use.

Two major operating systems have both included actions that violate this principle and rightly have been criticized for doing so. The Macintosh operating system originally had users drag the floppy disc icon onto the trash can icon in order to eject the disc: users just wanted their disc back, they did not want to throw it into the trash can. Also, both Windows and the Macintosh operating systems have users press the start button to shut down the operating system.

Finally, Fig. 6.14 gives an example of poor semantic mapping. OK normally means something is good. Cancel normally means stop doing something. In this example, OK means to cancel the email and cancel means do not cancel the email. A better design would be to have buttons that said 'cancel email' and 'continue with email'.

6.4.4 Known Influences on Decision Making

There are often differences between what people decide to do, and what an outside observer thinks that they rationally should decide to do. These differences are often predictable based on known biases and difficulties in reasoning. There are explanations for some of these effects showing how different assumptions by the user can lead to these effects. In other cases, these effects appear to be biases between otherwise equivalent choices, and sometimes they represent powerful tendencies that are not helpful most of the time.

Understanding people's biases—both your own and your users'—is important when designing software. It is important, albeit difficult, to counteract these biases when making decisions. Generally, these biases lead you to overestimate how good you are at something, and how much control you have over situations. You should design your system in such a way that it helps your users reason more accurately by highlighting information that they would otherwise naturally overlook or discount. The inherent biases described below are unique and have different causes, although in real-world settings several may apply at once, and even combine to make reasoning more complicated.

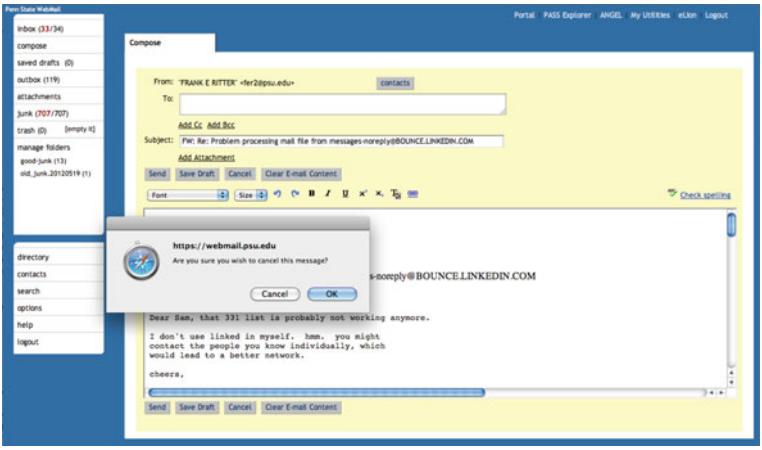

Fig. 6.14 A bad example of task-action mapping. Which button should you press to cancel the email, and which button to cancel the canceling of the email, or to OK the sending of the email?

6.4.4.1 Based on Mental Models

Problem solving and decision making are rooted in our world knowledge and world experiences. Users employ mental models in making decisions: if the mental model is wrong, the decision is also likely to be wrong. For example, if the user thinks that one cup of coffee helps thinking, they may reason that four cups might be four times better (which it patently is not). When you watch people use computer systems, you can often find examples of incorrect models such as the apocryphal tale of the novice user who looks for an "Any" key when asked to "Press any key to continue."

It is important to understand how mental models develop and evolve, and how they are used, so that you can support their development, evolution, and use. The system should provide the appropriate information needed to make a decision through displays, or help, or even in the surrounding context. If the information is difficult to retrieve from memory or to manipulate mentally, you should consider how you can either avoid or at least mitigate these effects. On solution would be to remove information that does not contribute to the decision, for example (Smallman and St. John 2005).

6.4.4.2 Confirmation Bias

Users tend to take the line of least effort. They want to find support in the world for their actions and reasoning steps that shows they were right. They therefore look for information that confirms their understanding of the situation. The corollary of

this is that they have greater difficulty seeing things that conflict with their understanding of the world. This confirmation bias is pervasive.

Confirmation bias is related to noticing the lack of objects or stimuli. We all seem to have difficulty seeing negative instances. That is, we have trouble noticing when something is not there that should be there. In software testing, for example, this can happen when a file is not written, or in project management it can happen when an email does not come from a colleague about a project update.

6.4.4.3 Regression to the Mean/Sample Sizes

Users tend to over-generalize. Given a single instance of a concept, they will assume that all such instances will have the same characteristics. This leads them to make assumptions that are unsupportable and which may turn out to be false.

An argument can be presented as to why restaurants are not as good the second time round. On the first visit to what you perceive as a good restaurant, they may have been having a very good day, or you may have been very hungry, which made the food taste better. Thus, the results were a bit uncharacteristically good. On your second visit, they may be having an average day, or you were not as hungry. These factors will lead to more average ratings.

These regression effects can be found with music on iPods by a band you are trying; they will occur when users use an application to do a task where the task types may vary, and in other situations where are you repeatedly evaluating something.

6.4.4.4 Availability Bias (Representativeness)

Users have an easier time remembering the first few items and the last few items in a list, as we noted when discussing memory. The primacy and recency effects apply across users. People will also remember some items such as those that stand out for a particular reason, such as being particularly anomalous, like an elephant in a list of candy bars.

When users are asked to reason about and make decisions, they typically retrieve and use memories that are easy to retrieve. These memories are not necessarily representative nor do they cover the distribution of memories.

It is true that users judge people and things fairly quickly. This applies in interviews and when opening a new gadget. These judgments are based on relatively little information, and will take longer to reverse if they are unfavorable or incorrect. These are the first memories.

People also retrieve memories based on their experiences rather than the facts of the world. For example, if you ask people in America which of each of the following pairs of countries is larger by population: Ireland or Indonesia, Canada or Vietnam, Bangladesh or South Korea, and Iraq or Iran, you will often get

Table 6.4 Example problems illustrating the framing problem

Assume that you are preparing a new manufacturing process that will cost $600,000 to create. Two alternatives have been created to help save money implementing this system. Assume that the exact scientific estimates of the alternatives are as follows
Program A: $200,000 will be saved
Program B: there is a 1/3 chance to save $600,000 will be saved, and a 2/3 chance that no money will be saved
Imagine that the Programs A and B are not available, but only C and D are
Program C: $400,000 will still have to be spent
Program D: There is a one in three chance that no money will have to be spent, and a two in three chance that the original estimate will have to be spent

astounding answers. The actual population sizes differ by a factor of at least two in all of these pairs.

6.4.4.5 Framing Effects

The way that outcomes are presented (framed) has a powerful influence on how users choose between alternatives. The framing effect (Tverksy and Kahneman 1981, 1986) notes that decision makers are quite sensitive to the decision's frame of reference. Consider the choices shown in Table 6.4. Which would you choose? The most common answers are given in Exercise 6.2.

Where the outcomes are noted in positive terms, lives are saved, money gained, and so on—people making decisions are risk aversive. They appear to act as if they wish to lock in their savings, choosing an outcome with certainty.

Where the outcomes are noted in negative terms, lives are lost, money lost, and so on—people making decisions are risk seeking. They make decisions based on trying to avoid the loss.

The choices in Table 6.4 have the same expected statistical values, but how you and your users will choose solutions will depend in some part on how the information is presented.

For example, in an experiment, Kahneman gave students mugs in class. When they were asked to sell them back, they wanted $7.12 for them. When they just got to look at the mugs, and could either collect a mug or get paid, they valued the mug at $3.50. This process had the same outcomes, mug or money, but the people holding the mug valued it much higher than the people who could get the mug, showing that how information is presented and the order in which it is presented can influence cognition.

This effect of order on having and valuing also plays out for users and software and software features. Losing a feature is seen by users as much more costly than not adding a feature. When a subscription to an online service is dropped it is likely to be valued higher than when adding it. This is useful to know for

marketing and for maintenance, although you are still likely to value the mug in hand more than the mug next to you.

6.4.4.6 Learning and Feedback

Dawes (1994) argues that decision making cannot improve without feedback. In his book, *House of cards*, he presents numerous examples of professionals some of whom do and some who do not get feedback about their decisions. Those that receive feedback have a chance of improving and usually do. Those that do not get feedback do not improve, although they do become more confident with practice, as they presumably get faster at it. He examines a wide range of tasks, but particularly takes psychotherapists to task, as very often in the past they provided treatment without receiving feedback on the outcomes of their treatments.

So, if you would like your users to improve their performance and to learn, you need to provide feedback. A failure to do so may only allow them to increase their confidence in their ability to do the task without increasing performance.

6.4.5 Larger Scale Decision Making Process: Expertise and RPDM

Decision making in the real world is also influenced by learning, mental models, and expertise. Work by Simon and his colleagues looked at how expertise influenced decision making. They showed that, in some cases, expertise led to faster decisions, but essentially the same decisions; for example, young business majors would make the same decisions for a business case, but would take longer (Simon 1997).

They also studied how people played chess as an analogue for other decision making tasks (Chase and Simon 1973; Gobet and Simon 1996b; Simon and Chase 1973). They found good chess players would consider more reorganization of the positions and better possible moves than would novices, who would have to do more problem solving. This expertise would also allow good players to remember positions more readily because they were not remembering all the pieces individually, but recognizing groups of pieces as patterns in a way similar to how ABC, 747, and HFE (and others noted in Table 5.1) might be considered as single objects instead of three separate characters (Gobet and Simon 1996a).

This approach has been implemented in computational models, including programs that solve problems (Ritter and Bibby 2008) and play chess (Gobet et al. 1997). These models start out with effortful problem solving or problem solving-like behavior that, with practice, becomes faster and uses recognition to drive its behavior. Sometimes the strategy does not change with improved learning (it is just faster), and sometimes the ability to recognize what to do changes how the

models perform the task (Ritter et al. 1998), mirroring how behavior changes with expertise in the task (Larkin et al. 1980).

Within human factors a body of work has developed that looks at decision making in the wild—naturalistic decision making—like fire-fighting, and search and rescue. Work on recognition-primed decision making (RPDM, Klein 1997), argues that experts do not do problem solving, but that they recognize the situation which directly leads them to the correct actions to take. This approach is consistent with previous work in psychology, and with Rasmussen's work on levels of expertise (Rasmussen 1983).

The work on naturalistic decision making encourages designers to provide information to users to help them make decisions. They should also provide the information in a way that supports recognition of the correct decisions, and explicitly acknowledges that experts may move more quickly to a solution using cues that relate the current situation to previously encountered situations.

6.4.6 Summary of Decision Making with Implications for System Design

Many decision-support systems try to support the process of formalizing the information contributing to the solution/decision. However, human problem solving is not always based on logic and rationality in terms of using the information to the full. This makes good decision support systems helpful, but difficult to design because often the problem is not with the quality of the information being presented to users.

Users do not make decisions the way you might think they will in several ways. They will take more time the more choices they have. With good design, these choices can help them and the choices can be made relatively quickly. With more choices, or worse design, the time to choose might become large. Users will be able to make more accurate and rapid choices if the choices (as described) match their mental models and the names of the tasks they are attempting to perform. So, provide time, reduce choices where appropriate, and have the choices match the mental model and terms that the user has.

The decisions users make will be based on their mental models, and they will often have different mental models than you. The cognitive architecture they can bring to solve the problem will limit their ability to make rational choices. They will often make choices consistent with previous choices because these choices confirm their previous choices. They will base their decisions on what they can retrieve from memory, and what they can retrieve first and easiest from memory is not always the most useful information for making their decisions. So, if your users often have to modify their behavior, you might wish to make repetitive behavior less easy. If they choices are likely to be consistent, then options like "Apply to all?" are helpful.

How they will make their decision will be influenced by their context and how the choices are framed. Decisions based on gains will be slightly risk-aversive; decisions to avoid a loss will lead to choosing riskier options. So be careful how you frame the options, either as a gain or a loss.

If the users do not get feedback about their choices, their choice quality will remain the same, but they will become more confident in their choices. If they do get feedback, they can learn how to make decisions more accurately and quicker using a variety of knowledge sources including recognizing a situation as being like a previous situation, and also recalling a useful answer. So, provide feedback in your systems when you can.

Finally, their choices can be situated within a large context, including geography and social relationships. Good designers will learn about these factors in a variety of ways. Good designs will take these multiple factors into account, and attempt to minimize the biases that will hurt the users, and take advantage of biases that can help users. The breakout box on biases in reasoning provides a set of examples applied to a simple task, and the breakout box on incompetence provides both serious and humorous insights into limitations of human reasoning. So when you can get context, use it to frame your interface's interactions.

6.4.6.1 Breakout Box: Biases in Reasoning

We can illustrate how biases affect reasoning using the classic Wason (1960) selection task. This is an example of an abstract deductive reasoning task that should be solved using rules of logic:

> Every card (shown in Fig. 6.15) has a letter on one side and a number on the other. Given a set of four cards as shown below, which cards would you need to turn over in order to prove or disprove the rule that "If a card has a vowel on one side, then it has an even number on the other side."

Most people opt for the A card and the 4 card. Selecting only the A and the 4 card only gathers evidence to prove the rule, however, and is an example of confirmation bias. In other words, often people only look for evidence that supports (or confirms) their hypothesis. In this case, the hypothesis is the rule that they were given. The correct solution, however, is that the A card and the 7 card should be turned over. If the A card has an even number on the other side that is evidence to support the rule; if it has an odd number it disproves the rule. The 7 card has to be turned over because if it has a vowel on the other side, this also disproves the rule. The 4 card does not have to be turned over because if there is a vowel on the other side, the rule is satisfied and if there is a consonant then the rule would not apply.

The way this and many problems are framed, however, has a major influence on people's performance. If the problem is made less abstract by using real objects that people are more likely to encounter in their daily lives, more people give the correct answer. Johnson-Laird et al. (1972), for example, framed the problem in a

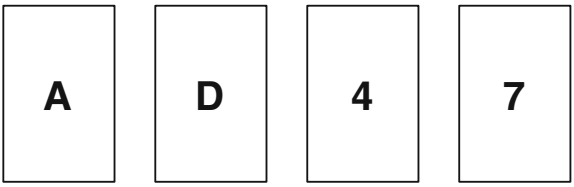

Fig. 6.15 Which cards do you need to turn over to verify that if a card has a vowel on one side then it has an even number on the other side? (Cards have a letter on one side and a number on the other)

more concrete way, using sealed/unsealed envelopes (so the flap was open/closed) which had stamps of different denominations on the other side. They then asked participants to prove or disprove the rule: "if an envelope is sealed then it has a 5p stamp on the other side." With a more concrete representation, 22 out of 24 people gave the correct answer. Making problems more concrete generally makes them easier.

For a more detailed psychological description of the issues surrounding performance on various versions of the Wason reasoning task see, for example, Quinlan and Dyson (2008).

6.4.6.2 Breakout Box: Why Don't Users Do What They Should?

At this point in your reading you may have realized that users do not always do what they should, that they do not always act in their own interests. It is one thing to make mistakes, slips, and errors (covered in the chapter on errors, and which you can infer from the memory chapter); it is another to hold quite wrong beliefs continuously contradicted by the world, and to continue to perform poorly based on these beliefs. Why does this happen?

Academics use other phrases, such as why smart people can be so foolish (Sternberg 2004) and how people cannot recognize their own incompetence (Dunning et al. 2003). Generally, this area can be examined as a type of metacognition—do you know what you know and can you judge your performance in quantitative ways (how good is it?) and qualitative ways (how is it good and how can it be improved?).

Sternberg (2004) notes five factors that can lead to foolishness—poor decision making and problem solving in leaders. These are as follows: (a) Unrealistic optimism, assuming things will work out or strategies will work. (b) Egocentrism, focusing on your own goals rather than a broader set of goals. Sternberg notes cases where focusing on short-term goals hurts long-term goals, and where not looking after others leads to significant problems arising for the leader. (c) A sense of omniscience, assuming you know all you need to know to make the decision. (d) A sense of omnipotence, assuming that you are competent in one domain and thus in all, and that the plans you put into action will work. (e) A sense of

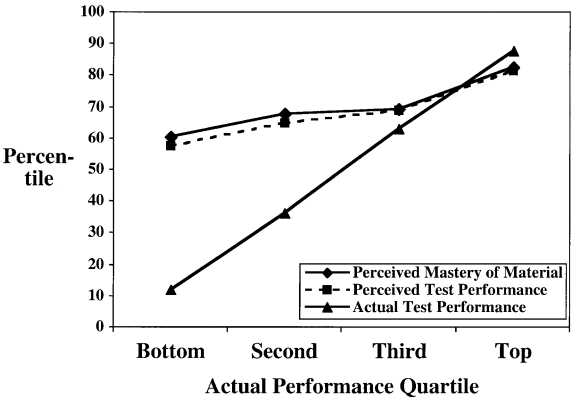

Fig. 6.16 Perceived percentile rankings for mastery of course material and test performance as a function of actual performance rank. Copied from Dunning et al. (2003) with permission

invulnerability, that plans cannot go badly wrong given the support you have. These biases in reasoning, however, are well known and occur in nearly everyone. They offer one set of reasons to explain why people are foolish at times.

When they study, students can be affected by these factors, and thus users who study manuals or are learning how to use software or systems would be equally vulnerable. Typically, students think they have learned more than they have through studying (Dunning et al. 2003; Winne and Jamieson-Noel 2002). This miscalibration can be based on judgments of the wrong measures. For example, some students measure the ease of reading the material. The real test is different, however. The real test is not how fast you can read the material, but how well can you recall it and apply it. Thus, students who stop studying when they can repeat the material or who write about the material will be better calibrated than students who make this judgment on factors not related to the test.

Figure 6.16 shows example results. Subjects who just took a sophomore psychology exam were asked how well they thought they had done. The subjects were grouped by quartiles by their exam score. The subjects who performed worst thought their scores were just above average (far left). The third quartile was relatively well calibrated in that their predictions matched their performance. The top quartile thought their performance would be a bit lower, showing modesty among the top performers.

Another problem is that the poor performers simply do not know any better. They lack the knowledge to judge that their performance is inadequate. Dunning et al. (2003) call it a double curse:

> The skills needed to produce correct responses are virtually identical to those needed to evaluate the accuracy of one's responses. The skills needed to produce logically sound arguments, for instance, are the same skills that are necessary to recognize when a logically sound argument has been made. Thus, if people lack the skills to produce correct answers, they are also cursed with an inability to know when their answers, or anyone else's, are right or wrong. They cannot recognize their responses as mistaken, or other people's responses as superior to their own. In short, incompetence means that people cannot successfully complete the task of metacognition, which, among its many meanings, refers to the ability to evaluate responses as correct or incorrect.

Thus, it is a rare and unhappy individual who knows that their answers are inadequate. For example, the knowledge to judge whether a sentence is grammatical, a word is correctly spelled, a diagram is well drawn, or program well written will be highly correlated with the ability to generate the correct behavior. So people who generate ungrammatical sentences or perform poorly on these other tasks are likely to be unable to know that they are ungrammatical.

What does this mean for users and designing for users? It means, on average, users will think that they know more than they do. They will be overconfident when performing tasks, and they will not always use the right cues to judge their knowledge.

Good design can help provide more feedback on performance, and could also provide education along the way about how to correct problems. Noting that "Error 23 occurred" does not help. Noting that "Removing disks without unmounting them can lead to disk errors" is a step in the right direction because it provides a way for users to learn from their errors through feedback.

6.5 Summary

This chapter has given you a broad understanding of how users think by providing high level descriptions of user behavior. Mental models, problem solving, and decision making all depend on the simple architectural mechanisms introduced in earlier chapters. When users interact with technology they depend on the high level cognition we have described. By understanding the capabilities and limitations of high level cognition you should be better equipped to understand some of the ways that users can represent systems and interfaces and how they interact with and use them.

Mental models are used to understand systems and to interact with systems. When the user's mental models are inaccurate, systems are hard to use. Designers can attempt to understand users' initial mental models. Designers can then either design to support users with those models, or they can help educate users to acquire more appropriate mental models.

Users problem solve when it is not clear what they should do next. Problem solving uses mental models, forms a basis for learning, and can be supported in a variety of ways. Systems that provide appropriate information to users can help with problem solving. When users are found to problem solve, it may be appropriate to modify the interface to support behavior to make it include less problem solving, to provide feedback to support problem solving such as distance to the goal, or to perform the task for the user.

Decision making is a more punctuated form of problem solving, made about and with systems. It is not always as accurate as outside observers would expect it to be. There are ways to support and improve decision making. Short-term, there are biases to avoid encouraging in the interface. Long term, providing feedback about past decisions and their accuracy can help users.

There are surprises for designers in each of these area, where folk psychology concepts and theories are inaccurate. The most general result and the one with the largest import is that, in each of these areas, users are likely to have different mental models, different problem solving strategies, and make decisions in different ways than system designers, so designers should study the users when these aspects of behavior are important. There is a rich literature on each of these areas in psychology and other areas. When you start to design interfaces that touch on other aspects of these areas, you should have enough background to go and learn more.

6.6 Other Resources

A classic text on human reasoning using mental models is Philip Johnson-Laird's book, *Mental Models*:

Johnson-Laird, P. N. (1989). *Mental models*. Cambridge, MA: The MIT Press.

You will find much material online that details some of the reasoning biases we have discussed in this chapter, but it is also worth reading some of the original texts, especially those by Kahneman and Tversky:

Kahneman, D. Slovic, P. and Tversky, A. eds. (1982). *Judgment under uncertainty: Heuristics and biases*. Cambridge, UK: Cambridge University Press.

A different approach to thinking about mental models is to think about the role of language in creating how we see the world. An interesting book to read for this approach is George Lakoff and Mark Johnson's 2008 book *Metaphors We Live By.*

Lakoff, G. and Johnson, M. (2008). *Metaphors we live by*. Chicago, IL: University of Chicago Press.

Hayes' (1981) book inspired parts of this book. It provides a good source for thinking about applied cognitive psychology and problem solving:

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

Another classic text is the 1972 book by Allen Newell and Herb Simon, and is worth reading for historical context as well as for their ideas about human problem solving specifically:

Newell, A. and Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

The journal *Psychological Science in the Public Interest* publishes papers about how results in psychology have public policy implications. The series is worth reading because it examines important topics. It is available online, free, as PDF files. For example, a review paper by Dunning et al. (2004) summarizes why users are often poorly calibrated in their decision making and provides more examples

and results. They have also published reviews on how to promote trust, how to improve learning in schools, and terrorism.

6.7 Exercises

6.1 Using a smartphone, explore it to find an ill-structured problem. This could be an application or function that has the features of an ill-structured problem. Discuss how you could improve this application, or note why the user would want it to be an ill-structured problem.

6.2 Find another "bias" in human problem solving or in decision making with particular attention paid to business problems. You might find implications in an earlier chapter, or you might find useful information in a book or journal on decision making in business. Compare this known bias in reasoning to the list shown here. Consider which are the more important biases.

6.3 Consider a web site that you know of medium complexity, such as your department's web site. Draw a map of it from memory (or, to be fairer, have some other people draw a map of it). Then, compare the maps with a map drawn from the web site. Compare the user's maps with the actual site, and draw conclusions about mental models.

6.4 Have some friends choose between the choices in Table 6.4. If your friends are like Tversky and Kahneman's subjects (Kahneman et al. 1982), they will prefer program A over program B (72–28%) and program C over program D (22–78%).

References

Bibby, P. A., & Payne, S. J. (1996). Instruction and practice in learning to use a device. *Cognitive Science, 20*(4), 539–578.

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science, 21*(1), 31–61.

Cacciabue, P. C., Decortis, F., Drozdowicz, B., Masson, M., & Nordvik, J. (1992). COSIMO: A cognitive simulation model of human decision making and behavior in accident management of complex plants. *IEEE Transactions on Systems, Man, and Cybernetics, 22*(5), 1058–1074.

Cacioppo, J. T., & Petty, R. E. (1982). The need for cognition. *Journal of Personality and Social Psychology, 42*(1), 116–131.

Carroll, J. M., & Olson, J. (Eds.). (1987). *Mental models in human-computer interaction: Research issues about what the user of software knows*. Washington, DC: National Academy Press.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology, 4*, 55–81.

Dawes, R. M. (1994). *House of cards: Psychology and psychotherapy built on myth*. New York, NY: The Free Press.

Dunning, D., Heath, C., & Suls, J. M. (2004). Flawed self-assessment: Implications for health, education, and the workplace. *Psychological Science in the Public Interest, 5*(3), 69–106.

Dunning, D., Johnson, K., Ehrlinger, J., & Kruger, J. (2003). Why people fail to recognize their own incompetence. *Current Directions in Psychological Science, 12*(3), 83–87.

Evans, J. St. B. T. (1990). *Biases in human reasoning*. Hove, UK: Lawrence Erlbaum.
Fogg, B. J. (2003). *Persuasive technology: Using computers to change what we think and do*. San Francisco, CA: Morgan Kaufmann.
Gentner, D., & Stevens, A. L. (Eds.). (1983). *Mental models*. Hillsdale, NJ: Erlbaum.
Gigerenzer, G., Todd, P. M., & the ABC Research Group (1999). *Simple heuristics that make us smart*. New York, NY: Oxford University Press.
Gobet, F., Richman, H., Staszewski, J., & Simon, H. A. (1997). Goals, representations, and strategies in a concept attainment task: The EPAM model. *The Psychology of Learning and Motivation, 37*, 265–290.
Gobet, F., & Simon, H. A. (1996a). Templates in chess memory: A mechanism for recalling several boards. *Cognitive Psychology, 31*, 1–40.
Gobet, F., & Simon, H. A. (1996b). The roles of recognition processes and look-ahead search in time-constrained expert problem solving: Evidence from grandmaster level chess. *Psychological Science, 7*, 52–55.
Hayes, J. R. (1981). *The complete problem solver*. Philadelphia: The Franklin Institute Press.
Hoffman, R. R., Crandall, B., & Shadbolt, N. R. (1998). Use of the critical decision method to elicit expert knowledge: A case study in the methodology of cognitive task analysis. *Human Factors, 40*, 254–276.
Johnson-Laird, P. N. (1983). *Mental models*. Cambridge: Cambridge University Press.
Johnson-Laird, P. N., Legrenzi, P., & Sonio Legrenzi, M. (1972). Reasoning and a sense of reality. *British Journal of Psychology, 63*, 395–400.
Kahneman, D. (2013). *Thinking, fast and slow*. New York, NY: Farrar, Straus and Giroux.
Kahneman, D., Slovic, P., & Tversky, A. (Eds.). (1982). *Judgment under uncertainty: Heuristics and biases*. Cambridge: Cambridge, UK.
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.
Klein, G. A. (1997). *Recognition-primed decision making: Looking back, looking forward*. Hillsdale, NJ: Erlbaum.
Krug, S. (2005). *Don't make me think: A common sense approach to web usability* (2nd ed.). Berkeley, CA: New Riders Press.
Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science, 208*, 1335–1342.
Lehto, M. R. (1997). Decision making. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (pp. 1201–1248). New York, NY: Wiley.
Levitt, S., & Dubner, S. J. (2005). *Freakonomics: A rogue economist explores the hidden side of everything*. New York, NY: William Morrow/HarperCollins.
Luchins, A. S. (1942). Mechanization in problem solving: The effect of Einstellung. *Psychological Monographs, 54*(6), 1–95.
Moray, N. (1996). A taxonomy and theory of mental models. In *Proceedings of the Human Factors and Ergonomics Society 40th Annual Meeting* (Vol. 1, pp. 164–168).
Moray, N. (1999). Mental models in theory and practice. In D. Gopher & A. Koriat (Eds.), *Attention and performance XVII: Cognitive regulation of performance: Interaction of theory and application* (pp. 223–258). Cambridge, MA: MIT Press.
Norman, D. A. (1983). Design rules based on analysis of human error. *Communications of the ACM, 26*(4), 254–258.
Norman, D. A. (1988). *The psychology of everyday things*. New York, NY: Basic Books.
Norman, D. A. (2013). *The design of everyday things*. New York, NY: Basic Books.
NTSB. (1990). *Aircraft accident report. United Airlines flight 232. Mc Donnell Douglas DC-10-10. Sioux Gateway airport. Sioux City, Iowa, July 19, 1989*. Washington DC: National Transportation Safety Board.
Payne, S. J. (1995). Naive judgments of stimulus-response compatibility. *Human Factors, 37*, 495–506.
Quinlan, P., & Dyson, B. (2008). *Cognitive psychology*. Harlow, UK: Pearson.

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

Revell, K. M. A., & Stanton, N. A. (2012). Models of models: Filtering and bias rings in depiction of knowledge structures and their implications for design. *Ergonomics, 55*(9), 1073–1092.

Ries, A., & Trout, J. (1986/2000). *Positioning: The battle for your mind*. New York, NY: McGraw-Hill International.

Ritter, F. E., & Bibby, P. A. (2008). Modeling how, when, and what learning happens in a diagrammatic reasoning task. *Cognitive Science, 32*, 862–892.

Ritter, F. E., Jones, R. M., & Baxter, G. D. (1998). Reusable models and graphical interfaces: Realising the potential of a unified theory of cognition. In U. Schmid, J. Krems, & F. Wysotzki (Eds.), *Mind modeling—a cognitive science approach to reasoning, learning and discovery* (pp. 83–109). Lengerich, Germany: Pabst Scientific Publishing.

Schwartz, B. (2004). *The paradox of choice: Why more is less*. New York, NY: Harper Perennial.

Simon, H. A. (1997). *Administrative behavior* (4th ed.). New York, NY: The Free Press.

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

Smallman, H. S., & St. John, M. (2005). Naïve realism: Misplaced faith in the utility of realistic displays. *Ergonomics in Design, 13*(Summer), 6–13.

Sternberg, R. J. (2004). Why smart people can be so foolish. *European Psychologist, 9*(3), 145–150.

Sternberg, R. J., & Davidson, J. E. (1995). *The nature of insight*. Cambridge, MA: MIT Press.

Tverksy, A., & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science, 211*, 453–458.

Tverksy, A., & Kahneman, D. (1986). Rational choice and the framing of decisions. *Journal of Business, 59*, 251–278.

Wason, P. C. (1960). On the failure to eliminate hypotheses in a conceptual task. *Quarterly Journal of Experimental Psychology, 12*, 129–140.

Wilson, J. R., & Rutherford, A. (1989). Mental models: Theory and application in human factors. *Human Factors, 31*, 617–634.

Winne, P. H., & Jamieson-Noel, D. (2002). Exploring students' calibration of self reports about study tactics and achievement. *Contemporary Educational Psychology, 27*(4), 551–572.