

THE
DESIGN
OF EVERYDAY
THINGS

THE PSYCHOPATHOLOGY OF EVERYDAY THINGS



If I were placed in the cockpit of a modern jet airliner, my inability to perform well would neither surprise nor bother me. But why should I have trouble with doors and light switches, water faucets and stoves? “Doors?” I can hear the reader saying. “You have trouble opening doors?” Yes. I push doors that are meant to be pulled, pull doors that should be pushed, and walk into doors that neither pull nor push, but slide. Moreover, I see others having the same troubles—unnecessary troubles. My problems with doors have become so well known that confusing doors are often called “Norman doors.” Imagine becoming famous for doors that don’t work right. I’m pretty sure that’s not what my parents planned for me. (Put “Norman doors” into your favorite search engine—be sure to include the quote marks: it makes for fascinating reading.)

How can such a simple thing as a door be so confusing? A door would seem to be about as simple a device as possible. There is not much you can do to a door: you can open it or shut it. Suppose you are in an office building, walking down a corridor. You come to a door. How does it open? Should you push or pull, on the left or the right? Maybe the door slides. If so, in which direction? I have seen doors that slide to the left, to the right, and even up into the ceiling.



FIGURE 1.1. Coffeepot for Masochists. The French artist Jacques Carelman in his series of books *Catalogue d'objets introuvables* (Catalog of unfindable objects) provides delightful examples of everyday things that are deliberately unworkable, outrageous, or otherwise ill-formed. One of my favorite items is what he calls “coffeepot for masochists.” The photograph shows a copy given to me by colleagues at the University of California, San Diego. It is one of my treasured art objects. (Photograph by Aymin Shamma for the author.)

The design of the door should indicate how to work it without any need for signs, certainly without any need for trial and error.

A friend told me of the time he got trapped in the doorway of a post office in a European city. The entrance was an imposing row of six glass swinging doors, followed immediately by a second, identical row. That’s a standard design: it helps reduce the airflow and thus maintain the indoor temperature of the building. There was no visible hardware: obviously the doors could swing in either direction: all a person had to do was push the side of the door and enter.

My friend pushed on one of the outer doors. It swung inward, and he entered the building. Then, before he could get to the next row of doors, he was distracted and turned around for an instant. He didn’t realize it at the time, but he had moved slightly to the right. So when he came to the next door and pushed it, nothing happened. “Hmm,” he thought, “must be locked.” So he pushed the side of the adjacent door. Nothing. Puzzled, my friend decided to go outside again. He turned around and pushed against the side of a door. Nothing. He pushed the adjacent door. Nothing. The door he had just entered no longer worked. He turned around once more and tried the inside doors again. Nothing. Concern, then mild panic. He was trapped! Just then, a group of people on the other side of the entranceway (to my friend’s right) passed easily through both sets of doors. My friend hurried over to follow their path.

How could such a thing happen? A swinging door has two sides. One contains the supporting pillar and the hinge, the other is unsupported. To open the door, you must push or pull on the unsupported edge. If you push on the hinge side, nothing happens. In my friend's case, he was in a building where the designer aimed for beauty, not utility. No distracting lines, no visible pillars, no visible hinges. So how can the ordinary user know which side to push on? While distracted, my friend had moved toward the (invisible) supporting pillar, so he was pushing the doors on the hinged side. No wonder nothing happened. Attractive doors. Stylish. Probably won a design prize.

Two of the most important characteristics of good design are *discoverability* and *understanding*. Discoverability: Is it possible to even figure out what actions are possible and where and how to perform them? Understanding: What does it all mean? How is the product supposed to be used? What do all the different controls and settings mean?

The doors in the story illustrate what happens when discoverability fails. Whether the device is a door or a stove, a mobile phone or a nuclear power plant, the relevant components must be visible, and they must communicate the correct message: What actions are possible? Where and how should they be done? With doors that push, the designer must provide signals that naturally indicate where to push. These need not destroy the aesthetics. Put a vertical plate on the side to be pushed. Or make the supporting pillars visible. The vertical plate and supporting pillars are natural signals, naturally interpreted, making it easy to know just what to do: no labels needed.

With complex devices, discoverability and understanding require the aid of manuals or personal instruction. We accept this if the device is indeed complex, but it should be unnecessary for simple things. Many products defy understanding simply because they have too many functions and controls. I don't think that simple home appliances—stoves, washing machines, audio and television sets—should look like Hollywood's idea of a spaceship control room. They already do, much to our consternation. Faced

with a bewildering array of controls and displays, we simply memorize one or two fixed settings to approximate what is desired.

In England I visited a home with a fancy new Italian washer-dryer combination, with super-duper multisymbol controls, all to do everything anyone could imagine doing with the washing and drying of clothes. The husband (an engineering psychologist) said he refused to go near it. The wife (a physician) said she had simply memorized one setting and tried to ignore the rest. I asked to see the manual: it was just as confusing as the device. The whole purpose of the design is lost.

The Complexity of Modern Devices

All artificial things are designed. Whether it is the layout of furniture in a room, the paths through a garden or forest, or the intricacies of an electronic device, some person or group of people had to decide upon the layout, operation, and mechanisms. Not all designed things involve physical structures. Services, lectures, rules and procedures, and the organizational structures of businesses and governments do not have physical mechanisms, but their rules of operation have to be designed, sometimes informally, sometimes precisely recorded and specified.

But even though people have designed things since prehistoric times, the field of design is relatively new, divided into many areas of specialty. Because everything is designed, the number of areas is enormous, ranging from clothes and furniture to complex control rooms and bridges. This book covers everyday things, focusing on the interplay between technology and people to ensure that the products actually fulfill human needs while being understandable and usable. In the best of cases, the products should also be delightful and enjoyable, which means that not only must the requirements of engineering, manufacturing, and ergonomics be satisfied, but attention must be paid to the entire experience, which means the aesthetics of form and the quality of interaction. The major areas of design relevant to this book are industrial design, interaction design, and experience design. None of the fields is well defined, but the focus of the efforts does vary, with industrial

designers emphasizing form and material, interactive designers emphasizing understandability and usability, and experience designers emphasizing the emotional impact. Thus:

Industrial design: The professional service of creating and developing concepts and specifications that optimize the function, value, and appearance of products and systems for the mutual benefit of both user and manufacturer (from the *Industrial Design Society of America's* website).

Interaction design: The focus is upon how people interact with technology. The goal is to enhance people's understanding of what can be done, what is happening, and what has just occurred. Interaction design draws upon principles of psychology, design, art, and emotion to ensure a positive, enjoyable experience.

Experience design: The practice of designing products, processes, services, events, and environments with a focus placed on the quality and enjoyment of the total experience.

Design is concerned with how things work, how they are controlled, and the nature of the interaction between people and technology. When done well, the results are brilliant, pleasurable products. When done badly, the products are unusable, leading to great frustration and irritation. Or they might be usable, but force us to behave the way the product wishes rather than as we wish.

Machines, after all, are conceived, designed, and constructed by people. By human standards, machines are pretty limited. They do not maintain the same kind of rich history of experiences that people have in common with one another, experiences that enable us to interact with others because of this shared understanding. Instead, machines usually follow rather simple, rigid rules of behavior. If we get the rules wrong even slightly, the machine does what it is told, no matter how insensible and illogical. People are imaginative and creative, filled with common sense; that is, a lot of valuable knowledge built up over years of experience. But instead of capitalizing on these strengths, machines require us to be precise and accurate, things we are not very good at. Machines have no

leeway or common sense. Moreover, many of the rules followed by a machine are known only by the machine and its designers.

When people fail to follow these bizarre, secret rules, and the machine does the wrong thing, its operators are blamed for not understanding the machine, for not following its rigid specifications. With everyday objects, the result is frustration. With complex devices and commercial and industrial processes, the resulting difficulties can lead to accidents, injuries, and even deaths. It is time to reverse the situation: to cast the blame upon the machines and their design. It is the machine and its design that are at fault. It is the duty of machines and those who design them to understand people. It is not our duty to understand the arbitrary, meaningless dictates of machines.

The reasons for the deficiencies in human-machine interaction are numerous. Some come from the limitations of today's technology. Some come from self-imposed restrictions by the designers, often to hold down cost. But most of the problems come from a complete lack of understanding of the design principles necessary for effective human-machine interaction. Why this deficiency? Because much of the design is done by engineers who are experts in technology but limited in their understanding of people. "We are people ourselves," they think, "so we understand people." But in fact, we humans are amazingly complex. Those who have not studied human behavior often think it is pretty simple. Engineers, moreover, make the mistake of thinking that logical explanation is sufficient: "If only people would read the instructions," they say, "everything would be all right."

Engineers are trained to think logically. As a result, they come to believe that all people must think this way, and they design their machines accordingly. When people have trouble, the engineers are upset, but often for the wrong reason. "What are these people doing?" they will wonder. "Why are they doing that?" The problem with the designs of most engineers is that they are too logical. We have to accept human behavior the way it is, not the way we would wish it to be.

I used to be an engineer, focused upon technical requirements, quite ignorant of people. Even after I switched into psychology and cognitive science, I still maintained my engineering emphasis upon logic and mechanism. It took a long time for me to realize that my understanding of human behavior was relevant to my interest in the design of technology. As I watched people struggle with technology, it became clear that the difficulties were caused by the technology, not the people.

I was called upon to help analyze the American nuclear power plant accident at Three Mile Island (the island name comes from the fact that it is located on a river, three miles south of Middletown in the state of Pennsylvania). In this incident, a rather simple mechanical failure was misdiagnosed. This led to several days of difficulties and confusion, total destruction of the reactor, and a very close call to a severe radiation release, all of which brought the American nuclear power industry to a complete halt. The operators were blamed for these failures: “human error” was the immediate analysis. But the committee I was on discovered that the plant’s control rooms were so poorly designed that error was inevitable: design was at fault, not the operators. The moral was simple: we were designing things for people, so we needed to understand both technology and people. But that’s a difficult step for many engineers: machines are so logical, so orderly. If we didn’t have people, everything would work so much better. Yup, that’s how I used to think.

My work with that committee changed my view of design. Today, I realize that design presents a fascinating interplay of technology and psychology, that the designers must understand both. Engineers still tend to believe in logic. They often explain to me in great, logical detail, why their designs are good, powerful, and wonderful. “Why are people having problems?” they wonder. “You are being too logical,” I say. “You are designing for people the way you would like them to be, not for the way they really are.”

When the engineers object, I ask whether they have ever made an error, perhaps turning on or off the wrong light, or the wrong

stove burner. “Oh yes,” they say, “but those were errors.” That’s the point: even experts make errors. So we must design our machines on the assumption that people will make errors. (Chapter 5 provides a detailed analysis of human error.)

Human-Centered Design

People are frustrated with everyday things. From the ever-increasing complexity of the automobile dashboard, to the increasing automation in the home with its internal networks, complex music, video, and game systems for entertainment and communication, and the increasing automation in the kitchen, everyday life sometimes seems like a never-ending fight against confusion, continued errors, frustration, and a continual cycle of updating and maintaining our belongings.

In the multiple decades that have elapsed since the first edition of this book was published, design has gotten better. There are now many books and courses on the topic. But even though much has improved, the rapid rate of technology change outpaces the advances in design. New technologies, new applications, and new methods of interaction are continually arising and evolving. New industries spring up. Each new development seems to repeat the mistakes of the earlier ones; each new field requires time before it, too, adopts the principles of good design. And each new invention of technology or interaction technique requires experimentation and study before the principles of good design can be fully integrated into practice. So, yes, things are getting better, but as a result, the challenges are ever present.

The solution is human-centered design (HCD), an approach that puts human needs, capabilities, and behavior first, then designs to accommodate those needs, capabilities, and ways of behaving. Good design starts with an understanding of psychology and technology. Good design requires good communication, especially from machine to person, indicating what actions are possible, what is happening, and what is about to happen. Communication is especially important when things go wrong. It is relatively easy to design things that work smoothly and harmoniously as

TABLE 1.1. The Role of HCD and Design Specializations

Experience design	These are areas of focus
Industrial design	
Interaction design	
Human-centered design	The process that ensures that the designs match the needs and capabilities of the people for whom they are intended

long as things go right. But as soon as there is a problem or a misunderstanding, the problems arise. This is where good design is essential. Designers need to focus their attention on the cases where things go wrong, not just on when things work as planned. Actually, this is where the most satisfaction can arise: when something goes wrong but the machine highlights the problems, then the person understands the issue, takes the proper actions, and the problem is solved. When this happens smoothly, the collaboration of person and device feels wonderful.

Human-centered design is a design philosophy. It means starting with a good understanding of people and the needs that the design is intended to meet. This understanding comes about primarily through observation, for people themselves are often unaware of their true needs, even unaware of the difficulties they are encountering. Getting the specification of the thing to be defined is one of the most difficult parts of the design, so much so that the HCD principle is to avoid specifying the problem as long as possible but instead to iterate upon repeated approximations. This is done through rapid tests of ideas, and after each test modifying the approach and the problem definition. The results can be products that truly meet the needs of people. Doing HCD within the rigid time, budget, and other constraints of industry can be a challenge: Chapter 6 examines these issues.

Where does HCD fit into the earlier discussion of the several different forms of design, especially the areas called industrial, interaction, and experience design? These are all compatible. HCD is a philosophy and a set of procedures, whereas the others are areas of focus (see Table 1.1). The philosophy and procedures of HCD add

deep consideration and study of human needs to the design process, whatever the product or service, whatever the major focus.

Fundamental Principles of Interaction

Great designers produce pleasurable experiences. *Experience*: note the word. Engineers tend not to like it; it is too subjective. But when I ask them about their favorite automobile or test equipment, they will smile delightedly as they discuss the fit and finish, the sensation of power during acceleration, their ease of control while shifting or steering, or the wonderful feel of the knobs and switches on the instrument. Those are experiences.

Experience is critical, for it determines how fondly people remember their interactions. Was the overall experience positive, or was it frustrating and confusing? When our home technology behaves in an uninterpretable fashion we can become confused, frustrated, and even angry—all strong negative emotions. When there is understanding it can lead to a feeling of control, of mastery, and of satisfaction or even pride—all strong positive emotions. Cognition and emotion are tightly intertwined, which means that the designers must design with both in mind.

When we interact with a product, we need to figure out how to work it. This means discovering what it does, how it works, and what operations are possible: discoverability. Discoverability results from appropriate application of five fundamental psychological concepts covered in the next few chapters: *affordances*, *signifiers*, *constraints*, *mappings*, and *feedback*. But there is a sixth principle, perhaps most important of all: the *conceptual model* of the system. It is the conceptual model that provides true understanding. So I now turn to these fundamental principles, starting with affordances, signifiers, mappings, and feedback, then moving to conceptual models. Constraints are covered in Chapters 3 and 4.

AFFORDANCES

We live in a world filled with objects, many natural, the rest artificial. Every day we encounter thousands of objects, many of them new to us. Many of the new objects are similar to ones we already

know, but many are unique, yet we manage quite well. How do we do this? Why is it that when we encounter many unusual natural objects, we know how to interact with them? Why is this true with many of the artificial, human-made objects we encounter? The answer lies with a few basic principles. Some of the most important of these principles come from a consideration of affordances.

The term *affordance* refers to the relationship between a physical object and a person (or for that matter, any interacting agent, whether animal or human, or even machines and robots). An affordance is a relationship between the properties of an object and the capabilities of the agent that determine just how the object could possibly be used. A chair affords (“is for”) support and, therefore, affords sitting. Most chairs can also be carried by a single person (they afford lifting), but some can only be lifted by a strong person or by a team of people. If young or relatively weak people cannot lift a chair, then for these people, the chair does not have that affordance, it does not afford lifting.

The presence of an affordance is jointly determined by the qualities of the object and the abilities of the agent that is interacting. This relational definition of affordance gives considerable difficulty to many people. We are used to thinking that properties are associated with objects. But affordance is not a property. An affordance is a relationship. Whether an affordance exists depends upon the properties of both the object and the agent.

Glass affords transparency. At the same time, its physical structure blocks the passage of most physical objects. As a result, glass affords seeing through and support, but not the passage of air or most physical objects (atomic particles can pass through glass). The blockage of passage can be considered an anti-affordance—the prevention of interaction. To be effective, affordances and anti-affordances have to be discoverable—perceivable. This poses a difficulty with glass. The reason we like glass is its relative invisibility, but this aspect, so useful in the normal window, also hides its anti-affordance property of blocking passage. As a result, birds often try to fly through windows. And every year, numerous people injure themselves when they walk (or run) through closed glass

doors or large picture windows. If an affordance or anti-affordance cannot be perceived, some means of signaling its presence is required: I call this property a *signifier* (discussed in the next section).

The notion of affordance and the insights it provides originated with J. J. Gibson, an eminent psychologist who provided many advances to our understanding of human perception. I had interacted with him over many years, sometimes in formal conferences and seminars, but most fruitfully over many bottles of beer, late at night, just talking. We disagreed about almost everything. I was an engineer who became a cognitive psychologist, trying to understand how the mind works. He started off as a Gestalt psychologist, but then developed an approach that is today named after him: Gibsonian psychology, an ecological approach to perception. He argued that the world contained the clues and that people simply picked them up through “direct perception.” I argued that nothing could be direct: the brain had to process the information arriving at the sense organs to put together a coherent interpretation. “Nonsense,” he loudly proclaimed; “it requires no interpretation: it is directly perceived.” And then he would put his hand to his ears, and with a triumphant flourish, turn off his hearing aids: my counterarguments would fall upon deaf ears—literally.

When I pondered my question—how do people know how to act when confronted with a novel situation—I realized that a large part of the answer lay in Gibson’s work. He pointed out that all the senses work together, that we pick up information about the world by the combined result of all of them. “Information pickup” was one of his favorite phrases, and Gibson believed that the combined information picked up by all of our sensory apparatus—sight, sound, smell, touch, balance, kinesthetic, acceleration, body position—determines our perceptions without the need for internal processing or cognition. Although he and I disagreed about the role played by the brain’s internal processing, his brilliance was in focusing attention on the rich amount of information present in the world. Moreover, the physical objects conveyed important information about how people could interact with them, a property he named “affordance.”

Affordances exist even if they are not visible. For designers, their visibility is critical: visible affordances provide strong clues to the operations of things. A flat plate mounted on a door affords pushing. Knobs afford turning, pushing, and pulling. Slots are for inserting things into. Balls are for throwing or bouncing. Perceived affordances help people figure out what actions are possible without the need for labels or instructions. I call the signaling component of affordances *signifiers*.

SIGNIFIERS

Are affordances important to designers? The first edition of this book introduced the term *affordances* to the world of design. The design community loved the concept and affordances soon propagated into the instruction and writing about design. I soon found mention of the term everywhere. Alas, the term became used in ways that had nothing to do with the original.

Many people find affordances difficult to understand because they are relationships, not properties. Designers deal with fixed properties, so there is a temptation to say that the property is an affordance. But that is not the only problem with the concept of affordances.

Designers have practical problems. They need to know how to design things to make them understandable. They soon discovered that when working with the graphical designs for electronic displays, they needed a way to designate which parts could be touched, slid upward, downward, or sideways, or tapped upon. The actions could be done with a mouse, stylus, or fingers. Some systems responded to body motions, gestures, and spoken words, with no touching of any physical device. How could designers describe what they were doing? There was no word that fit, so they took the closest existing word—*affordance*. Soon designers were saying such things as, “I put an affordance there,” to describe why they displayed a circle on a screen to indicate where the person should touch, whether by mouse or by finger. “No,” I said, “that is not an affordance. That is a way of communicating where the touch should be. You are communicating where to do the touching: the

affordance of touching exists on the entire screen: you are trying to signify *where* the touch should take place. That's not the same thing as saying *what* action is possible."

Not only did my explanation fail to satisfy the design community, but I myself was unhappy. Eventually I gave up: designers needed a word to describe what they were doing, so they chose *affordance*. What alternative did they have? I decided to provide a better answer: *signifiers*. Affordances determine what actions are possible. Signifiers communicate where the action should take place. We need both.

People need some way of understanding the product or service they wish to use, some sign of what it is for, what is happening, and what the alternative actions are. People search for clues, for any sign that might help them cope and understand. It is the sign that is important, anything that might signify meaningful information. Designers need to provide these clues. What people need, and what designers must provide, are signifiers. Good design requires, among other things, good communication of the purpose, structure, and operation of the device to the people who use it. That is the role of the signifier.

The term *signifier* has had a long and illustrious career in the exotic field of semiotics, the study of signs and symbols. But just as I appropriated *affordance* to use in design in a manner somewhat different than its inventor had intended, I use *signifier* in a somewhat different way than it is used in semiotics. For me, the term *signifier* refers to any mark or sound, any perceivable indicator that communicates appropriate behavior to a person.

Signifiers can be deliberate and intentional, such as the sign PUSH on a door, but they may also be accidental and unintentional, such as our use of the visible trail made by previous people walking through a field or over a snow-covered terrain to determine the best path. Or how we might use the presence or absence of people waiting at a train station to determine whether we have missed the train. (I explain these ideas in more detail in my book *Living with Complexity*.)



FIGURE 1.2. Problem Doors: Signifiers Are Needed. Door hardware can signal whether to push or pull without signs, but the hardware of the two doors in the upper photo, A, are identical even though one should be pushed, the other pulled. The flat, ribbed horizontal bar has the obvious perceived affordance of pushing, but as the signs indicate, the door on the left is to be pulled, the one on the right is to be pushed. In the bottom pair of photos, B and C, there are no visible signifiers or affordances. How does one know which side to push? Trial and error. When external signifiers—signs—have to be added to something as simple as a door, it indicates bad design. (Photographs by the author.)

The signifier is an important communication device to the recipient, whether or not communication was intended. It doesn't matter whether the useful signal was deliberately placed or whether it is incidental: there is no necessary distinction. Why should it matter whether a flag was placed as a deliberate clue to wind direction (as is done at airports or on the masts of sailboats) or was there as an

advertisement or symbol of pride in one's country (as is done on public buildings). Once I interpret a flag's motion to indicate wind direction, it does not matter why it was placed there.

Consider a bookmark, a deliberately placed signifier of one's place in reading a book. But the physical nature of books also makes a bookmark an accidental signifier, for its placement also indicates how much of the book remains. Most readers have learned to use this accidental signifier to aid in their enjoyment of the reading. With few pages left, we know the end is near. And if the reading is torturous, as in a school assignment, one can always console oneself by knowing there are "only a few more pages to get through." Electronic book readers do not have the physical structure of paper books, so unless the software designer deliberately provides a clue, they do not convey any signal about the amount of text remaining.

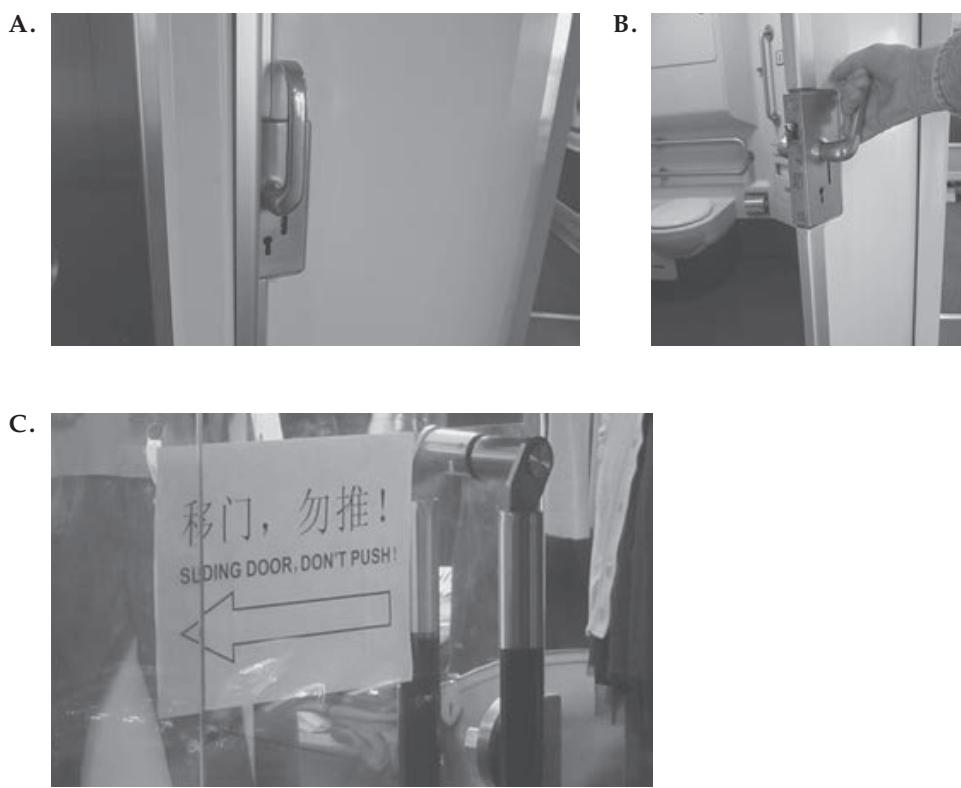


FIGURE 1.3. Sliding Doors: Seldom Done Well. Sliding doors are seldom signified properly. The top two photographs show the sliding door to the toilet on an Amtrak train in the United States. The handle clearly signifies "pull," but in fact, it needs to be rotated and the door slid to the right. The owner of the store in Shanghai, China, Photo C, solved the problem with a sign. "DON'T PUSH!" it says, in both English and Chinese. Amtrak's toilet door could have used a similar kind of sign. (Photographs by the author.)

Whatever their nature, planned or accidental, signifiers provide valuable clues as to the nature of the world and of social activities. For us to function in this social, technological world, we need to develop internal models of what things mean, of how they operate. We seek all the clues we can find to help in this enterprise, and in this way, we are detectives, searching for whatever guidance we might find. If we are fortunate, thoughtful designers provide the clues for us. Otherwise, we must use our own creativity and imagination.

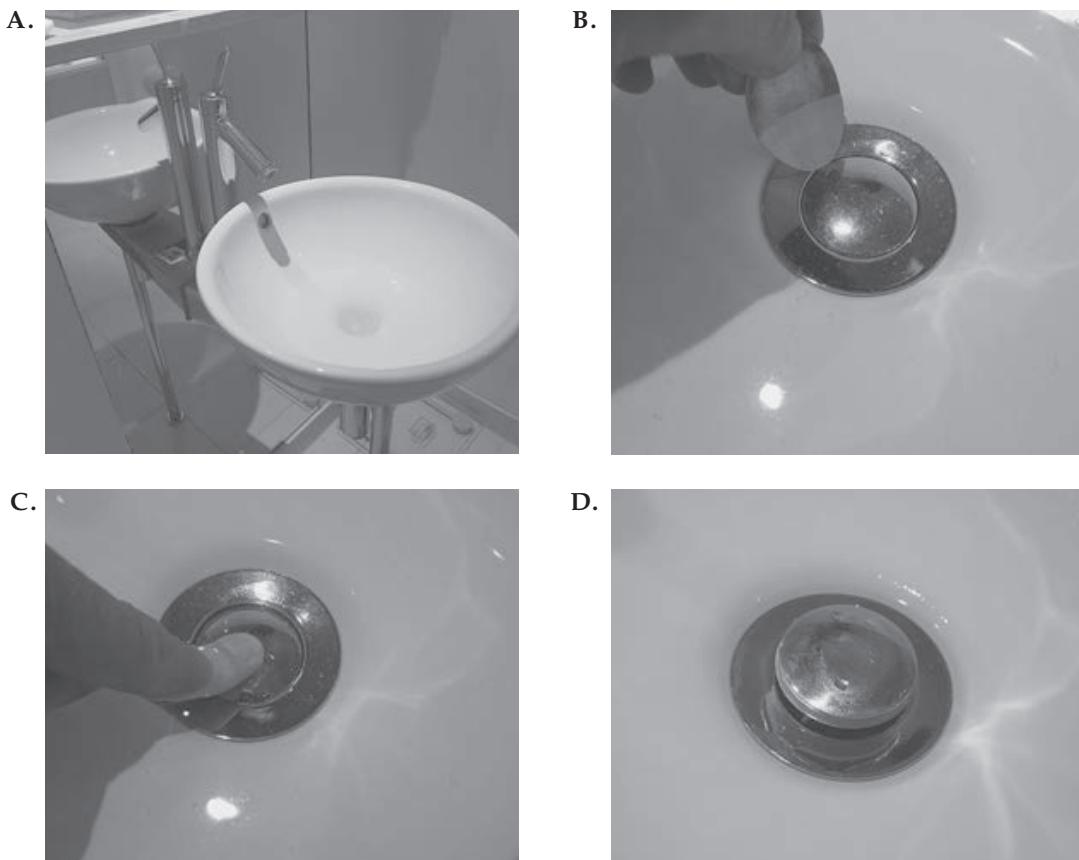


FIGURE 1.4. The Sink That Would Not Drain: Where Signifiers Fail. I washed my hands in my hotel sink in London, but then, as shown in Photo A, was left with the question of how to empty the sink of the dirty water. I searched all over for a control: none. I tried prying open the sink stopper with a spoon (Photo B): failure. I finally left my hotel room and went to the front desk to ask for instructions. (Yes, I actually did.) “Push down on the stopper,” I was told. Yes, it worked (Photos C and D). But how was anyone to ever discover this? And why should I have to put my clean hands back into the dirty water to empty the sink? The problem here is not just the lack of signifier, it is the faulty decision to produce a stopper that requires people to dirty their clean hands to use it. (Photographs by the author.)

Affordances, perceived affordances, and signifiers have much in common, so let me pause to ensure that the distinctions are clear.

Affordances represent the possibilities in the world for how an agent (a person, animal, or machine) can interact with something. Some affordances are perceivable, others are invisible. Signifiers are signals. Some signifiers are signs, labels, and drawings placed in the world, such as the signs labeled “push,” “pull,” or “exit” on doors, or arrows and diagrams indicating what is to be acted upon or in which direction to gesture, or other instructions. Some signifiers are simply the perceived affordances, such as the handle of a door or the physical structure of a switch. Note that some perceived affordances may not be real: they may look like doors or places to push, or an impediment to entry, when in fact they are not. These are misleading signifiers, oftentimes accidental but sometimes purposeful, as when trying to keep people from doing actions for which they are not qualified, or in games, where one of the challenges is to figure out what is real and what is not.

FIGURE 1.5. Accidental Affordances Can Become Strong Signifiers. This wall, at the Industrial Design department of KAIST, in Korea, provides an anti-affordance, preventing people from falling down the stair shaft. Its top is flat, an accidental by-product of the design. But flat surfaces afford support, and as soon as one person discovers it can be used to dispose of empty drink containers, the discarded container becomes a signifier, telling others that it is permissible to discard their items there. (Photographs by the author.)



My favorite example of a misleading signifier is a row of vertical pipes across a service road that I once saw in a public park. The pipes obviously blocked cars and trucks from driving on that road: they were good examples of anti-affordances. But to my great surprise, I saw a park vehicle simply go through the pipes. Huh? I walked over and examined them: the pipes were made of rubber, so vehicles could simply drive right over them. A very clever signifier, signaling a blocked road (via an apparent anti-affordance) to the average person, but permitting passage for those who knew.

To summarize:

- Affordances are the possible interactions between people and the environment. Some affordances are perceivable, others are not.
- Perceived affordances often act as signifiers, but they can be ambiguous.
- Signifiers signal things, in particular what actions are possible and how they should be done. Signifiers must be perceivable, else they fail to function.

In design, signifiers are more important than affordances, for they communicate how to use the design. A signifier can be words, a graphical illustration, or just a device whose perceived affordances are unambiguous. Creative designers incorporate the signifying part of the design into a cohesive experience. For the most part, designers can focus upon signifiers.

Because affordances and signifiers are fundamentally important principles of good design, they show up frequently in the pages of this book. Whenever you see hand-lettered signs pasted on doors, switches, or products, trying to explain how to work them, what to do and what not to do, you are also looking at poor design.

AFFORDANCES AND SIGNIFIERS: A CONVERSATION

A designer approaches his mentor. He is working on a system that recommends restaurants to people, based upon their preferences and those of their friends. But in his tests, he discovered that people never used all of the features. “Why not?” he asks his mentor.

(With apologies to Socrates.)

DESIGNER	MENTOR
I'm frustrated; people aren't using our application properly.	Can you tell me about it?
The screen shows the restaurant that we recommend. It matches their preferences, and their friends like it as well. If they want to see other recommendations, all they have to do is swipe left or right. To learn more about a place, just swipe up for a menu or down to see if any friends are there now. People seem to find the other recommendations, but not the menus or their friends? I don't understand.	Why do you think this might be?
I don't know. Should I add some affordances? Suppose I put an arrow on each edge and add a label saying what they do.	That is very nice. But why do you call these affordances? They could already do the actions. Weren't the affordances already there?
Yes, you have a point. But the affordances weren't visible. I made them visible.	Very true. You added a signal of what to do.
Yes, isn't that what I said?	Not quite—you called them affordances even though they afford nothing new: they signify what to do and where to do it. So call them by their right name: " <i>signifiers</i> ."
Oh, I see. But then why do designers care about affordances? Perhaps we should focus our attention on signifiers.	You speak wisely. Communication is a key to good design. And a key to communication is the signifier.
Oh. Now I understand my confusion. Yes, a signifier is what signifies. It is a sign. Now it seems perfectly obvious.	Profound ideas are always obvious once they are understood.

MAPPING

Mapping is a technical term, borrowed from mathematics, meaning the relationship between the elements of two sets of things. Suppose there are many lights in the ceiling of a classroom or auditorium and a row of light switches on the wall at the front of the



FIGURE 1.6. Signifiers on a Touch Screen.

The arrows and icons are signifiers: they provide signals about the permissible operations for this restaurant guide. Swiping left or right brings up new restaurant recommendations. Swiping up reveals the menu for the restaurant being displayed; swiping down, friends who recommend the restaurant.

room. The mapping of switches to lights specifies which switch controls which light.

Mapping is an important concept in the design and layout of controls and displays. When the mapping uses spatial correspondence between the layout of the controls and the devices being controlled, it is easy to determine how to use them. In steering a car, we rotate the steering wheel clockwise to cause the car to turn right: the top of the wheel moves in the same direction as the car. Note that other choices could have been made. In early cars, steering was controlled by a variety of devices, including tillers, handlebars, and reins. Today, some vehicles use joysticks, much as in a computer game. In cars that used tillers, steering was done much as one steers a boat: move the tiller to the left to turn to the right. Tractors, construction equipment such as bulldozers and cranes, and military tanks that have tracks instead of wheels use separate controls for the speed and direction of each track: to turn right, the left track is increased in speed, while the right track is slowed or even reversed. This is also how a wheelchair is steered.

All of these mappings for the control of vehicles work because each has a compelling conceptual model of how the operation of the control affects the vehicle. Thus, if we speed up the left wheel of a wheelchair while stopping the right wheel, it is easy to imagine the chair's pivoting on the right wheel, circling to the right. In

a small boat, we can understand the tiller by realizing that pushing the tiller to the left causes the ship's rudder to move to the right and the resulting force of the water on the rudder slows down the right side of the boat, so that the boat rotates to the right. It doesn't matter whether these conceptual models are accurate: what matters is that they provide a clear way of remembering and understanding the mappings. The relationship between a control and its results is easiest to learn wherever there is an understandable mapping between the controls, the actions, and the intended result.

Natural mapping, by which I mean taking advantage of spatial analogies, leads to immediate understanding. For example, to move an object up, move the control up. To make it easy to determine which control works which light in a large room or auditorium, arrange the controls in the same pattern as the lights. Some natural mappings are cultural or biological, as in the universal standard that moving the hand up signifies more, moving it down signifies less, which is why it is appropriate to use vertical position to represent intensity or amount. Other natural mappings follow from the principles of perception and allow for the natural grouping or patterning of controls and feedback. Groupings and proximity are important principles from Gestalt psychology that can be used to map controls to function: related controls should be grouped together. Controls should be close to the item being controlled.

Note that there are many mappings that feel "natural" but in fact are specific to a particular culture: what is natural for one culture is not necessarily natural for another. In Chapter 3, I discuss how



FIGURE 1.7. Good Mapping: Automobile Seat Adjustment Control. This is an excellent example of natural mapping. The control is in the shape of the seat itself: the mapping is straightforward. To move the front edge of the seat higher, lift up on the front part of the button. To make the seat back recline, move the button back. The same principle could be applied to much more common objects. This particular control is from Mercedes-Benz, but this form of mapping is now used by many automobile companies. (Photograph by the author.)

different cultures view time, which has important implications for some kinds of mappings.

A device is easy to use when the set of possible actions is visible, when the controls and displays exploit natural mappings. The principles are simple but rarely incorporated into design. Good design takes care, planning, thought, and an understanding of how people behave.

FEEDBACK

Ever watch people at an elevator repeatedly push the Up button, or repeatedly push the pedestrian button at a street crossing? Ever drive to a traffic intersection and wait an inordinate amount of time for the signals to change, wondering all the time whether the detection circuits noticed your vehicle (a common problem with bicycles)? What is missing in all these cases is feedback: some way of letting you know that the system is working on your request.

Feedback—communicating the results of an action—is a well-known concept from the science of control and information theory. Imagine trying to hit a target with a ball when you cannot see the target. Even as simple a task as picking up a glass with the hand requires feedback to aim the hand properly, to grasp the glass, and to lift it. A misplaced hand will spill the contents, too hard a grip will break the glass, and too weak a grip will allow it to fall. The human nervous system is equipped with numerous feedback mechanisms, including visual, auditory, and touch sensors, as well as vestibular and proprioceptive systems that monitor body position and muscle and limb movements. Given the importance of feedback, it is amazing how many products ignore it.

Feedback must be immediate: even a delay of a tenth of a second can be disconcerting. If the delay is too long, people often give up, going off to do other activities. This is annoying to the people, but it can also be wasteful of resources when the system spends considerable time and effort to satisfy the request, only to find that the intended recipient is no longer there. Feedback must also be informative. Many companies try to save money by using inexpensive lights or sound generators for feedback. These simple light flashes

or beeps are usually more annoying than useful. They tell us that something has happened, but convey very little information about what has happened, and then nothing about what we should do about it. When the signal is auditory, in many cases we cannot even be certain which device has created the sound. If the signal is a light, we may miss it unless our eyes are on the correct spot at the correct time. Poor feedback can be worse than no feedback at all, because it is distracting, uninformative, and in many cases irritating and anxiety-provoking.

Too much feedback can be even more annoying than too little. My dishwasher likes to beep at three a.m. to tell me that the wash is done, defeating my goal of having it work in the middle of the night so as not to disturb anyone (and to use less expensive electricity). But worst of all is inappropriate, uninterpretable feedback. The irritation caused by a “backseat driver” is well enough known that it is the staple of numerous jokes. Backseat drivers are often correct, but their remarks and comments can be so numerous and continuous that instead of helping, they become an irritating distraction. Machines that give too much feedback are like backseat drivers. Not only is it distracting to be subjected to continual flashing lights, text announcements, spoken voices, or beeps and boops, but it can be dangerous. Too many announcements cause people to ignore all of them, or wherever possible, disable all of them, which means that critical and important ones are apt to be missed. Feedback is essential, but not when it gets in the way of other things, including a calm and relaxing environment.

Poor design of feedback can be the result of decisions aimed at reducing costs, even if they make life more difficult for people. Rather than use multiple signal lights, informative displays, or rich, musical sounds with varying patterns, the focus upon cost reduction forces the design to use a single light or sound to convey multiple types of information. If the choice is to use a light, then one flash might mean one thing; two rapid flashes, something else. A long flash might signal yet another state; and a long flash followed by a brief one, yet another. If the choice is to use a sound, quite often the least expensive sound device is selected, one that

can only produce a high-frequency beep. Just as with the lights, the only way to signal different states of the machine is by beeping different patterns. What do all these different patterns mean? How can we possibly learn and remember them? It doesn't help that every different machine uses a different pattern of lights or beeps, sometimes with the same patterns meaning contradictory things for different machines. All the beeps sound alike, so it often isn't even possible to know which machine is talking to us.

Feedback has to be planned. All actions need to be confirmed, but in a manner that is unobtrusive. Feedback must also be prioritized, so that unimportant information is presented in an unobtrusive fashion, but important signals are presented in a way that does capture attention. When there are major emergencies, then even important signals have to be prioritized. When every device is signaling a major emergency, nothing is gained by the resulting cacophony. The continual beeps and alarms of equipment can be dangerous. In many emergencies, workers have to spend valuable time turning off all the alarms because the sounds interfere with the concentration required to solve the problem. Hospital operating rooms, emergency wards. Nuclear power control plants. Airplane cockpits. All can become confusing, irritating, and life-endangering places because of excessive feedback, excessive alarms, and incompatible message coding. Feedback is essential, but it has to be done correctly. Appropriately.

CONCEPTUAL MODELS

A conceptual model is an explanation, usually highly simplified, of how something works. It doesn't have to be complete or even accurate as long as it is useful. The files, folders, and icons you see displayed on a computer screen help people create the conceptual model of documents and folders inside the computer, or of apps or applications residing on the screen, waiting to be summoned. In fact, there are no folders inside the computer—those are effective conceptualizations designed to make them easier to use. Sometimes these depictions can add to the confusion, however. When reading e-mail or visiting a website, the material appears to be on

the device, for that is where it is displayed and manipulated. But in fact, in many cases the actual material is “in the cloud,” located on some distant machine. The conceptual model is of one, coherent image, whereas it may actually consist of parts, each located on different machines that could be almost anywhere in the world. This simplified model is helpful for normal usage, but if the network connection to the cloud services is interrupted, the result can be confusing. Information is still on their screen, but users can no longer save it or retrieve new things: their conceptual model offers no explanation. Simplified models are valuable only as long as the assumptions that support them hold true.

There are often multiple conceptual models of a product or device. People’s conceptual models for the way that regenerative braking in a hybrid or electrically powered automobile works are quite different for average drivers than for technically sophisticated drivers, different again for whoever must service the system, and yet different again for those who designed the system.

Conceptual models found in technical manuals and books for technical use can be detailed and complex. The ones we are concerned with here are simpler: they reside in the minds of the people who are using the product, so they are also “mental models.” Mental models, as the name implies, are the conceptual models in people’s minds that represent their understanding of how things work. Different people may hold different mental models of the same item. Indeed, a single person might have multiple models of the same item, each dealing with a different aspect of its operation: the models can even be in conflict.

Conceptual models are often inferred from the device itself. Some models are passed on from person to person. Some come from manuals. Usually the device itself offers very little assistance, so the model is constructed by experience. Quite often these models are erroneous, and therefore lead to difficulties in using the device.

The major clues to how things work come from their perceived structure—in particular from signifiers, affordances, constraints, and mappings. Hand tools for the shop, gardening, and the house tend to make their critical parts sufficiently visible that concep-



FIGURE 1.8. Junghans Mega 1000 Digital Radio Controlled Watch. There is no good conceptual model for understanding the operation of my watch. It has five buttons with no hints as to what each one does. And yes, the buttons do different things in their different modes. But it is a very nice-looking watch, and always has the exact time because it checks official radio time stations. (The top row of the display is the date: Wednesday, February 20, the eighth week of the year.) (Photograph by the author.)

tual models of their operation and function are readily derived. Consider a pair of scissors: you can see that the number of possible actions is limited. The holes are clearly there to put something into, and the only logical things that will fit are fingers. The holes are both affordances—they allow the fingers to be inserted—and signifiers—they indicate where the fingers are to go. The sizes of the holes provide constraints to limit the possible fingers: a big hole suggests several fingers; a small hole, only one. The mapping between holes and fingers—the set of possible operations—is signified and constrained by the holes. Moreover, the operation is not sensitive to finger placement: if you use the wrong fingers (or the wrong hand), the scissors still work, although not as comfortably. You can figure out the scissors because their operating parts are visible and the implications clear. The conceptual model is obvious, and there is effective use of signifiers, affordances, and constraints.

What happens when the device does not suggest a good conceptual model? Consider my digital watch with five buttons: two along the top, two along the bottom, and one on the left side (Figure 1.8). What is each button for? How would you set the time? There is no way to tell—no evident relationship between the operating controls and the functions, no constraints, no apparent mappings. Moreover, the buttons have multiple ways of being used. Two of the buttons do different things when pushed quickly or when kept depressed for several seconds. Some operations require simultaneous depression of several of the buttons. The only way to tell how to work the watch is to read the manual, over and over again. With the scissors, moving the handle makes the blades move. The watch provides no

visible relationship between the buttons and the possible actions, no discernible relationship between the actions and the end results. I really like the watch: too bad I can't remember all the functions.

Conceptual models are valuable in providing understanding, in predicting how things will behave, and in figuring out what to do when things do not go as planned. A good conceptual model allows us to predict the effects of our actions. Without a good model, we operate by rote, blindly; we do operations as we were told to do them; we can't fully appreciate why, what effects to expect, or what to do if things go wrong. As long as things work properly, we can manage. When things go wrong, however, or when we come upon a novel situation, then we need a deeper understanding, a good model.

For everyday things, conceptual models need not be very complex. After all, scissors, pens, and light switches are pretty simple devices. There is no need to understand the underlying physics or chemistry of each device we own, just the relationship between the controls and the outcomes. When the model presented to us is inadequate or wrong (or, worse, nonexistent), we can have difficulties. Let me tell you about my refrigerator.

I used to own an ordinary, two-compartment refrigerator—nothing very fancy about it. The problem was that I couldn't set the temperature properly. There were only two things to do: adjust the temperature of the freezer compartment and adjust the tempera-



FIGURE 1.9. Refrigerator Controls. Two compartments—fresh food and freezer—and two controls (in the fresh food unit). Your task: Suppose the freezer is too cold, the fresh food section just right. How would you adjust the controls so as to make the freezer warmer and keep the fresh food the same? (Photograph by the author.)

ture of the fresh food compartment. And there were two controls, one labeled “freezer,” the other “refrigerator.” What’s the problem?

Oh, perhaps I’d better warn you. The two controls are not independent. The freezer control also affects the fresh food temperature, and the fresh food control also affects the freezer. Moreover, the manual warns that one should “always allow twenty-four (24) hours for the temperature to stabilize whether setting the controls for the first time or making an adjustment.”

It was extremely difficult to regulate the temperature of my old refrigerator. Why? Because the controls suggest a false conceptual model. Two compartments, two controls, which implies that each control is responsible for the temperature of the compartment that carries its name: this conceptual model is shown in Figure 1.10A. It is wrong. In fact, there is only one thermostat and only one cooling mechanism. One control adjusts the thermostat setting, the other the relative proportion of cold air sent to each of the two compartments of the refrigerator. This is why the two controls interact: this conceptual model is shown in Figure 1.10B. In addition, there must be a temperature sensor, but there is no way of knowing where it is located. With the conceptual model suggested by the controls,

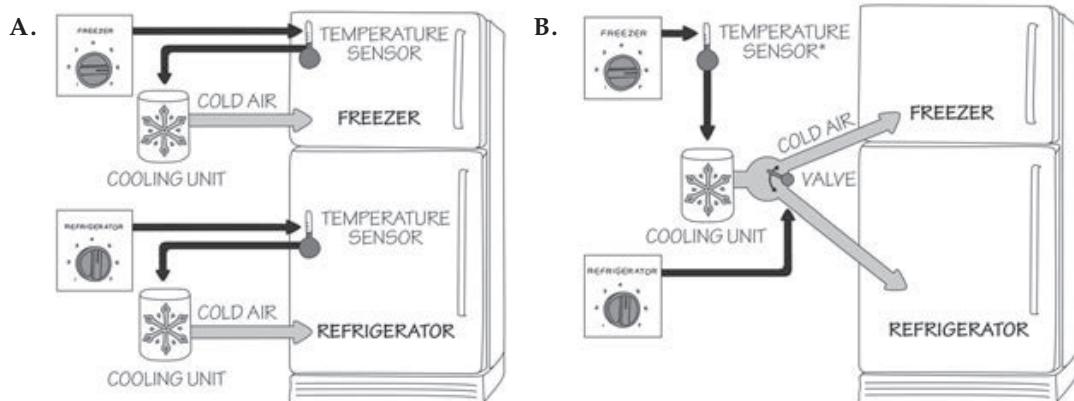


FIGURE 1.10. Two Conceptual Models for a Refrigerator. The conceptual model A is provided by the system image of the refrigerator as gleaned from the controls. Each control determines the temperature of the named part of the refrigerator. This means that each compartment has its own temperature sensor and cooling unit. This is wrong. The correct conceptual model is shown in B. There is no way of knowing where the temperature sensor is located so it is shown outside the refrigerator. The freezer control determines the freezer temperature (so is this where the sensor is located?). The refrigerator control determines how much of the cold air goes to the freezer and how much to the refrigerator.

adjusting the temperatures is almost impossible and always frustrating. Given the correct model, life would be much easier.

Why did the manufacturer suggest the wrong conceptual model? We will never know. In the twenty-five years since the publication of the first edition of this book, I have had many letters from people thanking me for explaining their confusing refrigerator, but never any communication from the manufacturer (General Electric). Perhaps the designers thought the correct model was too complex, that the model they were giving was easier to understand. But with the wrong conceptual model, it was impossible to set the controls. And even though I am convinced I knew the correct model, I still couldn't accurately adjust the temperatures because the refrigerator design made it impossible to discover which control was for the temperature sensor, which for the relative proportion of cold air, and in which compartment the sensor was located. The lack of immediate feedback for the actions did not help: it took twenty-four hours to see whether the new setting was appropriate. I shouldn't have to keep a laboratory notebook and do controlled experiments just to set the temperature of my refrigerator.

I am happy to say that I no longer own that refrigerator. Instead I have one that has two separate controls, one in the fresh food compartment, one in the freezer compartment. Each control is nicely calibrated in degrees and labeled with the name of the compartment it controls. The two compartments are independent: setting the temperature in one has no effect on the temperature in the other. This solution, although ideal, does cost more. But far less expensive solutions are possible. With today's inexpensive sensors and motors, it should be possible to have a single cooling unit with a motor-controlled valve controlling the relative proportion of cold air diverted to each compartment. A simple, inexpensive computer chip could regulate the cooling unit and valve position so that the temperatures in the two compartments match their targets. A bit more work for the engineering design team? Yes, but the results would be worth it. Alas, General Electric is still selling refrigerators with the very same controls and mechanisms that cause so much

confusion. The photograph in Figure 1.9 is from a contemporary refrigerator, photographed in a store while preparing this book.

The System Image

People create mental models of themselves, others, the environment, and the things with which they interact. These are conceptual models formed through experience, training, and instruction. These models serve as guides to help achieve our goals and in understanding the world.

How do we form an appropriate conceptual model for the devices we interact with? We cannot talk to the designer, so we rely upon whatever information is available to us: what the device looks like, what we know from using similar things in the past, what was told to us in the sales literature, by salespeople and advertisements, by articles we may have read, by the product website and instruction manuals. I call the combined information available to us the *system image*. When the system image is incoherent or inappropriate, as in the case of the refrigerator, then the user cannot easily use the device. If it is incomplete or contradictory, there will be trouble.

As illustrated in Figure 1.11, the designer of the product and the person using the product form somewhat disconnected vertices of a triangle. The designer's conceptual model is the designer's conception of the product, occupying one vertex of the triangle. The product itself is no longer with the designer, so it is isolated as a second vertex, perhaps sitting on the user's kitchen counter. The system image is what can be perceived from the physical structure that has been built (including documentation, instructions, signifiers, and any information available from websites and help lines). The user's conceptual model comes from the system image, through interaction with the product, reading, searching for online information, and from whatever manuals are provided. The designer expects the user's model to be identical to the design model, but because designers cannot communicate directly with users, the entire burden of communication is on the system image.

FIGURE 1.11. The Designer's Model, the User's Model, and the System Image. The designer's conceptual model is the designer's conception of the look, feel, and operation of a product. The system image is what can be derived from the physical structure that has been built (including documentation). The user's mental model is developed through interaction with the product and the system image. Designers expect the user's model to be identical to their own, but because they cannot communicate directly with the user, the burden of communication is with the system image.



Figure 1.11 indicates why communication is such an important aspect of good design. No matter how brilliant the product, if people cannot use it, it will receive poor reviews. It is up to the designer to provide the appropriate information to make the product understandable and usable. Most important is the provision of a good conceptual model that guides the user when things go wrong. With a good conceptual model, people can figure out what has happened and correct the things that went wrong. Without a good model, they struggle, often making matters worse.

Good conceptual models are the key to understandable, enjoyable products: good communication is the key to good conceptual models.

The Paradox of Technology

Technology offers the potential to make life easier and more enjoyable; each new technology provides increased benefits. At the same time, added complexities increase our difficulty and frustration with technology. The design problem posed by technological advances is enormous. Consider the wristwatch. A few decades ago, watches were simple. All you had to do was set the time and keep the watch wound. The standard control was the stem: a knob at the side of the watch. Turning the knob would wind the spring that provided power to the watch movement. Pulling out the knob and turning it rotated the hands. The operations were easy to learn and easy to do. There was a reasonable relationship between the

turning of the knob and the resulting turning of the hands. The design even took into account human error. In its normal position, turning the stem wound the mainspring of the clock. The stem had to be pulled before it would engage the gears for setting the time. Accidental turns of the stem did no harm.

Watches in olden times were expensive instruments, manufactured by hand. They were sold in jewelry stores. Over time, with the introduction of digital technology, the cost of watches decreased rapidly, while their accuracy and reliability increased. Watches became tools, available in a wide variety of styles and shapes and with an ever-increasing number of functions. Watches were sold everywhere, from local shops to sporting goods stores to electronic stores. Moreover, accurate clocks were incorporated in many appliances, from phones to musical keyboards: many people no longer felt the need to wear a watch. Watches became inexpensive enough that the average person could own multiple watches. They became fashion accessories, where one changed the watch with each change in activity and each change of clothes.

In the modern digital watch, instead of winding the spring, we change the battery, or in the case of a solar-powered watch, ensure that it gets its weekly dose of light. The technology has allowed more functions: the watch can give the day of the week, the month, and the year; it can act as a stopwatch (which itself has several functions), a countdown timer, and an alarm clock (or two); it has the ability to show the time for different time zones; it can act as a counter and even as a calculator. My watch, shown in Figure 1.8, has many functions. It even has a radio receiver to allow it to set its time with official time stations around the world. Even so, it is far less complex than many that are available. Some watches have built-in compasses and barometers, accelerometers, and temperature gauges. Some have GPS and Internet receivers so they can display the weather and news, e-mail messages, and the latest from social networks. Some have built-in cameras. Some work with buttons, knobs, motion, or speech. Some detect gestures. The watch is no longer just an instrument for telling time: it has become a platform for enhancing multiple activities and lifestyles.

The added functions cause problems: How can all these functions fit into a small, wearable size? There are no easy answers. Many people have solved the problem by not using a watch. They use their phone instead. A cell phone performs all the functions much better than the tiny watch, while also displaying the time.

Now imagine a future where instead of the phone replacing the watch, the two will merge, perhaps worn on the wrist, perhaps on the head like glasses, complete with display screen. The phone, watch, and components of a computer will all form one unit. We will have flexible displays that show only a tiny amount of information in their normal state, but that can unroll to considerable size. Projectors will be so small and light that they can be built into watches or phones (or perhaps rings and other jewelry), projecting their images onto any convenient surface. Or perhaps our devices won't have displays, but will quietly whisper the results into our ears, or simply use whatever display happens to be available: the display in the seatback of cars or airplanes, hotel room televisions, whatever is nearby. The devices will be able to do many useful things, but I fear they will also frustrate: so many things to control, so little space for controls or signifiers. The obvious solution is to use exotic gestures or spoken commands, but how will we learn, and then remember, them? As I discuss later, the best solution is for there to be agreed upon standards, so we need learn the controls only once. But as I also discuss, agreeing upon these is a complex process, with many competing forces hindering rapid resolution. We will see.

The same technology that simplifies life by providing more functions in each device also complicates life by making the device harder to learn, harder to use. This is the paradox of technology and the challenge for the designer.

The Design Challenge

Design requires the cooperative efforts of multiple disciplines. The number of different disciplines required to produce a successful product is staggering. Great design requires great designers, but that isn't enough: it also requires great management, because the

hardest part of producing a product is coordinating all the many, separate disciplines, each with different goals and priorities. Each discipline has a different perspective of the relative importance of the many factors that make up a product. One discipline argues that it must be usable and understandable, another that it must be attractive, yet another that it has to be affordable. Moreover, the device has to be reliable, be able to be manufactured and serviced. It must be distinguishable from competing products and superior in critical dimensions such as price, reliability, appearance, and the functions it provides. Finally, people have to actually purchase it. It doesn't matter how good a product is if, in the end, nobody uses it.

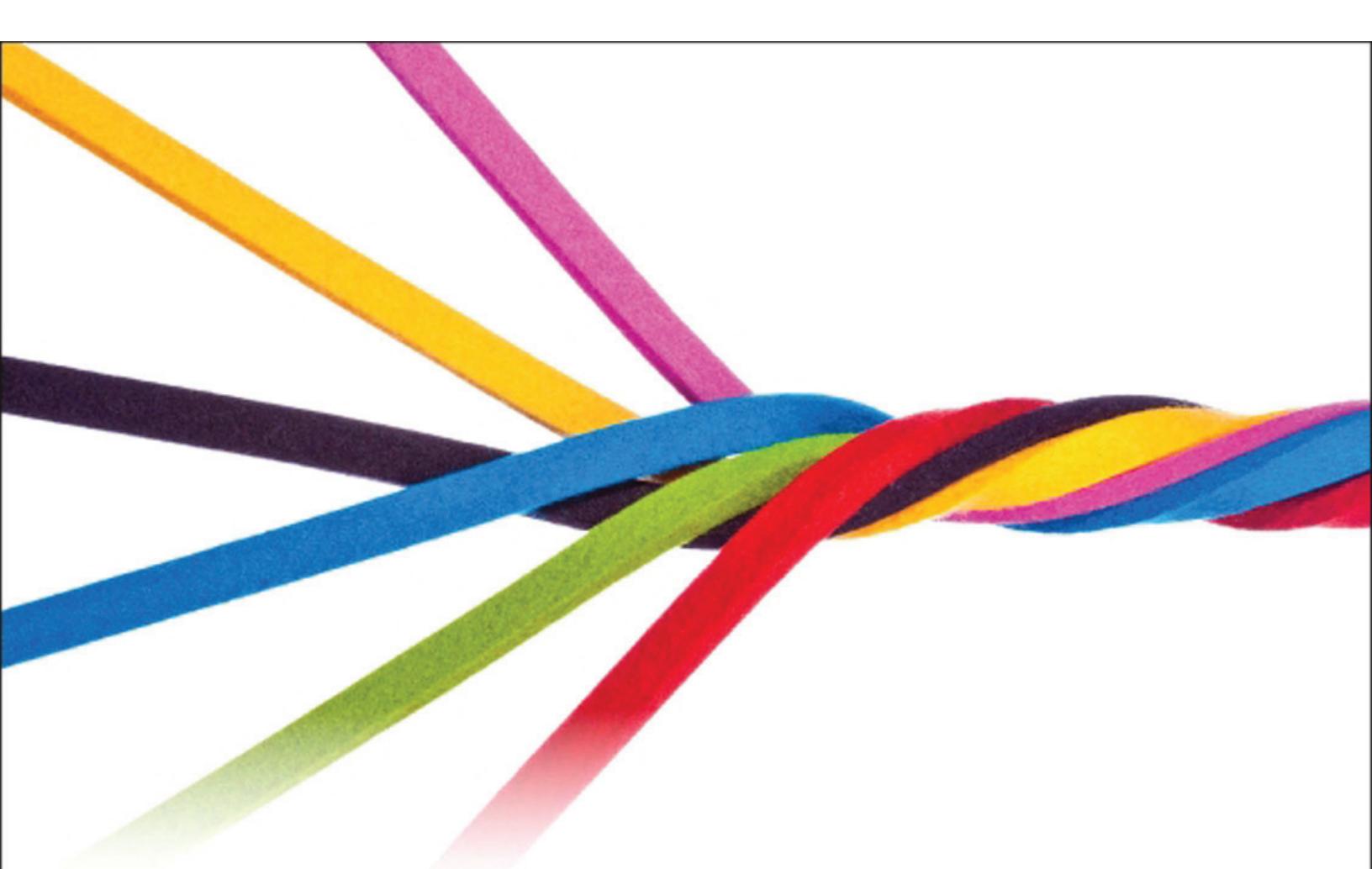
Quite often each discipline believes its distinct contribution to be most important: "Price," argues the marketing representative, "price plus these features." "Reliable," insist the engineers. "We have to be able to manufacture it in our existing plants," say the manufacturing representatives. "We keep getting service calls," say the support people; "we need to solve those problems in the design." "You can't put all that together and still have a reasonable product," says the design team. Who is right? Everyone is right. The successful product has to satisfy all these requirements.

The hard part is to convince people to understand the viewpoints of the others, to abandon their disciplinary viewpoint and to think of the design from the viewpoints of the person who buys the product and those who use it, often different people. The viewpoint of the business is also important, because it does not matter how wonderful the product is if not enough people buy it. If a product does not sell, the company must often stop producing it, even if it is a great product. Few companies can sustain the huge cost of keeping an unprofitable product alive long enough for its sales to reach profitability—with new products, this period is usually measured in years, and sometimes, as with the adoption of high-definition television, decades.

Designing well is not easy. The manufacturer wants something that can be produced economically. The store wants something that will be attractive to its customers. The purchaser has several

demands. In the store, the purchaser focuses on price and appearance, and perhaps on prestige value. At home, the same person will pay more attention to functionality and usability. The repair service cares about maintainability: how easy is the device to take apart, diagnose, and service? The needs of those concerned are different and often conflict. Nonetheless, if the design team has representatives from all the constituencies present at the same time, it is often possible to reach satisfactory solutions for all the needs. It is when the disciplines operate independently of one another that major clashes and deficiencies occur. The challenge is to use the principles of human-centered design to produce positive results, products that enhance lives and add to our pleasure and enjoyment. The goal is to produce a great product, one that is successful, and that customers love. It can be done.





Human-Computer Interaction

An Empirical Research Perspective



I. Scott MacKenzie

Human-Computer Interaction

An Empirical Research Perspective

I. Scott MacKenzie



ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier



Historical Context

1

Human-computer interaction. In the beginning, there were humans. In the 1940s came computers. Then in the 1980s came interaction. Wait! What happened between 1940 and 1980? Were humans not *interacting* with computers then? Well, yes, but not just any human. Computers in those days were too precious, too complicated, to allow the average human to mess with them. Computers were carefully guarded. They lived a secluded life in large air-conditioned rooms with raised floors and locked doors in corporate or university research labs or government facilities. The rooms often had glass walls to show off the unique status of the behemoths within.

If you were of that breed of human who was permitted access, you were probably an engineer or a scientist—specifically, a computer scientist. And you knew what to do. Whether it was connecting relays with patch cords on an ENIAC (1940s), changing a magnetic memory drum on a UNIVAC (1950s), adjusting the JCL stack on a System/360 (1960s), or *grep*ing and *awking* around the *unix* command set on a PDP-11 (1970s), you were on home turf. *Unix* commands like *grep*, for *global regular expression print*, were obvious enough. Why consult the manual? You probably wrote it! As for *unix*'s *vi* editor, if some poor soul was stupid enough to start typing text while in command mode, well, he got what he deserved.¹ Who gave him a login account, anyway? And what's all this talk about *make the state of the system visible to the user*? What user? Sounds a bit like ... well ... socialism!

Interaction was not on the minds of the engineers and scientists who designed, built, configured, and programmed the early computers. But by the 1980s interaction was an issue. The new computers were not only powerful, they were useable—by anyone! With usability added, computers moved from their earlier secure confines onto people's desks in workplaces and, more important, into people's homes. One reason human-computer interaction (HCI) is so exciting is that the field's emergence and progress are aligned with, and in good measure responsible for, this dramatic shift in computing practices.

¹One of the classic UI foibles—told and re-told by HCI educators around the world—is the *vi* editor's lack of feedback when switching between modes. Many a user made the mistake of providing input while in *command mode* or entering a command while in *input mode*.

This book is about research in human-computer interaction. As in all fields, research in HCI is the force underlying advances that migrate into products and processes that people use, whether for work or pleasure. While HCI itself is broad and includes a substantial applied component—most notably in design—the focus in this book is narrow. The focus is on research—the what, the why, and the how—with a few stories to tell along the way.

Many people associate research in HCI with developing a new or improved interaction or interface and testing it in a user study. The term “user study” sometimes refers to an informal evaluation of a user interface. But this book takes a more formal approach, where a user study is “an experiment with human participants.” HCI experiments are discussed throughout the book. The word *empirical* is added to this book’s title to give weight to the value of experimental research. The research espoused here is empirical because it is based on observation and experience and is carried out and reported on in a manner that allows results to be verified or refuted through the efforts of other researchers. In this way, each item of HCI research joins a large body of work that, taken as a whole, defines the field and sets the context for applying HCI knowledge in real products or processes.

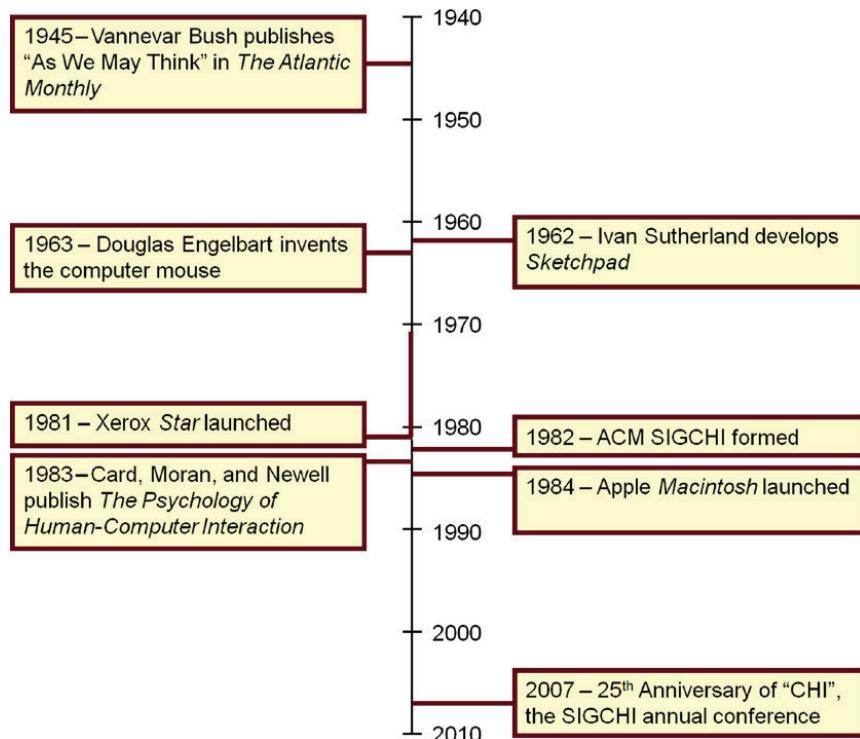
1.1 Introduction

Although HCI emerged in the 1980s, it owes a lot to older disciplines. The most central of these is the field of *human factors*, or *ergonomics*. Indeed, the name of the preeminent annual conference in HCI—the Association for Computing Machinery Conference on Human Factors in Computing Systems (ACM SIGCHI)—uses that term. SIGCHI is the special interest group on computer-human interaction sponsored by the ACM.²

Human factors is both a science and a field of engineering. It is concerned with human capabilities, limitations, and performance, and with the design of systems that are efficient, safe, comfortable, and even enjoyable for the humans who use them. It is also an art in the sense of respecting and promoting creative ways for practitioners to apply their skills in designing systems. One need only change *systems* in that statement to *computer systems* to make the leap from human factors to HCI. HCI, then, is human factors, but narrowly focused on human interaction with computing technology of some sort.

That said, HCI itself does not feel “narrowly focused.” On the contrary, HCI is tremendously broad in scope. It draws upon interests and expertise in disciplines such as psychology (particularly cognitive psychology and experimental psychology), sociology, anthropology, cognitive science, computer science, and linguistics.

²The Association of Computing Machinery (ACM), founded in 1947, is the world’s leading educational and scientific computing society, with over 95,000 members. The ACM is organized into over 150 special interest groups, or “SIGs.” Among the services offered is the ACM Digital Library, a repository of online publications which includes 45+ ACM journals, 85+ ACM conference proceedings, and numerous other publications from affiliated organizations. See www.acm.org.

**FIGURE 1.1**

Timeline of notable events in the history of human–computer interaction HCI.

Figure 1.1 presents a timeline of a few notable events leading to the birth and emergence of HCI as a field of study, beginning in the 1940s.

1.2 Vannevar Bush's "as we may think" (1945)

Vannevar Bush's prophetic essay "As We May Think," published in the *Atlantic Monthly* in July, 1945 (Bush, 1945), is required reading in many HCI courses even today. The article has garnered 4,000+ citations in scholarly publications.³ Attesting to the importance of Bush's vision to HCI is the 1996 reprint of the entire essay in the ACM's *interactions* magazine, complete with annotations, sketches, and biographical notes.

Bush (see Figure 1.2) was the U.S. government's Director of the Office of Scientific Research and a scientific advisor to President Franklin D. Roosevelt. During World War II, he was charged with leading some 6,000 American scientists in the application of science to warfare. But Bush was keenly aware of the possibilities that lay ahead in peacetime in applying science to more lofty and humane

³Google Scholar search using *author: "v bush."*

**FIGURE 1.2**

Vannevar Bush at work (circa 1940–1944).

pursuits. His essay concerned the dissemination, storage, and access to scholarly knowledge. Bush wrote:

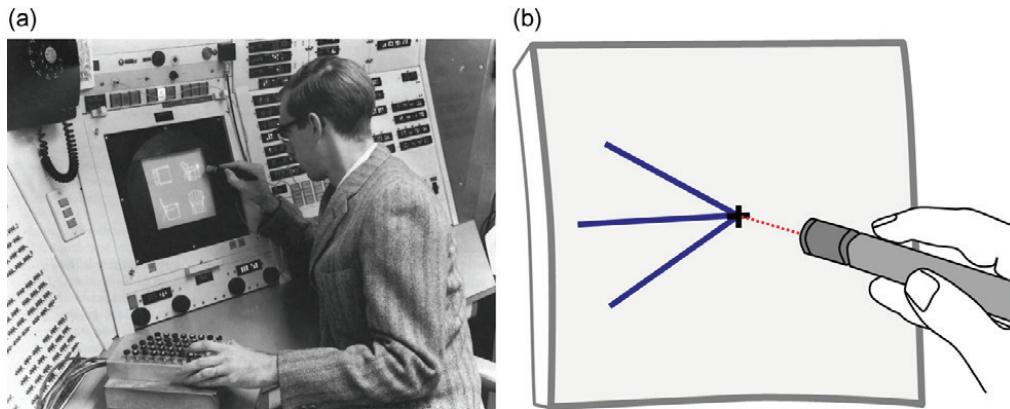
the summation of human experience is being expanded at a prodigious rate, and the means we use for threading through the consequent maze to the momentarily important item is the same as was used in the days of square-rigged ships (p. 37).⁴

Aside from the reference to antiquated square-rigged ships, what Bush says we can fully relate to today, especially his mention of the *expanding human experience* in relation to HCI. For most people, nothing short of Olympian talent is needed to keep abreast of the latest advances in the information age. Bush's *consequent maze* is today's *information overload* or *lost in hyperspace*. Bush's *momentarily important item* sounds a bit like a blog posting or a tweet. Although blogs and tweets didn't exist in 1945, Bush clearly anticipated them.

Bush proposed navigating the knowledge maze with a device he called *memex*. Among the features of memex is *associative indexing*, whereby points of interest can be connected and joined so that selecting one item immediately and automatically selects another: "When the user is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard" (Bush, 1945, p. 44). This sounds like a description of hyperlinks and bookmarks. Although today it is easy to equate memex with hypertext and the World Wide Web, Bush's inspiration for this idea came from the contemporary telephone exchange, which he described as a "spider web of metal, sealed in a thin glass container" (viz. vacuum tubes) (p. 38). The maze of connections in a telephone exchange gave rise to Bush's more general theme of a spider web of connections for the information in one's mind, linking one's experiences.

It is not surprising that some of Bush's ideas, for instance, *dry photography*, today seem naïve. Yet the ideas are naïve only when juxtaposed with Bush's

⁴For convenience, page references are to the March 1996 reprint in the ACM's *interactions*.

**FIGURE 1.3**

(a) Demo of Ivan Sutherland's Sketchpad. (b) A light pen dragging ("rubber banding") lines, subject to constraints.

brilliant foretelling of a world we are still struggling with and are still fine-tuning and perfecting.

1.3 Ivan Sutherland's Sketchpad (1962)

Ivan Sutherland developed Sketchpad in the early 1960s as part of his PhD research in electrical engineering at the Massachusetts Institute of Technology (M.I.T.). Sketchpad was a graphics system that supported the manipulation of geometric shapes and lines (*objects*) on a display using a light pen. To appreciate the inferior usability in the computers available to Sutherland at the time of his studies, consider these introductory comments in a paper he published in 1963:

Heretofore, most interaction between man and computers has been slowed by the need to reduce all communication to written statements that can be typed. In the past we have been writing letters to, rather than conferring with, our computers (Sutherland, 1963, p. 329).

With Sketchpad, commands were not typed. Users did not "write letters to" the computer. Instead, objects were drawn, resized, grabbed and moved, extended, deleted—directly, using the light pen (see Figure 1.3). Object manipulations worked with constraints to maintain the geometric relationships and properties of objects.

The use of a pointing device for input makes Sketchpad the first *direct manipulation* interface—a sign of things to come. The term "direct manipulation" was coined many years later by Ben Shneiderman at the University of Maryland to provide a psychological context for a suite of related features that naturally came together in this new genre of human-computer interface (Shneiderman, 1983). These features included visibility of objects, incremental action, rapid feedback, reversibility, exploration, syntactic correctness of all actions, and replacing language with action. While Sutherland's Sketchpad was one of the earliest examples

of a direct manipulation system, others soon followed, most notably the *Dynabook* concept system by Alan Kay of the Xerox Palo Alto Research Center (PARC) (Kay and Goldberg, 1977). I will say more about Xerox PARC throughout this chapter.

Sutherland's work was presented at the Institute of Electrical and Electronics Engineers (IEEE) conference in Detroit in 1963 and subsequently published in its proceedings (Sutherland, 1963). The article is available in the ACM Digital Library (<http://portal.acm.org>). Demo videos of Sketchpad are available on YouTube (www.youtube.com). Not surprisingly, a user study of Sketchpad was not conducted, since Sutherland was a student of electrical engineering. Had his work taken place in the field of industrial engineering (where human factors is studied), user testing would have been more likely.

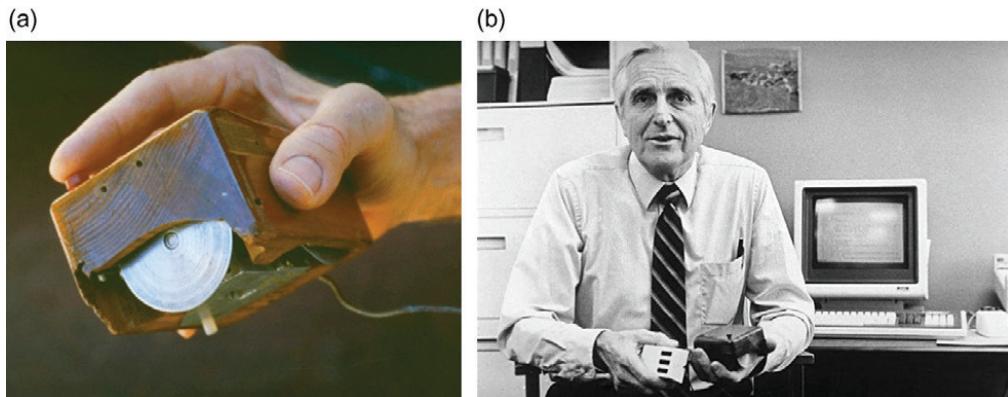
1.4 Invention of the mouse (1963)

If there is one device that symbolizes the emergence of HCI, it is the computer mouse. Invented by Douglas Engelbart in 1963, the mouse was destined to fundamentally change the way humans interact with computers.⁵ Instead of typing commands, a user could manipulate a mouse to control an on-screen tracking symbol, or cursor. With the cursor positioned over a graphic image representing the command, the command is issued with a select operation—pressing and releasing a button on the mouse.

Engelbart was among a group of researchers at the Stanford Research Institute (SRI) in Menlo Park, California. An early hypertext system called NLS, for oN-Line System, was the project for which an improved pointing device was needed. Specifically, the light pen needed to be replaced. The light pen was an established technology, but it was awkward. The user held the pen in the air in front of the display. After a few minutes of interaction, fatigue would set in. A more natural and comfortable device might be something on the desktop, something in close proximity to the keyboard. The keyboard is where the user's hands are normally situated, so a device beside the keyboard made the most sense. Engelbart's invention met this requirement.

The first prototype mouse is seen in Figure 1.4a. The device included two potentiometers positioned at right angles to each other. Large metal wheels were attached to the shafts of the potentiometers and protruded slightly from the base of the housing. The wheels rotated as the device was moved across a surface. Side-to-side motion rotated one wheel; to-and-fro motion rotated the other. With diagonal movement, both wheels rotated, in accordance with the amount of movement in each direction. The amount of rotation of each wheel altered the voltage at the wiper terminal of the potentiometer. The voltages were passed on to the host system for processing. The *x* and *y* positions of an on-screen object or cursor were indirectly

⁵Engelbart's patent for the mouse was filed on June 21, 1967 and issued on November 17, 1970 (Engelbart, 1970). U.S. patent laws allow one year between public disclosure and filing; thus, it can be assumed that prior to June 21, 1966, Engelbart's invention was not disclosed to the public.

**FIGURE 1.4**

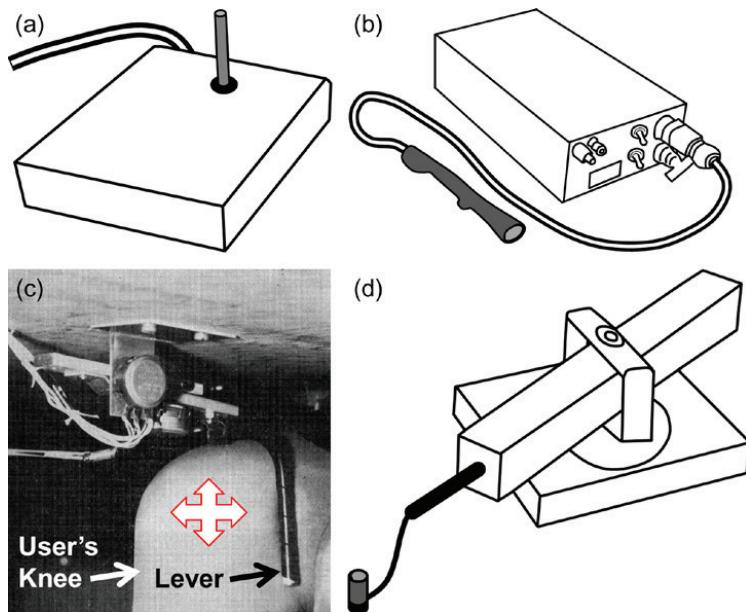
(a) The first mouse. (b) Inventor Douglas Engelbart holding his invention in his left hand and an early three-button variation in his right hand.

controlled by the two voltage signals. In Figure 1.4a, a selection button can be seen under the user's index finger. In Figure 1.4b, Engelbart is shown with his invention in his left hand and a three-button version of a mouse, which was developed much later, in his right.

Initial testing of the mouse focused on selecting and manipulating text, rather than drawing and manipulating graphic objects. Engelbart was second author of the first published evaluation of the mouse. This was, arguably, HCI's first user study, so a few words are in order here. Engelbart, along with English and Berman conducted a controlled experiment comparing several input devices capable of both selection and *x-y* position control of an on-screen cursor (English, Engelbart, and Berman, 1967). Besides the mouse, the comparison included a *light pen*, a *joystick*, a *knee-controlled lever*, and a *Grafacan*. The joystick (Figure 1.5a) had a moving stick and was operated in two control modes. In absolute or position-control mode, the cursor's position on the display had an absolute correspondence to the position of the stick. In rate-control mode, the cursor's velocity was determined by the amount of stick deflection, while the direction of the cursor's motion was determined by the direction of the stick. An embedded switch was included for selection and was activated by pressing down on the stick.

The light pen (Figure 1.5b) was operated much like the pen used by Sutherland (see Figure 1.3). The device was picked up and moved to the display surface with the pen pointing at the desired object. A projected circle of orange light indicated the target to the lens system. Selection involved pressing a switch on the barrel of the pen.

The knee-controlled lever (Figure 1.5c) was connected to two potentiometers. Side-to-side knee motion controlled side-to-side (*x*-axis) cursor movement; up-and-down knee motion controlled up-and-down (*y*-axis) cursor movement. Up-and-down knee motion was achieved by a "rocking motion on the ball of the foot" (p. 7). The device did not include an integrated method for selection. Instead, a key on the system's keyboard was used.

**FIGURE 1.5**

Additional devices used in the first comparative evaluation of a mouse: (a) Joystick.

(b) Lightpen. (c) Knee-controlled lever. (d) Grafacon.

(Source: a, b, d, adapted from English et al., 1967; c, 1967 IEEE. Reprinted with permission)

The Grafacon (Figure 1.5d) was a commercial device used for tracing curves. As noted, the device consisted “of an extensible arm connected to a linear potentiometer, with the housing for the linear potentiometer pivoted on an angular potentiometer” (1967, 6). Originally, there was a pen at the end of the arm; however, this was replaced with a knob-and-switch assembly (see Figure 1.5). The user gripped the knob and moved it about to control the on-screen cursor. Pressing the knob caused a selection.

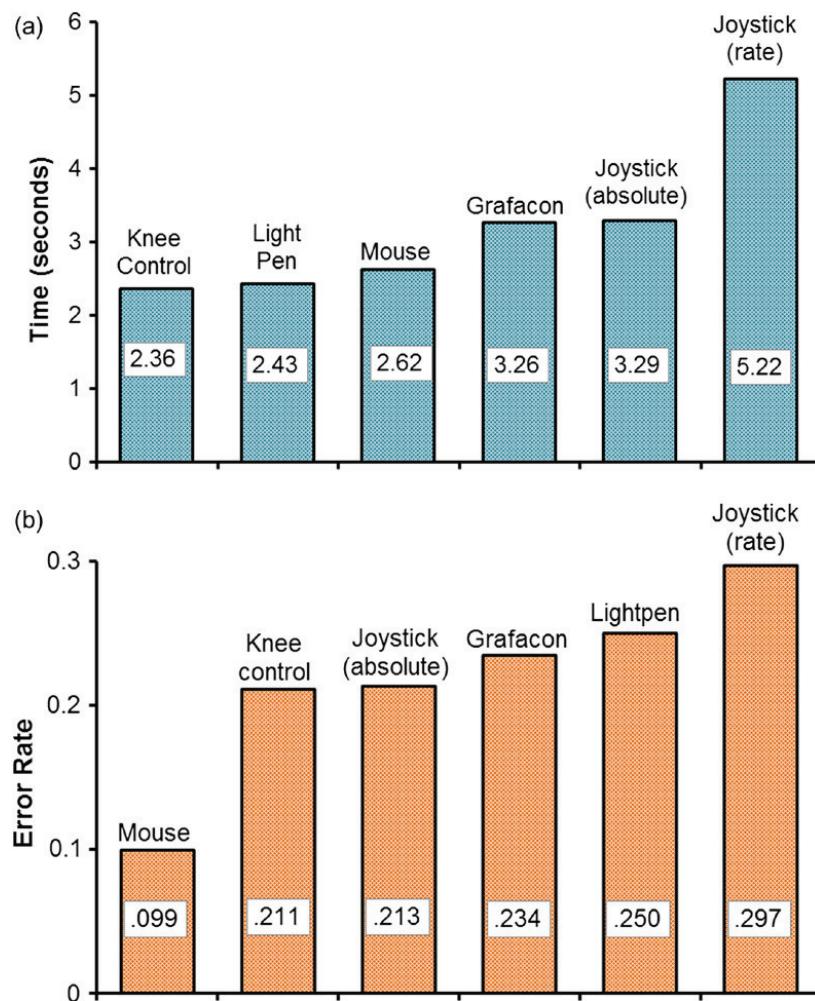
The knee-controlled lever and Grafacon are interesting alternatives to the mouse. They illustrate and suggest the processes involved in empirical research. It is not likely that Engelbart simply woke up one morning and invented the mouse. While it may be true that novel ideas sometimes arise through “eureka” moments, typically there is more to the process of invention. Refining ideas—deciding what works and what doesn’t—is an iterative process that involves a good deal of trial and error. No doubt, Engelbart and colleagues knew from the outset that they needed a device that would involve some form of human action as input and would produce two channels (x - y) of analog positioning data as output. A select operation was also needed to produce a command or generate closure at the end of a positioning operation. Of course, we know this today as a *point-select*, or *point-and-click*, operation. Operating the device away from the display meant some form of on-screen tracker (a *cursor*) was needed to establish correspondence between the *device space* and the *display space*. While this seems obvious today, it was a newly emerging form of human-to-computer interaction in the 1960s.

In the comparative evaluation, English et al. (1967) measured users' *access time* (the time to move the hand from the keyboard to the device) and *motion time* (the time from the onset of cursor movement to the final selection). The evaluation included 13 participants (eight experienced in working with the devices and three inexperienced). For each trial, a character target (with surrounding distracter targets) appeared on the display. The trial began with the participant pressing and releasing the spacebar on the system's keyboard, whereupon a cursor appeared on the display. The participant moved his or her hand to the input device and then manipulated the device to move the cursor to the target. With the cursor over the target, a selection was made using the method associated with the device. Examples of the test results from the inexperienced participants are shown in Figure 1.6. Each bar represents the mean for ten sequences. Every sequence consists of eight target-patterns. Results are shown for the mean task completion time (Figure 1.6a) and error rate (Figure 1.6b), where the error rate is the ratio of missed target selections to all selections.

While it might appear that the knee-controlled lever is the best device in terms of time, each bar in Figure 1.6a includes both the access time and the motion time. The access time for the knee-controlled lever is, of course, zero. The authors noted that considering motion time only, the knee-controlled lever "no longer shows up so favorably" (p. 12). At 2.43 seconds per trial, the light pen had a slight advantage over the mouse at 2.62 seconds per trial; however, this must be viewed with consideration for the inevitable discomfort in continued use of a light pen, which is operated in the air at the surface of the display. Besides, the mouse was the clear winner in terms of accuracy. The mouse error rate was less than half that of any other device condition in the evaluation (see Figure 1.6b).

The mouse evaluation by English et al. (1967) marks an important milestone in empirical research in HCI. The methodology was empirical and the write-up included most of what is expected today in a user study destined for presentation at a conference and publication in a conference proceedings. For example, the write-up contained a detailed description of the participants, the apparatus, and the procedure. The study could be reproduced if other researchers wished to verify or refute the findings. Of course, reproducing the evaluation today would be difficult, as the devices are no longer available. The evaluation included an independent variable, input method, with six levels: mouse, light pen, joystick (position-control), joystick (rate-control), and knee-controlled lever. There were two dependent variables, task completion time and error rate. The order of administering the device conditions was different for each participant, a practice known today as counterbalancing. While testing for statistically significant differences using an analysis of variance (ANOVA) was not done, it is important to remember that the authors did not have at their disposal the many tools taken for granted today, such as spreadsheets and statistics applications.

The next published comparative evaluation involving a mouse was by Card, English, and Burr (1978), about 10 years later. Card et al.'s work was carried out at Xerox PARC and was part of a larger effort that eventually produced the first windows-based graphical user interface, or GUI (see next section). The mouse

**FIGURE 1.6**

Results of first comparative evaluation of a computer mouse: (a) Task completion time in seconds. (b) Error rate as the ratio of missed selections to all selections.

(Adapted from English et al., 1967)

underwent considerable refining and reengineering at PARC. Most notably, the potentiometer wheels were replaced with a rolling ball assembly, developed by Rider (1974). The advantage of the refined mouse over competing devices was reconfirmed by Card et al. (1978) and has been demonstrated in countless comparative evaluations since and throughout the history of HCI. It was becoming clear that Engelbart's invention was changing the face of human-computer interaction.

Years later, Engelbart would receive the ACM Turing Award (1997) and the ACM SIGCHI Lifetime Achievement Award (1998; 1st recipient). It is interesting that Engelbart's seminal invention dates to the early 1960s, yet commercialization of the mouse did not occur until 1981, when the Xerox Star was launched.

1.5 Xerox star (1981)

There was a buzz around the floor of the National Computer Conference (NCC) in May 1981. In those days, the NCC was *the* yearly conference for computing. It was both a gathering of researchers (sponsored by the American Federation of Information Processing Societies, or AFIPS) and a trade show. The trade show was huge.⁶ All the players were there. There were big players, like IBM, and little players, like Qupro Data Systems of Kitchener, Ontario, Canada. I was there, “working the booth” for Qupro. Our main product was a small desktop computer system based on a single-board computer known as the *Pascal MicroEngine*.

The buzz at the NCC wasn’t about Qupro. It wasn’t about IBM, either. The buzz was about Xerox. “Have you been to the Xerox booth?” I would hear. “You gotta check it out. It’s really cool.” And indeed it was. The Xerox booth had a substantial crowd gathered around it throughout the duration of the conference. There were scripted demonstrations every hour or so, and the crowd was clearly excited by what they were seeing. The demos were of the Star, or the *Xerox 8100 Star Information System*, as it was formally named. The excitement was well deserved, as the 1981 launch of the Xerox Star at the NCC marks a watershed moment in the history of computing. The Star was the first commercially released computer system with a GUI. It had windows, icons, menus, and a pointing device (WIMP). It supported direct manipulation and what-you-see-is-what-you-get (WYSIWYG) interaction. The Star had what was needed to bring computing to the people.

The story of the Star began around 1970, when Xerox established its research center, PARC, in Palo Alto, California. The following year, Xerox signed an agreement with SRI licensing Xerox to use Engelbart’s invention, the mouse (Johnson et al., 1989, p. 22). Over the next 10 years, development proceeded along a number of fronts. The most relevant development for this discussion is that of the *Alto*, the Star’s predecessor, which began in 1973. The Alto also included a GUI and mouse. It was used widely at Xerox and at a few external test sites. However, the Alto was never released commercially—a missed opportunity on a grand scale, according to some (D. K. Smith and Alexander, 1988).

Figure 1.7 shows the Star workstation, which is unremarkable by today’s standards. The graphical nature of the information on the system’s display can be seen in the image. This was novel at the time. The display was bit-mapped, meaning images were formed by mapping bits in memory to pixels on the display. Most systems at the time used character-mapped displays, meaning the screen image was composed of sequences of characters, each limited to a fixed pattern (e.g., 7×10 pixels) retrieved from read-only memory. Character-mapped displays required considerably less memory, but limited the richness of the display image. The mouse—a two-button variety—is featured by the system’s keyboard.

⁶Attendance figures for 1981 are unavailable, but the NCC was truly huge. In 1983, NCC attendance exceeded 100,000 (Abrahams, 1987).

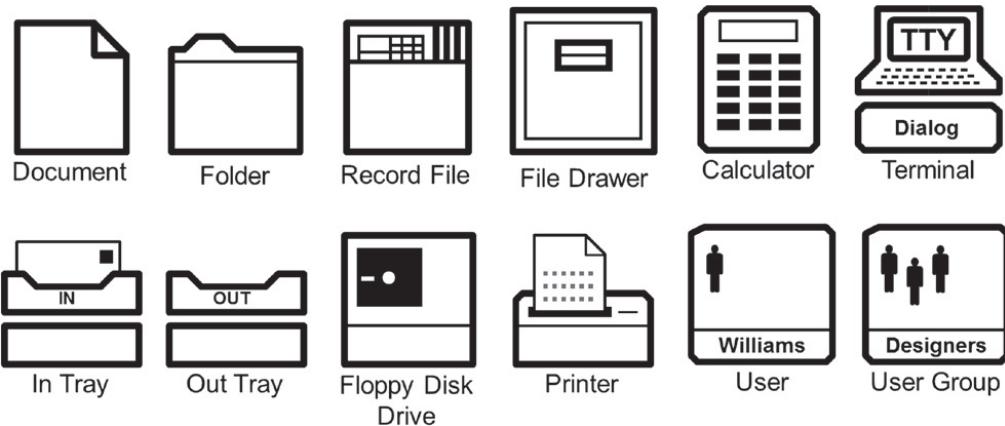
**FIGURE 1.7**

Xerox Star workstation.

As the designers noted, the Star was intended as an *office automation system* (Johnson et al., 1989). Business professionals would have Star workstations on their desks and would use them to create, modify, and manage documents, graphics tables, presentations, etc. The workstations were connected via high-speed Ethernet cables and shared centralized resources, such as printers and file servers. A key tenet in the Star philosophy was that workers wanted to get their work done, not fiddle with computers. Obviously, the computers had to be easy to use, or *invisible*, so to speak.

One novel feature of the Star was use of the *desktop metaphor*. Metaphors are important in HCI. When a metaphor is present, the user has a jump-start on knowing what to do. The user exploits existing knowledge from another domain. The desktop metaphor brings concepts from the office desktop to the system's display. On the display the user finds pictorial representations (*icons*) for things like documents, folders, trays, and accessories such as a calculator, printer, or notepad. A few examples of the Star's icons are seen in Figure 1.8. By using existing knowledge of a desktop, the user has an immediate sense of what to do and how things work. The Star designers, and others since, pushed the limits of the metaphor to the point where it is now more like an office metaphor than a desktop metaphor. There are windows, printers, and a trashcan on the display, but of course these artifacts are not found on an office desktop. However, the metaphor seemed to work, as we hear even today that the GUI is an example of the desktop metaphor. I will say more about metaphors again in Chapter 3.

In making the system usable (invisible), the Star developers created interactions that deal with files, not programs. So users “open a document,” rather than “invoke an editor.” This means that files are associated with applications, but these details are hidden from the user. Opening a spreadsheet document launches the spreadsheet application, while opening a text document opens a text editor.

**FIGURE 1.8**

Examples of icons appearing on the Xerox Star desktop.

(Adapted from Smith, Irby, Kimball, and Harslem, 1982)

With a GUI and point-select interaction, the Star interface was the archetype of direct manipulation. The enabling work on graphical interaction (e.g., Sutherland) and pointing devices (e.g., Engelbart) was complete. By comparison, previous command-line interfaces had a single channel of input. For every action, a command was needed to invoke it. The user had to learn and remember the syntax of the system's commands and type them in to get things done. Direct manipulation systems, like the Star, have numerous input channels, and each channel has a direct correspondence to a task. Furthermore, interaction with the channel is tailored to the properties of the task. A continuous property, such as display brightness or sound volume, has a continuous control, such as a slider. A discrete property, such as font size or family, has a discrete control, such as a multi-position switch or a menu item. Each control also has a dedicated location on the display and is engaged using a direct point-select operation. Johnson et al. (1989, 14) compares direct manipulation to driving a car. A gas pedal controls the speed, a lever controls the wiper blades, a knob controls the radio volume. Each control is a dedicated channel, each has a dedicated location, and each is operated according to the property it controls.

When operating a car, the driver can adjust the radio volume and then turn on the windshield wipers. Or the driver can first turn on the windshield wipers and then adjust the radio volume. The car is capable of responding to the driver's inputs in any order, according to the driver's wishes. In computing, direct manipulation brings the same flexibility. This is no small feat. Command-line interfaces, by comparison, are simple. They follow a software paradigm known as *sequential programming*. Every action occurs in a sequence under the system's control. When the system needs a specific input, the user is prompted to enter it. Direct manipulation interfaces require a different approach because they must accept the user's actions according to the user's wishes. While manipulating *hello* in a text editor, for example, the user might change the font to Courier (*hello*) and then change the style to bold (**hello**). Or the user might first set the style to bold (**hello**) and then

change the font to Courier (**hello**). The result is the same, but the order of actions differs. The point here is that the user is in control, not the system. To support this, direct manipulation systems are designed using a software paradigm known as *event-driven programming*, which is substantially more complicated than sequential programming. Although event-driven programming was not new (it was, and still is, used in process-control to respond to *sensor events*), designing systems that responded asynchronously to *user events* was new in the early 1970s when work began on the Star. Of course, from the user's perspective, this detail is irrelevant (remember the invisible computer). We mention it here only to give credit to the Herculean effort that was invested in designing the Star and bringing it to market.

Designing the Star was not simply a matter of building an interface using windows, icons, menus, and a pointing device (WIMP), it was about designing a system on which these components could exist and work. A team at PARC led by Alan Kay developed such a system beginning around 1970. The central ingredients were a new object-oriented programming language known as Smalltalk and a software architecture known as Model-View-Controller. This was a complex programming environment that evolved in parallel with the design of the Star. It is not surprising, then, that the development of the Star spanned about 10 years, since the designers were not only inventing a new style of human-computer interaction, they were inventing the architecture on which this new style was built.

In the end, the Star was not a commercial success. While many have speculated on why (e.g., D. K. Smith and Alexander, 1988), probably the most significant reason is that the Star was not a personal computer. In the article by the Star interface designers Johnson et al. (1989), there are numerous references to the Star as a *personal computer*. But it seems they had a different view of "personal." They viewed the Star as a beefed-up version of a terminal connected to a central server, "a collection of personal computers" (p. 12). In another article, designers Smith and Irby call the Star "a personal computer designed for office professionals" (1998, 17). "Personal"? Maybe, but without a doubt the Star was, first and foremost, a networked workstation connected to a server and intended for an office environment. And it was expensive: \$16,000 for the workstation alone. That's a distant world from personal computing as we know it today. It was also a distant world from personal computing as it existed in the late 1970s and early 1980s. Yes, even then personal computing was flourishing. The *Apple II*, introduced in 1977 by Apple Computer, was hugely successful. It was the platform on which *VisiCalc*, the first spreadsheet application, was developed. *VisiCalc* eventually sold over 700,000 copies and became known as the first "killer app." Notably, the Star did not have a spreadsheet application, nor could it run any spreadsheet or other application available in the market place. The Star architecture was "closed"—it could only run applications developed by Xerox.

Other popular personal computer systems available around the same time were the *PET*, *VIC-20*, and *Commodore 64*, all by Commodore Business Machines, and the *TRS-80* by Tandy Corp. These systems were truly personal. Most of them were located in people's homes. But the *user interface* was terrible. These systems worked with a traditional command-line interface. The operating system—if you

could call it that—usually consisted of a BASIC-language interpreter and a console prompt. LOAD, SAVE, RUN, EDIT, and a few other commands were about the extent of it. Although these systems were indeed personal, a typical user was a hobbyist, computer enthusiast, or anyone with enough technical skill to connect components together and negotiate the inevitable software and hardware hiccups. But users loved them, and they were cheap. However, they were tricky to use. So while the direct manipulation user interface of the Star may have been intuitive and had the potential to be used by people with no technical skill (or interest in having it!), the system just didn't reach the right audience.

1.6 Birth of HCI (1983)

Nineteen eighty-three is a good year to peg as the birth of HCI. There are at least three key events as markers: the first ACM SIGCHI conference, the publication of Card, Moran, and Newell's *The Psychology of Human-Computer Interaction* (1983), and the arrival of the Apple Macintosh, pre-announced with flyers in December 1983. The *Mac* launch was in January 1984, but I'll include it here anyway.

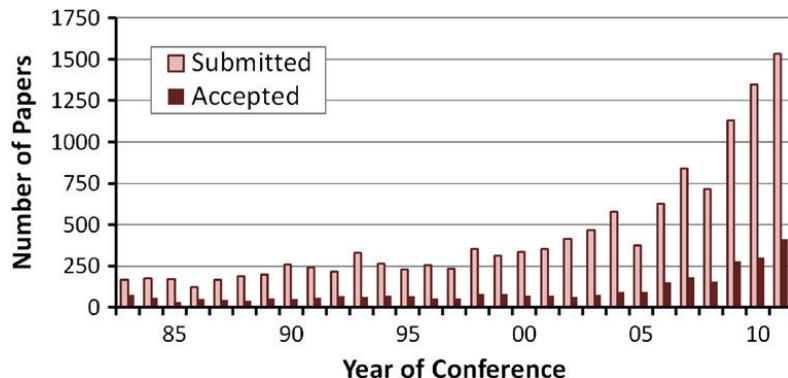
1.6.1 First ACM SIGCHI conference (1983)

Human-computer interaction's roots reach as early as 1969, when ACM's Special Interest Group on Social and Behavioral Computing (SIGSOC) was formed. (Borman, 1996). Initially, SIGSOC focused on computers in the social sciences. However, emphasis soon shifted to the needs and behavioral characteristics of the users, with talk about the user interface or the human factors of computing. Beginning in 1978, SIGSOC lobbied the ACM for a name change. This happened at the 1982 *Conference on Human Factors in Computing Systems* in Gaithersburg, Maryland, where the formation of the ACM Special Interest Group on Computer-Human Interaction (SIGCHI) was first publicly announced. Today, the ACM provides the following articulate statement of SIGCHI and its mission:

The ACM Special Interest Group on Computer-Human Interaction is the world's largest association of professionals who work in the research and practice of computer-human interaction. This interdisciplinary group is composed of computer scientists, software engineers, psychologists, interaction designers, graphic designers, sociologists, and anthropologists, just to name some of the domains whose special expertise come to bear in this area. They are brought together by a shared understanding that designing useful and usable technology is an interdisciplinary process, and believe that when done properly it has the power to transform persons' lives.⁷

The interdisciplinary nature of the field is clearly evident in the list of disciplines that contribute to, and have a stake in, HCI.

⁷Retrieved from <http://www.acm.org/sigs#026> on September 10, 2012.

**FIGURE 1.9**

Number of papers submitted and accepted by year for the ACM SIGCHI Conference on Human Factors in Computing Systems (“CHI”). Statistics from the ACM Digital Library.

In the following year, 1983, the first SIGCHI conference was held in Boston. Fifty-nine technical papers were presented. The conference adopted a slightly modified name to reflect its new stature: *ACM SIGCHI Conference on Human Factors in Computing Systems*. “CHI,” as it is known (pronounced with a hard “k” sound), has been held yearly ever since and in recent years has had an attendance of about 2,500 people.

The CHI conference brings together both researchers and practitioners. The researchers are there for the technical program (presentation of papers), while the practitioners are there to learn about the latest themes of research in academia and industry. Actually, both groups are also there to network (meet and socialize) with like-minded HCI enthusiasts from around the world. Simply put, CHI is *the* event in HCI, and the yearly pilgrimage to attend is often the most important entry in the calendar for those who consider HCI their field.

The technical program is competitive. Research papers are peer reviewed, and acceptance requires rising above a relatively high bar for quality. Statistics compiled from 1982 to 2011 indicate a total of 12,671 paper submissions with 3,018 acceptances, for an overall acceptance rate of 24 percent. Figure 1.9 shows the breakdown by year, as provided on the ACM Digital Library website.⁸ The technical program is growing rapidly. For example, the number of accepted contributions in 2011 (410) exceeded the number of submissions in 2005 (372).

Once accepted, researchers present their work at the conference, usually in a 15–20 minute talk augmented with visual slides and perhaps a video demonstration of the research. Acceptance also means the final submitted paper is published in the conference proceedings and archived in the ACM Digital Library. Some tips on writing and publishing a research paper are presented in Chapter 8.

⁸Data retrieved from <http://portal.acm.org>. (Click on “Proceedings,” scroll down to any CHI conference proceedings, click on it, then click on the “Publication” tab.)

CHI papers have high visibility, meaning they reach a large community of researchers and practitioners in the field. One indication of the quality of the work is *impact*, the number of citations credited to a paper. Since the standards for acceptance are high, one might expect CHI papers to have high impact on the field of HCI. And indeed this is the case (MacKenzie, 2009a). I will say more about research impact in Chapter 4.

Although the annual CHI conference is SIGCHI's flagship event, other conferences are sponsored or co-sponsored by SIGCHI. These include the annual *ACM Symposium on User Interface Software and Technology (UIST)*, specialized conferences such as the *ACM Symposium on Eye Tracking Research and Applications (ETRA)* and the *ACM Conference on Computers and Accessibility (ASSETS)*, and regional conferences such as the *Nordic Conference on Computer-Human Interaction (NordiCHI)*.

1.6.2 The psychology of human-computer interaction (1983)

If two HCI researchers speaking of “Card, Moran, and Newell” are overheard, there is a good chance they are talking about *The Psychology of Human-Computer Interaction*—the book published in 1983 and co-authored by Stuart Card, Tom Moran, and Allen Newell. (See Figure 1.10.) The book emerged from work done at Xerox PARC. Card and Moran arrived at PARC in 1974 and soon after joined PARC’s *Applied Information-Processing Psychology Project (AIP)*. Newell, a professor of computer science and cognitive psychology at Carnegie Mellon University in Pittsburgh, Pennsylvania, was a consultant to the project. The AIP mission was “to create an applied psychology of human-computer interaction by conducting requisite basic research within a context of application” (Card et al., 1983, p. ix).

The book contains 13 chapters organized roughly as follows: scientific foundation (100 pages), text editing examples (150 pages), modeling (80 pages), and extensions and generalizations (100 pages). So what is an “applied psychology of human-computer interaction”? Applied psychology is built upon basic research in psychology. The first 100 or so pages in the book provide a comprehensive overview of core knowledge in basic psychology as it pertains to the human sensory, cognitive, and motor systems. In the 1980s, many computer science students (and professionals) were challenged with building simple and intuitive interfaces for computer systems, particularly in view of emerging interaction styles based on a GUI. For many students, Card, Moran, and Newell’s book was their first formalized exposure to human perceptual input (e.g., the time to visually perceive a stimulus), cognition (e.g., the time to decide on the appropriate reaction), and motor output (e.g., the time to react and move the hand or cursor to a target). Of course, research in human sensory, cognitive, and motor behavior was well developed at the time. What Card, Moran, and Newell did was connect low-level human processes with the seemingly innocuous interactions humans have with computers (e.g., typing or using a mouse). The framework for this was the *model human processor (MHP)*. (See Figure 1.11.) The MHP had an eye and an ear (for sensory input to a perceptual processor), a

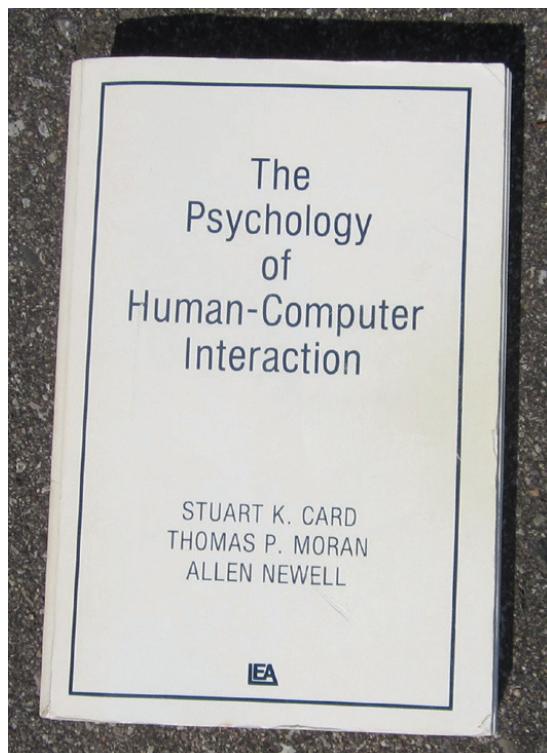


FIGURE 1.10

Card, Moran, and Newell's *The Psychology of Human-Computer Interaction*.

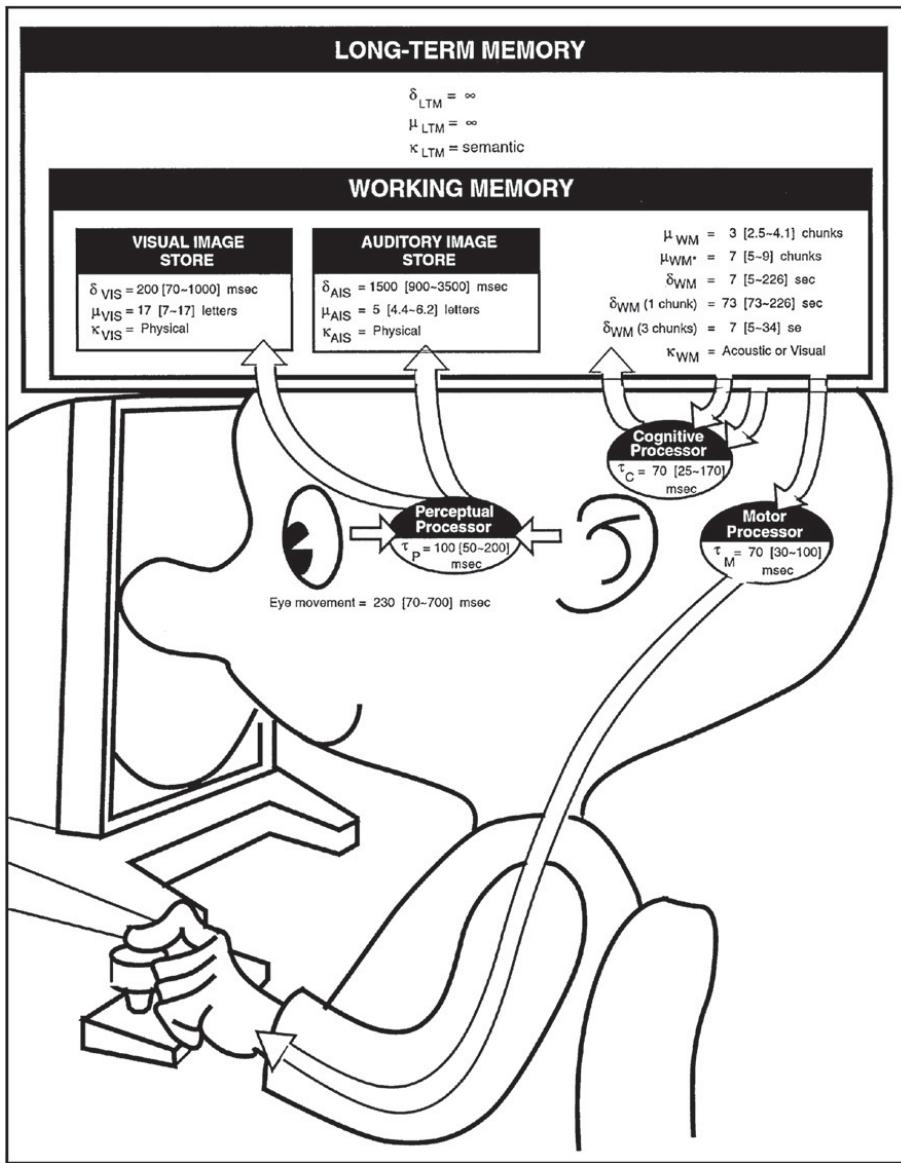
(Published by Erlbaum in 1983)

brain (with a cognitive processor, short-term memory, and long-term memory), and an arm, hand, and finger (for motor responses).

The application selected to frame the analyses in the book was text editing. This might seem odd today, but it is important to remember that 1983 predates the World Wide Web and most of today's computing environments such as mobile computing, touch-based input, virtual reality, texting, tweeting, and so on. Text editing seemed like the right framework in which to develop an applied psychology of human-computer interaction.⁹ Fortunately, all the issues pertinent to text editing are applicable across a broad spectrum of human-computer interaction.

An interesting synergy between psychology and computer science—and it is well represented in the book—is the notion that human behavior can be understood, even modeled, as an information processing activity. In the 1940s and 1950s the work of Shannon (1949), Huffman (1952), and others, on the transmission of information through electronic channels, was quickly picked up by psychologists like Miller (1956), Fitts (1954), and Welford (1968) as a way to characterize human perceptual, cognitive, and motor behavior. Card, Moran, and Newell

⁹At a panel session at CHI 2008, Moran noted that the choice was between text editing and programming.

**FIGURE 1.11**

The model human processor (MHP) (Card et al., 1983, p. 26).

adapted information processing models of human behavior to interactive systems. The two most prominent examples in the book are Hick's law for choice reaction time (Hick, 1952) and Fitts' law for rapid aimed movement (Fitts, 1954). I will say more about these in Chapter 7, Modeling Interaction.

Newell later reflected on the objectives of *The Psychology of Human-Computer Interaction*:

We had in mind the need for a theory for designers of interfaces. The design of the interface is the leverage point in human-computer interaction. The classical emphasis of human factors and man-machine psychology on experimental

analysis requires that the system or a suitable mock-up be available for experimentation, but by the time such a concrete system exists, most of the important degrees of freedom in the interface have been bound. What is needed are tools for thought for the designer—so at design time the properties and constraints of the user can be brought to bear in making the important choices. Our objective was to develop an engineering-style theory of the user that permitted approximate, back-of-the-envelope calculations of how the user would interact with the computer when operating at a terminal. (Newell, 1990, pp. 29–30)

There are some interesting points here. For one, Newell astutely identifies a dilemma in the field: experimentation cannot be done until it is too late. As he put it, the system is built and the degrees of freedom are bound. This is an overstatement, perhaps, but it is true that novel interactions in new products always seem to be followed by a flurry of research papers identifying weaknesses and suggesting and evaluating improvements. There is more to the story, however. Consider the Apple iPhone’s two-finger gestures, the Nintendo Wii’s acceleration sensing flicks, the Microsoft IntelliMouse’s scrolling wheel, or the Palm Pilot’s text-input gestures (aka *Graffiti*). These “innovations” were not fresh ideas born out of engineering or design brilliance. These breakthroughs, and many more, have context, and that context is the milieu of basic research in human-computer interaction and related fields.¹⁰ For the examples just cited, the research preceded commercialization. Research by its very nature requires dissemination through publication. It is not surprising, then, that conferences like CHI and books like *The Psychology of Human-Computer Interaction* are fertile ground for discovering and spawning new and exciting interaction techniques.

Newell also notes that an objective in the book was to generate “tools for thought.” This is a casual reference to models—models of interaction. The models may be quantitative and predictive or qualitative and descriptive. Either way, they are tools, the carver’s knife, the cobbler’s needle. Whether generating quantitative predictions across alternative design choices or delimiting a problem space to reveal new relationships, a model’s purpose is to tease out strengths and weaknesses in a hypothetical design and to elicit opportunities to improve the design. The book includes exemplars, such as the keystroke-level model (KLM) and the goals, operators, methods, and selection rules model (GOMS). Both of these models were presented in earlier work (Card, Moran, and Newell, 1980), but were presented again in the book, with additional discussion and analysis. The book’s main contribution on modeling, however, was to convincingly demonstrate why and how models are important and to teach us how to build them. For this, HCI’s debt to

¹⁰Of the four examples cited, research papers anticipating each are found in the HCI literature. On multi-touch finger gestures, there is Rekimoto’s “pick-and-drop” (1997), Dietz and Leigh’s DiamondTouch (Dietz and Leigh, 2001), or, much earlier, Herot and Weinzapfel’s two-finger rotation gesture on a touchscreen (Herot and Weinzapfel, 1978). On acceleration sensing, there is Harrison et al.’s “tilt me!” (1998). On the wheel mouse, there is Venolia’s “roller mouse” (Venolia, 1993). On single-stroke handwriting, there is Goldberg and Richardson’s “Unistrokes” (1993).

Card, Moran, and Newell is considerable. I will discuss descriptive and predictive models further in Chapter 7, Modeling Interaction.

Newell suggests using approximate “back of the envelope” calculations as a convenient way to describe or predict user interaction. In *The Psychology of Human-Computer Interaction*, these appear, among other ways, through a series of 19 interaction examples in Chapter 2 (pp. 23–97). The examples are presented as questions about a user interaction. The solutions use rough calculations but are based on data and concepts gleaned from basic research in experimental psychology. Example 10 is typical:

A user is presented with two symbols, one at a time. If the second symbol is identical to the first, he is to push the key labeled YES. Otherwise he is to push NO. What is the time between signal and response for the YES case? (Card et al., 1983, p. 66)

Before giving the solution, let us consider a modern context for the example. Suppose a user is texting a friend and is entering the word *hello* on a mobile phone using predictive text entry (T9). Since the mobile phone keypad is ambiguous for text entry, the correct word does not always appear. After entering 4(GHI), 3(DEF), 5(JKL), 5(JKL), 6(MNO), a word appears on the display. This is the *signal* in the example (see above). There are two possible responses. If the word is *hello*, it matches the word in the user’s mind and the user presses 0(SPACE) to accept the word and append a space. This is the YES response in the example. If the display shows some other word, a collision has occurred, meaning there are multiple candidates for the key sequence. The user presses *(NEXT) to display the next word in the ambiguous set. This is the NO response in the example. As elaborated by Card, Moran, and Newell, the interaction just described is a type of *simple decision* known as *physical matching*. The reader is walked through the solution using the model human processor to illustrate each step, from stimulus to cognitive processing to motor response. The solution is approximate. There is a nominal prediction accompanied by a *fastman* prediction and a *slowman* prediction. Here’s the solution:

$$\begin{aligned} \text{Reaction time} &= t_p + 2 \times t_c + t_M \\ &= 100[30 \sim 200] + 2 \times (70[25 \sim 170]) + 70[30 \sim 100] \quad (1) \\ &= 310[130 \sim 640]\text{ms} \end{aligned}$$

(Card et al., 1983, p. 69). There are four low-level processing cycles: a perceptual processor cycle (t_p), two cognitive processor cycles (t_c), and a motor processor cycle (t_M). For each, the nominal value is bracketed by an expected minimum and maximum. The values in Equation 1 are obtained from basic research in experimental psychology, as cited in the book. The fastman–slowman range is large and demonstrates the difficulty in accurately predicting human behavior. The book has many other examples like this. There are also modern contexts for the examples, just waiting to be found and applied.

It might not be apparent that predicting the time for a task that takes only one-third of a second is relevant to the bigger picture of designing interactive systems.

But don't be fooled. If a complex task can be deconstructed into primitive actions, there is a good chance the time to do the task can be predicted by dividing the task into a series of motor actions interlaced with perceptual and cognitive processing cycles. This idea is presented in Card, Moran, and Newell's book as a *keystroke-level model* (KLM), which I will address again in Chapter 7.

The Psychology of Human-Computer Interaction is still available (see <http://www.amazon.com>) and is regularly and highly cited in research papers (5,000+ citations according to Google Scholar). At the ACM SIGCHI conference in Florence, Italy in 2008, there was a panel session celebrating the book's 25th anniversary. Both Card and Moran spoke on the book's history and on the challenges they faced in bringing a psychological science to the design of interactive computing systems. Others spoke on how the book affected and influenced their own research in human-computer interaction.

1.6.3 Launch of the Apple Macintosh (1984)

January 22, 1984 was a big day in sports. It was the day of Super Bowl XVIII, the championship game of the National Football League in the United States. It was also a big day in advertising. With a television audience of millions, companies were jockeying (and paying!) to deliver brief jolts of hype to viewers who were hungry for entertainment and primed to purchase the latest must-have products. One ad—played during the third quarter—was a 60-second stint for the Apple Macintosh (the Mac) personal computer. The ad, which is viewable on YouTube, used Orwell's *Nineteen Eighty-Four* as a theme, portraying the Mac as a computer that would shatter the conventional image of the home computer.¹¹ The ad climaxed with a female athlete running toward, and tossing a sledgehammer through, the face of Big Brother. The disintegration of Big Brother signaled the triumph of the human spirit over the tyranny and oppression of the corporation. Directed by Ridley Scott,¹² the ad was a hit and was even named the 1980s Commercial of the Decade by *Advertising Age* magazine.¹³ It never aired again.

The ad worked. Soon afterward, computer enthusiasts scooped up the Mac. It was sleek and sported the latest input device, a computer mouse. (See Figure 1.12.) The operating system and applications software heralded the new age of the GUI with direct manipulation and point-select interaction. The Mac was not only cool, the interface was simple and intuitive. Anyone could use it. Part of the simplicity was its one-button mouse. With one button, there was no confusion on which button to press.

There are plenty of sources chronicling the history of Apple and the events leading to the release of the Mac (Levy, 1995; Linzmayer, 2004; Moritz, 1984). Unfortunately, along with the larger-than-life stature of Apple and its flamboyant

¹¹Search using “1984 Apple Macintosh commercial.”

¹²Known for his striking visual style, Scott directed many off-beat feature-length films such as *Alien* (1979), *Blade Runner* (1982), *Thelma and Louise* (1991), and *Gladiator* (2000).

¹³[http://en.wikipedia.org/wiki/1984_\(advertisement\)](http://en.wikipedia.org/wiki/1984_(advertisement)).

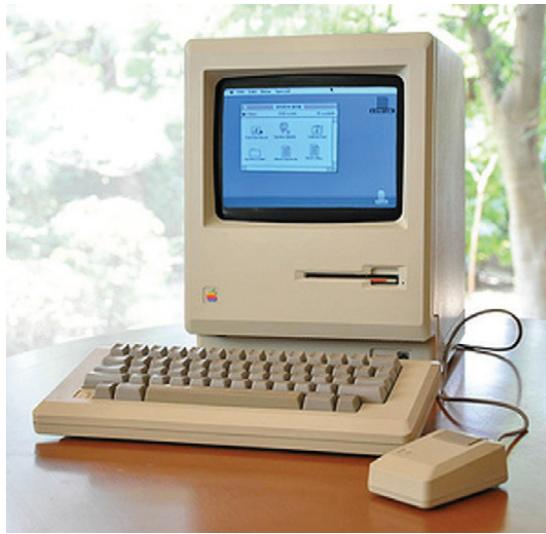


FIGURE 1.12 The Apple Macintosh.

leaders comes plenty of folklore to untangle. A few notable events are listed in Figure 1.13. Names of the key players are deliberately omitted.

1.7 Growth of HCI and graphical user interfaces (GUIs)

With the formation of ACM SIGCHI in 1983 and the release and success of the Apple Macintosh in 1984, human-computer interaction was off and running. GUIs entered the mainstream and, consequently, a much broader community of users and researchers were exposed to this new genre of interaction. Microsoft was a latecomer in GUIs. Early versions of Microsoft *Windows* appeared in 1985, but it was not until the release of *Windows 3.0* (1990) and in particular *Windows 3.1* (1992) that Microsoft Windows was considered a serious alternative to the Macintosh operating system. Microsoft increased its market share with improved versions of Windows, most notably *Windows 95* (1995), *Windows 98* (1998), *Windows XP* (2001), and *Windows 7* (2009). Today, Microsoft operating systems for desktop computers have a market share of about 84 percent, compared to 15 percent for Apple.¹⁴

With advancing interest in human-computer interaction, all major universities introduced courses in HCI or user interface (UI) design, with graduate students often choosing a topic in HCI for their thesis research. Many such programs of study were in computer science departments; however, HCI also emerged as a legitimate and popular focus in other areas such as psychology, cognitive science, industrial engineering, information systems, and sociology. And it wasn't just universities that recognized the importance of the emerging field. Companies soon

¹⁴www.statowl.com.

1976	April – Apple Computer Inc. founded in Cupertino, California.
1977	Launch of Apple II. Sells for \$1300 U.S. with 4KB RAM. Hugely successful (more than one million units sold). Works with a text-based command-line interface.
1978	<i>Lisa</i> project started . Goal of producing a powerful (and expensive!) personal computer.
1979	September – <i>Macintosh</i> project started. Goal of producing a low-cost easy-to-use computer for the average consumer. December – Apple and Xerox sign an agreement that allows Xerox to invest in Apple. In return Apple's engineers visit Xerox PARC and see the Xerox <i>Alto</i> . The GUI ideas in the <i>Alto</i> influence <i>Lisa</i> and <i>Macintosh</i> development.
1980	December – Apple goes public through initial public offering (IPO) of its stock.
1981	May – Xerox <i>Star</i> launched at the National Computer Conference (NCC) in Chicago. Members of the <i>Lisa</i> design team are present and see the <i>Star</i> demo. They decide to re-vamp the <i>Lisa</i> interface to be icon-based. August – IBM PC announced. Highly successful, but embodies traditional text-based command-line interface.
1982	<i>Lisa</i> and <i>Macintosh</i> development continue. Within Apple, there is an atmosphere of competition between the two projects.
1983	January – <i>Lisa</i> released. <i>Lisa</i> incorporates a GUI and mouse input. Sells for \$10,000 U.S. In the end, <i>Lisa</i> is a commercial failure. December – brochures distributed in magazines (e.g., <i>Time</i>) pre-announcing the <i>Macintosh</i> .
1984	January 22 – <i>Macintosh</i> ad plays during Super Bowl XVIII. January 24 – <i>Macintosh</i> released. Sells for \$2500 U.S.

FIGURE 1.13

Some notable events leading to the release of the Apple Macintosh.¹⁵

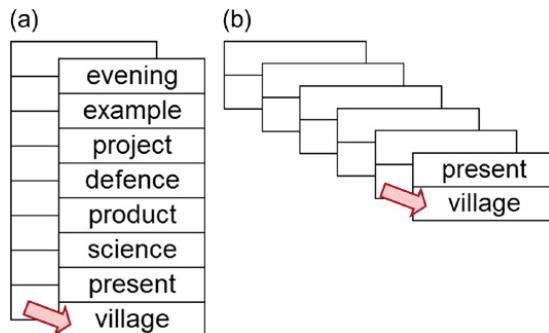
realized that designing good user interfaces was good business. But it wasn't easy. Stories of bad UIs are legion in HCI (e.g., Cooper, 1999; Johnson, 2007; Norman, 1988). So there was work to be done. Practitioners—that is, specialists applying HCI principles in industry—are important members of the HCI community, and they form a significant contingent at many HCI conferences today.

1.8 Growth of HCI research

Research interest in human-computer interaction, at least initially, was in the quality, effectiveness, and efficiency of the interface. How quickly and accurately can people do common tasks using a GUI versus a text-based command-line interface? Or, given two or more variations in a GUI implementation, which one is quicker or more accurate? These or similar questions formed the basis of much empirical research in the early days of HCI. The same is still true today.

A classic example of a research topic in HCI is the design of menus. With a GUI, the user issues a command to the computer by selecting the command from a menu rather than typing it on the keyboard. Menus require *recognition*; typing

¹⁵www.theapplemuseum.com, http://en.wikipedia.org/wiki/History_of_Apple, and www.guidebookgallery.org/articles/lisainterview, with various other sources to confirm dates and events.

**FIGURE 1.14**

Breadth versus depth in menu design: (a) 8×8 choices in a broad hierarchy.
 (b) $2 \times 2 \times 2 \times 2 \times 2 \times 2$ choices in a deep hierarchy.

requires *recall*. It is known that recognition is preferred over recall in user interfaces (Bailey, 1996, p. 144; Hodgson and Ruth, 1985; Howes and Payne, 1990), at least for novices, but a new problem then surfaces. If there are numerous commands in a menu, how should they be organized? One approach is to organize menu commands in a hierarchy that includes depth and breadth. The question arises: what is the best structure for the hierarchy? Consider the case of 64 commands organized in a menu. The menu could be organized with depth = 8 and breadth = 2, or with depth = 2 and breadth = 6. Both structures provide access to 64 menu items. The breadth-emphasis case gives $8^2 = 64$ choices (Figure 1.14a). The depth-emphasis case gives $2^6 = 64$ choices (Figure 1.14b). Which organization is better? Is another organization better still (e.g., $4^3 = 64$)? Given these questions, it is not surprising that menu design issues were actively pursued as research topics in the early days of HCI (e.g., Card, 1982; Kiger, 1984; Landauer and Nachbar, 1985; D. P. Miller, 1981; Snowberry, Parkinson, and Sisson, 1983; Tullis, 1985).

Depth versus breadth is not the only research issue in menu design; there are many others. Should items be ordered alphabetically or by function (Card, 1982; Mehlenbacher, Duffy, and Palmer, 1989)? Does the presence of a title on a submenu improve menu access (J. Gray, 1986)? Is access improved if an icon is added to the label (Hemenway, 1982)? Do people in different age groups respond differently to broad versus deep menu hierarchies (Zaphiris, Kurniawan, and Ellis, 2003)? Is there a depth versus breadth advantage for menus on mobile devices (Geven, Sefelin, and Tschelig, 2006)? Does auditory feedback improve menu access (Zhao, Dragicevic, Chignell, Balakrishnan, and Baudisch, 2007)? Can the tilt of a mobile phone be used for menu navigation (Rekimoto, 1996)? Can menu lists be pie shaped, rather than linear (Callahan, Hopkins, Weiser, and Shneiderman, 1988)? Can pie menus be used for text entry (D. Venolia and Neiberg, 1994)?

The answers to these research questions can be found in the papers cited. They are examples of the kinds of research questions that create opportunities for empirical research in HCI. There are countless such topics of research in HCI. While we've seen many in this chapter, we will find many more in the chapters to come.

1.9 Other readings

Two other papers considered important in the history of HCI are:

- “Personal Dynamic Media” by A. Kay and A. Goldberg (1977). This article describes *Dynabook*. Although never built, Dynabook provided the conceptual basis for laptop computers, tablet PCs, and e-books.
- “The Computer for the 21st Century” by M. Weiser (1991). This is the essay that presaged ubiquitous computing. Weiser begins, “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it” (p. 94).

Other sources taking a historical view of human-computer interaction include: Baecker, Grudin, Buxton, and Greenberg, 1995; Erickson and McDonald, 2007; Grudin, 2012; Myers, 1998.

1.10 Resources

The following online resources are useful for conducting research in human-computer interaction:

- Google Scholar: <http://scholar.google.ca>
- ACM Digital Library: <http://portal.acm.org>
- HCI Bibliography: <http://hcibib.org>

This website is available as a resource accompanying this book:

- www.yorku.ca/mack/HCIbook
Many downloads are available to accompany the examples presented herein.
-

STUDENT EXERCISES

- 1-1.** The characteristics of direct manipulation include visibility of objects, incremental action, rapid feedback, reversibility, exploration, syntactic correctness of all actions, and replacing language with action. For each characteristic consider and discuss an example task performed with modern GUIs. Contrast the task with the same task as performed in a command-line environment such as unix, linux, or DOS.

CHAPTER 3

Cognitive Engineering

DONALD A. NORMAN

PROLOGUE

cognitive Engineering, a term invented to reflect the enterprise I find myself engaged in: neither Cognitive Psychology, nor Cognitive Science, nor Human Factors. It is a type of applied Cognitive Science, trying to apply what is known from science to the design and construction of machines. It is a surprising business. On the one hand, there actually is quite a lot known in Cognitive Science that can be applied. But on the other hand, our lack of knowledge is appalling. On the one hand, computers are ridiculously difficult to use. On the other hand, many devices are difficult to use—the problem is not restricted to computers, there are fundamental difficulties in understanding and using most complex devices. So the goal of Cognitive Engineering is to come to understand the issues, to show how to make better choices when they exist, and to show what the tradeoffs are when, as is the usual case, an improvement in one domain leads to deficits in another.

In this chapter I address some of the problems of applications that have been of primary concern to me over the past few years and that have guided the selection of contributors and themes of this book. The chapter is not intended to be a coherent discourse on Cognitive Engineering. Instead, I discuss a few issues that seem central to the

way that people interact with machines. The goal is to determine what are the critical phenomena: The details can come later. Overall, I have two major goals:

1. To understand the fundamental principles behind human action and performance that are relevant for the development of engineering principles of design.
2. To devise systems that are pleasant to use—the goal is neither efficiency nor ease nor power, although these are all to be desired, but rather systems that are pleasant, even fun: to produce what Laurel calls “pleasurable engagement” (Chapter 4).

AN ANALYSIS OF TASK COMPLEXITY

Start with an elementary example: how a person performs a simple task. Suppose there are two variables to be controlled. How should we build a device to control these variables? The control question seems trivial: If there are two variables to be controlled, why not simply have two controls, one for each? What is the problem? It turns out that there is more to be considered than is obvious at first thought. Even the task of controlling a single variable by means of a single control mechanism raises a score of interesting issues.

One has only to watch a novice sailor attempt to steer a small boat to a compass course to appreciate how difficult it can be to use a single control mechanism (the tiller) to affect a single outcome (boat direction). The mapping from tiller motion to boat direction is the opposite of what novice sailors sometimes expect. And the mapping of compass movement to boat movement is similarly confusing. If the sailor attempts to control the boat by examining the compass, determining in which direction to move the boat, and only then moving the tiller, the task can be extremely difficult.

Experienced sailors will point out that this formulation puts the problem in its clumsiest, most difficult form: With the right formulation, or the right conceptual model, the task is not complex. That comment makes two points. First, the description I gave is a reasonable one for many novice sailors: The task is quite difficult for them. The point is not that there are simpler ways of viewing the task, but that even a task that has but a single mechanism to control a single variable can be difficult to understand, to learn, and to do. Second, the comment reveals the power of the proper conceptual model of the

situation: The correct conceptual model can transform confusing, difficult tasks into simple, straightforward ones. This is an important point that forms the theme of a later section.

Psychological Variables Differ From Physical Variables

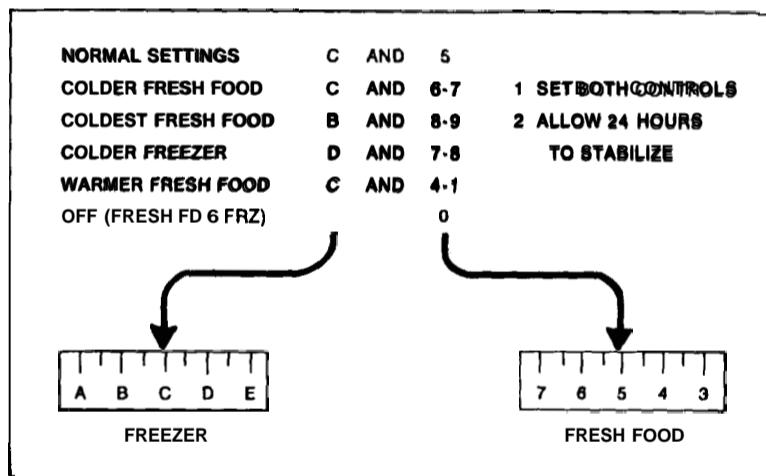
There is a discrepancy between the person's *psychologically* expressed goals and the *physical* controls and variables of the task. The person starts with goals and intentions. These are *psychological* variables. They exist in the mind of the person and they relate directly to the needs and concerns of the person. However, the task is to be performed on a *physical* system, with physical mechanisms to be manipulated, resulting in changes to the physical variables and system state. Thus, the person must interpret the physical variables into terms relevant to the psychological goals and must translate the psychological intentions into physical actions upon the mechanisms. This means that there must be a stage of interpretation that relates physical and psychological variables, as well as functions that relate the manipulation of the physical variables to the resulting change in physical state.

In many situations the variables that can easily be controlled are not those that the person cares about. Consider the example of bathtub water control. The person wants to control rate of total water flow and temperature. But water arrives through two pipes: hot and cold. The easiest system to build has two faucets and two spouts. As a result, the physical mechanisms control rate of hot water and rate of cold water. Thus, the variables of interest to the user interact with the two physical variables: Rate of total flow is the sum of the two physical variables; temperature is a function of their difference (or ratio). The problems come from several sources:

1. *Mapping problems.* Which control is hot, which is cold? Which way should each control be turned to increase or decrease the flow? (Despite the appearance of universal standards for these mappings, there are sufficient variations in the standards, idiosyncratic layouts, and violations of expectations, that each new faucet poses potential problems.)
2. *Ease of control.* To make the water hotter while maintaining total rate constant requires simultaneous manipulation of both faucets.
3. *Evaluation.* With two spouts, it is sometimes difficult to determine if the correct outcome has been reached.

Faucet technology evolved to solve the problem. First, mixing spouts were devised that aided the evaluation problem. Then, "single control" faucets were devised that varied the psychological factors directly: One dimension of movement of the control affects rate of flow, another orthogonal dimension affects temperature. These controls are clearly superior to use. They still do have a mapping problem—knowing what kind of movement to which part of the mechanism controls which variable—and because the mechanism is no longer as visible as in the two-faucet case, they are not quite so easy to understand for the first-time user. Still, faucet design can be used as a positive example of how technology has responded to provide control over the variables of psychological interest rather than over the physical variables that are easier and more obvious.

It is surprisingly easy to find other examples of the two-variable—two-control task. The water faucets is one example. The loudness and balance controls on some audio sets is another. The temperature controls of some refrigerator-freezer units is another. Let me examine this latter example, for it illustrates a few more issues that need to be considered, including the invisibility of the control mechanisms and a long time delay between adjustment of the control and the resulting change of temperature.



There are two variables of concern to the user: the temperature of the freezer compartment and the temperature of the regular "fresh

food" compartment. At first, this seems just like the water control example, but there is a difference. Consider the refrigerator that I own. It has two compartments, a freezer and a fresh foods one, and two controls, both located in the fresh foods section. One control is labeled "freezer," the other "fresh food," and there is an associated instruction plate (see the illustration). But what does each control do? What is the mapping between their settings and my goal? The labels seem clear enough, but if you read the "instructions" confusion can rapidly set in. Experience suggests that the action is not as labeled: The two controls interact with one another. The problems introduced by this example seem to exist at almost every level:

1. Matching the psychological variables of interest to the physical variables being controlled. Although the labels on the control mechanisms indicate some relationship to the desired psychological variables, in fact, they do not control those variables directly.
2. The mapping relationships. There is clearly strong interaction between the two controls, making simple mapping between control and function or control and outcome difficult.
3. Feedback. Very slow, so that by the time one is able to determine the result of an action, so much time has passed that the action is no longer remembered, making "correction" of the action difficult.
4. Conceptual model. None. The instructions seem deliberately opaque and nondescriptive of the actual operations.

I suspect that this problem results from the way this refrigerator's cooling mechanism is constructed. The two variables of psychological interest cannot be controlled directly. Instead, there is only one cooling mechanism and one thermostat, which therefore, must be located in either the "fresh food" section or in the freezer, but not both. A good description of this mechanism, stating which control affected which function would probably make matters workable. If one mechanism were clearly shown to control the thermostat and the other to control the relative proportion of cold air directed toward the freezer and fresh foods section, the task would be

much easier. The user would be able to get a clear conceptual model of the operation. Without a conceptual model, with a 24-hour delay between setting the controls and determining the results, it is almost impossible to determine how to operate the controls. Two variables: two controls. Who could believe that it would be so difficult?

Even Simple Tasks Involve a Large Number of Aspects

The conclusion to draw from these examples is that even with two variables, the number of aspects that must be considered is surprisingly large. Thus, suppose the person has two psychological goals, G_1 and G_2 . These give rise to two intentions, I_1 and I_2 , to satisfy the goals. The system has some physical state, S , realized through the values of its variables: For convenience, let there be two variables of interest, V_1 and V_2 . And let there be two mechanisms that control the system, M_1 and M_2 . So we have the psychological goals and intentions (G and I) and the physical state, mechanisms, and variables (S , M , and V). First, the person must examine the current system state, S , and evaluate it with respect to the goals, G . This requires translating the physical state of the system into a form consistent with the psychological goal. Thus, in the case of steering a boat, the goal is to reach some target, but the physical state is the numerical compass heading. In writing a paper, the goal may be a particular appearance of the manuscript, but the physical state may be the presence of formatting commands in the midst of the text. The difference between desired goal and current state gives rise to an intention, again stated in psychological terms. This must get translated into an action sequence, the specification of what physical acts will be performed upon the mechanisms of the system. To go from intention to action specification requires consideration of the mapping between physical mechanisms and system state, and between system state and the resulting psychological interpretation. There may not be a simple mapping between the mechanisms and the resulting physical variables, nor between the physical variables and the resulting psychological states. Thus, each physical variable might be affected by an interaction of the control mechanisms: $V_1 = f(M_1, M_2)$ and $V_2 = g(M_1, M_2)$. In turn, the system state, S is a function of all its variables: $S = h(V_1, V_2)$. And finally, the mapping between system state and psychological interpretation is complex. All in all, the two variable–two mechanism situation can involve a surprising number of aspects. The list of aspects is shown and defined in Table 3.1.

TABLE 3.1
ASPECTS OF A TASK

Aspect	Description
Goals and intentions.	A goal is the state the person wishes to achieve; an intention is the decision to act so as to achieve the goal.
Specification of the action sequence.	The psychological process of determining the psychological representation of the actions that are to be executed by the user on the mechanisms of the system.
Mapping from psychological goals and intentions to action sequence.	In order to specify the action sequence, the user must translate the psychological goals and intentions into the desired system state, then determine what settings of the control mechanisms will yield that state, and then determine what physical manipulations of the mechanisms are required. The result is the internal, mental specification of the actions that are to be executed.
Physical state of the system.	The physical state of the system, determined by the values of all its physical variables.
Control mechanisms.	The physical devices that control the physical variables.
Mapping between the physical mechanisms and system state.	The relationship between the settings of the mechanisms of the system and the system state.
Interpretation of system state.	The relationship between the physical state of the system and the psychological goals of the user can only be determined by first translating the physical state into psychological states (perception), then interpreting the perceived system state in terms of the psychological variables of interest.
Evaluating the outcome.	Evaluation of the system state requires comparing the interpretation of the perceived system state with the desired goals. This often leads to a new set of goals and intentions.

TOWARD A THEORY OF ACTION

It seems clear that we need to develop theoretical tools to understand what the user is doing. We need to know more about how people actually do things, which means a theory of action. There isn't any realistic hope of getting *the* theory of action, at least for a long time, but

certainly we should be able to develop approximate theories.¹ And that is what follows: an approximate theory for action which distinguishes among different stages of activities, not necessarily always used nor applied in that order, but different kinds of activities that appear to capture the critical aspects of doing things. The stages have proved to be useful in analyzing systems and in guiding design. The essential components of the theory have already been introduced in Table 3.1.

In the theory of action to be considered here, a person interacts with a system, in this case a computer. Recall that the person's goals are expressed in terms relevant to the person—in psychological terms—and the system's mechanisms and states are expressed in terms relative to it—in physical terms. The discrepancy between psychological and physical variables creates the major issues that must be addressed in the design, analysis, and use of systems. I represent the discrepancies as two gulfs that must be bridged: the *Gulf of Execution* and the *Gulf of Evaluation*, both shown in Figure 3.1.²

The Gulfs of Execution and Evaluation

The user of the system starts off with goals expressed in psychological terms. The system, however, presents its current state in physical terms. Goals and system state differ significantly in form and content, creating the Gulfs that need to be bridged if the system can be used (Figure 3.1). The Gulfs can be bridged by starting in either direction. The designer can bridge the Gulfs by starting at the system side and moving closer to the person by constructing the input and output characteristics of the interface so as to make better matches to the

¹ There is little prior work in psychology that can act as a guide. Some of the principles come from the study of servomechanisms and cybernetics. The first study known to me in psychology—and in many ways still the most important analysis—is the book *Plans and the Structure of Behavior* by Miller, Galanter, and Pribram (1960) early in the history of information processing psychology. Powers (1973) applied concepts from control theory to cognitive concerns. In the work most relevant to the study of Human-Computer Interaction, Card, Moran, and Newell (1983), analyzed the cycle of activities from Goal through Selection: the GOMS model (*Goal, Operator, Methods, Selection*). Their work is closely related to the approach given here. This is an issue that has concerned me for some time, so some of my own work is relevant: the analysis of errors, of typing, and of the attentional control of actions (Norman, 1981a, 1984b, 1986; Norman & Shallice, 1985; Rumelhart & Norman, 1982).

² The emphasis on the the discrepancy between the user and the system, and the suggestion that we should conceive of the discrepancy as a Gulf that must be bridged by the user and the system designer, came from Jim Hollan and Ed Hutchins during one of the many revisions of the Direct Manipulation chapter (Chapter 5).

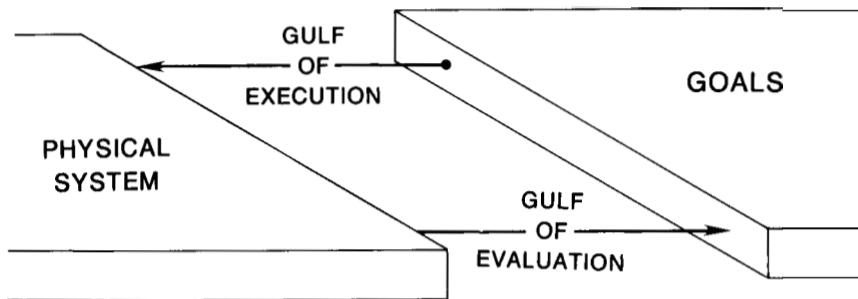


FIGURE 3.1. The Gulfs of Execution and Evaluation. Each Gulf is unidirectional: The Gulf of Execution goes from Goals to Physical System; the Gulf of Evaluation goes from Physical System to Goals.

psychological needs of the user. The user can bridge the Gulfs by creating plans, action sequences, and interpretations that move the normal description of the goals and intentions closer to the description required by the physical system (Figure 3.2).

Bridging the Gulf of Execution. The gap from goals to physical system is bridged in four segments: intention formation, specifying the action sequence, executing the action, and, finally, making contact with the input mechanisms of the interface. The intention is the first step, and it starts to bridge the gulf, in part because the interaction language demanded by the physical system comes to color the thoughts of the person, a point expanded upon in Chapter 5 by Hutchins, Hollan, and Norman. Specifying the action sequence is a nontrivial exercise in planning (see Riley & O'Malley, 1985). It is what Moran calls matching the internal specification to the external (Moran, 1983). In the terms of the aspects listed in Table 3.1, specifying the action requires translating the psychological goals of the intention into the changes to be made to the physical variables actually under control of the system. This, in turn, requires following the mapping between the psychological intentions and the physical actions permitted on the mechanisms of the system, as well as the mapping between the physical mechanisms and the resulting physical state variables, and between the physical state of the system and the psychological goals and intentions.

After an appropriate action sequence is determined, the actions must be executed. Execution is the first physical action in this sequence: Forming the goals and intentions and specifying the action sequence were all mental events, Execution of an action means to do something, whether it is just to say something or to perform a complex motor

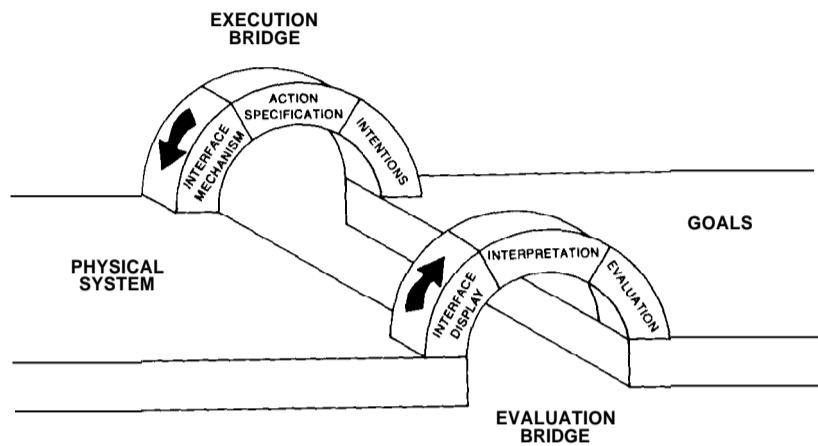


FIGURE 3.2. Bridging the Gulfs of Execution and Evaluation. The Gulf of *Execution* is bridged from the psychology side by the user's formation of intentions relevant to the system and the determination of an action sequence. It is bridged from the system side when the designer of the system builds the input characteristics of the interface. The Gulf of *Evaluation* is bridged from the psychology side by the user's perception of the system state and the interpretation placed on that perception, which is then evaluated by comparing it with the original goals and intentions. It is bridged from the system side when the designer builds the output characteristics of the interface.

sequence. Just what physical actions are required is determined by the choice of input devices on the system, and this can make a major difference in the usability of the system. Because some physical actions are more difficult than others, the choice of input devices can affect the selection of actions, which in turn affects how well the system matches with intentions. On the whole, theorists in this business tend to ignore the input devices, but in fact, the choice of input device can often make an important impact on the usability of a system. (See Chapter 15 by Buxton for a discussion of this frequently overlooked point.)

Bridging the Gulf of Evaluation. Evaluation requires comparing the interpretation of system state with the original goals and intentions. One problem is to determine what the system state is, a task that can be assisted by appropriate output displays by the system itself. The outcomes are likely to be expressed in terms of physical variables that bear complex relationships to the psychological variables of concern to the user and in which the intentions were formulated. The gap from system to user is bridged in four segments: starting with the output

displays of the interface, moving to the perceptual processing of those displays, to its interpretation, and finally, to the evaluation—the comparison of the interpretation of system state with the original goals and intention. But in doing all this, there is one more problem, one just beginning to be understood, and one not assisted by the usual forms of displays: the problem of level. There may be many levels of outcomes that must be matched with different levels of intentions (see Norman, 1981a; Rasmussen in press; Rasmussen & Lind, 1981). And, finally, if the change in system state does not occur immediately following the execution of the action sequence, the resulting delay can severely impede the process of evaluation, for the user may no longer remember the details of the intentions or the action sequence.

Stages of User Activities

A convenient summary of the analysis of tasks is that the process of performing and evaluating an action can be approximated by seven stages of user activity³ (Figure 3.3):

- Establishing the Goal
- Forming the Intention
- Specifying the Action Sequence
- Executing the Action
- Perceiving the System State
- Interpreting the State
- Evaluating the System State with respect to the Goals and Intentions

³ The last two times I spoke of an approximate theory of action (Norman, 1984a, 1985) I spoke of four stages. Now I speak of seven. An explanation seems to be in order. The answer really is simple. The full theory of action is not yet in existence, but whatever its form, it involves a continuum of stages on both the action/execution side and the perception/evaluation side. The notion of stages is a simplification of the underlying theory: I do not believe that there really are clean, separable stages. However, for practical application, approximating the activity into stages seems reasonable and useful. Just what division of stages should be made, however, seems less clear. In my original formulations, I suggested four stages: intention, action sequence, execution, and evaluation. In this chapter I separated goals and intentions and expanded the analysis of evaluation by adding perception and interpretation, thus making the stages of evaluation correspond better with the stages of execution: Perception is the evaluatory equivalent of execution, interpretation the equivalent of the action sequence, and evaluation the equivalent of forming the intention. The present formulation seems a richer, more satisfactory analysis.

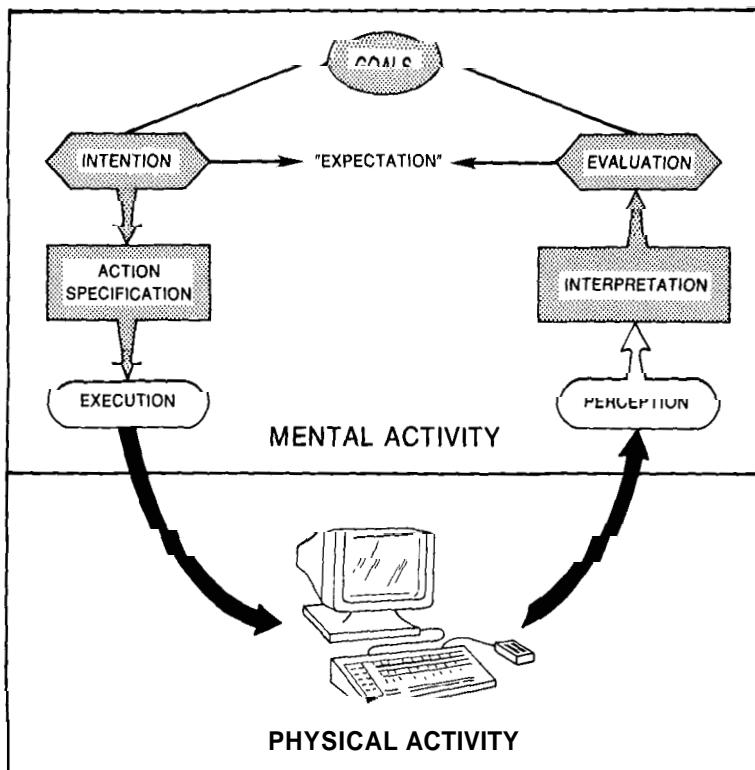


FIGURE 3.3. Seven stages of user activities involved in the performance of a task. The primary, central stage is the establishment of the goal. Then, to carry out an action requires three stages: forming the intention, specifying the action sequence, and executing the action. To assess the effect of the action also requires three stages, each in some sense complementary to the three stages of carrying out the action: perceiving the system state, interpreting the state, and evaluating the interpreted state with respect to the original goals and intentions.

Real activity does not progress as a simple sequence of stages. Stages appear out of order, some may be skipped, some repeated. Even the analysis of relatively simple tasks demonstrates the complexities. Moreover, in some situations, the person is reactive—event or data driven—responding to events, as opposed to starting with goals and intentions. Consider the task of monitoring a complex, ongoing operation. The person's task is to respond to observations about the state of the system. Thus, when an indicator starts to move a bit out of range, or when something goes wrong and an alarm is triggered, the operator

must diagnose the situation and respond appropriately. The diagnosis leads to the formation of goals and intentions: Evaluation includes not only checking on whether the intended actions were executed properly and intentions satisfied, but whether the original diagnosis was appropriate. Thus, although the stage analysis is relevant, it must be used in ways appropriate to the situation.

Consider the example of someone who has written a letter on a computer word-processing system. The overall goal is to convey a message to the intended recipient. Along the way, the person prints a draft of the letter. Suppose the person decides that the draft, shown in Figure 3.4A, doesn't look right: The person, therefore, establishes the intention "Improve the appearance of the letter." Call this first intention *intention₁*. Note that this intention gives little hint of how the task is to be accomplished. As a result, some problem solving is required, perhaps ending with *intention₂*: "Change the indented paragraphs to block paragraphs." To do this requires *intention₃*: "Change the occurrences of .pp in the source code for the letter to .sp." This in turn requires the person to generate an action sequence appropriate for the text editor, and then, finally, to execute the actions on the computer keyboard. Now, to evaluate the results of the operation requires still further operations, including generation of a fourth intention, *intention₄*: "Format the file" (in order to see whether *intention₂* and *intention₁* were satisfied). The entire sequence of stages is shown in Figure 3.4B. The final product, the reformatted letter, is shown in Figure 3.4C. Even intentions that appear to be quite simple (e.g., *intention*₁: "Approve the appearance of the letter") lead to numerous subintentions. The intermediary stages may require generating some new subintentions.

Practical Implications

The existence of the two gulfs points out a critical requirement for the design of the interface: to bridge the gap between goals and system. Moreover, as we have seen, there are only two ways to do this: move the system closer to the user; move the user closer to the system. Moving from the system to the user means providing an interface that matches the user's needs, in a form that can be readily interpreted and manipulated. This confronts the designer with a large number of issues. Not only do users differ in their knowledge, skills, and needs, but for even a single user the requirements for one stage of activity can conflict with the requirements for another. Thus, menus can be thought of as information to assist in the stages of intention formation and action specification, but they frequently make execution more

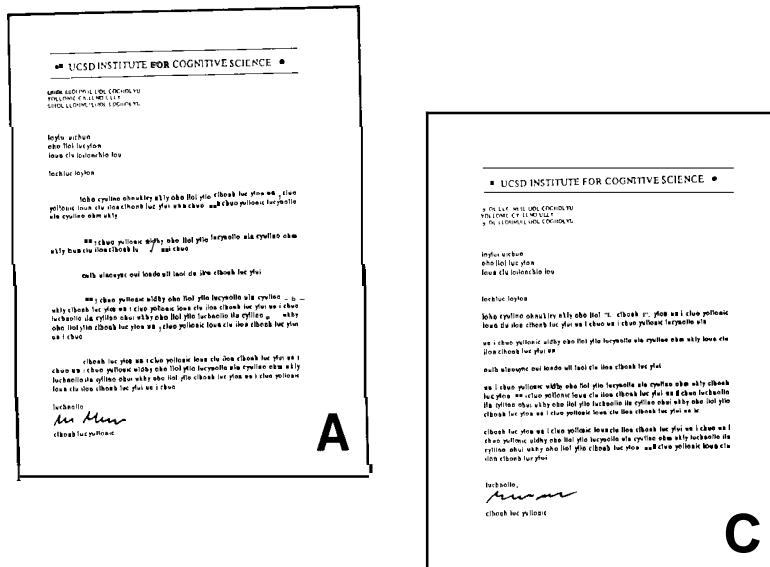


FIGURE 3.4. Sequence of stages in a typical task. (A) The starting point. The letter doesn't "look right," so the initial intention is "improve the appearance of the letter." (B) The sequence of stages necessary to make the appropriate changes to the source file of the manuscript, then to get a printed, formatted copy of the letter, and finally, to evaluate the outcome against the several levels of intentions. (C) The final product, the reformatted letter.

difficult. The attempt to aid evaluation by presenting extra information can impair intention selection, in part by providing distractions. On the other hand, failure to provide information can make life more complex for the user, making it harder to get the job done and adding to the frustrations with the system if the user is left bewildered, not knowing what options are available or what is happening.

Many systems can be characterized by how well they support the different stages. The argument over whether action specification should be done by command language or by pointing at menu options or icons turns out to be an argument over the relative merits of support for the stages of *Execution* and *Action Specification*.

Visual presence can aid the various stages of activity. Thus, we give support to the generation of intentions by reminding the user of what is possible. We support action selection because the visible items act as a direct translation into possible actions. We aid execution, especially if execution by pointing (throwing switches) is possible. And we aid evaluation by making it possible to provide visual reminders of what was done. Visual structure can aid in the interpretation. Thus, for some purposes, graphs, pictures, and moving images will be superior to words: In other situations words will be superior.

Moving from psychological variables to physical variables can take effort. The user must translate goals conceived in psychological terms to actions suitable for the system. Then, when the system responds, the user must interpret the output, translating the physical display of the interface back into psychological terms. The major responsibility should rest with the system designer to assist the user in understanding the system. This means providing a good, coherent design model and a consistent, relevant system image.

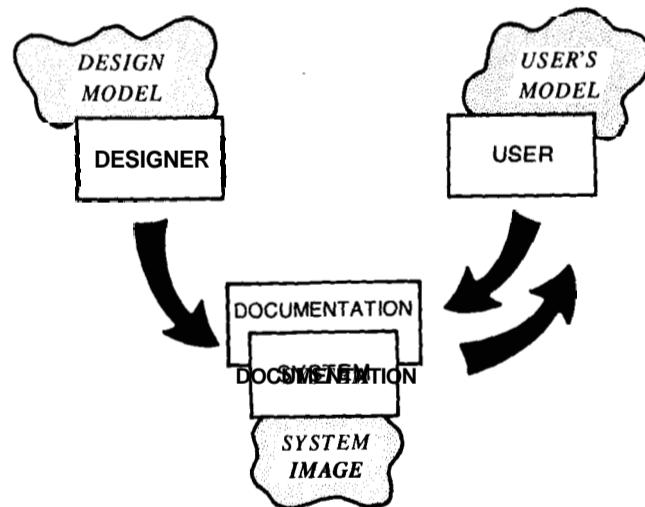
CONCEPTUAL MODELS AND THE SYSTEM IMAGE

There are two sides to the interface: the system side and the human side. The stages of execution and perception mediate between psychological and physical representations. And the input mechanism and output displays of the system mediate between the psychological and physical representations. We change the interface at the system side through proper design. We change the interface at the human side through training and experience. In the ideal case, no psychological effort is required to bridge the gulfs. But such a situation occurs only either with simple situations or with experienced, expert users. With complex tasks or with nonexpert users, the user must engage in a planning process to go from intentions to action sequence. This planning process, oftentimes involving active problem solving, is aided when the

person has a good conceptual understanding of the physical system, an argument developed more fully by Riley in Chapter 7.

Think of a conceptual model of the system as providing a scaffolding upon which to build the bridges across the gulfs. The scaffoldings provided by these conceptual models are probably only important during learning and trouble-shooting. But for these situations they are essential. Expert users can usually do without them. They allow the user to derive possible courses of action and possible system responses. The problem is to design the system so that, first, it follows a consistent, coherent conceptualization—a design model—and, second, so that the user can develop a mental model of that system—a user model—consistent with the design model.

Mental models seem a pervasive property of humans. I believe that people form internal, mental models of themselves and of the things and people with whom they interact. These models provide predictive and explanatory power for understanding the interaction. Mental models evolve naturally through interaction with the world and with the particular system under consideration (see Owen's description in Chapter 9 and the discussion by Riley, Chapter 7). These models are highly affected by the nature of the interaction, coupled with the person's prior knowledge and understanding. The models are neither complete nor accurate (see Norman, 1983c), but nonetheless they function to guide much human behavior.



There really are three different concepts to be considered: two mental, one physical. First, there is the conceptualization of the system held by designer; second, there is the conceptual model constructed by the user; and third, there is the physical image of the system from which the users develop their conceptual models. Both of the conceptual models are what have been called "mental models," but to separate the several different meanings of that term, I refer to these two aspects by different terms. I call the conceptual model held by the designer the *Design Model*, and the conceptual model formed by the user the *User's Model*. The third concept is the image resulting from the physical structure that has been built (including the documentation and instructions): I call that the *System Image*.

The Design Model is the conceptual model of the system to be built. Ideally, this conceptualization is based on the user's task, requirements, and capabilities. The conceptualization must also consider the user's background, experience, and the powers and limitations of the user's information processing mechanisms, most especially processing resources and short-term memory limits.

The user develops a mental model of the system—the *User's Model*. Note that the user model is not formed from the Design Model: It results from the way the user interprets the *System Image*. Thus, in many ways, the primary task of the designer is to construct an appropriate System Image, realizing that everything the user interacts with helps to form that image: the physical knobs, dials, keyboards, and displays, and the documentation, including instruction manuals, help facilities, text input and output, and error messages. The designer should want the User's Model to be compatible with the underlying conceptual model, the Design Model. And this can only happen through interaction with the System Image. These comments place a severe burden on the designer. If one hopes for the user to understand a system, to use it properly, and to enjoy using it, then it is up to the designer to make the System Image explicit, intelligible, consistent. And this goes for everything associated with the system. Remember too that people do not always read documentation, and so the major (perhaps entire) burden is placed on the image that the system projects.⁴

⁴ The story is actually more complex. The "user's model" can refer to two distinctive things: the individual user's own personal, idiosyncratic model (which is the meaning I intended); or the generalized "typical user" model that is what the designer develops to help in the formulation of the "Design Model." I jumped between these two different meanings in this paragraph. Finally, there is yet another model to worry about: the model that an intelligent program might construct of the person with which it is interacting. This too has been called a user model and is discussed by Mark in Chapter 11.

There do exist good examples of systems that present a System Image to the user in a clear, consistent fashion, following a carefully chosen conceptual model in such a way that the User's Model matches the Design Model. One example is the spreadsheet programs (starting with VISICALC), systems that match the conceptualizations of the targeted user, the accountant or budget planner. Another good example is the stack calculator, especially the early designs from Hewlett Packard. And a third example is the "office desk" metaphor followed in the Xerox Star, Apple Lisa and Macintosh workstations.

It is easier to design consistent Design Models for some things than for others. In general, the more specialized the tool, the higher the level at which a system operates, the easier the task. Spreadsheets are relatively straightforward. General purpose operating systems or programming languages are not. Whenever there is one single task and one set of users, the task of developing the conceptual model is much simplified. When the system is general purpose, with a relatively unlimited set of users and power, then the task becomes complex, perhaps undoable. In this case, it may be necessary to have conceptualizations that depend on the use to which the system is being put.

This discussion is meant to introduce the importance and the difficulties of conceptual models.⁵ Further discussion of these issues occurs throughout this book, but most especially in the chapters by diSessa (Chapter 10), Mark (Chapter 11), Owen (Chapter 9), and Riley (Chapter 7).

ON THE QUALITY OF HUMAN-COMPUTER INTERACTION

The theme of quality of the interaction and "conviviality" of the interface is important, a theme worth speaking of with force. So for the moment, let me move from a discussion of theories of action and

⁵ There has been a lot said, but little accomplished, on the nature and importance of mental models in the use of complex systems. The book, *Mental Models*, edited by Gentner and Stevens (1983) is perhaps the first attempt to spell out some of the issues. And Johnson-Laird's book (1983), with the same title, gets at one possible theoretical understanding of the mental models that people create and use in everyday life. At the time this is being written, the best publication on the role of a mental model in learning and using a complex system is the paper by Kieras and Bovair (1984).

conceptual models and speak of the qualitative nature of human-computer interaction. The details of the interaction matter, ease of use matters, but I want more than correct details, more than a system that is easy to learn or to use: I want a system that is enjoyable to use.

This is an important, dominating design philosophy, easier to say than to do. It implies developing systems that provide a strong sense of understanding and control. This means tools that reveal their underlying conceptual model and allow for interaction, tools that emphasize comfort, ease, and pleasure of use: for what Illich (1973) has called *convivial fools*. A major factor in this debate is the feeling of control that the user has over the operations that are being performed. A "powerful," "intelligent" system can lead to the well documented problems of "overautomation," causing the user to be a passive observer of operations, no longer in control of either what operations take place, or of how they are done. On the other hand, systems that are not sufficiently powerful or intelligent can leave too large a gap in the mappings from intention to action execution and from system state to psychological interpretation. The result is that operation and interpretation are complex and difficult, and the user again feels out of control, distanced from the system.

Laurel approaches this issue of control over one's activities from the perspective of drama in her chapter, *Interface as Mimesis* (Chapter 4). To Laurel, the critical aspect is "pleasurable engagement," by which she means the complete and full engagement of the person in pursuit of the "end cause" of the activity. The computer should be invisible to the user, acting as the means by which the person enters into the engagement, but avoiding intrusion into the ongoing thoughts and activities.

The Power of Tools

When I look around at instances of good system design—systems that I think have had profound influence upon the users, I find that what seems more important than anything else is that they are viewed as tools. That is, the system is deemed useful because it offers powerful tools that the user is able to apply constructively and creatively, with understanding. Here is a partial list of system innovations that follow these principles:

- *Smalltalk*. This language—and more importantly, the design philosophy used in getting there—emphasize the development of tools at an appropriate conceptual level, with object-oriented, message-passing software, where new instances or procedures

are derived from old instances, with derived (inherited) conditions and values, and with the operations visible as graphic objects, if you so want them to be (Goldberg, 1984; Tesler, 1981).

- *The Xerox Star computer.* A carefully done, psychologically motivated approach to the user interface, emphasizing a consistent, well-thought-through user model (Smith, Irby, Kimball, Verplank, & Harslem, 1982). The implementation has changed how we think of interfaces. The Star was heavily influenced by Smalltalk and it, in turn, led to the Apple *Lisa* and *Macintosh*.
- *UNIX.* The underlying philosophy is to provide a number of small, carefully crafted operations that can be combined in a flexible manner under the control of the user to do the task at hand. It is something like a construction set of computational procedures. The mechanisms that make this possible are a consistent data structure and the ability to concatenate programs (via "pipes" and input-output redirection). The interface suffers multiple flaws and is easily made the subject of much ridicule. But the interface has good ideas: aliases, shell scripts, pipes, terminal independence, and an emphasis on shared files and learning by browsing. Elsewhere I have scolded it for its shortcomings (Compton, 1984; Norman, 1981b), but we should not overlook its strengths.
- *Interlisp (and the Lisp machines).* Providing a powerful environment for Lisp program development, integrating editor, debugger, compiler, and interpreter, nowadays coupled with graphics and windows. To say nothing of DWIM — *Do What I Mean* (See Teitelman & Masinter, 1981).
- *Spreadsheets.* Merging the computational power of the computer with a clean, useful conceptual model, allowing the interface to drive the entire system, providing just the right tools for a surprising variety of applications.
- *Steamer.* A teaching system based on the concept of intelligent graphics that make visible to the student the operations of an otherwise abstract and complex steam generator system for large ships. (Hollan, Hutchins, & Weizman, 1984).

- *Bill Budge's Pinball Construction Set* (Budge, 1983). A game, but one that illustrates the toolkit notion of interface, for the user can manipulate the structures at will to create the game of choice. It is easy to learn, easy to use, yet powerful. There is no such thing as an illegal operation, there are no error messages—and no need for any. Errors are simply situations where the operation is not what is desired. No new concepts are in this game over those illustrated by the other items on this list, but the other examples require powerful computers, whereas this works on home machines such as the Apple II, thus bringing the concept to the home.

This list is idiosyncratic. It leaves out some important examples in favor of ones of lesser importance. Nonetheless, these are the items that have affected me the most. The major thing all these systems offer is a set of powerful tools to the user.

The Problem With Tools

The *Pinball Construction Set* illustrates some of the conflicts that tools present, especially conflict over how much intelligence should be present. Much as I enjoy manipulating the parts of the pinball sets, much as my 4-year-old son could learn to work it with almost no training or bother, neither of us are any good at constructing pinball sets. I can't quite get the parts in the right places: When I stretch a part to change its shape, I usually end up with an unworkable part. Balls get stuck in weird corners. The action is either too fast or too slow. Yes, it is easy to change each problem as it is discovered, but the number seems endless. I wish the tools were more intelligent—do as I am intending, not as I am doing. (This point is examined in more detail in Chapter 5 by Hutchins, Hollan, and Norman.)

Simple tools have problems because they can require too much skill from the user. Intelligent tools can have problems if they fail to give any indication of how they operate and of what they are doing. The user can feel like a bystander, watching while unexplained operations take place. The result is a feeling of lack of control over events. This is a serious problem, one that is well known to students of social psychology. It is a problem whether it occurs to the individual while interacting with colleagues, while a passenger in a runaway vehicle, or while using a computer. If we take the notion of "conviviality" seriously, we will develop tools that make visible their operations and assumptions. The argument really comes down to presenting an appropriate system image to the user, to assist the user's understanding

of what is going on: to keep the user in control. These are topics discussed in Mark's chapter (Chapter 11). They require, among other things, developing a good model of the user. In addition, the user must have a good user's model of the system.

When systems take too much control of the environment, they can cause serious social problems. Many observers have commented on the dehumanizing results of automation in the workplace. In part, this automatically results from the systems that take control away from the users. As Ehn and Kyng (1984) put it, such a result follows naturally when the office or workplace is thought of as a system, so that the computer reduces "the jobs of the workers to algorithmic procedures' minimizing the need for skill or control, and thereby the attractiveness of the workplace. The alternative view, that of tools, offers more control to the worker. For Eng and Kyng, tools "are under complete and continuous manual control of the worker, are fashioned for the use of the skilled worker to create products of good use quality, and are extensions of the accumulated knowledge of tools and materials of a given labour process." The problem arises over and over again as various workplaces become automated, whether it is the factory, the office, or the aviation cockpit. I believe the difficulties arise from the tension between the natural desire to want intelligent systems that can compensate for our inadequacies and the desire to feel in control of the outcome. Proponents of automatic systems do not wish to make the workplace less pleasant. On the contrary, they wish to improve it. And proponents of tools often wish for the power of the automated systems. (See Chapters 2, 19, and 21 by Bannon for further discussion of these issues.)

The Gulfs of Execution and Evaluation, Revisited

The stages of action play important roles in the analysis of the interface, for they define the psychological stages that need support from the interface. Moreover, the quality of the interaction probably depends heavily upon the "directness" of the relationship between the psychological and physical variables: just how the Gulfs of Figure 3.1 are bridged. The theory suggests that two of the mappings of Table 3.1 play critical roles: (a) the mapping from the psychological variables in which the goals are stated to the physical variables upon which the

control is actually exerted; (b) the mapping from the physical variables of the system to psychological variables. The easier and more direct these two mappings, the easier and more pleasant the learning and use of the interface, at least so goes the theory.⁶ In many ways, the design efforts must focus upon the mappings much more than the stages. This issue forms the focus of much of the discussion in the chapter by Hutchins, Hollan, and Norman (Chapter 5), where it is the mappings that are discussed explicitly as helping bridge the gulf between the demands of the machine and the thought processes and actions of the user. In that chapter the discussion soon turns to the qualitative feeling of control that can develop when one perceives that manipulation is directly operating upon the objects of concern to the user: The actions and the results occur instantaneously upon the same object. That chapter provides a start toward a more formal analysis of these qualitative feelings of "conviviality" or what Hutchins, Hollan, and Norman call "direct engagement" with the task.

The problem of level. *A major issue in the development of tools is to determine the proper level. Tools that are too primitive, no matter how much their power, are difficult to work with. The primitive commands of a Turing machine are of sufficient power to do any task doable on a computer, but who would ever want to program any real task with them? This is the "Turing tarpit" discussed in Chapter 5 by Hutchins, Hollan, and Norman. When I program a computer, I want a language that matches my level of thought or action. A programming language is precisely in the spirit of a tool: It is a set of operations and construction procedures that allows a machine to do anything doable, unrestricted by conventions or preconceived notions. The power of computers comes about in part because their languages do follow the tool formulation. But not everyone should do this kind of programming. Most people need higher-level tools, tools where the components are already closely matched to the task. On the other hand, tools that are at too high a level are too specialized. An apple-peeler is well matched to its purpose, but it has a restricted set of uses. Spelling checkers are powerful tools, but of little aid outside their domain. Specialized tools are invaluable when*

⁶ Streitz (1985) has expressed a similar view, stating that "An interactive computer system (ICS) is the more user-oriented the less discrepancies do exist between the relevant knowledge representations on the user's side and on the side of the ICS."

they match the level and intentions of the user, frustrating when they do not.

How do we determine the proper level of a tool? That is a topic that needs more study. There are strong and legitimate arguments against systems that are too specialized. Equally, there are strong arguments against tools that are too primitive, that operate at too low a level. We want higher-level tools that are crafted to the task. We need lower-level tools in order to create and modify higher-level ones. The level of the tool has to match the level of the intention. Again, easier to say than to do.

DESIGN ISSUES

Designing computer systems for people is especially difficult for a number of reasons. First, the number of variables and potential actions is large, possibly in the thousands. Second, the technology available today is limited: limited in the nature of what kinds of input mechanisms exist; limited in the form and variety of output; limited in the amount of affordable memory and computational power. This means that the various mappings (see Table 3.1) are particularly arbitrary. On the other hand, the computer has the potential to make visible much more of the operation of the system and, more importantly, to translate the system's operations into psychologically meaningful variables and displays than any other machine. But, as the opening sections of this chapter attempted to demonstrate, the problem is intrinsically difficult: It isn't just computers that are difficult to use, interaction with any complex device is difficult.

Any real system is the result of a series of tradeoffs that balance one design decision against another, that take into account time, effort, and expense. Almost always the benefits of a design decision along one dimension lead to deficits along some other dimension. The designer must consider the wide class of users, the physical limitations, the constraints caused by time and economics, and the limitations of the technology. Moreover, the science and engineering disciplines necessary for a proper design of the interface do not yet exist. So what is the designer to **do**? What do those of us who are developing the design principles need to do? In this section I review some of the issues, starting with a discussion of the need for approximate theory, moving to a discussion of the general nature of tradeoffs, and then to an exhortation to attend first to the first-order issues. In all of this, the goal is a

User-Centered Interface, which means providing intelligent, understandable, tools that bridge the gap between people and systems: convivial tools.

What Is It We Want in Computer Design?

Approximate science. In part we need a combined science and engineering discipline that guides the design, construction, and use of systems. An important point to realize is that *approximate methods suffice*, at least for most applications. This is true of most applied disciplines, from the linear model of transistor circuits to the stress analysis of bridges and buildings: The engineering models are only approximations to reality, but the answers are precise enough for the purpose. Note, of course, that the designer must know both the approximate model and its limits.

Consider an example from Psychology: the nature of short-term memory (STM). Even though there is still not an agreed upon theory of memory, and even though the exact nature of STM is still in doubt, quite a bit is known about the phenomena of STM. The following approximation captures a large portion of the phenomena of STM and is, therefore, a valuable tool for many purposes:

The five-slot approximate model of STM. *Short-term memory consists of 5 slots, each capable of holding one item (which might be a pointer to a complex memory structure). Each item decays with a half-life of 15 seconds. Most information is lost from STM as a result of interference, new information that takes up the available slots.*

Although the approximate model is clearly wrong in all its details, in most practical applications the details of STM do not matter: This approximate model can be very valuable. Other approximate models are easy to find. The time to find something can be approximated by assuming that one object can be examined within the fovea at any one time, and that saccades take place at approximately 5 per second. Reaction and decision times can be approximated by cycles of 100 milliseconds. The book by Card, Moran, and Newell (1983) provides sophisticated examples of the power of approximate models of human cognition. All these models can be criticized at the theoretical level. **But** they all provide numerical assessment of behavior that will be accurate enough for almost all applications.

Tradeoffs

Design is a series of tradeoffs: Assistance for one stage is apt to interfere with another. Any single design technique is apt to have its virtues along one dimension compensated by deficiencies along another. Each technique provides a set of tradeoffs. The lesson applies to almost any aspect of design. Add extra help for the unskilled user and you run the risk of frustrating the experienced user. Make the display screen larger and some tasks get better, but others get more confused. Display more information, and the time to paint the display goes up, the memory requirement goes up, programs become larger, bulkier, slower. It is well known that different tasks and classes of users have different needs and requirements.

The design choices depend on the technology being used, the class of users, and the goals of the design. The designers must decide which aspects of the interface should gain, which can be left wanting. This focus on the tradeoffs emphasizes that the design problem must be looked at as a whole, not in isolated pieces, for the optimal choice for one part of the problem will probably not be optimal for another. According to this view, there are no correct answers, only tradeoffs among alternatives.

It might be useful to point out that although there may not be any best solution to a problem in which the needs of different parts conflict, there is a worst solution. And even if no design is "best" along all dimensions, some designs are clearly better than others—along all dimensions. It clearly is possible to design a bad system. Equally, it is possible to avoid bad design.

The prototypical tradeoff: information versus time. One basic tradeoff pervades many design issues: *Factors that increase informativeness tend to decrease the amount of available workspace and system responsiveness.* On the one hand, the more informative and complete the display, the more useful when the user has doubts or lacks understanding. On the other hand, the more complete the display, the longer it takes to be displayed and the more space it must occupy physically. This tradeoff of amount of information versus space and time appears in many guises and is one of the major interface issues that must be handled (Norman, 1983a). To appreciate its importance, one has only to examine a few recent commercial offerings, highly touted for their innovative (and impressive) human factors design that were intended

to make the system easy and pleasurable to use, but which so degraded system response time that serious user complaints resulted. The term "user friendly" has taken on a negative meaning as a result of badly engineered tradeoffs, sacrificing utility, efficiency, and ease of use for the benefit of some hypothetical, ill-informed, first-time user.

It is often stated that current computer systems do not provide beginning users with sufficient information. However, the long, informative displays or sequence of questions, options, or menus that may make a system usable by the beginner are disruptive to the experienced user who knows exactly what action is to be specified and wishes to minimize the time and mental effort required to do the specification. The tradeoff here is not only between different needs, but between different stages of activity. After all, the extra information required by the beginner would not bother the experienced users if they could ignore it. However, this information usually cannot be ignored. It is apt to take excess time to be displayed or to use up valuable space on the display, in either case impeding the experienced users in executing and evaluating their actions. We pit the experienced user's requirement for ease of specification against the beginner's requirement for knowledge.

First- and second-order issues. One major tradeoff concerns just which aspects of the system will be worked on. With limited time and people, the design team has to make decisions: Some parts of the system will receive careful attention, others will not. Each different aspect of the design takes time, energy, and resources, none of which is apt to be readily available. Therefore, it is important to be able to distinguish the first order effects from the secondary effects—the big issues from the little issues.

I argue that it is the conceptual models that are of primary importance: the design model, the system image, the user's model. If you don't have the right design model, then all else fades to insignificance. Get the major issue right first—the Design Model and the System Image. Then, and only then, worry about the second order issues.

Example: *VISICALC*. At the time VISICALC was introduced, it represented a significant breakthrough in design. Bookkeepers and accountants were often wary of computers, especially those who were involved in small and medium size enterprises where they had to work alone, without the assistance of corps of programmers and computer specialists. VISICALC changed all this. It let the users work on their own terms, putting together a "spreadsheet" of figures, readily changing the numbers and watching the implications appear in the relevant spots.

It would be useful to explore the various design issues involved in the construction of **VISICALC**. The designers not only were faced with the creation of a conceptualization unlike anything else that existed, but they chose to do it on a relatively small and limited machine, one in which the two major languages available were **BASIC** and Assembler code, which could only display 24 rows of 40 columns worth of upper-case letters and digits. Yet, spreadsheets require matrices with hundreds of rows and columns of numerals. The success of **VISICALC** was due both to the power of the original conceptualization and the clever use of design techniques to overcome the limitations of the machine. Probably an important key to its success was that the design team consisted of just two people, one a user (at the time, he was a student in the Harvard Business School who needed a tool to do business analyses and projections), the other a programmer.

But look at the command structure used in **VISICALC**: cryptic, obscure, and unmeaningful. It is easy to make errors, difficult to remember the appropriate operations. The choice of command names could be used as an exercise in how not to do things, for they appear to be the typical conventions chosen by computer programmers, for computer programmers. The point of this is to note that **VISICALC** was a success story, despite the poor choice of command structure. Yes, **VISICALC** would have been much improved had the commands been better. People would have liked it better, users would have been happier. But the commands were a second-order issue. The designers of **VISICALC** were working with limited time, manpower, and budget: They were wise in concentrating on the important conceptualizations and letting the problems of command names go for later. I certainly do not wish to advocate the use of poor commands, but the names are second-order issues.

Why was the command structure less important than the overall conceptual structure? Two factors helped:

- The system was self-contained.
- The typical user was a frequent user.

First, **VISICALC** was a self-contained system. That is, many users of **VISICALC**, especially the first wave of users, used only **VISICALC**. They put the floppy disk containing **VISICALC** into the computer, turned it on, did their work, and then turned off the computer. Therefore, there were no conflicts between the command choices used by **VISICALC** and other programs. This eliminated one major source of difficulty. Second, most users of **VISICALC** were practiced, experienced users of the system. The prime audience of the system was the professional who worked with spreadsheet computations on a regular

basis. Therefore, the commands would be expected to be used frequently. And whenever there is much experience and practice, lack of meaning and consistency is not so important. Yes, the learning time might be long, but it only need take place once and then, once the commands have been learned well, they become automatic, causing no further difficulty. Choices of command names are especially critical when many different systems are to be used, each with its own cryptic, idiosyncratic choice of names. Problems arise when different systems are involved, oftentimes with similar functions that have different names and conventions, and with similar names that have different meanings. When a system is heavily used by beginners or casual users, then command names take on added significance.

Prescriptions for Design Principles

What is it that we need to do? What should we accomplish? What is the function of *Cognitive Engineering*? The list of things is long, for here we speak of creating an entirely new discipline, one moreover that combines two already complex fields: psychology and computer science. Moreover, it requires breaking new ground, for our knowledge of what fosters good interactions among people and between people and devices is young, without a well-developed foundation. We are going to need a good, solid technical grounding in the principles of human processing. In addition, we need to understand the more global issues that determine the essence of interaction. We need to understand the way that hardware affects the interaction: As Chapter 15 by Buxton points out, even subtle changes in hardware can make large changes in the usability of a system. And we need to explore the technology into far richer and more expressive domains than has so far been done.

On the one hand, we do need to go deeper into the details of the design. On the other hand, we need to determine some of the higher, overriding principles. The analysis of the stages of interaction moves us in the former direction, into the details of interaction. In this chapter I have raised a number of the issues relevant to the second issue: the higher, more global concerns of human–machine interaction. The general ideas and the global framework lead to a set of overriding design guidelines, not for guiding specific details of the design, but for structuring how the design process might proceed. Here are some prescriptions for design:

- *Create a science of user-centered design.* For this, we need principles that can be applied at the time of the design, principles that get the design to a pretty good state the first time around.

This requires sufficient design principles and simulation tools for establishing the design of an interface *before* constructing it. There will still have to be continual iterations, testing, and refinement of the interface—all areas of design need that—but the first pass ought to be close.

- *Take interface design seriously as an independent and important problem.* It takes at least three kinds of special knowledge to design an interface: first, knowledge of design, of programming and of the technology; second, knowledge of people, of the principles of mental computation, of communication, and of interaction; and third, expert knowledge of the task that is to be accomplished. Most programmers and designers of computer systems have the first kind of knowledge, but not the second or third. Most psychologists have the second, but not the first or third. And the potential user is apt to have the third, but not the first or second. As a result, if a computer system is to be constructed with a truly user-centered design, it will have to be done in collaboration with people trained in all these areas. We need either especially trained interface specialists or teams of designers, some members expert in the topic domain of the device, some expert in the mechanics of the device, and some expert about people. (This procedure is already in use by a number of companies: often those with the best interfaces, I might add.)
- *Separate the design of the interface from the design of the system.* This is the principle of modularization in design. It allows the previous point to work. Today, in most systems, everyone has access to control of the screen or mouse. This means that even the deepest, darkest, most technical systems programmer can send a message to the user when trouble arises: Hence arises my favorite mystical error message: "longjmp botch, core dump" or du Boulay's favorite compiler error message: "Fatal error in pass zero" (Draper & Norman, 1984; du Boulay & Matthew, 1984). It is only the interface module that should be in communication with the user, for it is only this module that can know which messages to give, which to defer, to know where on the screen messages should go without interfering with the main task, or to know the associated information that should be provided. Messages are interruptions (and sometimes reminders), in the sense described in the chapters by Cypher (Chapter 12) and Miyata and Norman (Chapter 13).

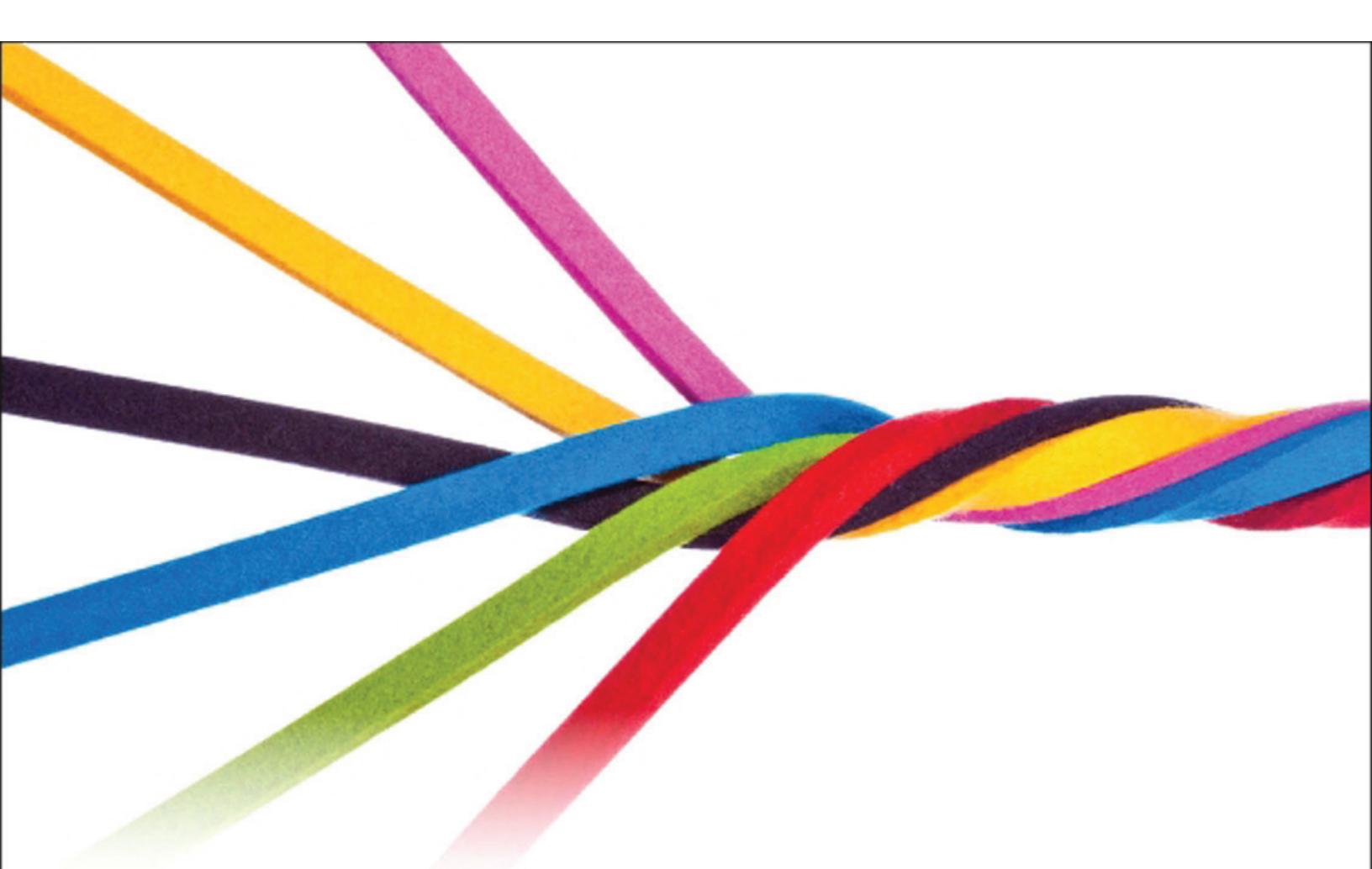
Because they affect the ongoing task, they have to be presented at the right time, at the right level of specification.

Modularity also allows for change: The system can change without affecting the interface; the interface can change without affecting the system. Different users may need different interfaces, even for the same task and the same system. Evaluations of the usability of the interface may lead to changes—the principle of iterative, interactive design—and this should be possible without disruption to the rest of the system. This is not possible if user interaction is scattered throughout the system: It is possible if the interface is a separate, independent module.

- ***Do user-centered system design: Start with the needs of the user.*** From the point of view of the user, the interface *is* the system. Concern for the nature of the interaction and for the user—these are the things that should force the design. Let the requirements for the interaction drive the design of the interface, let ideas about the interface drive the technology. The final design is a collaborative effort among many different disciplines, trading off the virtues and deficits of many different design approaches. But user-centered design emphasizes that the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming. The needs of the users should dominate the design of the interface, and the needs of the interface should dominate the design of the rest of the system.

ACKNOWLEDGMENTS

The chapter has been much aided by the comments of numerous people. I thank Eileen Conway for her aid with the illustrations. Julie Norman and Sondra Buffett provided extensive editorial comments for each of the numerous revisions. Liam Bannon, Steve Draper, and Dave Owen provided a number of useful comments and suggestions. Jonathan Grudin was most savage of the lot, and therefore the most helpful. And the Asilomar Workshop group provided a thorough reading, followed by two hours of intensive commentary. All this effort on the part of the critics led to major revision and reorganization. For all this assistance, I am grateful.



Human-Computer Interaction

An Empirical Research Perspective



I. Scott MacKenzie

Human-Computer Interaction

Scientific Foundations

4

In the last chapter, we examined a variety of interaction topics in HCI. By and large, the research methodology for studying these topics is empirical and scientific. Ideas are conceived, developed, and implemented and then framed as hypotheses that are tested in experiments. This chapter presents the enabling features of this methodology. Our goal is to establish the what, why, and how of research, with a focus on research that is both empirical and experimental. While much of the discussion is general, the examples are directed at HCI. We begin with the terminology surrounding research and empirical research.

4.1 What is research?

Research means different things to different people. “Being a researcher” or “conducting research” carries a certain elevated status in universities, colleges, and corporations. Consequently, the term research is bantered around in a myriad of situations. Often, the word is used simply to add weight to an assertion (“Our research shows that …”). While writing an early draft of this chapter, a television ad for an Internet service provider was airing in southern Ontario. The ad proclaimed, “Independent research proves [name_of_product] is the fastest and most reliable—period.”¹ One might wonder about the nature of the research, or of the independence and impartiality of the work. Of course, forwarding assertions to promote facts, observations, hypotheses, and the like is often the goal. But what is research? Surely, it is more than just a word to add force to a statement or opinion. To rise above conjecture, we demand evidence—evidence meeting a standard of credibility such that the statement is beyond dispute. Providing such credibility is the goal of research.

Returning to the word itself, research has at least three definitions. First, conducting research can be an exercise as simple as *careful or diligent search*.² So carefully searching one’s garden to find and remove weeds meets one standard of

¹Advertisement by Rogers Communications Inc. airing on television in southern Ontario during the winter of 2008/2009.

²www.merriam-webster.com.

conducting research. Or perhaps one undertakes a search on a computer to locate all files modified on a certain date. That's research. It's not the stuff of MSc or PhD theses, but it meets one definition of research.

The second definition of research is *collecting information about a particular subject*. So surveying voters to collect information on political opinions is conducting research. In HCI we might observe people interacting with an interface and collect information about their interactions, such as the number of times they consulted the manual, clicked the wrong button, retried an operation, or uttered an expletive. That's research.

The third definition is more elaborate: *research is investigation or experimentation aimed at the discovery and interpretation of facts and revision of accepted theories or laws in light of new facts*.

In this definition we find several key elements of research that motivate discussions in this book. We find the idea of *experimentation*. Conducting experiments is a central activity in a lot of HCI research. I will say more about this in the next chapter. In HCI research, an experiment is sometimes called a *user study*. The methodology is sometimes formal, sometimes ad hoc. A formal and standardized methodology is generally preferred because it brings consistency to a body of work and facilitates the review and comparison of research from different studies. One objective of this book is to promote the use of a consistent methodology for experimental research in HCI.

To be fair, the title of this book changed a few times on the way to press. Is the book about *experimental research*? Well, yes, a lot of it is, but there are important forms of HCI research that are non-experimental. So as not to exclude these, the focus shifted to *empirical research*, a broader term that encompasses both experimental and non-experimental methodologies. Among the latter is building and testing models of interaction, which we examine formally in Chapter 7.

Returning to research, the third definition speaks of *facts*. Facts are the building blocks of evidence, and it is evidence we seek in experimental research. For example, we might observe that a user committed three errors while entering a command with an interface. That's a fact. Of course, context is important. Did the user have prior experience with the interface, or with similar interfaces? Was the user a child or a computer expert? Perhaps we observed and counted the errors committed by a group of users while interacting with two different interfaces over a period of time. If they committed 15 percent more errors with one interface than with the other, the facts are more compelling (but, again, context is important). Collectively, the facts form an outward sign leading to evidence—evidence that one interface is better, or less error prone, than the other. Evidence testing is presented in more detail in Chapter 6, Hypothesis Testing. Note that *prove* or *proof* is not used here. In HCI research we don't prove things; we gather facts and formulate and test evidence.

The third definition mentions *theories* and *laws*. Theory has two common meanings. In the sense of Darwin's *theory of evolution* or Einstein's *theory of relativity*, the term theory is synonymous with *hypothesis*. In fact, one definition of theory is simply "a hypothesis assumed for the sake of argument or investigation." Of course,

through experimentation, these theories advanced beyond argument and investigation. The stringent demands of scientific inquiry confirmed the hypotheses of these great scientists. When confirmed through research, a theory becomes a *scientifically accepted body of principles that explain phenomena*.

A *law* is different from a theory. A law is more specific, more constraining, more formal, more binding. In the most exacting terms, a law is a relationship or phenomenon that is “invariable under given conditions.” Because variability is germane to human behavior, laws are of questionable relevance to HCI. Of course, HCI has laws. Take HCI’s best-known law as an example. *Fitts’ law* refers to a body of work, originally in human motor behavior (Fitts, 1954), but now widely used in HCI. Fitts’ work pertained to rapid-aimed movements, such as rapidly moving a cursor to an object and selecting it in a graphical user interface. Fitts himself never proposed a law. He proposed a model of human motor behavior. And by all accounts, that’s what Fitts’ law is—a model, a behavioral, descriptive, and predictive model. It includes equations and such for predicting the time to do point-select tasks. It is a law only in that other researchers took up the label as a celebration of the generality and importance of Fitts’ seminal work. We should all be so lucky. Fitts’ law is presented in more detail in Chapter 7.

Research, according to the third definition, involves *discovery*, *interpretation*, and *revision*. Discovery is obvious enough. That’s what we do—look for, or discover, things that are new and useful. Perhaps the discovery is a new style of interface or a new interaction technique. Interpretation and revision are central to research. Research does not proceed in a vacuum. Today’s research builds on what is already known or assumed. We interpret what is known; we revise and extend through discovery.

There are additional characteristics of research that are not encompassed in the dictionary definitions. Let’s examine a few of these.

4.1.1 Research must be published

Publication is the final step in research. It is also an essential step. Never has this rung as true as in the edict *publish or perish*. Researchers, particularly in academia, must publish. A weak or insufficient list of publications might spell disappointment when applying for research funds or for a tenure-track professorship at a university. Consequently, developing the skill to publish begins as a graduate student and continues throughout one’s career as a researcher, whether in academia or industry. The details and challenges in writing research papers are elaborated in Chapter 8.

Publishing is crucial, and for good reason. Until it is published, the knowledge gained through research cannot achieve its critical purpose—to extend, refine, or revise the existing body of knowledge in the field. This is so important that publication bearing a high standard of scrutiny is required. Not just any publication, but publication in archived peer-reviewed journals or conference proceedings. Research results are “written up,” submitted, and reviewed for their integrity, relevance, and contribution. The review is by peers—other researchers doing similar work. Are

the results novel and useful? Does the evidence support the conclusions? Is there a contribution to the field? Does the methodology meet the expected standards for research? If these questions are satisfactorily answered, the work has a good chance of acceptance and publication. Congratulations. In the end, the work is published and archived. *Archived* implies the work is added to the collection of related work accessible to other researchers throughout the world. This is the “existing body of knowledge” referred to earlier. The final step is complete.

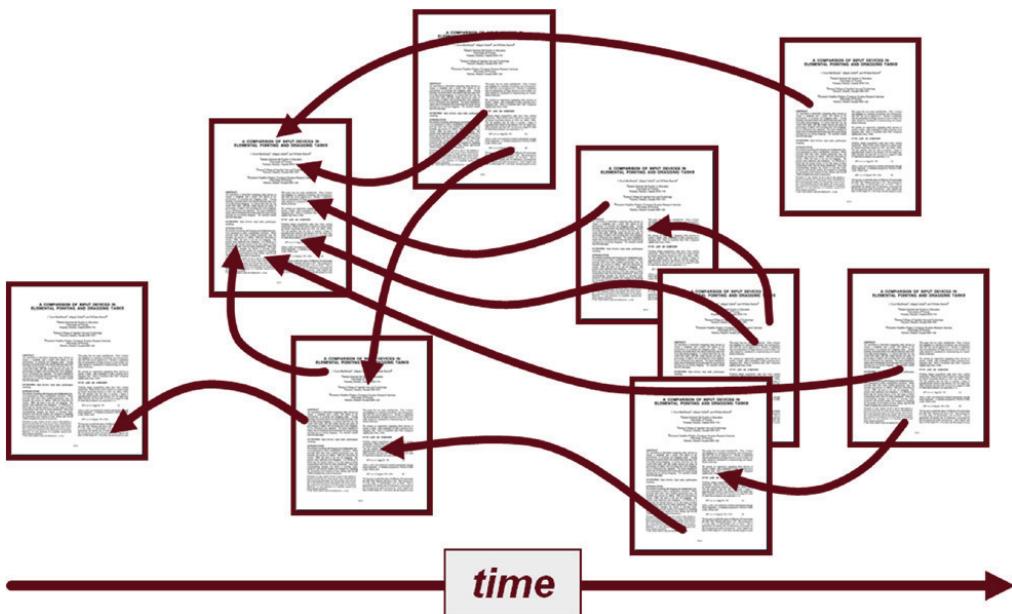
Research results are sometimes developed into bona fide inventions. If an individual or a company wishes to profit from their invention, then patenting is an option. The invention is disclosed in a patent application, which also describes previous related work (prior art), how the invention addresses a need, and the best mode of implementation. If the application is successful, the patent is granted and the inventor or company thereafter owns the rights to the invention. If another company wishes to use the invention for commercial purpose, they must enter into a license agreement with the patent holder. This side note is included only to make a small point: a patent is a publication. By patenting, the individual or company is not only retaining ownership of the invention but is also making it public through publication of the patent. Thus, patents meet the must-publish criterion for research.

4.1.2 Citations, references, impact

Imagine the World Wide Web without hyperlinks. Web pages would live in isolation, without connections between them. Hyperlinks provide the essential pathways that connect web pages to other web pages, thus providing structure and cohesion to a topic or theme. Similarly, it is hard to imagine the world’s body of published research without *citations* and *references*. Citations, like hyperlinks, connect research papers to other research papers. Through citations, a body of research takes shape. The insights and lessons of early research inform and guide later research. The citation itself is just an abbreviated tag that appears in the body of a paper, for example, “... as noted in earlier research (Smith and Jones, 2003)” or “... as confirmed by Green et al. [5].” These two examples are formatted differently and follow the requirements of the conference or journal. The citation is expanded into a full bibliographic entry in the reference list at the end of the paper. Formatting of citations and references is discussed in Chapter 8.

Citations serve many purposes, including supporting intellectual honesty. By citing previous work, researchers acknowledge that their ideas continue, extend, or refine those in earlier research. Citations are also important to back up assertions that are otherwise questionable, for example, “the number of tablet computer users worldwide now exceeds two billion [9].” In the Results section of a research paper, citations are used to compare the current results with those from earlier research, for example, “the mean time to formulate a search query was about 15 percent less than the time reported by Smith and Jones [5].”

Figure 4.1 provides a schematic of a collection of research papers. Citations are shown as arrows. It incorporates a timeline, so all arrows point to the left, to earlier

**FIGURE 4.1**

A collection of research papers with citations to earlier papers.

papers. One of the papers seems to have quite a few citations to it. The number of citations to a research paper is a measure of the paper's *impact*. If many researchers cite a single paper, there is a good chance the work described in the cited paper is both of high quality and significant to the field. This point is often echoed in academic circles: "The only objective and transparent metric that is highly correlated with the quality of a paper is the number of citations."³ Interestingly enough, citation counts are only recently easily available. Before services like Google Scholar emerged, citation counts were difficult to obtain.

Since citation counts are available for individual papers, they are also easy to compile for individual researchers. Thus, impact can be assessed for researchers as well as for papers. The most accepted single measure of the impact of a researcher's publication record is the *H-index*. If a researcher's publications are ordered by the number of citations to each paper, the H-index is the point where the rank equals the number of citations. In other words, a researcher with $H\text{-index} = n$ has n publications each with n or more citations. Physicist J. Hirsch first proposed the H-index in 2005 (Hirsch, 2005). H-index quantifies in a single number both research productivity (number of publications) and overall impact of a body of work (number of citations). Some of the strengths and weaknesses of the H-index, as a measure of impact, are elaborated elsewhere (MacKenzie, 2009a).

³Dianne Murray, General Editor, *Interacting with Computers*. Posted to chi-announcements@acm.org on Oct 8, 2008.

4.1.3 Research must be reproducible

Research that cannot be replicated is useless. Achieving an expected standard of reproducibility, or repeatability, is therefore crucial. This is one reason for advancing a standardized methodology: it enforces a process for conducting and writing about the research that ensures sufficient detail is included to allow the results to be replicated. If skilled researchers care to test the claims, they will find sufficient guidance in the methodology to reproduce, or replicate, the original research. This is an essential characteristic of research.

Many great advances in science and research pertain to methodology. A significant contribution by Louis Pasteur (1822–1895), for example, was his use of a consistent methodology for his research in microbiology (Day and Gastel, 2006, pp. 8–9). Pasteur’s experimental findings on germs and diseases were, at the time, controversial. As Pasteur realized, the best way to fend off skepticism was to empower critics—other scientists—to see for themselves. Thus, he adopted a methodology that included a standardized and meticulous description of the materials and procedure. This allowed his experiments and findings to be replicated. A researcher questioning a result could redo the experiment and therefore verify or refute the result. This was a crucial advance in science. Today, reviewers of manuscripts submitted for publication are often asked to critique the work on this very point: “Is the work replicable?” “No” spells certain rejection.

One of the most cited papers in publishing history is a method paper. Lowry et al.’s, 1951 paper “Protein Measurement With the Folin Phenol Reagent” has garnered in excess of 200,000 citations (Lowry, Rosenbrough, Farr, and Randall, 1951).⁴ The paper describes a method for measuring proteins in fluids. In style, the paper reads much like a recipe. The method is easy to read, easy to follow, and, importantly, easy to reproduce.

4.1.4 Research versus engineering versus design

There are many ways to distinguish research from engineering and design. Researchers often work closely with engineers and designers, but the skills and contributions each brings are different. Engineers and designers are in the business of building things. They create products that strive to bring together the best in *form* (design emphasis) and *function* (engineering emphasis). One can imagine that there is certain tension, even a trade-off, between form and function. Finding the right balance is key. However, sometimes the balance tips one way or the other. When this occurs, the result is a product or a feature that achieves one (form or function) at the expense of the other. An example is shown in Figure 4.2a. The image shows part of a notebook computer, manufactured by a well-known computer company. By most accounts, it is a typical notebook computer. The image shows part of the keyboard and the built-in pointing device, a touchpad. The touchpad design (or is it engineering?) is interesting. It is seamlessly embedded in the system chassis.

⁴See <http://scholar.google.com>.

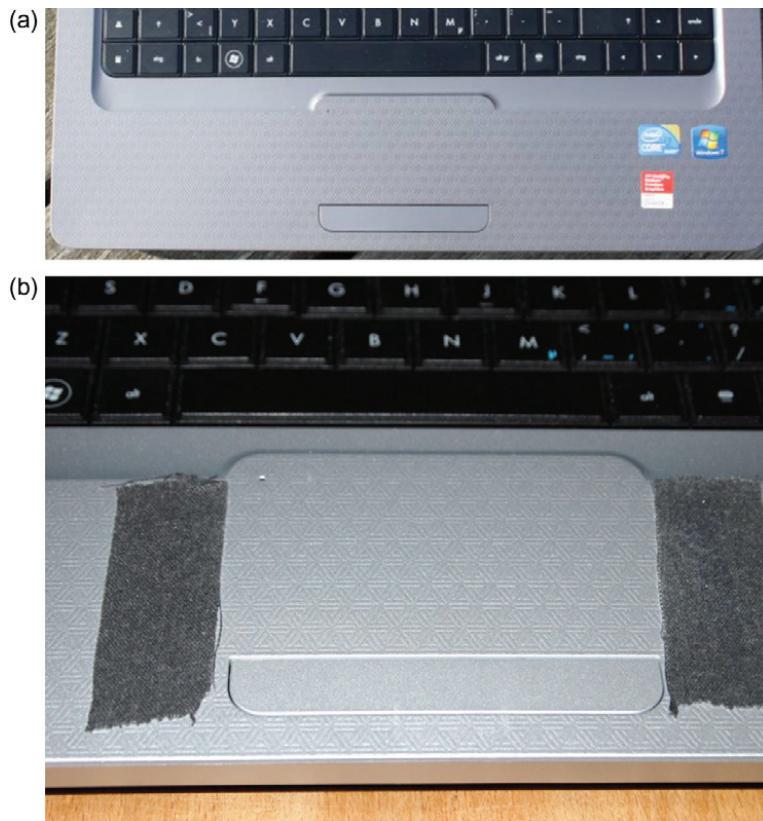


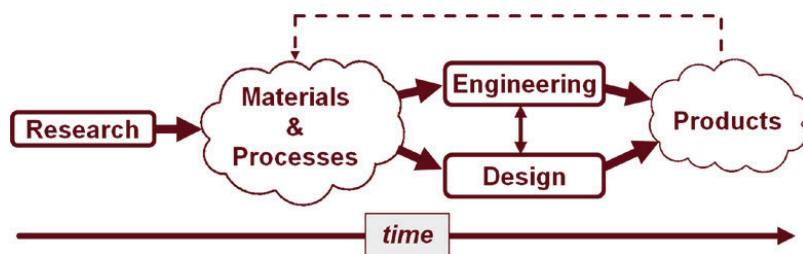
FIGURE 4.2

Form trumping function: (a) Notebook computer. (b) Duct tape provides tactile feedback indicating the edge of the touchpad.

The look is elegant—smooth, shiny, metallic. But something is wrong. Because the mounting is seamless and smooth, tactile feedback at the sides of the touchpad is missing. While positioning a cursor, the user has no sense of when his or her finger reaches the edge of the touchpad, except by observing that the cursor ceases to move. This is an example of form trumping function. One user's solution is shown in Figure 4.2b. Duct tape added on each side of the touchpad provides the all-important tactile feedback.⁵

Engineers and designers work in the world of products. The focus is on designing complete systems or products. Research is different. Research tends to be narrowly focused. Small ideas are conceived of, prototyped, tested, then advanced or discarded. New ideas build on previous ideas and, sooner or later, good ideas are refined into the building blocks—the materials and processes—that find their way into products. But research questions are generally small in scope. Research tends to be incremental, not monumental.

⁵For an amusing example of function trumping form, visit Google Images using “Rube Goldberg simple alarm clock.”

**FIGURE 4.3**

Timeline for research, engineering, and design.

Engineers and designers also work with prototypes, but the prototype is used to assess alternatives at a relatively late stage: as part of product development. A researcher's prototype is an early mock-up of an idea, and is unlikely to directly appear in a product. Yet the idea of using prototypes to inform or assess is remarkably similar, whether for research or for product development. The following characterization by Tim Brown (CEO of design firm IDEO) is directed at designers, but is well aligned with the use of prototypes for research:

Prototypes should command only as much time, effort, and investment as are needed to generate useful feedback and evolve an idea. The more “finished” a prototype seems, the less likely its creators will be to pay attention to and profit from feedback. The goal of prototyping isn’t to finish. It is to learn about the strengths and weaknesses of the idea and to identify new directions that further prototypes might take (Brown, 2008, p. 3).

One facet of research that differentiates it from engineering and design is the timeline. Research precedes engineering and design. Furthermore, the march forward for research is at a slower pace, without the shackles of deadlines. Figure 4.3 shows the timeline for research, engineering, and design. Products are the stuff of deadlines. Designers and engineers work within the corporate world, developing products that sell, and hopefully sell well. The raw materials for engineers and designers are materials and processes that already exist (dashed line in Figure 4.3) or emerge through research.

The computer mouse is a good example. It is a hugely successful product that, in many ways, defines a generation of computing, post 1981, when the Xerox Star was introduced. But in the 1960s the mouse was just an idea. As a prototype it worked well as an input controller to maneuver a tracking symbol on a graphics display. Engelbart's invention (English et al., 1967) took nearly 20 years to be engineered and designed into a successful product.

Similar stories are heard today. Apple Computer Inc., long known as a leader in innovation, is always building a better mousetrap. An example is the *iPhone*, introduced in June, 2007. And, evidently, the world has beaten a path to Apple's door.⁶ Notably, “with the iPhone, Apple successfully brought together decades

⁶The entire quotation is “Build a better mousetrap and the world will beat a path to your door” and is attributed to American essayist Ralph Waldo Emerson (1803–1882).

of research” (Selker, 2008). Many of the raw materials of this successful product came by way of low-level research, undertaken well before Apple’s engineers and designers set forth on their successfully journey. Among the iPhone’s interaction novelties is a two-finger *pinch* gesture for zooming in and out. New? Perhaps, but Apple’s engineers and designers no doubt were guided or inspired by research that came before them. For example, multi-touch gestures date back to at least the 1980s (Buxton, Hill, and Rowley, 1985; Hauptmann, 1989). What about changing the aspect ratio of the display when the device is tilted? New? Perhaps not. Tilt, as an interaction technique for user interfaces, dates back to the 1990s (B. Harrison et al., 1998; Hinckley et al., 2000; Rekimoto, 1996). These are just two examples of research ideas that, taken alone, are small scale. While engineers and designers strive to build better systems or products, in the broadest sense, researchers provide the raw materials and processes engineers and designers work with: stronger steel for bridges, a better mouse for pointing, a better algorithm for a search engine, a more natural touch interface for mobile phones.

4.2 What is empirical research?

By prefixing research with *empirical*, some powerful new ideas are added. According to one definition, empirical means *originating in or based on observation or experience*. Simple enough. Another definition holds that empirical means *relying on experience or observation alone, often without due regard for system and theory*. This is interesting. These words suggest researchers should be guided by direct observations and experiences about phenomena, without prejudice to, or even consideration of, existing theories. This powerful idea is a guiding principle in science—not to be blinded by preconceptions. Here’s an example. Prior to the 15th century, there was a prevailing *system* or *theory* that celestial bodies revolved around the earth. The Polish scientist Nicolas Copernicus (1473–1543) found evidence to the contrary. His work was empirical. It was based on observation without bias toward, influence by, or due regard to, existing theory. He observed, he collected data, he looked for patterns and relationships in the data, and he found evidence within the data that cut across contemporary thinking. His empirical evidence led to one of the great achievements in modern science—a heliocentric cosmology that placed the sun, rather than the earth, at the center of the solar system. Now that’s a nice *discovery* (see the third definition of research at the beginning of this chapter). In HCI and other fields of research, discoveries are usually more modest.

By another definition, empirical means *capable of being verified or disproved by observation or experiment*. These are strong words. An HCI research initiative is framed by hypotheses—assertions about the merits of an interface or an interaction technique. The assertions must be sufficiently clear and narrow to enable verification or disproval by gathering and testing evidence. This means using language in an assertion that speaks directly to empirical, observable, quantifiable aspects of the interaction. I will expand on this later in this chapter in the discussion on research questions.

4.3 Research methods

There are three common approaches, or methods, for conducting research in HCI and other disciplines in the natural and social sciences: the *observational method*, the *experimental method*, and the *correlational method*. All three are empirical as they are based on observation or experience. But there are differences and these follow from the objectives of the research and from the expertise and style of the researcher. Let's examine each method.

4.3.1 Observational method

Observation is the starting point for this method. In conducting empirical research in HCI, it is essential to observe humans interacting with computers or computer-embedded technology of some sort. The observational method encompasses a collection of common techniques used in HCI research. These include interviews, field investigations, contextual inquiries, case studies, field studies, focus groups, think aloud protocols, storytelling, walkthroughs, cultural probes, and so on. The approach tends to be qualitative rather than quantitative. As a result, observational methods achieve *relevance* while sacrificing *precision* (Sheskin, 2011, p. 76). Behaviors are studied by directly observing phenomena in a natural setting, as opposed to crafting constrained behaviors in an artificial laboratory setting. Real world phenomena are high in relevance, but lack the precision available in controlled laboratory experiments.

Observational methods are generally concerned with discovering and explaining the reasons underlying human behavior. In HCI, this is the *why* or *how* of the interaction, as opposed to the *what*, *where*, or *when*. The methods focus on human thought, feeling, attitude, emotion, passion, sensation, reflection, expression, sentiment, opinion, mood, outlook, manner, style, approach, strategy, and so on. These human qualities can be studied through observational methods, but they are difficult to measure. The observations are more likely to involve note-taking, photographs, videos, or audio recordings rather than measurement. Measurements, if gathered, tend to use categorical data or simple counts of phenomena. Put another way, observational methods tend to examine and record the quality of interaction rather than quantifiable human performance.

4.3.2 Experimental method

With the experimental method (also called the *scientific method*), knowledge is acquired through controlled experiments conducted in laboratory settings. Acquiring knowledge may imply gathering new knowledge, but it may also mean studying existing knowledge for the purpose of verifying, refuting, correcting, integrating, or extending. In the relevance-precision dichotomy, it is clear where controlled experiments lie. Since the tasks are artificial and occur in a controlled laboratory setting, relevance is diminished. However, the control inherent in the

methodology brings precision, since extraneous factors—the diversity and chaos of the real world—are reduced or eliminated.

A controlled experiment requires at least two variables: a *manipulated variable* and a *response variable*. In HCI, the manipulated variable is typically a property of an interface or interaction technique that is presented to participants in different configurations. Manipulating the variable simply refers to systematically exposing participants to different configurations of the interface or interaction technique. To qualify as a controlled experiment, at least two configurations are required. Thus, comparison is germane to the experimental method. This point deserves further elaboration. In HCI, we often hear of a system or design undergoing a “usability evaluation” or “user testing.” Although these terms often have different meanings in different contexts, such evaluations or tests generally do not follow the experimental method. The reason is simple: there is no manipulated variable. This is mentioned only to distinguish a usability evaluation from a *user study*. Undertaking a user study typically implies conducting a controlled experiment where different configurations of a system are tested and compared. A “usability evaluation,” on the other hand, usually involves assessing a single user interface for strengths and weaknesses. The evaluation might qualify as research (“collecting information about a particular subject”), but it is not experimental research. I will return to this point shortly. A manipulated variable is also called an *independent variable* or *factor*.

A response variable is a property of human behavior that is observable, quantifiable, and measurable. The most common response variable is time, often called *task completion time* or some variation thereof. Given a task, how long do participants take to do the task under each of the configurations tested? There are, of course, a multitude of other behaviors that qualify as response variables. Which ones are used depend on the characteristics of the interface or interaction technique studied in the research. A response variable is also called a *dependent variable*. Independent variables and dependent variables are explored in greater detail in Chapter 5.

HCI experiments involve humans, so the methodology employed is borrowed from experimental psychology, a field with a long history of research involving humans. In a sense, HCI is the beneficiary of this more mature field. The circumstances manipulated in a psychology experiment are often quite different from those manipulated in an HCI experiment, however. HCI is narrowly focused on the interaction between humans and computing technology, while experimental psychology covers a much broader range of the human experience.

It is naïve to think we can simply choose to focus on the experimental method and ignore qualities of interaction that are outside the scope of the experimental procedure. A full and proper user study—an experiment with human participants— involves more than just measuring and analyzing human performance. We engage observational methods by soliciting comments, thoughts, and opinions from participants. Even though a task may be performed quickly and with little or no error, if participants experience fatigue, frustration, discomfort, or another *quality* of interaction, we want to know about it. These qualities of interaction may not appear in the numbers, but they cannot be ignored.

One final point about the experimental method deserves mention. A controlled experiment, if designed and conducted properly, often allows a powerful form of conclusion to be drawn from the data and analyses. The relationship between the independent variable and the dependent variable is one of *cause and effect*; that is, the manipulations in the interface or interaction techniques are said to have *caused* the observed differences in the response variable. This point is elaborated in greater detail shortly. Cause-and-effect conclusions are not possible in research using the observational method or the correlational method.

4.3.3 Correlational method

The correlational method involves looking for relationships between variables. For example, a researcher might be interested in knowing if users' privacy settings in a social networking application are related to their personality, IQ, level of education, employment status, age, gender, income, and so on. Data are collected on each item (privacy settings, personality, etc.) and then relationships are examined. For example, it might be apparent in the data that users with certain personality traits tend to use more stringent privacy settings than users with other personality traits.

The correlational method is characterized by quantification since the magnitude of variables must be ascertained (e.g., age, income, number of privacy settings). For nominal-scale variables, categories are established (e.g., personality type, gender). The data may be collected through a variety of methods, such as observation, interviews, on-line surveys, questionnaires, or measurement. Correlational methods often accompany experimental methods, if questionnaires are included in the experimental procedure. Do the measurements on response variables suggest relationships by gender, by age, by level of experience, and so on?

Correlational methods provide a balance between relevance and precision. Since the data were not collected in a controlled setting, precision is sacrificed. However, data collected using informal techniques, such as interviews, bring relevance—a connection to real-life experiences. Finally, the data obtained using correlational methods are circumstantial, not causal. I will return to this point shortly.

This book is primarily directed at the experimental method for HCI research. However, it is clear in the discussions above that the experimental method will often include observational methods and correlational methods.

4.4 Observe and measure

Let's return to the foundation of empirical research: observation.

4.4.1 Observation

The starting point for empirical research in HCI is to observe humans interacting with computers. But how are observations made? There are two possibilities. Either

another human is the observer or an apparatus is the observer. A human observer is the experimenter or investigator, not the human interacting with the computer. Observation is the precursor to *measurement*, and if the investigator is the observer, then measurements are collected manually. This could involve using a log sheet or notebook to jot down the number of events of interest observed. Events of interest might include the number of times the user clicked a button or moved his or her hand from the keyboard to the mouse. It might involve observing users in a public space and counting those who are using mobile phones in a certain way, for example, while walking, while driving, or while paying for groceries at a checkout counter. The observations may be broken down by gender or some other attribute of interest.

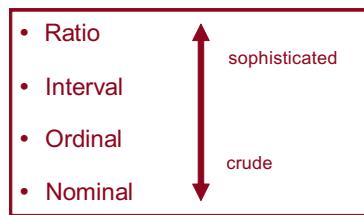
Manual observation could also involve timing by hand the duration of activities, such as the time to type a phrase of text or the time to enter a search query. One can imagine the difficulty in manually gathering measurements as just described, not to mention the inaccuracy in the measurements. Nevertheless, manual timing is useful for preliminary testing, sometimes called *pilot testing*.

More often in empirical research, the task of observing is delegated to the apparatus—the computer. Of course, this is a challenge in some situations. As an example, if the interaction is with a digital sports watch or automated teller machine (ATM), it is not possible to embed data collection software in the apparatus. Even if the apparatus is a conventional desktop computer, some behaviors of interest are difficult to detect. For example, consider measuring the number of times the user's attention switches from the display to the keyboard while doing a task. The computer is not capable of detecting this behavior. In this case, perhaps an eye tracking apparatus or camera could be used, but that adds complexity to the experimental apparatus. Another example is clutching with a mouse—lifting and repositioning the device. The data transmitted from a mouse to a host computer do not include information on clutching, so a conventional host system is not capable of observing and recording this behavior. Again, some additional apparatus or sensing technology may be devised, but this complicates the apparatus. Or a human observer can be used. So depending on the behaviors of interest, some ingenuity might be required to build an apparatus and collect the appropriate measurements.

If the apparatus includes custom software implementing an interface or interaction technique, then it is usually straightforward to record events such as key presses, mouse movement, selections, finger touches, or finger swipes and the associated timestamps. These data are stored in a file for follow-up analyses.

4.4.2 Measurement scales

Observation alone is of limited value. Consider observations about rain and flowers. In some locales, there is ample rain but very few flowers in April. This is followed by less rain and a full-blown field of flowers in May. The observations may inspire anecdote (*April showers bring May flowers*), but a serious examination of patterns for rain and flowers requires measurement. In this case, an observer located in a garden would observe, measure, and record the amount of rain and

**FIGURE 4.4**

Scales of measurement: nominal, ordinal, interval, and ratio. Nominal measurements are considered simple, while ratio measurements are sophisticated.

the number of flowers in bloom. The measurements might be recorded each day during April and May, perhaps by several observers in several gardens. The measurements are collected, together with the means, tallied by month and analyzed for “significant differences” (see Chapter 6). With measurement, anecdotes turn to empirical evidence. The observer is now in a position to quantify the amount of rain and the number of flowers in bloom, separately for April and May. The added value of measurement is essential for science. In the words of engineer and physicist Lord Kelvin (1824–1907), after whom the Kelvin scale of temperature is named, “[Without measurement] your knowledge of it is of a meager and unsatisfactory kind.”⁷

As elaborated in many textbooks on statistics, there are four scales of measurement: nominal, ordinal, interval, and ratio. Organizing this discussion by these four scales will help. Figure 4.4 shows the scales along a continuum with nominal scale measurements as the least sophisticated and ratio-scale measurements as the most sophisticated. This follows from the types of computations possible with each measurement, as elaborated below.

The nature, limitations, and abilities of each scale determine the sort of information and analyses possible in a research setting. Each is briefly defined below.

4.4.3 Nominal

A measurement on the nominal scale involves arbitrarily assigning a code to an attribute or a category. The measurement is so arbitrary that the code needn’t be a number (although it could be). Examples are automobile license plate numbers, codes for postal zones, job classifications, military ranks, etc. Clearly, mathematical manipulations on nominal data are meaningless. It is nonsense, for example, to

⁷The exact and full quote, according to several online sources, is “When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge of it is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced it to the stage of science.”

compute the mean of several license plate numbers. Nominal data identify mutually exclusive categories. Membership or exclusivity is meaningful, but little else. The only relationship that holds is equivalence, which exists between entities in the same class. Nominal data are also called *categorical data*.

If we are interested in knowing whether males and females differ in their use of mobile phones, we might begin our investigation by observing people and assigning each a code of “M” for male, “F” for female. Here, the attribute is gender and the code is M or F. If we are interested in handedness, we might observe the writing habits of users and assign codes of “LH” for left-handers and “RH” for right-handers. If we are interested in scrolling strategies, we might observe users interacting with a GUI application and categorize them according to their scrolling methods, for example as “MW” for mouse wheel, “CD” for clicking and dragging the scroll-bar, or “KB” for keyboard.

Nominal data are often used with frequencies or counts—the number of occurrences of each attribute. In this case, our research is likely concerned with the difference in the counts between categories: “Are males or females more likely to ...?”, “Do left handers or right handers have more difficulty with ...?”, or “Are Mac or PC users more inclined to ...?” Bear in mind that while the attribute is categorical, the count is a ratio-scale measurement (discussed shortly).

Here is an example of nominal scale attributes using real data. Attendees of an HCI research course were dispatched to several locations on a university campus. Their task was to observe, categorize, and count students walking between classes. Each student was categorized by gender (male, female) and by whether he or she was using a mobile phone (not using, using). The results are shown in Figure 4.5. A total of 1,527 students were observed. The split by gender was roughly equal (51.1% male, 48.9% female). By mobile phone usage, 13.1 percent of the students (200) were observed using their mobile phone while walking.

The research question in Figure 4.5 is as follows: are males or females more likely to use a mobile phone as they walk about a university campus? I will demonstrate how to answer this question in Chapter 6 on Hypothesis Testing.

Gender	Mobile Phone Usage		Total	%
	Not Using	Using		
Male	683	98	781	51.1%
Female	644	102	746	48.9%
Total	1327	200	1527	
%	86.9%	13.1%		

FIGURE 4.5

Two examples of nominal scale data: gender (male, female) and mobile phone usage (not using, using).

How many email messages do you receive each day?

1. None (I don't use email)
2. 1-5 per day
3. 6-25 per day
4. 26-100 per day
5. More than 100 per day

FIGURE 4.6

Example of a questionnaire item soliciting an ordinal response.

4.4.4 Ordinal data

Ordinal scale measurements provide an order or ranking to an attribute. The attribute can be any characteristic or circumstance of interest. For example, users might be asked to try three global positioning systems (GPS) for a period of time and then rank the systems by preference: first choice, second choice, third choice. Or users could be asked to consider properties of a mobile phone such as price, features, cool-appeal, and usability, and then order the features by personal importance. One user might choose usability (first), cool-appeal (second), price (third), and then features (fourth). The main limitation of ordinal data is that the interval is not intrinsically equal between successive points on the scale. In the example just cited, there is no innate sense of how much more important usability is over cool-appeal or whether the difference is greater or less than that between, for example, cool-appeal and price.

If we are interested in studying users' e-mail habits, we might use a questionnaire to collect data. Figure 4.6 gives an example of a questionnaire item soliciting ordinal data. There are five rankings according to the number of e-mail messages received per day. It is a matter of choice whether to solicit data in this manner or, in the alternative, to ask for an estimate of the number of e-mail messages received per day. It will depend on how the data are used and analyzed.

Ordinal data are slightly more sophisticated than nominal data since comparisons of *greater than* or *less than* are possible. However, it is not valid to compute the mean of ordinal data.

4.4.5 Interval data

Moving up in sophistication, interval data have equal distances between adjacent values. However, there is no absolute zero. The classic example of interval data is temperature measured on the Fahrenheit or Celsius scale. Unlike ordinal data, it is meaningful to compute the mean of interval data, for example, the mean mid-day temperature during the month of July. Ratios of interval data are not meaningful, however. For example, one cannot say that 20°C is twice as warm as 10°C.

In HCI, interval data are commonly used in questionnaires where a response on a linear scale is solicited. An example is a Likert Scale (see Figure 4.7), where verbal responses are given a numeric code. In the example, verbal responses are

Please indicate your level of agreement with the following statements.					
	Strongly disagree	Mildly disagree	Neutral	Mildly agree	Strongly agree
It is safe to talk on a mobile phone while driving.	1	2	3	4	5
It is safe to read a text message on a mobile phone while driving.	1	2	3	4	5
It is safe to compose a text message on a mobile phone while driving.	1	2	3	4	5

FIGURE 4.7

A set of questionnaire items organized in a Likert Scale. The responses are examples of interval scale data.

symmetric about a neutral, central value with the gradations between responses more or less equal. It is this last quality—equal gradations between responses—that validates calculating the mean of the responses across multiple respondents.

There is some disagreement among researchers on the assumption of equal gradations between the items in Figure 4.7. Do respondents perceive the difference between, say, 1 and 2 (strongly disagree and mildly disagree) the same as the difference between, say, 2 and 3 (mildly disagree and neutral)? Attaching verbal tags to numbers is likely to bring qualitative and highly personal interpretations to the responses. There is evidence that respondents perceive items at the extremes of the scale as farther apart than items in the center (Kaptein, Nass, and Markopoulos, 2010). Nevertheless, the graduation between responses is much more similar here than between the five ordinal responses in Figure 4.6. One remedy for non-equal gradations in Likert-scale response items is simply to instruct respondents to interpret the items as equally spaced.

Examples of Likert Scale questionnaire items in HCI research papers are as follows: Bickmore and Picard, 2004; Dautenhahn et al., 2006; Garau et al., 2003; Guy, Ur, Ronen, Perer, and Jacovi, 2011; Wobbrock, Chau, and Myers, 2007.

4.4.6 Ratio data

Ratio-scale measurements are the most sophisticated of the four scales of measurement. Ratio data have an absolute zero and support a myriad of calculations to

summarize, compare, and test the data. Ratio data can be added, subtracted, multiplied, divided; means, standard deviations, and variances can be computed. In HCI, the most common ratio-scale measurement is time—the time to complete a task. But generally, all physical measurements are also ratio-scale, such as the distance or velocity of a cursor as it moves across a display, the force applied by a finger on a touchscreen, and so on. Many social variables are also ratio-scale, such as a user's age or years of computer experience.

Another common ratio-scale measurement is count (noted above). Often in HCI research, we count the number of occurrences of certain human activities, such as the number of button clicks, the number of corrective button clicks, the number of characters entered, the number of incorrect characters entered, the number of times an option is selected, the number of gaze shifts, the number of hand movements between the mouse and keyboard, the number of task retries, the number of words in a search query, etc. Although we tend to give time special attention, it too is a count—the number of seconds or minutes elapsed as an activity takes place. These are all ratio-scale measurements.

The expressive nature of a count is improved through *normalization*; that is, expressing the value as a count *per something*. So for example, knowing that a 10-word phrase was entered in 30 seconds is less revealing than knowing that the rate of entry was $10/0.5 = 20$ words per minute (wpm). The main benefit of normalizing counts is to improve comparisons. It is easy to compare 20 wpm for one method with 23 wpm for another method—the latter method is faster. It is much harder to compare 10 words entered in 30 seconds for one method with 14 words entered in 47 seconds for another method.

As another example, let's say two errors were committed while entering a 50-character phrase of text. Reporting the occurrence of two errors reveals very little, unless we also know the length of the phrase. Even so, comparisons with results from another study are difficult. (What if the other study used phrases of different lengths?) However, if the result is reported as a $2/50 = 4\%$ error rate, there is an immediate sense of the meaning, magnitude, and relevance of the human performance measured, and as convention has it, the other study likely reported error rates in much the same way. So where possible, normalize counts to make the measurements more meaningful and to facilitate comparisons.

An example in the literature is an experiment comparing five different text entry methods (Magerkurth and Stenzel, 2003). For speed, results were reported in “words per minute” (that’s fine); however, for accuracy, results were reported as the number of errors committed. Novice participants, for example, committed 24 errors while using multi-tap (Magerkurth and Stenzel, 2003, Table 2). While this number is useful for comparing results within the experiment, it provides no insight as to how the results compare with those in related research. The results would be more enlightening if normalized for the amount of text entered and reported as an “error rate (%),” computed as the number of character errors divided by the total number of characters entered times 100.

4.5 Research questions

In HCI, we conduct experimental research to answer (and raise!) questions about a new or existing user interface or interaction technique. Often the questions pertain to the relationship between two variables, where one variable is a circumstance or condition that is manipulated (an interface property) and the other is an observed and measured behavioral response (task performance).

The notion of posing or answering questions seems simple enough, but this is tricky because of the human element. Unlike an algorithm operating on a data set, where the time to search, sort, or whatever is the same with each try, people exhibit variability in their actions. This is true both from person to person and for a single person repeating a task. The result is always different! This variability affects the confidence with which we can answer *research questions*. To gauge the confidence of our answers, we use statistical techniques, as presented in Chapter 6, Hypothesis Testing.

Research questions emerge from an inquisitive process. The researcher has an idea and wishes to see if it has merit. Initial thoughts are fluid and informal:

- Is it viable?
- Is it as good as or better than current practice?
- What are its strengths and weaknesses?
- Which of several alternatives is best?
- What are the human performance limits and capabilities?
- Does it work well for novices, for experts?
- How much practice is required to become proficient?

These questions are unquestionably relevant, since they capture a researcher's thinking at the early stages of a research project. However, the questions above suffer a serious deficiency: They are not testable. The goal, then, is to move forward from the loose and informal questions above to questions more suitable for empirical and experimental enquiry.

I'll use an example to show how this is done. Perhaps a researcher is interested in text entry on touchscreen phones. Texting is something people do a lot. The researcher is experienced with the Qwerty soft keyboard on touchscreen phones, but finds it error prone and slow. Having thought about the problem for a while, an idea emerges for a new technique for entering text. Perhaps it's a good idea. Perhaps it's really good, better than the basic Qwerty soft keyboard (QSK). Being motivated to do research in HCI, the researcher builds a prototype of the entry technique and fiddles with the implementation until it works fine. The researcher decides to undertake some experimental research to evaluate the idea. What are the research questions? Perhaps the following capture the researcher's thinking:

- Is the new technique any good?
- Is the new technique better than QSK?

- Is the new technique faster than QSK?
- Is the new technique faster than QSK after a bit of practice?
- Is the measured entry speed (in words per minute) higher for the new technique than for a QSK after one hour of use?

From top to bottom, the questions are progressively narrower and more focused. Expressions like “any good” or “better than,” although well intentioned, are problematic for research. Remember observation and measurement? How does one measure “better than”? Farther down the list, the questions address qualities that are more easily observed and measured. Furthermore, since they are expressed across alternative designs, comparisons are possible. The last question speaks very specifically to entry speed measured in words per minute, to a comparison between two methods, and to a criterion for practice. This is a testable research question.

4.6 Internal validity and external validity

At this juncture we are in a position to consider two important properties of experimental research: *internal validity* and *external validity*. I’ll use the research questions above to frame the discussion. Two of the questions appear in the plot in Figure 4.8. The *x*-axis is labeled Breadth of Question or, alternatively, External Validity. The *y*-axis is labeled Accuracy of Answer or, alternatively, Internal Validity.

The question

Is the new technique better than QSK?

is positioned as high in breadth (that’s good!) yet answerable with low accuracy (that’s bad!). As already noted, this question is not testable in an empirical sense. Attempts to answer it directly are fraught with problems, because we lack a methodology to observe and measure “better than” (even though finding better interfaces is the final goal).

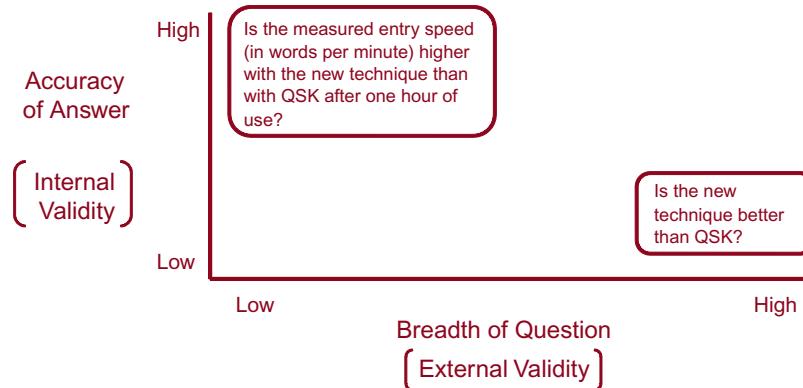


FIGURE 4.8

Graphical comparison of Internal Validity and External Validity.

The other, more detailed question

Is the measured entry speed (in words per minute) higher with the new technique than with QSK after one hour of use?

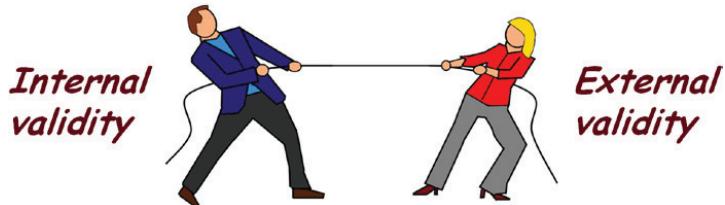
is positioned as low in breadth (that's bad!) yet answerable with high accuracy (that's good!). The question is testable, which means we can craft a methodology to answer it through observation and measurement. Unfortunately, the narrow scope of the question brings different problems. Focusing on entry speed is fine, but what about other aspects of the interaction? What about accuracy, effort, comfort, cognitive load, user satisfaction, practical use of the technique, and so on? The question excludes consideration of these, hence the low breadth rating.

The alternative labels for the axes in [Figure 4.8](#) are internal validity and external validity. In fact, the figure was designed to set up discussion on these important terms in experimental research.

Internal validity (definition) is the extent to which an effect observed is due to the test conditions. For the example, an *effect* is simply the difference in entry speed between the new technique and QSK. If we conduct an experiment to measure and compare the entry speed for the two techniques, we want confidence that the difference observed was actually due to inherent differences between the techniques. Internal validity captures this confidence. Perhaps the difference was due to something else, such as variability in the responses of the participants in the study. Humans differ. Some people are predisposed to be meticulous, while others are carefree, even reckless. Furthermore, human behavior—individually or between people—can change from one moment to the next, for no obvious reason. Were some participants tested early in the day, others late in the day? Were there any distractions, interruptions, or other environmental changes during testing? Suffice it to say that any source of variation beyond that due to the inherent properties of the test conditions tends to compromise internal validity. High internal validity means the effect observed really exists.

External validity (definition) is the extent to which experimental results are generalizable to other people and other situations. *Generalizable* clearly speaks to *breadth* in [Figure 4.8](#). To the extent the research pursues broadly framed questions, the results tend to be broadly applicable. But there is more. Research results that apply to “other people” imply that the participants involved were representative of a larger intended population. If the experiment used 18- to 25-year-old computer literate college students, the results might generalize to middle-aged computer literate professionals. But they might not generalize to middle-aged people without computer experience. And they likely would not apply to the elderly, to children, or to users with certain disabilities. In experimental research, random sampling is important for generalizability; that is, the participants selected for testing were drawn at random from the desired population.

Generalizable to “other situations” means the experimental *environment* and *procedures* were representative of real world situations where the interface or

**FIGURE 4.9**

There is tension between internal validity and external validity. Improving one comes at the expense of the other.

(Sketch courtesy of Bartosz Bajer)

technique will be used. If the research studied the usability of a GPS system for taxi drivers or delivery personnel and the experiment was conducted in a quiet, secluded research lab, there may be a problem with external validity. Perhaps a different experimental environment should be considered. Research on text entry where participants enter predetermined text phrases with no punctuation symbols, no uppercase characters, and without any ability to correct mistakes, may have problem with external validity. Again, a different experimental procedure should be considered.

The scenarios above are overly dogmatic. Experiment design is an exercise in compromise. While speaking in the strictest terms about high internal validity and high external validity, in practice one is achieved at the expense of the other, as characterized in Figure 4.9.

To appreciate the tension between internal and external validity, two additional examples are presented. The first pertains to the experimental environment. Consider an experiment that compares two remote pointing devices for presentation systems. To improve external validity, the experimental environment mimics expected usage. Participants are tested in a large room with a large presentation-size display, they stand, and they are positioned a few meters from the display. The other participants are engaged to act as an audience by attending and sitting around tables in the room during testing. There is no doubt this environment improves external validity. But what about internal validity? Some participants may be distracted or intimidated by the audience. Others might have a tendency to show off, impress, or act out. Such behaviors introduce sources of variation outside the realm of the devices under test, and thereby compromise internal validity. So our effort to improve external validity through environmental considerations may negatively impact internal validity.

A second example pertains to the experimental procedure. Consider an experiment comparing two methods of text entry. In an attempt to improve external validity, participants are instructed to enter whatever text they think of. The text may include punctuation symbols and uppercase and lowercase characters, and participants can edit the text and correct errors as they go. Again, external validity is improved since this is what people normally do when entering text. However, internal validity is compromised because behaviors are introduced that are not directly related to the text entry techniques—behaviors such as pondering (What should

I enter next?) and fiddling with commands (How do I move the cursor back and make a correction? How is overtype mode invoked?). Furthermore, since participants generate the text, errors are difficult to record since there is no “source text” with which to compare the entered text. So here again we see the compromise. The desire to improve external validity through procedural considerations may negatively impact internal validity.

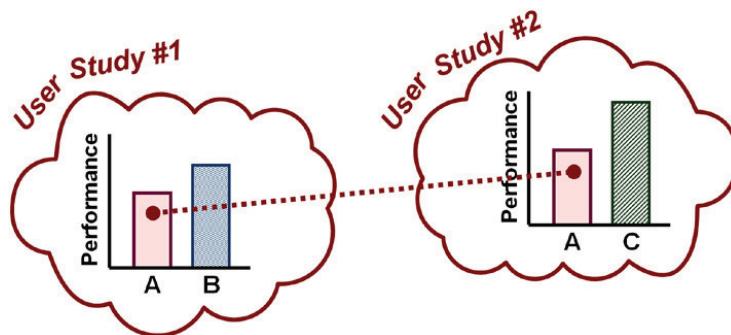
Unfortunately, there is no universal remedy for the tension between internal and external validity. At the very least, one must acknowledge the limitations. Formulating conclusions that are broader than what the results suggest is sure to raise the ire of reviewers. We can strive for the best of both worlds with a simple approach, however. Posing multiple narrow (testable) questions that cover the range of outcomes influencing the broader (untestable) questions will increase both internal and external validity. For example, a technique that is fast, accurate, easy to learn, easy to remember, and considered comfortable and enjoyable by users is generally better. Usually there is a positive correlation between the testable and untestable questions; i.e., participants generally find one UI better than another if it is faster and more accurate, takes fewer steps, is more enjoyable, is more comfortable, and so on.

Before moving on, it is worth mentioning *ecological validity*, a term closely related to external validity. The main distinction is in how the terms are used. Ecological validity refers to the methodology (using materials, tasks, and situations typical of the real world), whereas external validity refers to the outcome (obtaining results that generalize to a broad range of people and situations).

4.7 Comparative evaluations

Evaluating new ideas for user interfaces or interaction techniques is central to research in human-computer interaction. However, evaluations in HCI sometimes focus on a single idea or interface. The idea is conceived, designed, implemented, and evaluated—but not compared. The research component of such an evaluation is questionable. Or, to the extent the exercise is labeled research, it is more aligned with the second definition of research noted earlier: “collecting information about a particular subject.”

From a research perspective, our third definition is more appealing, since it includes the ideas of experimentation, discovery, and developing theories of interaction. Certainly, more meaningful and insightful results are obtained if a *comparative evaluation* is performed. In other words, a new user interface or interaction technique is designed and implemented and then compared with one or more alternative designs to determine which is faster, more accurate, less confusing, more preferred by users, etc. The alternatives may be variations in the new design, an established design (a baseline condition), or some combination of the two. In fact, the testable research questions above are crafted as comparisons (e.g., “Is Method A faster than Method B for …?”), and for good reason. A controlled experiment must include at least one independent variable and the independent variable must have at

**FIGURE 4.10**

Including a baseline condition serves as a check on the methodology and facilitates the comparison of results between user studies.

least two levels or test conditions. Comparison, then, is inherent in research following the experimental method discussed earlier. The design of HCI experiments is elaborated further in Chapter 5.

The idea of including an established design as a baseline condition is particularly appealing. There are two benefits. First, the baseline condition serves as a check on the methodology. Baseline conditions are well traveled in the research literature, so results in a new experiment are expected to align with previous results. Second, the baseline condition allows results to be compared with other studies. The general idea is shown in Figure 4.10. The results from two hypothetical user studies are shown. Both user studies are comparative evaluations and both include condition A as a baseline. Provided the methodology was more or less the same, the performance results in the two studies should be the same or similar for the baseline condition. This serves not only as a check on the methodology but also facilitates comparisons between the two user studies. A quick look at the charts suggests that condition C out-performs condition B. This is an interesting observation because condition C was evaluated in one study, condition B in another.

Consider the idea cited earlier of comparing two remote pointing devices for presentation systems. Such a study would benefit by including a conventional mouse as a baseline condition.⁸ If the results for the mouse are consistent with those found in other studies, then the methodology was probably okay, and the results for the remote pointing devices are likely valid. Furthermore, conclusions can often be expressed in terms of the known baseline condition, for example, “Device A was found to be about 8 percent slower than a conventional mouse.”

The value in conducting a comparative study was studied in research by Tohidi et al. (2006), who tested the hypothesis that a comparative evaluation yields more insight than a one-of evaluation. In their study, participants were assigned to groups and were asked to manually perform simple tasks with climate control interfaces

⁸The example cited earlier on remote pointing devices included a conventional mouse as a baseline condition (MacKenzie and Jusoh, 2001).

(i.e., thermostats). There were three different interfaces tested. Some of the participants interacted with just one interface, while others did the same tasks with all three interfaces. The participants interacting with all three interfaces consistently found more problems and were more critical of the interfaces. They were also less prone to inflate their subjective ratings. While this experiment was fully qualitative—human performance was not measured or quantified—the message is the same: a comparative evaluation yields more valuable and insightful results than a single-interface evaluation.

4.8 Relationships: circumstantial and causal

I noted above that looking for and explaining interesting relationships is part of what we do in HCI research. Often a controlled experiment is designed and conducted specifically for this purpose, and if done properly a particular type of conclusion is possible. We can often say that the condition manipulated in the experiment *caused* the changes in the human responses that were observed and measured. This is a *cause-and-effect relationship*, or simply a *causal relationship*.

In HCI, the variable manipulated is often a nominal-scale attribute of an interface, such as device, entry method, feedback modality, selection technique, menu depth, button layout, and so on. The variable measured is typically a ratio-scale human behavior, such as task completion time, error rate, or the number of button clicks, scrolling events, gaze shifts, etc.

Finding a causal relationship in an HCI experiment yields a powerful conclusion. If the human response measured is vital in HCI, such as the time it takes to do a common task, then knowing that a condition tested in the experiment reduces this time is a valuable outcome. If the condition is an implementation of a novel idea and it was compared with current practice, there may indeed be reason to celebrate. Not only has a causal relationship been found, but the new idea improves on existing practice. This is the sort of outcome that adds valuable knowledge to the discipline; it moves the state of the art forward.⁹ This is what HCI research is all about!

Finding a relationship does not necessarily mean a causal relationship exists. Many relationships are *circumstantial*. They exist, and they can be observed, measured, and quantified. But they are not causal, and any attempt to express the relationship as such is wrong. The classic example is the relationship between smoking and cancer. Suppose a research study tracks the habits and health of a large number of people over many years. This is an example of the correlational method of research mentioned earlier. In the end, a relationship is found between smoking and cancer: cancer is more prevalent in the people who smoked. Is it correct to conclude from the study that smoking *causes* cancer? No. The relationship observed is

⁹Reporting a non-significant outcome is also important, particularly if there is reason to believe a test condition might improve an interface or interaction technique. Reporting a non-significant outcome means that, at the very least, other researchers needn't pursue the idea further.

circumstantial, not causal. Consider this: when the data are examined more closely, it is discovered that the tendency to develop cancer is also related to other variables in the data set. It seems the people who developed cancer also tended to drink more alcohol, eat more fatty foods, sleep less, listen to rock music, and so on. Perhaps it was the increased consumption of alcohol that caused the cancer, or the consumption of fatty foods, or something else. The relationship is circumstantial, not causal. This is not to say that *circumstantial relationships* are not useful. Looking for and finding a circumstantial relationship is often the first step in further research, in part because it is relatively easy to collect data and look for circumstantial relationships.

Causal relationships emerge from controlled experiments. Looking for a causal relationship requires a study where, among other things, participants are selected randomly from a population and are randomly assigned to test conditions. A random assignment ensures that each group of participants is the same or similar in all respects except for the conditions under which each group is tested. Thus, the differences that emerge are more likely due to (*caused by*) the test conditions than to environmental or other circumstances. Sometimes participants are balanced into groups where the participants in each group are screened so that the groups are equal in terms of other relevant attributes. For example, an experiment testing two input controllers for games could randomly assign participants to groups or balance the groups to ensure the range of gaming experience is approximately equal.

Here is an HCI example similar to the smoking versus cancer example: A researcher is interested in comparing multi-tap and predictive input (*T9*) for text entry on a mobile phone. The researcher ventures into the world and approaches mobile phone users, asking for five minutes of their time. Many agree. They answer a few questions about experience and usage habits, including their preferred method of entering text messages. Fifteen multi-tap users and 15 *T9* users are found. The users are asked to enter a prescribed phrase of text while they are timed. Back in the lab, the data are analyzed. Evidently, the *T9* users were faster, entering at a rate of 18 words per minute, compared to 12 words per minute for the multi-tap users. That's 50 percent faster for the *T9* users! What is the conclusion? There is a relationship between method of entry and text entry speed; however, the relationship is circumstantial, not causal. It is reasonable to report what was done and what was found, but it is wrong to venture beyond what the methodology gives. Concluding from this simple study that *T9* is faster than multi-tap would be wrong. Upon inspecting the data more closely, it is discovered that the *T9* users tended to be more tech-savvy: they reported considerably more experience using mobile phones, and also reported sending considerably more text messages per day than the multi-tap users who, by and large, said they didn't like sending text messages and did so very infrequently.¹⁰ So the difference observed may be due to prior experience and usage habits, rather than to inherent differences in the text entry methods. If there is a genuine interest in determining if one text entry method

¹⁰Although it is more difficult to determine, perhaps technically savvy users were more willing to participate in the study. Perhaps the users who declined to participate were predominantly multi-tap users.

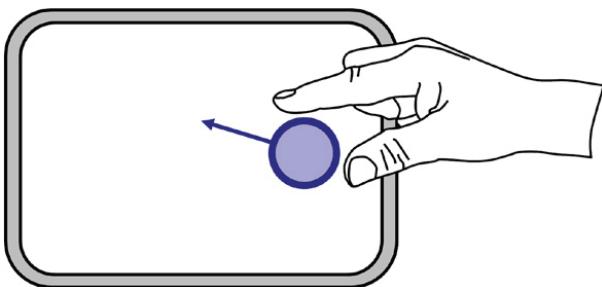
is faster than another, a controlled experiment is required. This is the topic of the next chapter.

One final point deserves mention. Cause and effect conclusions are not possible in certain types of controlled experiments. If the variable manipulated is a *naturally occurring attribute* of participants, then cause and effect conclusions are unreliable. Examples of naturally occurring attributes include gender (female, male), personality (extrovert, introvert), handedness (left, right), first language (e.g., English, French, Spanish), political viewpoint (left, right), and so on. These attributes are legitimate independent variables, but they cannot be manipulated, which is to say, they cannot be assigned to participants. In such cases, a cause and effect conclusion is not valid because it is not possible to avoid confounding variables (defined in Chapter 5). Being a male, being an extrovert, being left-handed, and so on always brings forth other attributes that systematically vary across levels of the independent variable. Cause and effect conclusions are unreliable in these cases because it is not possible to know whether the experimental effect was due to the independent variable or to the confounding variable.

4.9 Research topics

Most HCI research is not about designing products. It's not even about designing applications for products. In fact, it's not even about design or products. Research in HCI, like in most fields, tends to nip away at the edges. The march forward tends to be incremental. The truth is, most new research ideas tend to build on existing ideas and do so in modest ways. A small improvement to this, a little change to that. When big changes do arise, they usually involve bringing to market, through engineering and design, ideas that already exist in the research literature. Examples are the finger flick and two-finger gestures used on touchscreen phones. Most users likely encountered these for the first time with the Apple iPhone. The gestures seem like bold new advances in interaction, but, of course, they are not. The flick gesture dates at least to the 1960s. Flicks are clearly seen in use with a light pen in the videos of Sutherland's Sketchpad, viewable on YouTube. They are used to terminate a drawing command. Two-finger gestures date at least to the 1970s. Figure 4.11 shows Herot and Weinzapfel's (1978) two-finger gesture used to rotate a virtual knob on a touch-sensitive display. As reported, the knob can be rotated to within 5 degrees of a target position. So what might seem like a bold new advance is often a matter of good engineering and design, using ideas that already exist.

Finding a *research topic* is often the most challenging step for graduate students in HCI (and other fields). The expression “ABD” for “all but dissertation” is a sad reminder of this predicament. Graduate students sometimes find themselves in a position of having finished all degree requirements (e.g., coursework, a teaching practicum) without nailing down the big topic for dissertation research. Students might be surprised to learn that seasoned researchers in universities and industry also struggle for that next big idea. Akin to writer’s block, the harder one tries, the

**FIGURE 4.11**

A two-finger gesture on a touch-sensitive display is used to rotate a virtual knob.

(Adapted from Herot and Weinzapfel, 1978)

less likely is the idea to appear. I will present four tips to overcome “researcher’s block” later in this section. First, I present a few observations on ideas and how and where they arise.

4.9.1 Ideas

In the halcyon days after World War II, there was an American television show, a situation comedy, or sitcom, called *The Many Loves of Dobie Gillis* (1959–1963). Much like *Seinfeld* many years later, the show was largely about, well, nothing. Dobie’s leisurely life mostly focused on getting rich or on endearing a beautiful woman to his heart. Each episode began with an idea, a scheme. The opening scene often placed Dobie on a park bench beside *The Thinker*, the bronze and marble statue by French sculptor Auguste Rodin (1840–1917). (See Figure 4.12.) After some pensive moments by the statue, Dobie’s idea, his scheme, would come to him. It would be nice if research ideas in HCI were similarly available and with such assurance as were Dobie’s ideas. That they are not is no cause for concern, however. Dobie’s plans usually failed miserably, so we might question his approach to formulating his plans. Is it possible that *The Thinker*, in his pose, is more likely to inspire writer’s block than the idea so desperately sought? The answer may be yes, but there is little science here. We are dealing with human thought, inspiration, creativity, and a milieu of other human qualities that are poorly understood, at best.

If working hard to find a good idea doesn’t work, perhaps a better approach is to relax and just get on with one’s day. This seems to have worked for the ancient Greek scholar Archimedes (287–212 BC) who is said to have effortlessly come upon a brilliant idea as a solution to a problem. As a scientist, Archimedes was called upon to determine if King Hiero’s crown was pure gold or if it was compromised with a lesser alloy. One solution was to melt the crown, separating the constituent parts. This would destroy the crown—not a good idea. Archimedes’ idea was simple, and he is said to have discovered it while taking a bath. Yes, taking a bath, rather than sitting for hours in *The Thinker*’s pose. He realized—in an instant—that the volume of water displaced as he entered the bathtub must equal the volume

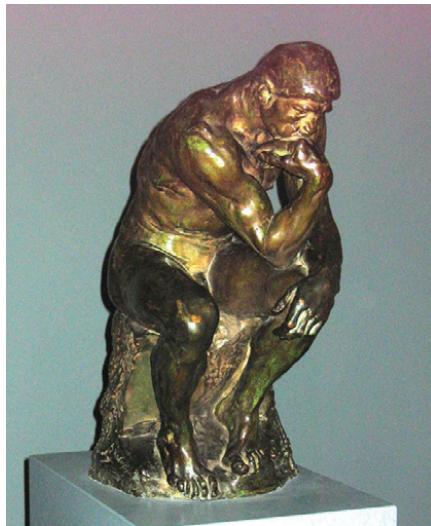
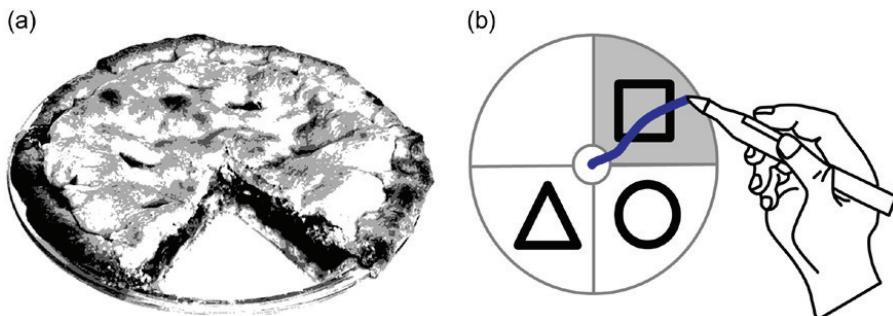


FIGURE 4.12

Rodin's *The Thinker* often appeared in the opening scenes of the American sitcom *The Many Loves of Dobie Gillis*.

of his body. Immersing the crown in water would similarly yield the crown's volume, and this, combined with the crown's weight, would reveal the crown's density. If the density of the crown equaled the known density of gold, the King's crown was pure gold—problem solved. According to the legend, Archimedes was so elated at his moment of revelation that he jumped from his bath and ran nude about the streets of Syracuse shouting “Eureka!” (“I found it!”).

While legends make good stories, we are not likely to be as fortunate as Archimedes in finding a good idea for an HCI research topic. Inspiration is not always the result of single moment of revelation. It is often gradual, with sources unknown or without a conscious and recognizable connection to the problem. Recall Vannevar Bush's memex, described in the opening chapter of this book. Memex was a concept. It was never built, even though Bush described the interaction with memex in considerable detail. We know memex today as hypertext and the World Wide Web. But where and how did Bush get his idea? The starting point is having a problem to solve. The problem of interest to Bush was coping with ever-expanding volumes of information. Scientists like Bush needed a convenient way to access this information. But how? It seems Bush's inspiration for memex came from... Let's pause for a moment, lest we infer Bush was engaged in a structured approach to problem solving. It is not likely that Bush went to work one morning intent on solving the problem of information access. More than likely, the idea came without deliberate effort. It may have come flittingly, in an instant, or gradually, over days, weeks, or months. Who knows? What is known, however, is that the idea did not arise from nothing. Ideas come from the human experience. This is why in HCI we often read about things like “knowledge in the head and knowledge in the world”

**FIGURE 4.13**

Pie menus in HCI: (a) The inspiration? (b) HCI example.

(Adapted from G. Kurtenbach, 1993)

(Norman, 1988, ch. 3) or metaphor and analogy (Carroll and Thomas, 1982). The context for inspiration is the human experience. So what was the source of Bush's inspiration for memex? The answer is in Bush's article, and also in Chapter 1.

Are there other examples relevant to HCI? Sure. Twitter co-founder Jack Dorsey is said to have come up with the idea for the popular micro-blogging site while sitting on a children's slide in a park eating Mexican food.¹¹ What about pie menus in graphical user interfaces? Pie menus, as an alternative to linear menus, were first proposed by Don Hopkins at the University of Maryland in 1988 (cited in Callahan et al., 1988). We might wonder about the source of Hopkins' inspiration (see Figure 4.13).

See also student exercises 4-2 and 4-3 at the end of this chapter.

4.9.2 Finding a topic

It is no small feat to find an interesting research topic. In the following paragraphs, four tips are offered on finding a topic suitable for research. As with the earlier discussion on the cost and frequency of errors (see Figure 3-46), there is little science to offer here. The ideas follow from personal experience and from working with students and other researchers in HCI.

4.9.3 Tip #1: Think small!

At a conference recently, I had an interesting conversation with a student. He was a graduate student in HCI. "Have you found a topic for your research," I asked. "Not really," he said. He had a topic, but only in a broad sense. Seems his supervisor had funding for a large research project related to aviation. The topic, in a general sense, was to develop an improved user interface for an air traffic control system. He was stuck. Where to begin? Did I have any ideas for him? Well, actually, no I didn't. Who wouldn't be stuck? The task of developing a UI for an air traffic control system is huge. Furthermore, the project mostly involves engineering and

¹¹New York Times Oct 30, 2010, p BU1.

design. Where is the research in designing an improved system of any sort? What are the research questions? What are the experimental variables? Unfortunately, graduate students are often saddled with similar big problems because a supervisor's funding source requires it. The rest of our discussion focused on narrowing the problem—in a big way. Not to some definable sub-system, but to a small aspect of the interface or interaction. The smaller, the better.

The point above is to think small. On finding that big idea, the advice is... forget it. Once you shed that innate desire to find something really significant and important, it's amazing what will follow. If you have a small idea, something that seems a little useful, it's probably worth pursuing as a research project. Pursue it and the next thing you know, three or four related interaction improvements come to mind. Soon enough, there's a dissertation topic in the works. So don't hesitate to think small.

4.9.4 Tip #2: Replicate!

An effective way to get started on research is to replicate an existing experiment from the HCI literature. This seems odd. Where is the research in simply replicating what has already been done? Of course, there is none. But there is a trick. Having taught HCI courses many times over many years, I know that students frequently get stuck finding a topic for the course's research project. Students frequently approach me for suggestions. If I have an idea that seems relevant to the student's interests, I'll suggest it. Quite often (usually!) I don't have any particular idea. If nothing comes to mind, I take another approach. The student is advised just to study the HCI literature—research papers from the CHI proceedings, for example—and find some experimental research on a topic of interest. Then just replicate the experiment. Is that okay, I am asked. Sure, no problem.

The trick is in the path to replicating. Replicating a research experiment requires a lot of work. The process of studying a research paper and precisely determining what was done, then implementing it, testing it, debugging it, doing an experiment around it, and so on will empower the student—the researcher—with a deep understanding of the issues, much deeper than simply reading the paper. This moves the line forward. The stage is set. Quite often, a new idea, a new twist, emerges. But it is important not to *require* something new. The pressure in that may backfire. Something new may emerge, but this might not happen until late in the process, or after the experiment is finished. So it is important to avoid a requirement for novelty. This is difficult, because it is germane to the human condition to strive for something new. Self-doubt may bring the process to a standstill. So keep the expectations low. A small tweak here, a little change there. Good enough. No pressure. Just replicate. You may be surprised with the outcome.

4.9.5 Tip #3: Know the literature!

It might seem obvious, but the process of reviewing research papers on a topic of interest is an excellent way to develop ideas for research projects. The starting

point is identifying the topic in a general sense. If one finds gaming of interest, then gaming is the topic. If one finds social networking of interest, then that's the topic. From there the task is to search out and aggressively study and analyze all published research on the topic. If there are too many publications, then narrow the topic. What, in particular, is the interest in gaming or social networking? Continue the search. Use Google Scholar, the ACM Digital Library, or whatever resource is conveniently available. Download all the papers, store them, organize them, study them, make notes, then open a spreadsheet file and start tabulating features from the papers. In the rows, identify the papers. In the columns, tabulate aspects of the interface or interaction technique, conditions tested, results obtained, and so on. Organize the table in whatever manner seems reasonable.

The process is chaotic at first. Where to begin? What are the issues? The task is daunting, at the very least, because of the divergence in reporting methods. But that's the point. The gain is in the process—bringing shape and structure to the chaos. The table will grow as more papers are found and analyzed. There are examples of such tables in published papers, albeit in a condensed summary form. [Figure 4.14](#) shows an example from a research paper on text entry using small keyboards. The table amounts to a mini literature review. Although the table is neat and tidy, don't be fooled. It emerged from a difficult and chaotic process of reviewing a collection of papers and finding common and relevant issues. The collection of notes in the right-hand column is evidence of the difficulty. This column is like a disclaimer, pointing out issues that complicate comparisons of the data in the other columns.

Are there research topics lurking within [Figure 4.14](#)? Probably. But the point is the process, not the product. Building such a table shapes the research area into relevant categories of inquiry. Similar tables are found in other research papers (e.g., Figure 11 and Figure 12 in MacKenzie, 1992; Table 3 and Table 4 in Soukoreff and MacKenzie, 2004). See also student exercise 4-4 at the end of this chapter.

4.9.6 Tip #4: Think inside the box!

The common expression “think outside the box” is a challenge to all. The idea is to dispense with accepted beliefs and assumptions (in the box) and to think in a new way that assumes nothing and challenges everything. However, there is a problem with the challenge. Contemporary, tech-savvy people, clever as they are, often believe they in fact do think outside the box, and that it is everyone else who is confined to life in the box. With this view, the challenge is lost before starting. If there is anything useful in tip #4, it begins with an unsavory precept: You are inside the box! All is not lost, however. Thinking inside the box, then, is thinking about and challenging one's own experiences—the experiences inside the box. The idea is simple. Just get on with your day, but at every juncture, every interaction, think and question. What happened? Why did it happen? Is there an alternative? Play the

Study (1 st author)	Number of Keys ^a	Direct/ Indirect	Scanning	Number of Participants	Speed ^b (wpm)	Notes
Bellman [2]	5	Indirect	No	11	11	4 cursors keys + SELECT key. Error rates not reported. No error correction method.
Dunlop [4]	4	Direct	No	12	8.90	4 letter keys + SPACE key. Error rates reported as "very low."
Dunlop [5]	4	Direct	No	20	12	4 letter keys + 1 key for SPACE/NEXT. Error rates not reported. No error correction method.
Tanaka-Ishii [25]	3	Direct	No	8	12+	4 letters keys + 4 keys for editing, and selecting. 5 hours training. Error rates not reported. Errors corrected using CLEAR key.
Gong [7]	3	Direct	No	32	8.01	3 letter keys + two additional keys. Error rate = 2.1%. Errors corrected using DELETE key.
MacKenzie [16]	3	Indirect	No	10	9.61	2 cursor keys + SELECT key. Error rate = 2.2%. No error correction method.
Baljko [1]	2	Indirect	Yes	12	3.08	1 SELECT key + BACKSPACE key. 43 virtual keys. RC scanning. Same phrase entered 4 times. Error rate = 18.5%. Scanning interval = 750 ms.
Simpson [24]	1	Indirect	Yes	4	4.48	1 SELECT key. 26 virtual keys. RC scanning. Excluded trials with selection errors or missed selections. No error correction. Scanning interval = 525 ms at end of study.
Koester [10]	1	Indirect	Yes	3	7.2	1 SELECT key. 33 virtual keys. RC scanning with word prediction. Dictionary size not given. Virtual BACKSPACE key. 10 blocks of trials. Error rates not reported. Included trials with selection errors or missed selections. Fastest participant: 8.4 wpm.

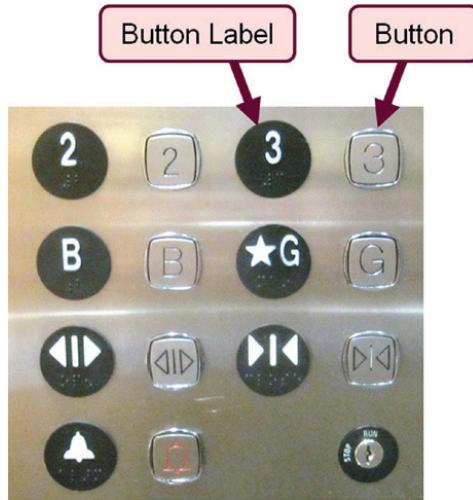
^a For "direct" entry, the value is the number of letter keys. For "indirect" entry, the value is the total number of keys.

^b The entry speed cited is the highest of the values reported in each source, taken from the last block if multiple blocks.

FIGURE 4.14

Table showing papers (rows) and relevant conditions or results (columns) from research papers on text entry using small keyboards.

(From MacKenzie, 2009b, Table 1; consult for full details on studies cited)

**FIGURE 4.15**

Elevator control panel. The button label is more prominent than the button.

role of both a participant (this is unavoidable) and an observer. Observe others, of course, but more importantly observe yourself. You are in the box, but have a look, study, and reconsider.

Here's an example, which on the surface seems trivial (but see tip #1). Recently, while at work at York University, I was walking to my class on Human-Computer Interaction. Being a bit late, I was in a hurry. The class was in a nearby building on the third floor and I was carrying some equipment. I entered the elevator and pushed the button—the wrong button. Apparently, for each floor the control panel has both a button label and a button. (See Figure 4.15.) I pushed the button label instead of the button. A second later I pushed the button, and my journey continued. End of story.

Of course, there is more. Why did I push the wrong button? Yes, I was in a hurry, but that's not the full reason. With a white number on a black background, the floor is identified more prominently by the button label than by the button. And the button label is round, like a button. On the button, the number is recessed in the metal and is barely visible. The error was minor, only a *slip* (right intention, wrong action; see Norman, 1988, ch. 5). Is there a research topic in this? Perhaps. Perhaps not. But experiencing, observing, and thinking about one's interactions with technology can generate ideas and promote a humbling yet questioning frame of thinking—thinking that moves forward into research topics. The truth is, I have numerous moments like this every day (and so do you!). Most amount to nothing, but the small foibles in interacting with technology are intriguing and worth thinking about.

In this chapter, we have examined the scientific foundations for research in human-computer interaction. With this, the next challenge is in designing

and conducting experiments using human participants (users) to evaluate new ideas for user interfaces and interaction techniques. We explore these topics in Chapter 5.

STUDENT EXERCISES

- 4-1.** Examine some published papers in HCI and find examples where results were reported as a raw count (e.g., number of errors) rather than as a count *per something* (e.g., percent errors). Find three examples and write a brief report (or prepare a brief presentation) detailing how the results were reported and the weakness or limitation in the method. Propose a better way to report the same results. Use charts or graphs where appropriate.
- 4-2.** What, in Vannevar Bush’s “human experience,” formed the inspiration for memex? (If needed, review Bush’s essay “As We May Think,” or see the discussion in Chapter 1.) What are the similarities between his inspiration and memex?
- 4-3.** A *fisheye lens* or *fisheye view* is a tool or concept in HCI whereby high-value information is presented in greater detail than low-value information. Furnas first introduced the idea in 1986 (Furnas, 1986). Although the motivation was to improve the visualization of large data sets, such as programs or databases, Furnas’ idea came from something altogether different. What was Furnas’ inspiration for fisheye views? Write a brief report describing the analogy offered by Furnas. Include in your report three examples of fisheye lenses, as described and implemented in subsequent research, noting in particular the background motivation.
- 4-4.** Here are some research themes: 3D gaming, mobile phone use while driving, privacy in social networking, location-aware user interfaces, tactile feedback in pointing and selecting, multi-touch tabletop interaction. Choose one of these topics (or another) and build a table similar to that in [Figure 4.14](#). Narrow the topic, if necessary (e.g., mobile phone *texting* while driving), and find at least five relevant research papers to include in the table. Organize the table identifying the papers in the rows and methods, relevant themes, and findings in the columns. Write a brief report about the table. Include citations and references to the selected papers.
- 4-5.** In Chapter 3, we used a 2D plot to illustrate the trade-off between the frequency of errors (x-axis) and the cost of errors (y-axis) (see [Figure 3.46](#)). The plot was just a sketch, since the analysis was informal. In this chapter, we discussed another trade-off, that between form and function. The

Survey Research in HCI

Hendrik Müller, Aaron Sedley, and Elizabeth Ferrall-Nunge

Short Description of the Method

A survey is a method of gathering information by asking questions to a subset of people, the results of which can be generalized to the wider target population. There are many different types of surveys, many ways to sample a population, and many ways to collect data from that population. Traditionally, surveys have been administered via mail, telephone, or in person. The Internet has become a popular mode for surveys due to the low cost of gathering data, ease and speed of survey administration, and its broadening reach across a variety of populations worldwide. Surveys in human-computer interaction (HCI) research can be useful to:

- Gather information about people's habits, interaction with technology, or behavior
- Get demographic or psychographic information to characterize a population
- Get feedback on people's experiences with a product, service, or application
- Collect people's attitudes and perceptions toward an application in the context of usage
- Understand people's intents and motivations for using an application
- Quantitatively measure task success with specific parts of an application
- Capture people's awareness of certain systems, services, theories, or features
- Compare people's attitudes, experiences, etc. over time and across dimensions

H. Müller (✉)

Google Australia Pty Ltd., Level 5, 48 Pirrama Road, Pyrmont, NSW 2009, Australia
e-mail: hendrik82@gmail.com

A. Sedley

Google, Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
e-mail: asedley@gmail.com

E. Ferrall-Nunge

Twitter, Inc., 1355 Market Street, Suite 900, San Francisco, CA 94103, USA
e-mail: enunge@gmail.com

While powerful for specific needs, surveys do not allow for observation of the respondents' context or follow-up questions. When conducting research into precise behaviors, underlying motivations, and the usability of systems, then other research methods may be more appropriate or needed as a complement.

This chapter reviews the history of surveys and appropriate uses of surveys and focuses on the best practices in survey design and execution.

History, Intellectual Tradition, Evolution

Since ancient times, societies have measured their populations via censuses for food planning, land distribution, taxation, and military conscription. Beginning in the nineteenth century, political polling was introduced in the USA to project election results and to measure citizens' sentiment on a range of public policy issues. At the emergence of contemporary psychology, Francis Galton pioneered the use of questionnaires to investigate the nature vs. nurture debate and differences between humans, the latter of which evolved into the field of psychometrics (Clauser, 2007). More recently, surveys have been used in HCI research to help answer a variety of questions related to people's attitudes, behaviors, and experiences with technology.

Though nineteenth-century political polls amplified public interest in surveys, it was not until the twentieth century that meaningful progress was made on survey-sampling methods and data representativeness. Following two incorrect predictions of the US presidential victors by major polls (Literary Digest for Landon in 1936 and Gallup for Dewey in 1948), sampling methods were assailed for misrepresenting the US electorate. Scrutiny of these polling failures; persuasive academic work by statisticians such as Kjaer, Bowley, and Neyman; and extensive experimentation by the US Census Bureau led to the acceptance of random sampling as the gold standard for surveys (Converse, 1987).

Roughly in parallel, social psychologists aimed to minimize questionnaire biases and optimize data collection. For example, in the 1920s and 1930s, Louis Thurstone and Rensis Likert demonstrated reliable methods for measuring attitudes (Edwards & Kenney, 1946); Likert's scaling approach is still widely used by survey practitioners. Stanley Payne's, 1951 classic "The Art of Asking Questions" was an early study of question wording. Subsequent academics scrutinized every aspect of survey design. Tourangeau (1984) articulated the four cognitive steps to survey responses, noting that people have to comprehend what is asked, retrieve the appropriate information, judge that information according to the question, and map the judgement onto the provided responses. Krosnick & Fabrigar (1997) studied many components of questionnaire design, such as scale length, text labels, and "no opinion" responses. Groves (1989) identified four types of survey-related error: coverage, sampling, measurement, and non-response. As online surveys grew in popularity, Couper (2008) and others studied bias from the visual design of Internet questionnaires.

The use of surveys for HCI research certainly predates the Internet, with efforts to understand users' experiences with computer hardware and software. In 1983, researchers at Carnegie Mellon University conducted an experiment comparing

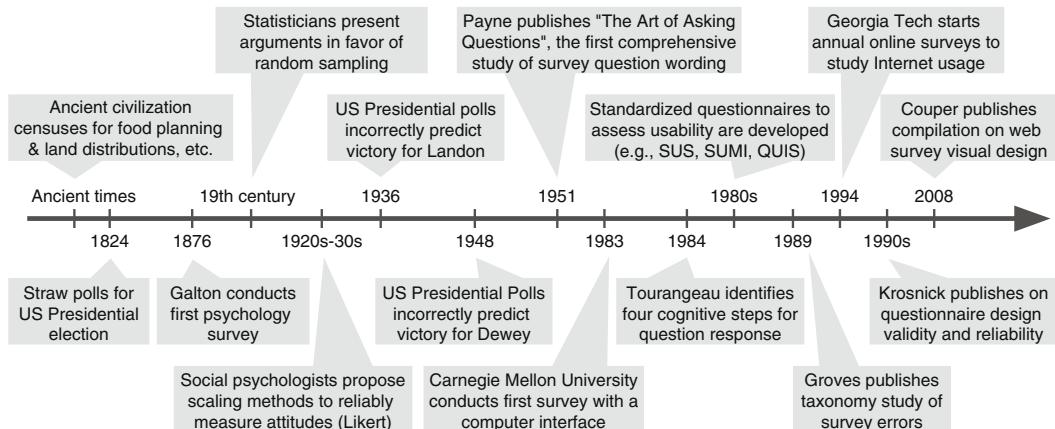


Fig. 1 Summary of the key stages in survey history

computer-collected survey responses with those from a printed questionnaire, finding less socially desirable responses in the digital survey and longer open-ended responses than in the printed questionnaire (Kiesler & Sproull, 1986). With the popularization of graphical user interfaces in the 1980s, surveys joined other methods for usability research. Several standardized questionnaires were developed to assess usability (e.g., SUS, QUIS, SUMI, summarized later in this chapter). Surveys are a direct means of measuring satisfaction; along with efficiency and effectiveness, satisfaction is a pillar of the ISO 9241, part 11, definition of usability (Abran et al., 2003). **User happiness** is fundamental to Google's HEART framework for user-centric measurement of Web applications (Rodden, Hutchinson, & Fu, 2010). In 1994, the Georgia Institute of Technology started annual online surveys to understand Internet usage and users and to explore Web-based survey research (Pitkow & Recker, 1994). As the Internet era progressed, online applications widely adopted surveys to measure users' satisfaction, unaddressed needs, and problems experienced, in addition to user profiling. See a summary of key stages in survey history in Fig. 1.

What Questions the Method Can Answer

When used appropriately, surveys can help inform application and user research strategies and provide insights into users' attitudes, experiences, intents, demographics, and psychographic characteristics. However, surveys are not the most appropriate method for many other HCI research goals. Ethnographic interviews, log data analysis, card sorts, usability studies, and other methods may be more appropriate. In some cases, surveys can be used with other research methods to holistically inform HCI development. This section explains survey appropriateness, when to avoid using surveys, as well as how survey research can complement other research methods.

When Surveys Are Appropriate

Overall, surveys are appropriate when needing to represent an entire population, to measure differences between groups of people, and to identify changes over time in people's attitudes and experiences. Below are examples of how survey data can be used in HCI research.

Attitudes. Surveys can accurately measure and reliably represent attitudes and perceptions of a population. While qualitative studies are able to gather attitudinal data, surveys provide statistically reliable metrics, allowing researchers to benchmark attitudes toward an application or an experience, to track changes in attitudes over time, and to tie self-reported attitudes to actual behavior (e.g., via log data). For example, surveys can be used to measure customer satisfaction with online banking immediately following their experiences.

Intent. Surveys can collect peoples' reasons for using an application at a specific time, allowing researchers to gauge the frequency across different objectives. Unlike other methods, surveys can be deployed while a person is actually using an application (i.e., an online intercept survey), minimizing the risk of imperfect recall on the respondent's part. Note that specific details and the context of one's intent may not be fully captured in a survey alone. For example, "Why did you visit this website?" could be answered in a survey, but qualitative research may be more appropriate in determining how well one understood specific application elements and what users' underlying motivations are in the context of their daily lives.

Task success. Similar to measuring intent, while HCI researchers can qualitatively observe task success through a lab or a field study, a survey can be used to reliably quantify levels of success. For example, respondents can be instructed to perform a certain task, enter results of the task, and report on their experiences while performing the task.

User experience feedback. Collecting open-ended feedback about a user's experience can be used to understand the user's interaction with technology or to inform system requirements and improvements. For example, by understanding the relative frequency of key product frustrations and benefits, project stakeholders can make informed decisions and trade-offs when allocating resources.

User characteristics. Surveys can be used to understand a system's users and to better serve their needs. Researchers can collect users' demographic information, technographic details such as system savviness or overall tech savviness, and psychographic variables such as openness to change and privacy orientation. Such data enables researchers to discover natural segments of users who may have different needs, motivations, attitudes, perceptions, and overall user experiences.

Interactions with technology. Surveys can be used to understand more broadly how people interact with technology and how technology influences social interactions with others by asking people to self-report on social, psychological, and demographic

variables while capturing their behaviors. Through the use of surveys, HCI researchers can glean insights into the effects technology has on the general population.

Awareness. Surveys can also help in understanding people's awareness of existing technologies or specific application features. Such data can, for example, help researchers determine whether low usage with an application is a result of poor awareness or other factors, such as usability issues. By quantifying how aware or unaware people are, researchers can decide whether efforts (e.g., marketing campaigns) are needed to increase overall awareness and thus use.

Comparisons. Surveys can be used to compare users' attitudes, perceptions, and experiences across user segments, time, geographies, and competing applications and between experimental and control versions. Such data enable researchers to explore whether user needs and experiences vary across geographies, assess an application's strengths and weaknesses among competing technologies and how each compares with their competitors' applications, and evaluate potential application improvements while aiding decision making between a variety of proposed designs.

When to Avoid Using a Survey

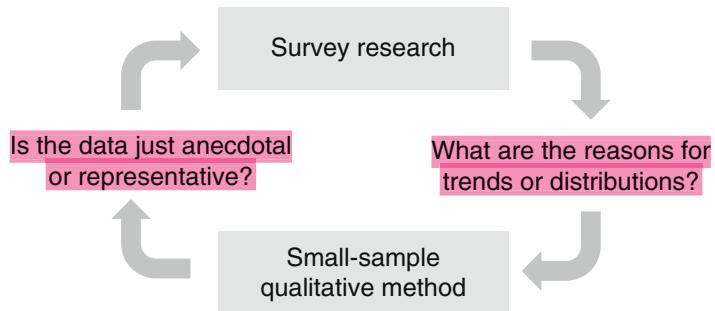
Because surveys are inexpensive and easy to deploy compared to other methods, many people choose survey research even when it is inappropriate for their needs. Such surveys can produce invalid or unreliable data, leading to an inaccurate understanding of a population and poor user experiences. Below are some HCI research needs that are better addressed with other methods.

Precise behaviors. While respondents can be asked to self-report their behaviors, gathering this information from log data, if available, will always be more accurate. This is particularly true when trying to understand precise user behaviors and flows, as users will struggle to recall their exact sequence of clicks or specific pages visited. For behaviors not captured in log data, a diary study, observational study, or experience sampling may gather more accurate results than a survey.

Underlying motivations. People often do not understand or are unable to explain why they take certain actions or prefer one thing over another. Someone may be able to report their intent in a survey but may not be aware of their subconscious motivations for specific actions. Exploratory research methods such as ethnography or contextual inquiry may be more appropriate than directly asking about underlying motivations in a survey.

Usability evaluations. Surveys are inappropriate for testing specific usability tasks and understanding of tools and application elements. As mentioned above, surveys can measure task success but may not explain why people cannot use a particular application, why they do not understand some aspect of a product, or why they do not identify missteps that caused the task failure. Furthermore, a user may still be able to complete a given task even though he or she encountered several confusions, which could not be uncovered through a survey. Task-based observational research and interview methods, such as usability studies, are better suited for such research goals.

Fig. 2 Employing survey research either before or after research using other methods



Using Surveys with Other Methods

Survey research may be especially beneficial when used in conjunction with other research methods (see Fig. 2). Surveys can follow previous qualitative studies to help quantify specific observations. For many surveys, up-front qualitative research may even be required to inform its content if no previous research exists. On the other hand, surveys can also be used to initially identify high-level insights that can be followed by in-depth research through more qualitative (meaning smaller sample) methods.

For example, if a usability study uncovers a specific problem, a survey can quantify the frequency of that problem across the population. Or a survey can be used first to identify the range of frustrations or goals, followed by qualitative interviews and observational research to gain deeper insights into self-reported behaviors and sources of frustration. Researchers may interview survey respondents to clarify responses (e.g., Yew, Shamma, & Churchill, 2011), interview another pool of participants in the same population for comparison (e.g., Froelich et al., 2012), or interview both survey respondents and new participants (e.g., Archambault & Grudin, 2012).

Surveys can also be used in conjunction with A/B experiments to aid comparative evaluations. For example, when researching two different versions of an application, the same survey can be used to assess both. By doing this, differences in variables such as satisfaction and self-reported task success can be measured and analyzed in parallel with behavioral differences observed in log data. Log data may show that one experimental version drives more traffic or engagement, but the survey may show that users were less satisfied or unable to complete a task. Moreover, log data can further validate insights from a previously conducted survey. For example, a social recommendation study by Chen, Geyer, Dugan, Muller, and Guy (2009) tested the quality of recommendations first in a survey and then through logging in a large field deployment. Psychophysiological data may be another objective accompaniment to survey data. For example, game researchers have combined surveys with data such as facial muscle and electrodermal activity (Nacke, Grimshaw, & Lindley, 2010) or attention and meditation as measured with EEG sensors (Schild, LaViola, & Masuch, 2012).

How to Do It: What Constitutes Good Work

This section breaks down survey research into the following six stages:

1. Research goals and constructs
2. Population and sampling
3. Questionnaire design and biases
4. Review and survey pretesting
5. Implementation and launch
6. Data analysis and reporting

Research Goals and Constructs

Before writing survey questions, researchers should first think about what they intend to measure, what kind of data needs to be collected, and how the data will be used to meet the research goals. When the survey-appropriate *research goals* have been identified, they should be matched to *constructs*, i.e., unidimensional attributes that cannot be directly observed. The identified constructs should then be converted into one or multiple survey questions. Constructs can be identified from prior primary research or literature reviews. Asking multiple questions about the same construct and analyzing the responses, e.g., through factor analysis, may help the researcher ensure the construct's validity.

An example will illustrate the process of converting constructs into questions. An overarching research goal may be to understand users' happiness with an online application, such as Google Search, a widely used Web search engine. Since happiness with an application is often multidimensional, it is important to separate it into measurable pieces—its constructs. Prior research might indicate that constructs such as “overall satisfaction,” “perceived speed,” and “perceived utility” contribute to users’ happiness with that application. When all the constructs have been identified, survey questions can be designed to measure each. To validate each construct, it is important to evaluate its unique relationship with the higher level goal, using correlation, regression, factor analysis, or other methods. Furthermore, a technique called cognitive pretesting can be used to determine whether respondents are interpreting the constructs as intended by the researcher (see more details in the pretesting section).

Once research goals and constructs are defined, there are several other considerations to help determine whether a survey is the most appropriate method and how to proceed:

- Do the survey constructs focus on results which will directly address research goals and inform stakeholders' decision making rather than providing merely informative data? An excess of “nice-to-know” questions increases survey length and the likelihood that respondents will not complete the questionnaire, diminishing the effectiveness of the survey results.

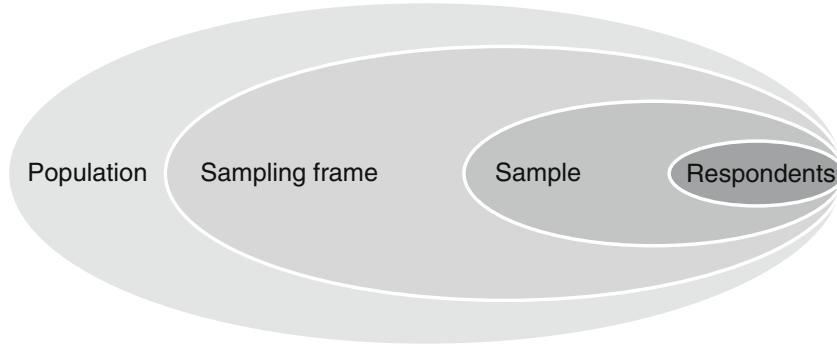


Fig. 3 The relationship between population, sampling frame, sample, and respondents

- Will the results be used for longitudinal comparisons or for one-time decisions? For longitudinal comparisons, researchers must plan on multiple survey deployments without exhausting available respondents.
- What is the number of responses needed to provide the appropriate level of precision for the insights needed? By calculating the number of responses needed (as described in detail in the following section), the researcher will ensure that key metrics and comparisons are statistically reliable. Once the target number is determined, researchers can then determine how many people to invite.

Population and Sampling

Key to effective survey research is determining who and how many people to survey. In order to do this, the survey's *population*, or set of individuals that meet certain criteria, and to whom researchers wish to generalize their results must first be defined. Reaching everyone in the population (i.e., a census) is typically impossible and unnecessary. Instead, researchers approximate the true population by creating a *sampling frame*, i.e., the set of people who the researcher is able to contact for the survey. The perfect sampling frame is identical to the population, but often a survey's sampling frame is only a portion of the population. The people from the sampling frame who are invited to take the survey are the *sample*, but only those who answer are *respondents*. See Fig. 3 illustrating these different groups.

For example, a survey can be deployed to understand the satisfaction of a product's or an application's users. In this case, the population includes everyone that uses the application, and the sampling frame consists of users that are actually reachable. The sampling frame may exclude those who have abandoned the application, anonymous users, and users who have not opted in to being contacted for research. Though the sampling frame may exclude many users, it could still include far more people than are needed to collect a statistically valid number of responses. However, if the sampling frame systematically excludes certain types of people (e.g., very dissatisfied or disengaged users), the survey will suffer from *coverage error* and its responses will misrepresent the population.

Probability Versus Non-probability Sampling

Sampling a population can be accomplished through probability- and non-probability-based methods. *Probability or random sampling* is considered the gold standard because every person in the sampling frame has an equal, nonzero chance of being chosen for the sample; essentially, the sample is selected completely randomly. This minimizes *sampling bias*, also known as *selection bias*, by randomly drawing the sample from individuals in the sampling frame and by inviting everyone in the sample in the same way. Examples of probability sampling methods include random digit telephone dialing, address-based mail surveys utilizing the US Postal Service Delivery Sequence File (DSF), and the use of a panel recruited through random sampling, those who have agreed in advance to receive surveys. For Internet surveys in particular, methods allowing for random sampling include intercept surveys for those who use a particular product (e.g., pop-up surveys or in-product links), list-based samples (e.g., for e-mail invitations), and pre-recruited probability-based panels (see Couper, 2000, for a thorough review). Another way to ensure probability sampling is to use a preexisting sampling frame, i.e., a list of candidates previously assembled using probability sampling methods. For example, Shklovski, Kraut, and Cummings' (2008) study of the effect of residential moves on communication with friends was drawn from a publicly available, highly relevant sampling frame, the National Change of Address (NCOA) database. Another approach is to analyze selected subsets of data from an existing representative survey like the General Social Survey (e.g., Wright & Randall, 2012).

While probability sampling is ideal, it is often impossible to reach and randomly select from the entire target population, especially when targeting small populations (e.g., users of a specialized enterprise product or experts in a particular field) or investigating sensitive or rare behavior. In these situations, researchers may use *non-probability sampling* methods such as volunteer opt-in panels, unrestricted self-selected surveys (e.g., links on blogs and social networks), *snowball recruiting* (i.e., asking for friends of friends), and *convenience samples* (i.e., targeting people readily available, such as mall shoppers) (Couper, 2000). However, non-probability methods are prone to high sampling bias and hence reduce representativeness compared to random sampling. One way representativeness can be assessed is by comparing key characteristics of the target population with those from the actual sample (for more details, refer to the analysis section).

Many academic surveys use convenience samples from an existing pool of the *university's psychology students*. Although not representative of most Americans, this type of sample is appropriate for investigating technology behavior among young people such as *sexting* (Drouin & Landgraff, 2012; Weisskirch & Delevi, 2011), instant messaging (Anandarajan, Zaman, Dai, & Arinze, 2010; Junco & Cotten, 2011; Zaman et al., 2010), and mobile phone use (Auter, 2007; Harrison, 2011; Turner, Love, & Howell, 2008). Convenience samples have also been used to identify special populations. For example, because identifying HIV and tuberculosis patients through official lists of names is difficult because of patient confidentiality, one study about the viability of using cell phones and text messages in HIV and tuberculosis education handed out surveys to potential respondents in health clinic

waiting rooms (Person, Blain, Jiang, Rasmussen, & Stout, 2011). Similarly, a study of Down's syndrome patients' use of computers invited participation through special interest listservs (Feng, Lazar, Kumin, & Ozok, 2010).

Determining the Appropriate Sample Size

No matter which sampling method is used, it is important to carefully determine the target sample size for the survey, i.e., the number of survey responses needed. If the sample size is too small, findings from the survey cannot be accurately generalized to the population and may fail to detect generalizable differences between groups. If the sample is larger than necessary, too many individuals are burdened with taking the survey, analysis time for the researcher may increase, or the sampling frame is used up too quickly. Hence, calculating the optimal sample size becomes crucial for every survey.

First, the researcher needs to determine approximately how many people make up the population being studied. Second, as the survey does not measure the entire population, the required level of precision must be chosen, which consists of the margin of error and the confidence level. The *margin of error* expresses the amount of sampling error in the survey, i.e., the range of uncertainty around an estimate of a population measure, assuming normally distributed data. For example, if 60 % of the sample claims to use a tablet computer, a 5 % margin of error would mean that actually 55–65 % of the population use tablet computers. Commonly used margin of errors are 5 and 3 %, but depending on the goals of the survey anywhere between 1 and 10 % may be appropriate. Using a margin of error higher than 10 % is not recommended, unless a low level of precision can meet the survey's goals. The *confidence level* indicates how likely the reported metric falls within the margin of error if the study were repeated. A 95 % confidence level, for example, would mean that 95 % of the time, observations from repeated sampling will fall within the interval defined by the margin of error. Commonly used confidence levels are 99, 95, and 90 %; using less than 90 % is not recommended.

There are various formulas for calculating the target sample size. Figure 4, based on Krejcie and Morgan's formula (1970), shows the appropriate sample size, given the population size, as well as the chosen margin of error and confidence level for your survey. Note that the table is based on a population proportion of 50 % for the response of interest, the most cautious estimation (i.e., when higher or lower than 50 %, the required sample size declines to achieve the same margin of error). For example, for a population larger than 100,000, a sample size of 384 is required to achieve a confidence level of 95 % and a margin of error of 5 %. Note that for population sizes over about 20,000, the required sample size does not significantly increase. Researchers may set the sample size to 500 to estimate a single population parameter, which yields a margin of error of about $\pm 4.4\%$ at a 95 % confidence level for large populations.

After having determined the target sample size for the survey, the researcher now needs to work backwards to estimate the number of people to actually invite to the

Confidence level Margin of error Size of population	90%				95%				99%			
	10%	5%	3%	1%	10%	5%	3%	1%	10%	5%	3%	1%
10	9	10	10	10	9	10	10	10	9	10	10	10
100	41	73	88	99	49	80	92	99	63	87	95	99
1000	63	213	429	871	88	278	516	906	142	399	648	943
10,000	67	263	699	4035	95	370	964	4899	163	622	1556	6239
100,000	68	270	746	6335	96	383	1056	8762	166	659	1810	14227
1,000,000	68	270	751	6718	96	384	1066	9512	166	663	1840	16317
100,000,000	68	271	752	6763	96	384	1067	9594	166	663	1843	16560

Fig. 4 Sample size as a function of population size and accuracy (confidence level and margin of error)

survey, taking into account the estimated size for each subgroup and the expected response rate. If a subgroup's incidence is very small, the total number of invitations must be increased to ensure the desired sample size for this subgroup. The **response rate** of a survey describes the percentage of those who completed the survey out of all those that were invited (for more details, see the later sections on monitoring survey paradata and maximizing response rates). If a similar survey has been conducted before, then its response rate is a good reference point for calculating the required sample size. If there is no prior response rate information, the survey can be sent out to a small number of people first to measure the response rate, which is then used to determine the total number of required invitations.

For example, assuming a 30 % response rate, a 50 % incidence rate for the group of interest, and the need for 384 complete responses from that group, 2,560 people should be invited to the survey. At this point, the calculation may determine that the researcher may require a sample that is actually larger than the sampling frame; hence, the researcher may need to consider more qualitative methods as an alternative.

Mode and Methods of Survey Invitation

To reach respondents, there are four basic survey modes: mail or written surveys, phone surveys, face-to-face or in-person surveys, and Internet surveys. Survey modes may also be used in combination. The survey mode needs to be chosen carefully as each mode has its own advantages and disadvantages, such as differences in typical response rates, introduced biases (Groves, 1989), required resources and costs, audience that can be reached, and respondents' level of anonymity.

Today, many HCI-related surveys are Internet based, as benefits often outweigh their disadvantages. Internet surveys have the following major advantages:

- Easy access to large geographic regions (including international reach)
- Simplicity of creating a survey by leveraging easily accessible commercial tools

- Cost savings during survey invitation (e.g., no paper and postage, simple implementation, insignificant cost increase for large sample sizes) and analysis (e.g., returned data is already in electronic format)
- Short fielding periods, as the data is collected immediately
- Lower bias due to respondent anonymity, as surveys are self-administered with no interviewer present
- Ability to customize the questionnaire to specific respondent groups using skip logic (i.e., asking respondents a different set of questions based on the answer to a previous question)

Internet surveys also have several disadvantages. The most discussed downside is the introduction of *coverage error*, i.e., a potential mismatch between the target population and the sampling frame (Couper, 2000; Groves, 1989). For example, online surveys fail to reach people without Internet or e-mail access. Furthermore, those invited to Internet surveys may be less motivated to respond or to provide accurate data because such surveys are less personal and can be ignored more easily. This survey mode also relies on the respondents' ability to use a computer and may only provide the researcher with minimal information about the survey respondents. (See chapter on “Crowdsourcing in HCI Research.”)

Questionnaire Design and Biases

Upon establishing the constructs to be measured and the appropriate sampling method, the first iteration of the survey questionnaire can be designed. It is important to carefully think through the design of each survey question (first acknowledged by Payne, 1951), as it is fairly easy to introduce biases that can have a substantial impact on the reliability and validity of the data collected. Poor questionnaire design may introduce *measurement error*, defined as the deviation of the respondents' answers from their true values on the measure. According to Couper (2000), measurement error in self-administered surveys can arise from the respondent (e.g., lack of motivation, comprehension problems, deliberate distortion) or from the instrument (e.g., poor wording or design, technical flaws). In most surveys, there is only one opportunity to deploy, and unlike qualitative research, no clarification or probing is possible. For these reasons, it is crucial that the questions accurately measure the constructs of interest.

Going forward, this section covers different types of survey questions, common questionnaire biases, questions to avoid, visual design considerations, reuse of established questionnaires, as well as visual survey design considerations.

Types of Survey Questions

There are two categories of survey questions—open- and closed-ended questions. Open-ended questions (Fig. 5) ask survey respondents to write in their own answers, whereas closed-ended questions (Fig. 6) provide a set of predefined answers to choose from.

What, if anything, do you find frustrating about your smartphone?

Fig. 5 Example of a typical open-ended question



Fig. 6 Example of a typical closed-ended question, a bipolar rating question in particular

Open-ended questions are appropriate when:

- The **universe** of possible answers is **unknown**, e.g., “What is your favorite smartphone application?”. However, once the universe of possible answers is identified, it may be appropriate to create a closed-ended version of the same question.
- There are so many **options** in the full list of possible answers that they **cannot** be easily **displayed**, e.g., “Which applications have you used on your smartphone in the last week?”.
- Measuring quantities with natural metrics (i.e., a construct with an inherent unit of measurement, such as age, length, or frequency), when being unable to access information from log data, such as time, frequency, and length, e.g., “How many times do you use your tablet in a typical week?” (using a text field that is restricted to numeric input, the answers to which can later be **bucketed flexibly**).
- **Measuring qualitative aspects of a user’s experience**, e.g., “What do you find most frustrating about using your smartphone?”.

Closed-ended questions are appropriate when:

- The universe of possible answers is **known** and **small** enough to be easily provided, e.g., “Which operating system do you use on your smartphone?” (with answer options including “Android” and “iOS”).
- Rating a single object on a **dimension**, e.g., “Overall, how satisfied or dissatisfied are you with your smartphone?” (on a **7-point scale** from “Extremely dissatisfied” to “Extremely satisfied”).
- Measuring quantities without natural metrics, such as importance, certainty, or degree, e.g., “How important is it to have your smartphone within reach 24 h a day?” (on a **5-point scale** from “Not at all important” to “Extremely important”).

What is the highest level of education you have completed?

- Less than High School
- High School
- Some College
- 2-year College Degree (Associates)
- 4-year College Degree (BA, BS)
- Master's Degree
- Doctoral Degree
- Professional Degree (MD, JD)

Fig. 7 Example of a single-choice question

Which of the following apps do you use daily on your smartphone?

Select all that apply.

- Gmail
- Maps
- Calendar
- Facebook
- Hangouts
- Drive

Fig. 8 Example of a multiple-choice question

Types of Closed-Ended Survey Questions

There are four basic types of closed-ended questions: single-choice, multiple-choice, rating, and ranking questions.

1. *Single-choice questions* work best when only one answer is possible for each respondent in the real world (Fig. 7).
2. *Multiple-choice questions* are appropriate when more than one answer may apply to the respondent. Frequently, multiple-choice questions are accompanied by “select all that apply” help text. The maximum number of selections may also be specified to force users to prioritize or express preferences among the answer options (Fig. 8).
3. *Ranking questions* are best when respondents must prioritize their choices given a real-world situation (Fig. 9).
4. *Rating questions* are appropriate when the respondent must judge an object on a continuum. To optimize reliability and minimize bias, scale points need to be

Rank the following smartphone manufacturers in order of your preference:
 Add a number to each row, 1 being the least preferred, 5 being the most preferred.

<input type="text"/>	Apple
<input type="text"/>	HTC
<input type="text"/>	Samsung
<input type="text"/>	Motorola
<input type="text"/>	Nokia

Fig. 9 Example of a ranking question

How important is it to you to make phone calls from your smartphone?

Not at all important <input type="radio"/>	Slightly important <input type="radio"/>	Moderately important <input type="radio"/>	Very important <input type="radio"/>	Extremely important <input type="radio"/>
---	---	---	---	--

Fig. 10 Example of a rating question, for a unipolar construct in particular

fully labeled instead of using numbers (Groves et al., 2004), and each scale point should be of equal width to avoid bias toward visually bigger response options (Tourangeau, Couper, & Conrad, 2004). Rating questions should use either a unipolar or a bipolar scale, depending on the construct being measured (Krosnick & Fabrigar, 1997; Schaeffer & Presser, 2003).

Unipolar constructs range from zero to an extreme amount and do not have a natural midpoint. They are best measured with a **5-point rating scale** (Krosnick & Fabrigar, 1997), which optimizes reliability while minimizing respondent burden, and with the following scale labels, which have been shown to be semantically equidistant from each other (Rohrmann, 2003): “**Not at all ...**,” “**Slightly ...**,” “**Moderately ...**,” “**Very ...**,” and “**Extremely ...**” Such constructs include importance (see Fig. 10), interest, usefulness, and relative frequency. **Bipolar constructs** range from an extreme negative to an extreme positive with a natural midpoint. Unlike unipolar constructs, they are best measured **with a 7-point rating scale to maximize reliability and data differentiation** (Krosnick & Fabrigar, 1997). Bipolar constructs may use the following scale labels: “**Extremely ...**,” “**Moderately ...**,” “**Slightly ...**,” “**Neither ... nor ...**,” “**Slightly ...**,” “**Moderately ...**,” and “**Extremely ...**” Such constructs include satisfaction (see Fig. 6, from dissatisfied to satisfied), perceived speed (from slow to fast), ease of use (from difficult to easy), and visual appeal (from unappealing to appealing).

When using a rating scale, the inclusion of a midpoint should be considered. While some may argue that including a midpoint provides an easy target for respondents who shortcut answering questions, others argue that the exclusion of a

midpoint forces people who truly are in the middle to choose an option that does not reflect their actual opinion. O’Muircheartaigh, Krosnick, and Helic (2001) found that having a midpoint on a rating scale increases reliability, has no effect on validity, and does not result in lower data quality. Additionally, people who look for shortcuts (“shortcutters”) are not more likely to select the midpoint when present. Omitting the midpoint, on the other hand, increases the amount of random measurement error, resulting in those who actually feel neutral to end up making a random choice on either side of the scale. These findings suggest that a midpoint should be included when using a rating scale.

Questionnaire Biases

After writing the first survey draft, it is crucial to check the phrasing of each question for potential biases that may bias the responses. The following section covers five common questionnaire biases: satisficing, acquiescence bias, social desirability, response order bias, and question order bias.

Satisficing

Satisficing occurs when respondents use a suboptimal amount of cognitive effort to answer questions. Instead, satisficers will typically pick what they consider to be the first acceptable response alternative (Krosnick, 1991; Simon, 1956). Satisficers compromise one or more of the following four cognitive steps for survey response as identified by Tourangeau (1984):

1. *Comprehension* of the question, instructions, and answer options
2. *Retrieval* of specific memories to aid with answering the question
3. *Judgement* of the retrieved information and its applicability to the question
4. *Mapping* of judgement onto the answer options

Satisficers shortcut this process by exerting less cognitive effort or by skipping one or more steps entirely; satisficers use less effort to understand the question, to thoroughly search their memories, to carefully integrate all retrieved information, or to accurately pick the proper response choice (i.e., they pick the next best choice).

Satisficing can take weak and strong forms (Krosnick, 1999). Weak satisficers make an attempt to answer correctly yet are less than thorough, while strong satisficers may not at all search their memory for relevant information and simply select answers at random in order to complete the survey quickly. In other words, weak satisficers carelessly process all four cognitive steps, while strong satisficers typically skip the retrieval and judgement steps.

Respondents are more likely to satisfice when (Krosnick, 1991):

- Cognitive ability to answer is low.
- Motivation to answer is low.
- Question difficulty is high at one of the four stages, resulting in cognitive exertion.

To minimize satisficing, the following may be considered:

- **Complex questions** that require an inordinate amount of cognitive exertion should be avoided.
- Answer options such as “no opinion,” “don’t know,” “not applicable,” or “unsure” should be avoided, since respondents with actual opinions will be tempted and select this option (Krosnick, 2002; Schaeffer & Presser, 2003). Instead, respondents should first be asked whether they have thought about the proposed question or issue enough to have an opinion; those that haven’t should be screened out.
- **Using the same rating scale in a series of back-to-back questions should be avoided.** Potential satisfiers may pick the same scale point for all answer options. This is known as straight-lining or item non-differentiation (Herzog & Bachman, 1981; Krosnick & Alwin, 1987, 1988).
- **Long questionnaires should be avoided,** since respondents will be less likely to optimally answer questions when they become increasingly **fatigued** and unmotivated (Cannell & Kahn, 1968; Herzog & Bachman, 1981).
- Respondent motivation can be increased by explaining the importance of the survey topic and that their responses are critical to the researcher (Krosnick, 1991).
- Respondents may be asked to justify their answer to the question that may exhibit satisficing.
- **Trap questions** (e.g., “Enter the number 5 in the following text box:”) can identify satisficers and fraudulent survey respondents.

Acquiescence Bias

When presented with agree/disagree, yes/no, or true/false statements, some respondents are more likely to concur with the statement independent of its substance. This tendency is known as acquiescence bias (Smith, 1967).

Respondents are more likely to acquiesce when:

- Cognitive ability is low (Krosnick, Narayan, & Smith, 1996) or motivation is low.
- Question difficulty is high (Stone, Gage, & Leavitt, 1957).
- **Personality** tendencies skew toward agreeableness (Costa & McCrae, 1988; Goldberg, 1990; Saris, Revilla, Krosnick, & Shaeffer, 2010).
- Social conventions suggest that a “yes” response is most polite (Saris et al., 2010).
- The respondent satisfies and only thinks of reasons why the statement is true, rather than expending cognitive effort to consider reasons for disagreement (Krosnick, 1991).
- Respondents with lower self-perceived status assume that the survey administrator agrees with the posed statement, resulting in deferential agreement bias (Saris et al., 2010).

To minimize acquiescence bias, the following may be considered:

- Avoid questions with agree/disagree, yes/no, true/false, or similar answer options (Krosnick & Presser, 2010).
- Where possible, ask construct-specific questions (i.e., questions that ask about the underlying construct in a neutral, non-leading way) instead of agreement statements (Saris et al., 2010).
- Use reverse-keyed constructs; i.e., the same construct is asked positively and negatively in the same survey. The raw scores of both responses are then combined to correct for acquiescence bias.

Social Desirability

Social desirability occurs when respondents answer questions in a manner they feel will be positively perceived by others (Goffman, 1959; Schlenker & Weigold, 1989). Favorable actions may be overreported, and unfavorable actions or views may be underreported. Topics that are especially prone to social desirability bias include voting behavior, religious beliefs, sexual activity, patriotism, bigotry, intellectual capabilities, illegal acts, acts of violence, and charitable acts.

Respondents are inclined to provide socially desirable answers when:

- Their behavior or views go against the social norm (Holbrook & Krosnick, 2010).
- Asked to provide information on sensitive topics, making the respondent feel uncomfortable or embarrassed about expressing their actual views (Holbrook & Krosnick, 2010).
- They perceive a threat of disclosure or consequences to answering truthfully (Tourangeau, Rips, & Rasinski, 2000).
- Their true identity (e.g., name, address, phone number) is captured in the survey (Paulhus, 1984).
- The data is directly collected by another person (e.g., in-person or phone surveys).

To minimize social desirability bias, respondents should be allowed to answer anonymously or the survey should be self-administered (Holbrook & Krosnick, 2010; Tourangeau & Smith, 1996; Tourangeau & Yan, 2007).

Response Order Bias

Response order bias is the tendency to select the items toward the beginning (i.e., primacy effect) or the end (i.e., recency effect) of an answer list or scale (Chan, 1991; Krosnick & Alwin, 1987; Payne, 1971). Respondents unconsciously interpret the ordering of listed answer options and assume that items near each other are related, top or left items are interpreted to be “first,” and middle answers in a scale without a natural order represent the typical value (Tourangeau et al., 2004). Primacy and recency effects are the strongest when the list of answer options is long (Schuman & Presser, 1981) or when they cannot be viewed as a whole (Couper et al., 2004).

To minimize response order effects, the following may be considered:

- Unrelated answer options should be randomly ordered across respondents (Krosnick & Presser, 2010).
- Rating scales should be ordered from negative to positive, with the most negative item first.
- The order of ordinal scales should be reversed randomly between respondents, and the raw scores of both scale versions should be averaged using the same value for each scale label. That way, the response order effects cancel each other out across respondents (e.g., Villar & Krosnick, 2011), unfortunately, at the cost of increasing variability.

Question Order Bias

Order effects also apply to the order of the questions in surveys. Each question in a survey has the potential to bias each subsequent question by priming respondents (Kinder & Iyengar, 1987; Landon, 1971).

The following guidelines may be considered:

- Questions should be ordered from broad to more specific (i.e., a funnel approach) to ensure that the survey follows conversational conventions.
- Early questions should be easy to answer and directly related to the survey topic (to help build rapport and engage respondents) (Dillman, 1978).
- Non-critical, complex, and sensitive questions should be included toward the end of the survey to avoid early drop-off and to ensure collection of critical data.
- Related questions need to be grouped to reduce context switching so that respondents can more easily and quickly access related information from memory, as opposed to disparate items.
- The questionnaire should be divided into multiple pages with distinct sections labeled for easier cognitive processing.

Other Types of Questions to Avoid

Beyond the five common questionnaire biases mentioned above, there are additional question types that can result in unreliable and invalid survey data. These include broad, leading, double-barreled, recall, prediction, hypothetical, and prioritization questions.

Broad questions lack focus and include items that are not clearly defined or those that can be interpreted in multiple ways. For example, “Describe the way you use your tablet computer” is too broad, as there are many aspects to using a tablet such as the purpose, applications being used, and its locations of use. Instead of relying on the respondent to decide on which aspects to report, the research goal as well as core construct(s) should be determined beforehand and asked about in a focused manner. A more focused set of questions for the example above could be “Which apps did you use on your tablet computer over the last week?” and “Describe the locations in which you used your tablet computer last week?”.

Leading questions manipulate respondents into giving a certain answer by providing biasing content or suggesting information the researcher is looking to have confirmed. For example, “**This application was recently ranked as number one in customer satisfaction. How satisfied are you with your experience today?**”. Another way that questions can lead the respondent toward a certain answer includes those that ask the respondent to agree or disagree with a given statement, as for example in “Do you agree or disagree with the following statement: I use my smartphone more often than my tablet computer.” Note that such questions can additionally result in acquiescence bias (as discussed above). To minimize the effects of leading questions, questions should be asked in a **fully neutral way without any examples or additional information** that may bias respondents toward a particular response.

Double-barreled questions ask about multiple items while only allowing for a single response, resulting in less reliable and valid data. Such questions can usually be detected by the existence of the word “and.” For example, when asked “How satisfied or dissatisfied are you with your smartphone and tablet computer?”, a respondent with differing attitudes toward the two devices will be forced to pick an attitude that either reflects just one device or the average across both devices. Questions with multiple items should be broken down into one question per construct or item.

Recall questions require the respondent to remember past attitudes and behaviors, leading to recall bias (Krosnick & Presser, 2010) and inaccurate recollections. When a respondent is asked “**How many times did you use an Internet search engine over the past 6 months?**”, they will try to rationalize a plausible number, because recalling a precise count is difficult or impossible. Similarly, asking questions that compare past attitudes to current attitudes, as in “Do you prefer the previous or current version of the interface?”, may result in skewed data due to difficulty remembering past attitudes. Instead, questions should focus on the present, as in “How satisfied or dissatisfied are you with your smartphone today?”, or use a recent time frame, for example, “In the past hour, how many times did you use an Internet search engine?”. If the research goal is to compare attitudes or behaviors across different product versions or over time, the researcher should field separate surveys for each product version or time period and make the comparison themselves.

Prediction questions ask survey respondents to anticipate future behavior or attitudes, resulting in biased and inaccurate responses. Such questions include “Over the next month, **how frequently will you use an Internet search engine?**”. Even more cognitively burdensome are *hypothetical questions*, i.e., asking the respondent to imagine a certain situation in the future and then predicting their attitude or behavior in that situation. For example, “Would you purchase more groceries if the store played your favorite music?” and “How much would you like this Website if it used blue instead of red for their color scheme?” are hypothetical questions. Other frequently used hypothetical questions are those that ask the respondent to prioritize a future feature set, as in “**Which of the following features would make you more satisfied with this product?**”. Even though the respondent may have a clear answer to this question, **their response does not predict actual future usage of or satisfaction with the product if that feature was added**. Such questions should be entirely excluded from surveys.

Leveraging Established Questionnaires

An alternative to constructing a brand new questionnaire is utilizing questionnaires developed by others. These usually benefit from prior validation and allow researchers to compare results with other studies that used the same questionnaire. When selecting an existing questionnaire, one should consider their particular research goals and study needs and adapt the questionnaire as appropriate. Below are commonly used HCI-related questionnaire instruments. Note that as survey research methodology has significantly advanced over time, each questionnaire should be assessed for potential sources of measurement error, such as the biases and the to-be-avoided question types mentioned previously.

- *NASA Task Load Index (NASA TLX)*. Originally developed for aircraft cockpits, this questionnaire allows researchers to subjectively assess the workload of operators working with human–machine systems. It measures mental demand, physical demand, temporal demand, performance, effort, and frustration (Hart & Staveland, 1988).
- *Questionnaire for User Interface Satisfaction (QUIS)*. This questionnaire assesses one's overall reaction to a system, including its software, screen, terminology, system information, and learnability (Chin, Diehl, & Norman, 1988).
- *Software Usability Measurement Inventory (SUMI)*. This questionnaire measures perceived software quality covering dimensions such as efficiency, affect, helpfulness, control, and learnability, which are then summarized into a single satisfaction score (Kirakowski & Corbett, 1993).
- *Computer System Usability Questionnaires (CSUQ)*. This questionnaire developed by IBM measures user satisfaction with system usability (Lewis, 1995).
- *System Usability Scale (SUS)*. As one of the most frequently used scales in user experience, SUS measures attitudes regarding the effectiveness, efficiency, and satisfaction with a system with ten questions, yielding a single score (Brooke, 1996).
- *Visual Aesthetics of Website Inventory (VisAwi)*. This survey measures perceived visual aesthetics of a Website on the four subscales of simplicity, diversity, colorfulness, and craftsmanship (Moshagen & Thielsch, 2010).

Visual Survey Design Considerations

Researchers should also take into account their survey's visual design, since specific choices, including the use of images, spacing, and progress bars, may unintentionally bias respondents. This section summarizes such visual design aspects; for more details, refer to Couper (2008).

While objective images (e.g., product screenshots) can help clarify questions, context-shaping images can influence a respondent's mindset. For example, when asking respondents to rate their level of health, presenting an image of someone in a hospital bed has a framing effect that results in higher health ratings compared to that of someone jogging (Couper, Conrad, & Tourangeau, 2007).

The visual treatment of response options also matters. When asking closed-ended questions, uneven spacing between horizontal scale options results in a higher selection rate for scale points with greater spacing; evenly spaced scale options are recommended (Tourangeau, Couper, & Conrad, 2004). Drop-down lists, compared to radio buttons, have been shown to be harder and slower to use and to result in more accidental selections (Couper, 2011). Lastly, larger text fields increase the amount of text entered (Couper, 2011) but may intimidate respondents, potentially causing higher break-offs (i.e., drop-out rates).

Survey questions can be presented one per page, multiple per page, or all on one page. Research into pagination effects on completion rates is inconclusive (Couper, 2011). However, questions appearing on the same page may have higher correlations with each other, a sign of measurement bias (Peytchev, Couper, McCabe, & Crawford, 2006). In practice, most Internet surveys with skip logic use multiple pages, whereas very short questionnaires are often presented on a single page.

While progress bars are generally preferred by respondents and are helpful for short surveys, their use in long surveys or surveys with skip logic can be misleading and intimidating. Progress between pages in long surveys may be small, resulting in increased break-off rates (Callegaro, Villar, & Yang, 2011). On the other hand, progress bars are likely to increase completion rates for short surveys, where substantial progress is shown between pages.

Review and Survey Pretesting

At this point in the survey life cycle, it is appropriate to have potential respondents take and evaluate the survey in order to identify any remaining points of confusion. For example, the phrase “mobile device” may be assumed to include mobile phones, tablets, and in-car devices by the researcher, while survey respondents may interpret it to be mobile phones only. Or, when asking for communication tools used by the respondent, the provided list of answer choices may not actually include all possible options needed to properly answer the question. Two established evaluation methods used to improve survey quality are cognitive pretesting and field testing the survey by launching it to a subset of the actual sample, as described more fully in the remainder of this section. By evaluating surveys early on, the researcher can identify disconnects between their own assumptions and how respondents will read, interpret, and answer questions.

Cognitive Pretesting

To conduct a cognitive pretest, a small set of potential respondents is invited to participate in an in-person interview where they are asked to take the survey while using the think-aloud protocol (similar to a usability study). A cognitive pretest assesses question interpretation, construct validity, and comprehension of survey

terminology and calls attention to missing answer options or entire questions (Bolton & Bronkhorst, 1995; Collins, 2003; Drennan, 2003; Presser et al., 2004). However, note that due to the testing environment, a cognitive pretest does not allow the researcher to understand contextual influences that may result in break-off or not filling out the survey in the first place.

As part of a pretest, participants are asked the following for each question:

1. “Read the entire question and describe it in your own words.”
2. “Select or write an answer while explaining your thought process.”
3. “Describe any confusing terminology or missing answer choices.”

During the interview, the researcher should observe participant reactions; identify misinterpretations of terms, questions, answer choices, or scale items; and gain insight into how respondents process questions and come up with their answers. The researcher then needs to analyze the collected information to improve problematic areas before fielding the final questionnaire. A questionnaire could go through several rounds of iteration before reaching the desired quality.

Field Testing

Piloting the survey with a small subset of the sample will help provide insights that cognitive pretests alone cannot (Collins, 2003; Presser et al., 2004). Through field testing, the researcher can assess the success of the sampling approach, look for common break-off points and long completion times, and examine answers to open-ended questions. High break-off rates and completion times may point to flaws in the survey design (see the following section), while unusual answers may suggest a disconnect between a question’s intention and respondents’ interpretation. To yield additional insights from the field test, a question can be added at the end of each page or at the end of the entire survey where respondents can provide explicit feedback on any points of confusion. Similar to cognitive pretests, field testing may lead to several rounds of questionnaire improvement as well as changes to the sampling method. Finally, once all concerns are addressed, the survey is ready to be fielded to the entire sample.

Implementation and Launch

When all questions are finalized, the survey is ready to be fielded based on the chosen sampling method. Respondents may be invited through e-mails to specifically named persons (e.g., respondents chosen from a panel), intercept pop-up dialogs while using a product or a site, or links placed directly in an application (see the sampling section for more details; Couper, 2000).

There are many platforms and tools that can be used to implement Internet surveys, such as ConfirmIt, Google Forms, Kinesis, LimeSurvey, SurveyGizmo, SurveyMonkey, UserZoom, Wufoo, and Zoomerang, to name just a few. When deciding on the appropriate platform, functionality, cost, and ease of use should be taken into consideration. The questionnaire may require a survey tool that supports functionality such as branching and conditionals, the ability to pass URL parameters, multiple languages, and a range of question types. Additionally, the researcher may want to customize the visual style of the survey or set up an automatic reporting dashboard, both of which may only be available on more sophisticated platforms.

Piping Behavioral Data into Surveys

Some platforms support the ability to combine survey responses with other log data, which is referred to as piping. Self-reported behaviors, such as frequency of use, feature usage, tenure, and platform usage, are less valid and reliable compared to generating the same metrics through log data. By merging survey responses with behavioral data, the researcher can more accurately understand the relationship between respondent characteristics and their behaviors or attitudes. For example, the researcher may find that certain types of users or the level of usage may correlate with higher reported satisfaction. Behavioral data can either be passed to the results database as a parameter in the survey invitation link or combined later via a unique identifier for each respondent.

Monitoring Survey Paradata

With the survey's launch, researchers should monitor the initial responses as well as survey paradata to identify potential mistakes in the survey design. Survey paradata is data collected about the survey response process, such as the devices from which the survey was accessed, time to survey completion, and various response-related rates. By monitoring such metrics, the survey researcher can quickly apply improvements before the entire sample has responded to the survey. The American Association for Public Opinion Research specified a set of definitions for commonly used paradata metrics (AAPOR, 2011):

- Click-through rate: Of those invited, how many opened the survey.
- Completion rate: Of those who opened the survey, how many finished the survey.
- Response rate: Of those invited, how many finished the survey.
- Break-off rate: Of those who started, how many dropped off on each page.
- Completion time: The time it took respondents to finish the entire survey.

Response rates are dependent on a variety of factors, the combination of which makes it difficult to specify an acceptable response rate in HCI survey research. A meta-analysis of 31 e-mail surveys from 1986 to 2000 showed that average response rates for e-mail surveys typically fall between 30 and 40 %, with follow-up

reminders significantly increasing response rates (Sheehan, 2001). Another review of 69 e-mail surveys showed that response rates averaged around 40 % (Cook, Heath, & Thompson, 2000). When inviting respondents through Internet intercept surveys (e.g., pop-up surveys or in-product links), response rates may be 15 % or lower (Couper, 2000). Meta-analyses of mailed surveys showed that their response rates are 40–50 % (Kerlinger, 1986) or 55 % (Baruch, 1999). In experimental comparisons to mailed surveys, response rates to Internet e-mail surveys were about 10 % lower (Kaplowitz, Hadlock, & Levine, 2004; Manfreda et al., 2008). Such meta reviews also showed that overall response rates have been declining over several decades (Baruch, 1999; Baruch & Holtom, 2008; Sheehan, 2001); however, this decline seems to have stagnated around 1995 (Baruch & Holtom, 2008).

Maximizing Response Rates

In order to gather enough responses to represent the target population with the desired level of precision, response rates should be maximized. Several factors affect response rates, including the respondents' interest in the subject matter, the perceived impact of responding to the survey, questionnaire length and difficulty, the presence and nature of incentives, and researchers' efforts to encourage response (Fan & Yan, 2010).

Based on experimentation with invitation processes for mail surveys, Dillman (1978) developed the “Total Design Method” to optimize response rates. This method, consistently achieving response rates averaging 70 % or better, consists of a timed sequence of four mailings: the initial request with the survey on week one, a reminder postcard on week two, a replacement survey to non-respondents on week four, and a second replacement survey to non-respondents by certified mail on week seven. Dillman incorporates social exchange theory into the Total Design Method by personalizing the invitation letters, using official stationery to increase trust in the survey's sponsorship, explaining the usefulness of the survey research and the importance of responding, assuring the confidentiality of respondents' data, and beginning the questionnaire with items directly related to the topic of the survey (1991). Recognizing the need to cover Internet and mixed-mode surveys, Dillman extended his prior work with the “Tailored Design Method.” With this update, he emphasized customizing processes and designs to fit each survey's topic, population, and sponsorship (2007).

Another component of optimizing response rates is getting as many complete responses as possible from those who start the survey. According to Peytchev (2009), causes of break-off may fall into the following three categories:

- Respondent factors (survey topic salience and cognitive ability)
- Survey design factors (length, progress indicators, and incentives)
- Question design factors (fatigue and intimidation from open-ended questions and lengthy grid questions)

The questionnaire design principles mentioned previously may help minimize break-off, such as making surveys as short as possible, having a minimum of required questions, using skip logic, and including progress bars for short surveys.

Providing an incentive to encourage survey responses may be advantageous in certain cases. Monetary incentives tend to increase response rates more than non-monetary incentives (Singer, 2002). In particular, non-contingent incentives, which are offered to all people in the sample, generally outperform contingent incentives, given only upon completion of the survey (Church, 1993). This is true even when a non-contingent incentive is considerably smaller than a contingent incentive. One strategy to maximize the benefit of incentives is to offer a small non-contingent award to all invitees, followed by a larger contingent award to initial non-respondents (Lavrakas, 2011). An alternate form of contingent incentive is a lottery, where a drawing is held among respondents for a small number of monetary awards or other prizes. However, the efficacy of such lotteries is unclear (Stevenson, Dykema, Cyffka, Klein, & Goldrick-Rab, 2012). Although incentives will typically increase response rates, it is much less certain whether they increase the representativeness of the results. Incentives are likely most valuable when facing a small population or sampling frame, and high response rates are required for sufficiently precise measurements. Another case where incentives may help is when some groups in the sample have low interest in the survey topic (Singer, 2002). Furthermore, when there is a cost to contact each potential respondent, as with door-to-door interviewing, incentives will decrease costs by lowering the number of people that need to be contacted.

Data Analysis and Reporting

Once all the necessary survey responses have been collected, it is time to start making sense of the data by:

1. Preparing and exploring the data
2. Thoroughly analyzing the data
3. Synthesizing insights for the target audience of this research

Data Preparation and Cleaning

Cleaning and preparing survey data before conducting a thorough analysis are essential to identify low-quality responses that may otherwise skew the results. When taking a pass through the data, survey researchers should look for signs of poor-quality responses. Such survey data can either be left as is, removed, or presented separately from trusted data. If the researcher decides to remove poor data, they must cautiously decide whether to remove data on the respondent level (i.e., listwise deletion), an individual question level (i.e., pairwise deletion), or only beyond a certain point in the survey where respondents' data quality is declined. The following are signals that survey researchers should look out for at the survey response level:

- **Duplicate responses.** In a self-administered survey, a respondent might be able to fill out the survey more than once. If possible, respondent information such as name, e-mail address, or any other unique identifier should be used to remove duplicate responses.
- **Speeders.** Respondents that complete the survey faster than possible, speeders, may have carelessly read and answered the questions, resulting in arbitrary responses. The researcher should examine the distribution of response times and remove any respondents that are suspiciously fast.
- **Straight-liners and other questionable patterns.** Respondents that always, or almost always, pick the same answer option across survey questions are referred to as straight-liners. Grid-style questions are particularly prone to respondent straight-lining (e.g., by always picking the first answer option when asked to rate a series of objects). Respondents may also try to hide the fact that they are randomly choosing responses by answering in a fixed pattern (e.g., by alternating between the first and second answer options across questions). If a respondent straight-lines through the entire survey, the researcher may decide to remove the respondent's data entirely. If a respondent starts straight-lining at a certain point, the researcher may keep data up until that point.
- **Missing data and break-offs.** Some respondents may finish a survey but skip several questions. Others may start the survey but break off at some point. Both result in missing data. It should first be determined whether those who did not respond to certain questions are different from those who did. A non-response study should be conducted to assess the amount of non-response bias for each survey question. If those who did not answer certain questions are not meaningfully different from those who did, the researcher can consider leaving the data as is; however, if there is a difference, the researcher may choose to impute plausible values based on similar respondents' answers (De Leeuw, Hox, & Huisman, 2003).

Furthermore, the following signals may need to be assessed at a question-by-question level:

- **Low inter-item reliability.** When multiple questions are used to measure a single construct, respondents' answers to these questions should be associated with each other. Respondents that give inconsistent or unreliable responses (e.g., selecting "very fast" and "very slow" for separate questions assessing the construct of speed) may not have carefully read the set of questions and should be considered for removal.
- **Outliers.** Answers that significantly deviate from the majority of responses are considered outliers and should be examined. For questions with numeric values, some consider outliers as the top and bottom 2 % of responses, while others calculate outliers as anything outside of **two or three standard deviations from the mean**. Survey researchers should determine how much of a difference keeping or removing the outliers has on variables' averages. If the impact is significant, the researcher may either remove such responses entirely or replace them with a value that equals two or three standard deviations from the mean. Another way to describe the central tendency while minimizing the effect of outliers is to use the median, rather than the mean.

- *Inadequate open-ended responses.* Due to the amount of effort required, open-ended questions may lead to low-quality responses. Obvious garbage and irrelevant answers, such as “**asdf**,” should be removed, and other answers from the same respondent should be examined to determine whether all their survey responses warrant removal.

Analysis of Closed-Ended Responses

To get an overview of what the survey data shows, *descriptive statistics* are fundamental. By looking at measures such as the frequency distribution, central tendency (e.g., **mean** or median), and data dispersion (e.g., **standard deviation**), emerging patterns can be uncovered. The frequency distribution shows the proportion of responses for each answer option. The central tendency measures the “central” position of a frequency distribution and is calculated using the mean, median, and mode. Dispersion examines the data spread around the central position through calculations such as standard deviation, variance, range, and interquartile range.

While descriptive statistics only describe the existing data set, *inferential statistics* can be used to draw inferences from the sample to the overall population in question. Inferential statistics consists of two areas: estimation statistics and hypothesis testing. Estimation statistics involves using the survey’s sample in order to approximate the population’s value. Either the margin of error or the confidence interval of the sample’s data needs to be determined for such estimation. To calculate the margin of error for an answer option’s proportion, only the sample size, the proportion, and a selected confidence level are needed. However, to determine the confidence interval for a mean, the standard error of the mean is required additionally. A confidence interval thus represents the estimated range of a population’s mean at a certain confidence level.

Hypothesis testing determines the probability of a hypothesis being true when comparing groups (e.g., means or proportions being the same or different) through the use of methods such as **t-test**, **ANOVA**, or Chi-square. The appropriate test is determined by the research question, type of prediction by the researcher, and type of variable (i.e., nominal, ordinal, interval, or ratio). An experienced quantitative researcher or statistician should be involved.

Inferential statistics can also be applied to identify connections among variables:

- **Bivariate correlations** are widely used to assess linear relationships between variables. For example, correlations can indicate which product dimensions (e.g., ease of use, speed, features) are most strongly associated with users’ overall satisfaction.
- **Linear regression analysis** indicates the proportion of variance in a continuous dependent variable that is explained by one or more independent variables and the amount of change explained by each unit of an independent variable.

- **Logistic regression** predicts the change in probability of getting a particular value in a binary variable, given a unit change in one or more independent variables.
- **Decision trees** assess the probabilities of reaching specific outcomes, considering relationships between variables.
- **Factor analysis** identifies groups of covariates and can be useful to reduce a large number of variables into a smaller set.
- **Cluster analysis** looks for related groups of respondents and is often used by market researchers to identify and categorize segments within a population.

There are many packages available to assist with survey analysis. Software such as Microsoft Excel, and even certain survey platforms such as SurveyMonkey or Google Forms, can be used for basic descriptive statistics and charts. More advanced packages such as SPSS, R, SAS, or Matlab can be used for complex modeling, calculations, and charting. Note that data cleaning often needs to be a precursor to conducting analysis using such tools.

Analysis of Open-Ended Comments

In addition to analyzing closed-ended responses, the review of open-ended comments contributes a more holistic understanding of the phenomena being studied. Analyzing a large set of open-ended comments may seem like a daunting task at first; however, if done correctly, it reveals important insights that cannot otherwise be extracted from closed-ended responses. The analysis of open-ended survey responses can be derived from the method of **grounded theory** (Böhm, 2004; Glaser & Strauss, 1967) (see chapter on “Grounded Theory Methods”).

An interpretive method, referred to as **coding** (Saldaña, 2009), is used to organize and transform qualitative data from open-ended questions to enable further quantitative analysis (e.g., preparing a frequency distribution of the codes or comparing the responses across groups). The core of such qualitative analysis is to assign one or several codes to each comment; each code consists of a word or a short phrase summarizing the essence of the response with regard to the objective of that survey question (e.g., described frustrations, behavior, sentiment, or user type). Available codes are chosen from a coding scheme, which may already be established by the community or from previous research or may need to be created by the researchers themselves. In most cases, as questions are customized to each individual survey, the researcher needs to establish the coding system using a **deductive** or an **inductive** approach.

When employing a **deductive approach**, the researcher defines the full list of possible codes in a top-down fashion; i.e., all codes are defined before reviewing the qualitative data and assigning those codes to comments. On the other hand, when using an **inductive** approach to coding, the codes are generated and constantly revised in a bottom-up approach; i.e., the data is coded according to categories

identified by reading and re-reading responses to the open-ended question. Bottom-up, inductive coding is recommended, as it has the benefit of capturing categories the researcher may not have thought of before reading the actual comments; however, it requires more coordination if multiple coders are involved. (See “Grounded Theory Method” chapter for an analogous discussion.)

To measure the reliability of both the developed coding system and the coding of the comments, either the same coder should partially repeat the coding or a second coder should be involved. *Intra-rater reliability* describes the degree of agreement when the data set is reanalyzed by the same researcher. *Inter-rater reliability* (Armstrong, Gosling, Weinman, & Marteau, 1997; Gwet, 2001) determines the agreement level of the coding results from at least two independent researchers (using correlations or Cohen’s kappa). If there is low agreement, the coding needs to be reviewed to identify the pattern behind the disagreement, coder training needs to be adjusted, or changes to codes need to be agreed upon to achieve consistent categorization. If the data set to be coded is too large and coding needs to be split up between researchers, inter-rater consistency can be measured by comparing results from coding an overlapping set of comments, by comparing the coding to a preestablished standard, or by including another researcher to review overlapping codes from the main coders.

After having analyzed all comments, the researcher may prepare descriptive statistics such as a frequency distribution of codes, conduct inferential statistical tests, summarize key themes, prepare necessary charts, and highlight specifics through the use of representative quotes. To compare results across groups, inferential analysis methods can be used as described above for closed-ended data (e.g., *t*-tests, ANOVA, or Chi-square).

Assessing Representativeness

A key criterion in any survey’s quality is the degree to which the results accurately represent the target population. If a survey’s sampling frame fully covers the population and the sample is randomly drawn from the sampling frame, a response rate of 100 % would ensure that the results are representative at a level of precision based on the sample size.

If, however, a survey has less than a 100 % response rate, those not responding might have provided a different answer distribution than those who did respond.

An example is a survey intended to measure attitudes and behaviors regarding a technology that became available recently. Since people who are early adopters of new technologies are usually very passionate about providing their thoughts and feedback, surveying users of this technology product would overestimate responses from early adopters (as compared to more occasional users) and the incidence of favorable attitudes toward that product. Thus, even a modest level of non-response can greatly affect the degree of non-response bias.

With response rates to major longitudinal surveys having decreased over time, much effort has been devoted to understanding non-response and its impact on data

quality as well as methods of adjusting results to mitigate non-response error. Traditional survey assumptions held that maximizing response rates minimized non-response bias (Groves, 2006). Therefore, the results of Groves' 2006 meta-analysis were both surprising and seminal, finding no meaningful correlation between response rates and non-response error across mail, telephone, and face-to-face surveys.

Reporting Survey Findings

Once the question-by-question analysis is completed, the researcher needs to synthesize findings across all questions to address the goals of the survey. Larger themes may be identified, and the initially defined research questions are answered, which are in turn translated into recommendations and broader HCI implications as appropriate. All calculations used for the data analysis should be reported with the necessary statistical rigor (e.g., sample sizes, p -values, margins of error, and confidence levels). Furthermore, it is important to list the survey's paradata and include response and break-off rates (see section on monitoring survey paradata).

Similar to other empirical research, it is important to not only report the results of the survey but also describe the original research goals and the used survey methodology. A detailed description of the survey methodology will explain the population being studied, sampling method, survey mode, survey invitation, fielding process, and response paradata. It should also include screenshots of the actual survey questions and explain techniques used to evaluate data quality. Furthermore, it is often necessary to include a discussion on how the respondents compare to the overall population. Lastly, any potential sources of survey bias, such as sampling biases or non-response bias, should be outlined.

Exercises

1. What are the differences between a survey and a questionnaire, both in concept and design?
2. In your own research area, create a survey and test it with five classmates. How long do you think it will take a classmate to fill it out? How long did it take them?

Acknowledgements We would like to thank our employers Google, Inc. and Twitter, Inc. for making it possible for us to work on this chapter. There are many that contributed to this effort, and we would like to call out the most significant ones: Carolyn Wei for identifying published papers that used survey methodology for their work, Sandra Lozano for her insights on analysis, Mario Callegaro for inspiration, Ed Chi and Robin Jeffries for reviewing several drafts of this document, and Professors Jon Krosnick from Stanford University and Mick Couper from the University of Michigan for laying the foundation of our survey knowledge and connecting us to the broader survey research community.

References

Overview Books

- Couper, M. (2008). *Designing effective Web surveys*. Cambridge, UK: Cambridge University Press.
- Fowler, F. J., Jr. (1995). *Improving survey questions: Design and evaluation* (Vol. 38). Thousand Oaks, CA: Sage. Incorporated.
- Groves, R. M. (1989). *Survey errors and survey costs*. Hoboken, NJ: Wiley.
- Groves, R. M. (2004). *Survey errors and survey costs* (Vol. 536). Hoboken, NJ: Wiley-Interscience.
- Groves, R. M., Fowler, F. J., Couper, M. P., Lepkowski, J. M., Singer, E., & Tourangeau, R. (2004). *Survey methodology*. Hoboken, NJ: Wiley.
- Marsden, P. V., & Wright, J. (Eds.). (2010). *Handbook of survey research* (2nd ed.). Bingley, UK: Emerald Publishing Group Limited.

Sampling Methods

- Aquilino, W. S. (1994). Interview mode effects in surveys of drug and alcohol use: A field experiment. *Public Opinion Quarterly*, 58(2), 210–240.
- Cochran, W. G. (1977). *Sampling techniques* (3rd ed.). New York, NY: Wiley.
- Couper, M. P. (2000). Web surveys: A review of issues and approaches. *Public Opinion Quarterly*, 64, 464–494.
- Kish, L. (1965). *Survey sampling*. New York, NY: Wiley.
- Krejcie, R. V., & Morgan, D. W. (1970). Determining sample size for research activities. *Educational and Psychological Measurement*, 30, 607–610.
- Lohr, S. L. (1999). *Sampling: Design and analysis*. Pacific Grove, CA: Duxbury Press.

Questionnaire Design

- Bradburn, N. M., Sudman, S., & Wansink, B. (2004). *Asking questions: The definitive guide to questionnaire design – for market research, political polls, and social and health questionnaires*. San Francisco, CA: Jossey-Bass. Revised.
- Cannell, C. F., & Kahn, R. L. (1968). Interviewing. *The Handbook of Social Psychology*, 2, 526–595.
- Chan, J. C. (1991). Response-order effects in Likert-type scales. *Educational and Psychological Measurement*, 51(3), 531–540.
- Costa, P. T., & McCrae, R. R. (1988). From catalog to classification: Murray's needs and the five-factor model. *Journal of Personality and Social Psychology*, 55(2), 258.
- Couper, M. P., Tourangeau, R., Conrad, F. G., & Crawford, S. D. (2004). What they see is what we get response options for web surveys. *Social Science Computer Review*, 22(1), 111–127.
- Edwards, A. L., & Kenney, K. C. (1946). A comparison of the Thurstone and Likert techniques of attitudes scale construction. *Journal of Applied Psychology*, 30, 72–83.
- Goffman, E. (1959). The presentation of self in everyday life, 1–17. Garden City, NY
- Goldberg, L. R. (1990). An alternative description of personality: The big-five factor structure. *Journal of Personality and Social Psychology*, 59(6), 1216.

- Herzog, A. R., & Bachman, J. G. (1981). Effects of questionnaire length on response quality. *Public Opinion Quarterly*, 45(4), 549–559.
- Holbrook, A. L., & Krosnick, J. A. (2010). Social desirability bias in voter turnout reports tests using the item count technique. *Public Opinion Quarterly*, 74(1), 37–67.
- Kinder, D. R., & Iyengar, S. (1987). *News That Matters: Television and American Opinion*. Chicago: University of Chicago Press.
- Krosnick, J. A. (1991). Response strategies for coping with the cognitive demands of attitude measures in surveys. *Applied Cognitive Psychology*, 5, 213–236.
- Krosnick, J. A. (1999). Survey research. *Annual review of psychology*, 50(1), 537–567.
- Krosnick, J. A. (2002). The causes of no-opinion responses to attitude measures in surveys: They are rarely what they appear to be. In R. Groves, D. Dillman, J. Eltinge, & R. Little (Eds.), *Survey non-response* (pp. 87–100). New York: Wiley.
- Krosnick, J. A., & Alwin, D. F. (1987). *Satisficing: A strategy for dealing with the demands of survey questions*. Columbus, OH: Ohio State University.
- Krosnick, J. A., & Alwin, D. F. (1988). A test of the form-resistant correlation hypothesis ratings, rankings, and the measurement of values. *Public Opinion Quarterly*, 52(4), 526–538.
- Krosnick, J. A., & Fabrigar, L. A. (1997). Designing rating scales for effective measurement in surveys. In L. Lyberg et al. (Eds.), *Survey measurement and process quality* (pp. 141–164). New York: Wiley.
- Krosnick, J. A., Narayan, S., & Smith, W. R. (1996). Satisficing in surveys: Initial evidence. *New Directions for Evaluation*, 1996(70), 29–44.
- Krosnick, J. A., & Presser, S. (2010). Question and questionnaire design. In P. V. Marsden & J. D. Wright (Eds.), *Handbook of survey research* (pp. 263–314). Bingley, UK: Emerald Group Publishing Limited.
- Landon, E. L. (1971). Order bias, the ideal rating, and the semantic differential. *Journal of Marketing Research*, 8(3), 375–378.
- O'Muircheartaigh, C. A., Krosnick, J. A., & Helic, A. (2001). Middle alternatives, acquiescence, and the quality of questionnaire data. In B. Irving (Ed.), *Harris Graduate School of Public Policy Studies*. Chicago, IL: University of Chicago.
- Paulhus, D. L. (1984). Two-component models of socially desirable responding. *Journal of Personality and Social Psychology*, 46(3), 598.
- Payne, S. L. (1951). *The art of asking questions*. Princeton, NJ: Princeton University Press.
- Payne, J. D. (1971). The effects of reversing the order of verbal rating scales in a postal survey. *Journal of the Marketing Research Society*, 14, 30–44.
- Rohrmann, B. (2003). Verbal qualifiers for rating scales: Sociolinguistic considerations and psychometric data. Project Report. Australia: University of Melbourne
- Saris, W. E., Revilla, M., Krosnick, J. A., & Schaeffer, E. M. (2010). Comparing questions with agree/disagree response options to questions with construct-specific response options. *Survey Research Methods*, 4(1), 61–79.
- Schaeffer, N. C., & Presser, S. (2003). The science of asking questions. *Annual Review of Sociology*, 29, 65–88.
- Schlenker, B. R., & Weigold, M. F. (1989). Goals and the self-identification process: Constructing desired identities. In L. Pervin (Ed.), *Goal concepts in personality and social psychology* (pp. 243–290). Hillsdale, NJ: Erlbaum.
- Schuman, H., & Presser, S. (1981). *Questions and answers in attitude surveys*. New York: Academic Press.
- Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, 63(2), 129–138.
- Smith, D. H. (1967). Correcting for social desirability response sets in opinion-attitude survey research. *Public Opinion Quarterly*, 31, 87–94.
- Stone, G. C., Gage, N. L., & Leavitt, G. S. (1957). Two kinds of accuracy in predicting another's responses. *The Journal of Social Psychology*, 45(2), 245–254.

- Tourangeau, R. (1984). *Cognitive science and survey methods. Cognitive aspects of survey methodology: Building a bridge between disciplines* (pp. 73–100). Washington, DC: National Academy Press.
- Tourangeau, R., Couper, M. P., & Conrad, F. (2004). Spacing, position, and order: Interpretive heuristics for visual features of survey questions. *Public Opinion Quarterly*, 68(3), 368–393.
- Tourangeau, R., Rips, L. J., & Rasinski, K. (2000). *The psychology of survey response*. Cambridge, UK: Cambridge University Press.
- Tourangeau, R., & Smith, T. W. (1996). Asking sensitive questions the impact of data collection mode, question format, and question context. *Public Opinion Quarterly*, 60(2), 275–304.
- Tourangeau, R., & Yan, T. (2007). Sensitive questions in surveys. *Psychological Bulletin*, 133(5), 859.
- Villar, A., & Krosnick, J. A. (2011). Global warming vs. climate change, taxes vs. prices: Does word choice matter? *Climatic change*, 105(1), 1–12.

Visual Survey Design

- Callegaro, M., Villar, A., & Yang, Y. (2011). A meta-analysis of experiments manipulating progress indicators in Web surveys. *Annual Meeting of the American Association for Public Opinion Research*, Phoenix
- Couper, M. (2011). Web survey methodology: Interface design, sampling and statistical inference. Presentation at *EUSTAT-The Basque Statistics Institute*, Vitoria-Gasteiz
- Couper, M. P., Conrad, F. G., & Tourangeau, R. (2007). Visual context effects in Web surveys. *Public Opinion Quarterly*, 71(4), 623–634.
- Peytchev, A., Couper, M. P., McCabe, S. E., & Crawford, S. D. (2006). Web survey design paging versus scrolling. *Public Opinion Quarterly*, 70(4), 596–607.
- Yan, T., Conrad, F. G., Tourangeau, R., & Couper, M. P. (2011). Should I stay or should I go: The effects of progress feedback, promised task duration, and length of questionnaire on completing Web surveys. *International Journal of Public Opinion Research*, 23(2), 131–147.

Established Questionnaire Instruments

- Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189, 194.
- Chin, J. P., Diehl, V. A., & Norman, K. L. (1988, May). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human factors in computing systems* (pp. 213–218). New York, NY: ACM
- Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Human Mental Workload*, 1, 139–183.
- Kirakowski, J., & Corbett, M. (1993). SUMI: The software usability measurement inventory. *British Journal of Educational Technology*, 24(3), 210–212.
- Lewis, J. R. (1995). IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1), 57–78.
- Moshagen, M., & Thielsch, M. T. (2010). Facets of visual aesthetics. *International Journal of Human-Computer Studies*, 68(10), 689–709.

Questionnaire Evaluation

- Bolton, R. N., & Bronkhorst, T. M. (1995). Questionnaire pretesting: Computer assisted coding of concurrent protocols. In N. Schwarz & S. Sudman (Eds.), *Answering questions* (pp. 37–64). San Francisco: Jossey-Bass.
- Collins, D. (2003). Pretesting survey instruments: An overview of cognitive methods. *Quality of Life Research an International Journal of Quality of Life Aspects of Treatment Care and Rehabilitation*, 12(3), 229–238.
- Drennan, J. (2003). Cognitive interviewing: Verbal data in the design and pretesting of questionnaires. *Journal of Advanced Nursing*, 42(1), 57–63.
- Presser, S., Couper, M. P., Lessler, J. T., Martin, E., Martin, J., Rothgeb, J. M., et al. (2004). Methods for testing and evaluating survey questions. *Public Opinion Quarterly*, 68(1), 109–130.

Survey Response Rates and Non-response

- American Association for Public Opinion Research, AAPOR. (2011). *Standard definitions: Final dispositions of case codes and outcome rates for surveys*. (7th ed). <http://aapor.org/Content/NavigationMenu/AboutAAPOR/StandardsampEthics/StandardDefinitions/StandardDefinitions2011.pdf>
- Baruch, Y. (1999). Response rates in academic studies: A comparative analysis. *Human Relations*, 52, 421–434.
- Baruch, Y., & Holtom, B. C. (2008). Survey response rate levels and trends in organizational research. *Human Relations*, 61(8), 1139–1160.
- Church, A. H. (1993). Estimating the effect of incentives on mail survey response rates: A meta-analysis. *Public Opinion Quarterly*, 57, 62–79.
- Cook, C., Heath, F., & Thompson, R. L. (2000). A meta-analysis of response rates in Web- or Internet-based surveys. *Educational and Psychological Measurement*, 60(6), 821–836.
- Dillman, D. A. (1978). *Mail and telephone surveys: The total design method*. New York: Wiley.
- Dillman, D. A. (1991). The design and administration of mail surveys. *Annual Review of Sociology*, 17, 225–249.
- Dillman, D. A. (2007). *Mail and Internet surveys: The tailored design method* (2nd ed.). Hoboken, NJ: Wiley.
- Fan, W., & Yan, Z. (2010). Factors affecting response rates of the web survey: A systematic review. *Computers in Human Behavior*, 26(2), 132–139.
- Groves, R. M. (2006). Non-response rates and non-response bias in household surveys. *Public Opinion Quarterly*, 70, 646–75.
- Groves, R. M., Presser, S., & Dipko, S. (2004). The role of topic interest in survey participation decisions. *Public Opinion Quarterly*, 68(1), 2–31.
- Kaplowitz, M. D., Hadlock, T. D., & Levine, R. (2004). A comparison of web and mail survey response rates. *Public Opinion Quarterly*, 68(1), 94–101.
- Kerlinger, F. N. (1986). *Foundations of behavioral research* (3rd ed.). New York: Holt, Rinehart & Winston.
- Kiesler, S., & Sproull, L. S. (1986). Response effects in the electronic survey. *Public Opinion Quarterly*, 50, 402–413.
- Lavrakas, P. J. (2011). The use of incentives in survey research. *66th Annual Conference of the American Association for Public Opinion Research*
- Lin, I., & Schaeffer, N. C. (1995). Using survey participants to estimate the impact of nonparticipation. *Public Opinion Quarterly*, 59(2), 236–258.

- Lu, H., & Gelman, A. (2003). A method for estimating design-based sampling variances for surveys with weighting, poststratification, and raking. *Journal of Official Statistics*, 19(2), 133–152.
- Manfreda, K. L., Bosnjak, M., Berzelak, J., Haas, I., Vehovar, V., & Berzelak, N. (2008). Web surveys versus other survey modes: A meta-analysis comparing response rates. *Journal of the Market Research Society*, 50(1), 79.
- Olson, K. (2006). Survey participation, non-response bias, measurement error bias, and total bias. *Public Opinion Quarterly*, 70(5), 737–758.
- Peytchev, A. (2009). Survey breakoff. *Public Opinion Quarterly*, 73(1), 74–97.
- Schonlau, M., Van Soest, A., Kapteyn, A., & Couper, M. (2009). Selection bias in web surveys and the use of propensity scores. *Sociological Methods & Research*, 37(3), 291–318.
- Sheehan, K. B. (2001). E-mail survey response rates: A review. *Journal of Computer Mediated Communication*, 6(2), 1–16.
- Singer, E. (2002). The use of incentives to reduce non-response in household surveys. In R. Groves, D. Dillman, J. Eltinge, & R. Little (Eds.), *Survey non-response* (pp. 87–100). New York: Wiley. 163–177.
- Stevenson, J., Dykema, J., Cyffka, C., Klein, L., & Goldrick-Rab, S. (2012). What are the odds? Lotteries versus cash incentives. Response rates, cost and data quality for a Web survey of low-income former and current college students. *67th Annual Conference of the American Association for Public Opinion Research*

Survey Analysis

- Armstrong, D., Gosling, A., Weinman, J., & Marteau, T. (1997). The place of inter-rater reliability in qualitative research: An empirical study. *Sociology*, 31(3), 597–606.
- Böhm, A. (2004). Theoretical coding: Text analysis in grounded theory. In *A companion to qualitative research*, London: SAGE. pp. 270–275.
- De Leeuw, E. D., Hox, J. J., & Huisman, M. (2003). Prevention and treatment of item nonresponse. *Journal of Official Statistics*, 19(2), 153–176.
- Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. Hawthorne, NY: Aldine de Gruyter.
- Gwet, K. L. (2001). *Handbook of inter-rater reliability*. Gaithersburg, MD: Advanced Analytics, LLC.
- Heeringa, S. G., West, B. T., & Berglund, P. A. (2010). *Applied survey data analysis*. Boca Raton, FL: Chapman & Hall/CRC.
- Lee, E. S., Forthofer, R. N., & Lorimor, R. J. (1989). *Analyzing complex survey data*. Newbury Park, CA: Sage.
- Saldaña, J. (2009). *The coding manual for qualitative researchers*. Thousand Oaks, CA: Sage Publications Limited.

Other References

- Abran, A., Khelifi, A., Suryn, W., & Seffah, A. (2003). Usability meanings and interpretations in ISO standards. *Software Quality Journal*, 11(4), 325–338.
- Anandarajan, M., Zaman, M., Dai, Q., & Arinze, B. (2010). Generation Y adoption of instant messaging: An examination of the impact of social usefulness and media richness on use richness. *IEEE Transactions on Professional Communication*, 53(2), 132–143.

- Archambault, A., & Grudin, J. (2012). A longitudinal study of facebook, linkedin, & twitter use. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI '12)* (pp. 2741–2750). New York: ACM
- Auter, P. J. (2007). Portable social groups: Willingness to communicate, interpersonal communication gratifications, and cell phone use among young adults. *International Journal of Mobile Communications*, 5(2), 139–156.
- Calfee, J. E., & Ringold, D. J. (1994). The 70 % majority: Enduring consumer beliefs about advertising. *Journal of Public Policy & Marketing*, 13(2).
- Chen, J., Geyer, W., Dugan, C., Muller, M., & Guy, I. (2009). Make new friends, but keep the old: Recommending people on social networking sites. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI '09)*, (pp. 201–210). New York: ACM
- Clauser, B. E. (2007). The life and labors of Francis Galton: A review of four recent books about the father of behavioral statistics. *Journal of Educational and Behavioral Statistics*, 32(4), 440–444.
- Converse, J. (1987). *Survey research in the United States: Roots and emergence 1890–1960*. Berkeley, CA: University of California Press.
- Drouin, M., & Landgraff, C. (2012). Texting, sexting, and attachment in college students' romantic relationships. *Computers in Human Behavior*, 28, 444–449.
- Feng, J., Lazar, J., Kumin, L., & Ozok, A. (2010). Computer usage by children with down syndrome: Challenges and future research. *ACM Transactions on Accessible Computing*, 2(3), 35–41.
- Froelich, J., Findlater, L., Ostergren, M., Ramanathan, S., Peterson, J., Wragg, I., et al. (2012). The design and evaluation of prototype eco-feedback displays for fixture-level water usage data. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI '12)* (pp. 2367–2376). New York: ACM
- Harrison, M. A. (2011). College students' prevalence and perceptions of text messaging while driving. *Accident Analysis and Prevention*, 43, 1516–1520.
- Junco, R., & Cotten, S. R. (2011). Perceived academic effects of instant messaging use. *Computers & Education*, 56, 370–378.
- Katosh, J. P., & Traugott, M. W. (1981). The consequences of validated and self-reported voting measures. *Public Opinion Quarterly*, 45(4), 519–535.
- Nacke, L. E., Grimshaw, M. N., & Lindley, C. A. (2010). More than a feeling: Measurement of sonic user experience and psychophysiology in a first-person shooter game. *Interacting with Computers*, 22(5), 336–343.
- Obermiller, C., & Spangenberg, E. R. (1998). Development of a scale to measure consumer skepticism toward advertising. *Journal of Consumer Psychology*, 7(2), 159–186.
- Obermiller, C., & Spangenberg, E. R. (2000). On the origin and distinctiveness of skepticism toward advertising. *Marketing Letters*, 11, 311–322.
- Person, A. K., Blain, M. L. M., Jiang, H., Rasmussen, P. W., & Stout, J. E. (2011). Text messaging for enhancement of testing and treatment for tuberculosis, human immunodeficiency virus, and syphilis: A survey of attitudes toward cellular phones and healthcare. *Telemedicine Journal and e-Health*, 17(3), 189–195.
- Pitkow, J. E., & Recker, M. (1994). Results from the first World-Wide web user survey. *Computer Networks and ISDN Systems*, 27(2), 243–254.
- Rodden, R., Hutchinson, H., & Fu, X. (2010). Measuring the user experience on a large scale: User-centered metrics for web applications. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI '10)* (pp. 2395–2398) ACM, New York, NY, USA
- Schild, J., LaViola, J., & Masuch, M. (2012). Understanding user experience in stereoscopic 3D games. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI '12)* (pp. 89–98). New York: ACM
- Shklovski, I., Kraut, R., & Cummings, J. (2008). Keeping in touch by technology: Maintaining friendships after a residential move. In *Proceedings of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI '08)* (pp. 807–816). New York: ACM

- Turner, M., Love, S., & Howell, M. (2008). Understanding emotions experienced when using a mobile phone in public: The social usability of mobile (cellular) telephones. *Telematics and Informatics*, 25, 201–215.
- Weisskirch, R. S., & Delevi, R. (2011). “Sexting” and adult romantic attachment. *Computers in Human Behavior*, 27, 1697–1701.
- Wright, P. J., & Randall, A. K. (2012). Internet pornography exposure and risky sexual behavior among adult males in the United States. *Computers in Human Behavior*, 28, 1410–1416.
- Yew, J., Shamma, D. A., & Churchill, E. F. (2011). Knowing funny: Genre perception and categorization in social video sharing. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI '11)* (pp. 297–306). New York: ACM
- Zaman, M., Rajan, M. A., & Dai, Q. (2010). Experiencing flow with instant messaging and its facilitating role on creative behaviors. *Computers in Human Behavior*, 26, 1009–1018.

THE
DESIGN
OF EVERYDAY
THINGS

THE PSYCHOLOGY OF EVERYDAY ACTIONS

During my family's stay in England, we rented a furnished house while the owners were away. One day, our landlady returned to the house to get some personal papers. She walked over to the old, metal filing cabinet and attempted to open the top drawer. It wouldn't open. She pushed it forward and backward, right and left, up and down, without success. I offered to help. I wiggled the drawer. Then I twisted the front panel, pushed down hard, and banged the front with the palm of one hand. The cabinet drawer slid open. "Oh," she said, "I'm sorry. I am so bad at mechanical things." No, she had it backward. It is the mechanical thing that should be apologizing, perhaps saying, "I'm sorry. I am so bad with people."



My landlady had two problems. First, although she had a clear goal (retrieve some personal papers) and even a plan for achieving that goal (open the top drawer of the filing cabinet, where those papers are kept), once that plan failed, she had no idea of what to do. But she also had a second problem: she thought the problem lay in her own lack of ability: she blamed herself, falsely.

How was I able to help? First, I refused to accept the false accusation that it was the fault of the landlady: to me, it was clearly a fault in the mechanics of the old filing cabinet that prevented the drawer from opening. Second, I had a conceptual model of how the cabinet worked, with an internal mechanism that held the door shut in normal usage, and the belief that the drawer mechanism was probably out of alignment. This conceptual model gave me a plan: wiggle the drawer. That failed. That caused me to modify

my plan: wiggling may have been appropriate but not forceful enough, so I resorted to brute force to try to twist the cabinet back into its proper alignment. This felt good to me—the cabinet drawer moved slightly—but it still didn’t open. So I resorted to the most powerful tool employed by experts the world around—I banged on the cabinet. And yes, it opened. In my mind, I decided (without any evidence) that my hit had jarred the mechanism sufficiently to allow the drawer to open.

This example highlights the themes of this chapter. First, how do people do things? It is easy to learn a few basic steps to perform operations with our technologies (and yes, even filing cabinets are technology). But what happens when things go wrong? How do we detect that they aren’t working, and then how do we know what to do? To help understand this, I first delve into human psychology and a simple conceptual model of how people select and then evaluate their actions. This leads the discussion to the role of understanding (via a conceptual model) and of emotions: pleasure when things work smoothly and frustration when our plans are thwarted. Finally, I conclude with a summary of how the lessons of this chapter translate into principles of design.



How People Do Things: The Gulfs of Execution and Evaluation

When people use something, they face two gulfs: the Gulf of Execution, where they try to figure out how it operates, and the Gulf of Evaluation, where they try to figure out what happened (Figure 2.1). The role of the designer is to help people bridge the two gulfs.

In the case of the filing cabinet, there were visible elements that helped bridge the Gulf of Execution when everything was working perfectly. The drawer handle clearly signified that it should be pulled and the slider on the handle indicated how to release the catch that normally held the drawer in place. But when these operations failed, there then loomed a big gulf: what other operations could be done to open the drawer?

The Gulf of Evaluation was easily bridged, at first. That is, the catch was released, the drawer handle pulled, yet nothing happened. The lack of action signified a failure to reach the goal. But when other operations were tried, such as my twisting and pulling, the filing cabinet provided no more information about whether I was getting closer to the goal.

The Gulf of Evaluation reflects the amount of effort that the person must make to interpret the physical state of the device and to determine how well the expectations and intentions have been met. The gulf is small when the device provides information about its state in a form that is easy to get, is easy to interpret, and matches the way the person thinks about the system. What are the major design elements that help bridge the Gulf of Evaluation? Feedback and a good conceptual model.

The gulfs are present for many devices. Interestingly, many people do experience difficulties, but explain them away by blaming themselves. In the case of things they believe they should be capable of using—water faucets, refrigerator temperature controls, stove tops—they simply think, “I’m being stupid.” Alternatively, for complicated-looking devices—sewing machines, washing machines, digital watches, or almost any digital controls—they simply give up, deciding that they are incapable of understanding them. Both explanations are wrong. These are the things of everyday household use. None of them has a complex underlying structure. The difficulties reside in their design, not in the people attempting to use them.

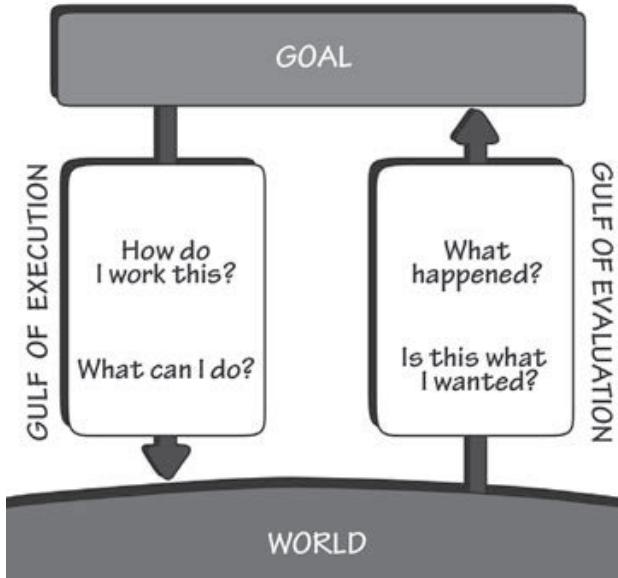


FIGURE 2.1. The Gulfs of Execution and Evaluation. When people encounter a device, they face two gulfs: the Gulf of Execution, where they try to figure out how to use it, and the Gulf of Evaluation, where they try to figure out what state it is in and whether their actions got them to their goal.

How can the designer help bridge the two gulfs? To answer that question, we need to delve more deeply into the psychology of human action. But the basic tools have already been discussed: We bridge the Gulf of Execution through the use of signifiers, constraints, mappings, and a conceptual model. We bridge the Gulf of Evaluation through the use of feedback and a conceptual model.

The Seven Stages of Action

There are two parts to an action: executing the action and then evaluating the results: doing and interpreting. Both execution and evaluation require understanding: how the item works and what results it produces. Both execution and evaluation can affect our emotional state.

Suppose I am sitting in my armchair, reading a book. It is dusk, and the light is getting dimmer and dimmer. My current activity is reading, but that goal is starting to fail because of the decreasing illumination. This realization triggers a new goal: get more light. How do I do that? I have many choices. I could open the curtains, move so that I sit where there is more light, or perhaps turn on a nearby light. This is the planning stage, determining which of the many possible plans of action to follow. But even when I decide to turn on the nearby light, I still have to determine how to get it done. I could ask someone to do it for me, I could use my left hand or my right. Even after I have decided upon a plan, I still have to specify how I will do it. Finally, I must execute—do—the action. When I am doing a frequent act, one for which I am quite experienced and skilled, most of these stages are subconscious. When I am still learning how to do it, determining the plan, specifying the sequence, and interpreting the result are conscious.

Suppose I am driving in my car and my action plan requires me to make a left turn at a street intersection. If I am a skilled driver, I don't have to give much conscious attention to specify or perform the action sequence. I think "left" and smoothly execute the required action sequence. But if I am just learning to drive, I have to think about each separate component of the action. I must apply the brakes and check for cars behind and around me, cars and

pedestrians in front of me, and whether there are traffic signs or signals that I have to obey. I must move my feet back and forth between pedals and my hands to the turn signals and back to the steering wheel (while I try to remember just how my instructor told me I should position my hands while making a turn), and my visual attention is divided among all the activity around me, sometimes looking directly, sometimes rotating my head, and sometimes using the rear- and side-view mirrors. To the skilled driver, it is all easy and straightforward. To the beginning driver, the task seems impossible.

The specific actions bridge the gap between what we would like to have done (our goals) and all possible physical actions to achieve those goals. After we specify what actions to make, we must actually do them—the stages of execution. There are three stages of execution that follow from the goal: plan, specify, and perform (the left side of Figure 2.2). Evaluating what happened has three stages: first, perceiving what happened in the world; second, trying to make sense of it (interpreting it); and, finally, comparing what happened with what was wanted (the right side of Figure 2.2).

There we have it. Seven stages of action: one for goals, three for execution, and three for evaluation (Figure 2.2).

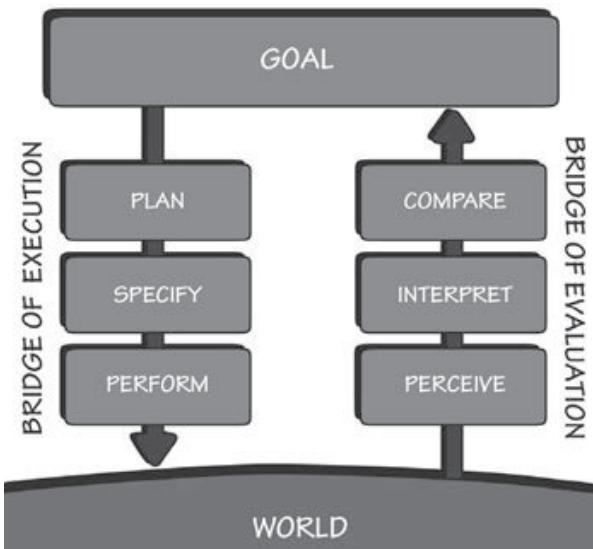


FIGURE 2.2. The Seven Stages of the Action Cycle. Putting all the stages together yields the three stages of execution (plan, specify, and perform), three stages of evaluation (perceive, interpret, and compare), and, of course, the goal: seven stages in all.

1. **Goal** (form the goal)
2. **Plan** (the action)
3. **Specify** (an action sequence)
4. **Perform** (the action sequence)
5. **Perceive** (the state of the world)
6. **Interpret** (the perception)
7. **Compare** (the outcome with the goal)

The seven-stage action cycle is simplified, but it provides a useful framework for understanding human action and for guiding design. It has proven to be helpful in designing interaction. Not all of the activity in the stages is conscious. Goals tend to be, but even they may be subconscious. We can do many actions, repeatedly cycling through the stages while being blissfully unaware that we are doing so. It is only when we come across something new or reach some impasse, some problem that disrupts the normal flow of activity, that conscious attention is required.

Most behavior does not require going through all stages in sequence; however, most activities will not be satisfied by single actions. There must be numerous sequences, and the whole activity may last hours or even days. There are multiple feedback loops in which the results of one activity are used to direct further ones, in which goals lead to subgoals, and plans lead to subplans. There are activities in which goals are forgotten, discarded, or reformulated.

Let's go back to my act of turning on the light. This is a case of event-driven behavior: the sequence starts with the world, causing evaluation of the state and the formulation of a goal. The trigger was an environmental event: the lack of light, which made reading difficult. This led to a violation of the goal of reading, so it led to a subgoal—get more light. But reading was not the high-level goal. For each goal, one has to ask, "Why is that the goal?" Why was I reading? I was trying to prepare a meal using a new recipe, so I needed to reread it before I started. Reading was thus a subgoal. But cooking was itself a subgoal. I was cooking in order to eat, which had the goal of satisfying my hunger. So the hierarchy of goals is roughly: satisfy hunger; eat; cook; read cookbook; get more light. This is called a root cause analysis: asking "Why?" until the ultimate, fundamental cause of the activity is reached.

The action cycle can start from the top, by establishing a new goal, in which case we call it goal-driven behavior. In this situation, the cycle starts with the goal and then goes through the three stages of execution. But the action cycle can also start from the bottom, triggered by some event in the world, in which case we

call it either data-driven or event-driven behavior. In this situation, the cycle starts with the environment, the world, and then goes through the three stages of evaluation.

For many everyday tasks, goals and intentions are not well specified: they are opportunistic rather than planned. Opportunistic actions are those in which the behavior takes advantage of circumstances. Rather than engage in extensive planning and analysis, we go about the day's activities and do things as opportunities arise. Thus, we may not have planned to try a new café or to ask a question of a friend. Rather, we go through the day's activities, and if we find ourselves near the café or encountering the friend, then we allow the opportunity to trigger the appropriate activity. Otherwise, we might never get to that café or ask our friend the question. For crucial tasks we make special efforts to ensure that they get done. Opportunistic actions are less precise and certain than specified goals and intentions, but they result in less mental effort, less inconvenience, and perhaps more interest. Some of us adjust our lives around the expectation of opportunities. And sometimes, even for goal-driven behavior, we try to create world events that will ensure that the sequence gets completed. For example, sometimes when I must do an important task, I ask someone to set a deadline for me. I use the approach of that deadline to trigger the work. It may only be a few hours before the deadline that I actually get to work and do the job, but the important point is that it does get done. This self-triggering of external drivers is fully compatible with the seven-stage analysis.

The seven stages provide a guideline for developing new products or services. The gulfs are obvious places to start, for either gulf, whether of execution or evaluation, is an opportunity for product enhancement. The trick is to develop observational skills to detect them. Most innovation is done as an incremental enhancement of existing products. What about radical ideas, ones that introduce new product categories to the marketplace? These come about by reconsidering the goals, and always asking what the real goal is: what is called the *root cause* analysis.

Harvard Business School marketing professor Theodore Levitt once pointed out, "People don't want to buy a quarter-inch drill.

They want a quarter-inch hole!” Levitt’s example of the drill implying that the goal is really a hole is only partially correct, however. When people go to a store to buy a drill, that is not their real goal. But why would anyone want a quarter-inch hole? Clearly that is an intermediate goal. Perhaps they wanted to hang shelves on the wall. Levitt stopped too soon.

Once you realize that they don’t really want the drill, you realize that perhaps they don’t really want the hole, either: they want to install their bookshelves. Why not develop methods that don’t require holes? Or perhaps books that don’t require bookshelves. (Yes, I know: electronic books, e-books.)

Human Thought: Mostly Subconscious

Why do we need to know about the human mind? Because things are designed to be used by people, and without a deep understanding of people, the designs are apt to be faulty, difficult to use, difficult to understand. That is why it is useful to consider the seven stages of action. The mind is more difficult to comprehend than actions. Most of us start by believing we already understand both human behavior and the human mind. After all, we are all human: we have all lived with ourselves all of our lives, and we like to think we understand ourselves. But the truth is, we don’t. Most of human behavior is a result of subconscious processes. We are unaware of them. As a result, many of our beliefs about how people behave—including beliefs about ourselves—are wrong. That is why we have the multiple social and behavioral sciences, with a good dash of mathematics, economics, computer science, information science, and neuroscience.

Consider the following simple experiment. Do all three steps:

1. Wiggle the second finger of your hand.
2. Wiggle the third finger of the same hand.
3. Describe what you did differently those two times.

On the surface, the answer seems simple: I thought about moving my fingers and they moved. The difference is that I thought

about a different finger each time. Yes, that's true. But how did that thought get transmitted into action, into the commands that caused different muscles in the arm to control the tendons that wiggled the fingers? This is completely hidden from consciousness.

The human mind is immensely complex, having evolved over a long period with many specialized structures. The study of the mind is the subject of multiple disciplines, including the behavioral and social sciences, cognitive science, neuroscience, philosophy, and the information and computer sciences. Despite many advances in our understanding, much still remains mysterious, yet to be learned. One of the mysteries concerns the nature of and distinction between those activities that are conscious and those that are not. Most of the brain's operations are subconscious, hidden beneath our awareness. It is only the highest level, what I call *reflective*, that is conscious.

Conscious attention is necessary to learn most things, but after the initial learning, continued practice and study, sometimes for thousands of hours over a period of years, produces what psychologists call "overlearning." Once skills have been overlearned, performance appears to be effortless, done automatically, with little or no awareness. For example, answer these questions:

What is the phone number of a friend?

What is Beethoven's phone number?

What is the capital of:

- Brazil?
- Wales?
- The United States?
- Estonia?

Think about how you answered these questions. The answers you knew come immediately to mind, but with no awareness of how that happened. You simply "know" the answer. Even the ones you got wrong came to mind without any awareness. You might have been aware of some doubt, but not of how the name entered your consciousness. As for the countries for which you didn't

know the answer, you probably knew you didn't know those immediately, without effort. Even if you knew you knew, but couldn't quite recall it, you didn't know how you knew that, or what was happening as you tried to remember.

You might have had trouble with the phone number of a friend because most of us have turned over to our technology the job of remembering phone numbers. I don't know anybody's phone number—I barely remember my own. When I wish to call someone, I just do a quick search in my contact list and have the telephone place the call. Or I just push the "2" button on the phone for a few seconds, which autodials my home. Or in my auto, I can simply speak: "Call home." What's the number? I don't know: my technology knows. Do we count our technology as an extension of our memory systems? Of our thought processes? Of our mind?

What about Beethoven's phone number? If I asked my computer, it would take a long time, because it would have to search all the people I know to see whether any one of them was Beethoven. But you immediately discarded the question as nonsensical. You don't personally know Beethoven. And anyway, he is dead. Besides, he died in the early 1800s and the phone wasn't invented until the late 1800s. How do we know what we do not know so rapidly? Yet some things that we do know can take a long time to retrieve. For example, answer this:

*In the house you lived in three houses ago, as you entered the front door,
was the doorknob on the left or right?*

Now you have to engage in conscious, reflective problem solving, first to retrieve just which house is being talked about, and then what the correct answer is. Most people can determine the house, but have difficulty answering the question because they can readily imagine the doorknob on both sides of the door. The way to solve this problem is to imagine doing some activity, such as walking up to the front door while carrying heavy packages with both hands: how do you open the door? Alternatively, visualize yourself inside the house, rushing to the front door to open it for a visitor.

Usually one of these imagined scenarios provides the answer. But note how different the memory retrieval for this question was from the retrieval for the others. All these questions involved long-term memory, but in very different ways. The earlier questions were memory for factual information, what is called *declarative memory*. The last question could have been answered factually, but is usually most easily answered by recalling the activities performed to open the door. This is called *procedural memory*. I return to a discussion of human memory in Chapter 3.

Walking, talking, reading. Riding a bicycle or driving a car. Singing. All of these skills take considerable time and practice to master, but once mastered, they are often done quite automatically. For experts, only especially difficult or unexpected situations require conscious attention.

Because we are only aware of the reflective level of conscious processing, we tend to believe that all human thought is conscious. But it isn't. We also tend to believe that thought can be separated from emotion. This is also false. Cognition and emotion cannot be separated. Cognitive thoughts lead to emotions: emotions drive cognitive thoughts. The brain is structured to act upon the world, and every action carries with it expectations, and these expectations drive emotions. That is why much of language is based on physical metaphors, why the body and its interaction with the environment are essential components of human thought.

Emotion is highly underrated. In fact, the emotional system is a powerful information processing system that works in tandem with cognition. Cognition attempts to make sense of the world: emotion assigns value. It is the emotional system that determines whether a situation is safe or threatening, whether something that is happening is good or bad, desirable or not. Cognition provides understanding: emotion provides value judgments. A human without a working emotional system has difficulty making choices. A human without a cognitive system is dysfunctional.

Because much human behavior is subconscious—that is, it occurs without conscious awareness—we often don't know what we are about to do, say, or think until after we have done it. It's as

if we had two minds: the subconscious and the conscious, which don't always talk to each other. Not what you have been taught? True, nonetheless. More and more evidence is accumulating that we use logic and reason after the fact, to justify our decisions to ourselves (to our conscious minds) and to others. Bizarre? Yes, but don't protest: enjoy it.

Subconscious thought matches patterns, finding the best possible match of one's past experience to the current one. It proceeds rapidly and automatically, without effort. Subconscious processing is one of our strengths. It is good at detecting general trends, at recognizing the relationship between what we now experience and what has happened in the past. And it is good at generalizing, at making predictions about the general trend, based on few examples. But subconscious thought can find matches that are inappropriate or wrong, and it may not distinguish the common from the rare. Subconscious thought is biased toward regularity and structure, and it is limited in formal power. It may not be capable of symbolic manipulation, of careful reasoning through a sequence of steps.

Conscious thought is quite different. It is slow and labored. Here is where we slowly ponder decisions, think through alternatives, compare different choices. Conscious thought considers first this approach, then that—comparing, rationalizing, finding explanations. Formal logic, mathematics, decision theory: these are the tools of conscious thought. Both conscious and subconscious modes of thought are powerful and essential aspects of human life. Both can provide insightful leaps and creative moments. And both are subject to errors, misconceptions, and failures.

Emotion interacts with cognition biochemically, bathing the brain with hormones, transmitted either through the bloodstream or through ducts in the brain, modifying the behavior of brain cells. Hormones exert powerful biases on brain operation. Thus, in tense, threatening situations, the emotional system triggers the release of hormones that bias the brain to focus upon relevant parts of the environment. The muscles tense in preparation for action. In calm, nonthreatening situations, the emotional system triggers the release of hormones that relax the muscles and bias the brain toward explo-

TABLE 2.1. Subconscious and Conscious Systems of Cognition

Subconscious	Conscious
Fast	Slow
Automatic	Controlled
Multiple resources	Limited resources
Controls skilled behavior	Invoked for novel situations: when learning, when in danger, when things go wrong

ration and creativity. Now the brain is more apt to notice changes in the environment, to be distracted by events, and to piece together events and knowledge that might have seemed unrelated earlier.

A positive emotional state is ideal for creative thought, but it is not very well suited for getting things done. Too much, and we call the person scatterbrained, flitting from one topic to another, unable to finish one thought before another comes to mind. A brain in a negative emotional state provides focus: precisely what is needed to maintain attention on a task and finish it. Too much, however, and we get tunnel vision, where people are unable to look beyond their narrow point of view. Both the positive, relaxed state and the anxious, negative, and tense state are valuable and powerful tools for human creativity and action. The extremes of both states, however, can be dangerous.

Human Cognition and Emotion

The mind and brain are complex entities, still the topic of considerable scientific research. One valuable explanation of the levels of processing within the brain, applicable to both cognitive and emotional processing, is to think of three different levels of processing, each quite different from the other, but all working together in concert. Although this is a gross oversimplification of the actual processing, it is a good enough approximation to provide guidance in understanding human behavior. The approach I use here comes from my book *Emotional Design*. There, I suggested

that a useful approximate model of human cognition and emotion is to consider three levels of processing: visceral, behavioral, and reflective.

THE VISCERAL LEVEL

The most basic level of processing is called *visceral*. This is sometimes referred to as “the lizard brain.” All people have the same basic visceral responses. These are part of the basic protective mechanisms of the human affective system, making quick judgments about the environment: good or bad, safe or dangerous. The visceral system allows us to respond quickly and subconsciously, without

conscious awareness or control. The basic biology of the visceral system minimizes its ability to learn. Visceral learning takes place primarily by sensitization or desensitization through such mechanisms as adaptation and classical conditioning. Visceral responses are fast and automatic.

Three Levels of Processing



FIGURE 2.3. Three Levels of Processing: Visceral, Behavioral, and Reflective. Visceral and behavioral levels are subconscious and the home of basic emotions. The reflective level is where conscious thought and decision-making reside, as well as the highest level of emotions.

They give rise to the startle reflex for novel, unexpected events; for such genetically programmed behavior as fear of heights, dislike of the dark or very noisy environments, dislike of bitter

tastes and the liking of sweet tastes, and so on. Note that the visceral level responds to the immediate present and produces an affective state, relatively unaffected by context or history. It simply assesses the situation: no cause is assigned, no blame, and no credit.

The visceral level is tightly coupled to the body’s musculature—the motor system. This is what causes animals to fight or flee, or to relax. An animal’s (or person’s) visceral state can often be read by analyzing the tension of the body: tense means a negative state; relaxed, a positive state. Note, too, that we often determine our own body state by noting our own musculature. A common self-report

might be something like, “I was tense, my fists clenched, and I was sweating.”

Visceral responses are fast and completely subconscious. They are sensitive only to the current state of things. Most scientists do not call these emotions: they are precursors to emotion. Stand at the edge of a cliff and you will experience a visceral response. Or bask in the warm, comforting glow after a pleasant experience, perhaps a nice meal.

For designers, the visceral response is about immediate perception: the pleasantness of a mellow, harmonious sound or the jarring, irritating scratch of fingernails on a rough surface. Here is where the style matters: appearances, whether sound or sight, touch or smell, drive the visceral response. This has nothing to do with how usable, effective, or understandable the product is. It is all about attraction or repulsion. Great designers use their aesthetic sensibilities to drive these visceral responses.

Engineers and other logical people tend to dismiss the visceral response as irrelevant. Engineers are proud of the inherent quality of their work and dismayed when inferior products sell better “just because they look better.” But all of us make these kinds of judgments, even those very logical engineers. That’s why they love some of their tools and dislike others. Visceral responses matter.

THE BEHAVIORAL LEVEL

The *behavioral* level is the home of learned skills, triggered by situations that match the appropriate patterns. Actions and analyses at this level are largely subconscious. Even though we are usually aware of our actions, we are often unaware of the details. When we speak, we often do not know what we are about to say until our conscious mind (the reflective part of the mind) hears ourselves uttering the words. When we play a sport, we are prepared for action, but our responses occur far too quickly for conscious control: it is the behavioral level that takes control.

When we perform a well-learned action, all we have to do is think of the goal and the behavioral level handles all the details: the conscious mind has little or no awareness beyond creating the

desire to act. It's actually interesting to keep trying it. Move the left hand, then the right. Stick out your tongue, or open your mouth. What did you do? You don't know. All you know is that you "willed" the action and the correct thing happened. You can even make the actions more complex. Pick up a cup, and then with the same hand, pick up several more items. You automatically adjust the fingers and the hand's orientation to make the task possible. You only need to pay conscious attention if the cup holds some liquid that you wish to avoid spilling. But even in that case, the actual control of the muscles is beneath conscious perception: concentrate on not spilling and the hands automatically adjust.

For designers, the most critical aspect of the behavioral level is that every action is associated with an expectation. Expect a positive outcome and the result is a positive affective response (a "positive valence," in the scientific literature). Expect a negative outcome and the result is a negative affective response (a negative valence): dread and hope, anxiety and anticipation. The information in the feedback loop of evaluation confirms or disconfirms the expectations, resulting in satisfaction or relief, disappointment or frustration.

Behavioral states are learned. They give rise to a feeling of control when there is good understanding and knowledge of results, and frustration and anger when things do not go as planned, and especially when neither the reason nor the possible remedies are known. Feedback provides reassurance, even when it indicates a negative result. A lack of feedback creates a feeling of lack of control, which can be unsettling. Feedback is critical to managing expectations, and good design provides this. Feedback—knowledge of results—is how expectations are resolved and is critical to learning and the development of skilled behavior.

Expectations play an important role in our emotional lives. This is why drivers tense when trying to get through an intersection before the light turns red, or students become highly anxious before an exam. The release of the tension of expectation creates a sense of relief. The emotional system is especially responsive to changes in states—so an upward change is interpreted positively even if it is only from a very bad state to a not-so-bad state, just as a change is

interpreted negatively even if it is from an extremely positive state to one only somewhat less positive.

THE REFLECTIVE LEVEL

The *reflective* level is the home of conscious cognition. As a consequence, this is where deep understanding develops, where reasoning and conscious decision-making take place. The visceral and behavioral levels are subconscious and, as a result, they respond rapidly, but without much analysis. Reflection is cognitive, deep, and slow. It often occurs after the events have happened. It is a reflection or looking back over them, evaluating the circumstances, actions, and outcomes, often assessing blame or responsibility. The highest levels of emotions come from the reflective level, for it is here that causes are assigned and where predictions of the future take place. Adding causal elements to experienced events leads to such emotional states as guilt and pride (when we assume ourselves to be the cause) and blame and praise (when others are thought to be the cause). Most of us have probably experienced the extreme highs and lows of anticipated future events, all imagined by a runaway reflective cognitive system but intense enough to create the physiological responses associated with extreme anger or pleasure. Emotion and cognition are tightly intertwined.

DESIGN MUST TAKE PLACE AT ALL LEVELS: VISCERAL, BEHAVIORAL, AND REFLECTIVE

To the designer, reflection is perhaps the most important of the levels of processing. Reflection is conscious, and the emotions produced at this level are the most protracted: those that assign agency and cause, such as guilt and blame or praise and pride. Reflective responses are part of our memory of events. Memories last far longer than the immediate experience or the period of usage, which are the domains of the visceral and behavioral levels. It is reflection that drives us to recommend a product, to recommend that others use it—or perhaps to avoid it.

Reflective memories are often more important than reality. If we have a strongly positive visceral response but disappointing

usability problems at the behavioral level, when we reflect back upon the product, the reflective level might very well weigh the positive response strongly enough to overlook the severe behavioral difficulties (hence the phrase, “Attractive things work better”). Similarly, too much frustration, especially toward the ending stage of use, and our reflections about the experience might overlook the positive visceral qualities. Advertisers hope that the strong reflective value associated with a well-known, highly prestigious brand might overwhelm our judgment, despite a frustrating experience in using the product. Vacations are often remembered with fondness, despite the evidence from diaries of repeated discomfort and anguish.

All three levels of processing work together. All play essential roles in determining a person’s like or dislike of a product or service. One nasty experience with a service provider can spoil all future experiences. One superb experience can make up for past deficiencies. The behavioral level, which is the home of interaction, is also the home of all expectation-based emotions, of hope and joy, frustration and anger. Understanding arises at a combination of the behavioral and reflective levels. Enjoyment requires all three. Designing at all three levels is so important that I devote an entire book to the topic, *Emotional Design*.

In psychology, there has been a long debate about which happens first: emotion or cognition. Do we run and flee because some event happened that made us afraid? Or are we afraid because our conscious, reflective mind notices that we are running? The three-level analysis shows that both of these ideas can be correct. Sometimes the emotion comes first. An unexpected loud noise can cause automatic visceral and behavioral responses that make us flee. Then, the reflective system observes itself fleeing and deduces that it is afraid. The actions of running and fleeing occur first and set off the interpretation of fear.

But sometimes cognition occurs first. Suppose the street where we are walking leads to a dark and narrow section. Our reflective system might conjure numerous imagined threats that await us. At some point, the imagined depiction of potential harm is large

enough to trigger the behavioral system, causing us to turn, run, and flee. Here is where the cognition sets off the fear and the action.

Most products do not cause fear, running, or fleeing, but badly designed devices can induce frustration and anger, a feeling of helplessness and despair, and possibly even hate. Well-designed devices can induce pride and enjoyment, a feeling of being in control and pleasure—possibly even love and attachment. Amusement parks are experts at balancing the conflicting responses of the emotional stages, providing rides and fun houses that trigger fear responses from the visceral and behavioral levels, while all the time providing reassurance at the reflective level that the park would never subject anyone to real danger.

All three levels of processing work together to determine a person's cognitive and emotional state. High-level reflective cognition can trigger lower-level emotions. Lower-level emotions can trigger higher-level reflective cognition.

The Seven Stages of Action and the Three Levels of Processing

The stages of action can readily be associated with the three different levels of processing, as shown in Figure 2.4. At the lowest level are the visceral levels of calmness or anxiety when approaching a task or evaluating the state of the world. Then, in the middle level, are the behavioral ones driven by expectations on the execution side—for example, hope and fear—and emotions driven by the confirmation of those expectations on the evaluation side—for example, relief or despair. At the highest level are the reflective emotions, ones that assess the results in terms of the presumed causal agents and the consequences, both immediate and long-term. Here is where satisfaction and pride occur, or perhaps blame and anger.

One important emotional state is the one that accompanies complete immersion into an activity, a state that the social scientist Mihaly Csikszentmihalyi has labeled “flow.” Csikszentmihalyi has long studied how people interact with their work and play, and how their lives reflect this intermix of activities. When in the flow state, people lose track of time and the outside environment.

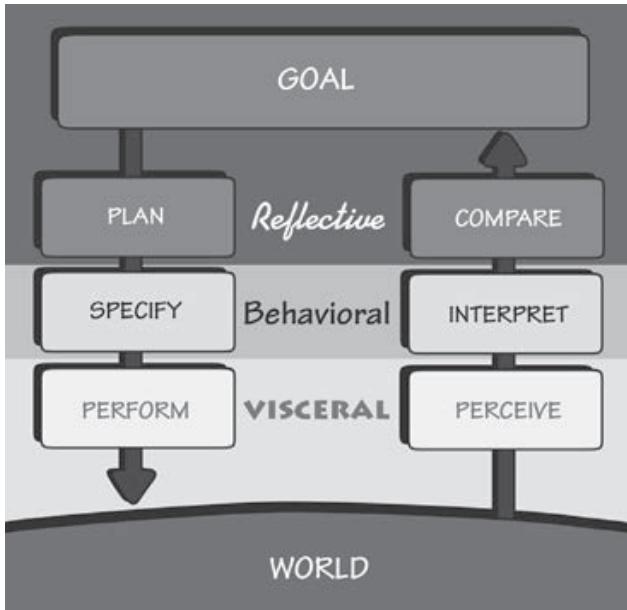


FIGURE 2.4. Levels of Processing and the Stages of the Action Cycle. Visceral response is at the lowest level: the control of simple muscles and sensing the state of the world and body. The behavioral level is about expectations, so it is sensitive to the expectations of the action sequence and then the interpretations of the feedback. The reflective level is a part of the goal- and plan-setting activity as well as affected by the comparison of expectations with what has actually happened.

They are at one with the task they are performing. The task, moreover, is at just the proper level of difficulty: difficult enough to provide a challenge and require continued attention, but not so difficult that it invokes frustration and anxiety.

Csikszentmihalyi's work shows how the behavioral level creates a powerful set of emotional responses. Here, the subconscious expectations established by the execution side of the action cycle set up emotional states dependent upon those expectations. When the results of our actions are evaluated against expectations, the resulting emotions affect our feelings as we continue through

the many cycles of action. An easy task, far below our skill level, makes it so easy to meet expectations that there is no challenge. Very little or no processing effort is required, which leads to apathy or boredom. A difficult task, far above our skill, leads to so many failed expectations that it causes frustration, anxiety, and helplessness. The flow state occurs when the challenge of the activity just slightly exceeds our skill level, so full attention is continually required. Flow requires that the activity be neither too easy nor too difficult relative to our level of skill. The constant tension coupled with continual progress and success can be an engaging, immersive experience sometimes lasting for hours.

People as Storytellers

Now that we have explored the way that actions get done and the three different levels of processing that integrate cognition and emotion, we are ready to look at some of the implications.

People are innately disposed to look for causes of events, to form explanations and stories. That is one reason storytelling is such a persuasive medium. Stories resonate with our experiences and provide examples of new instances. From our experiences and the stories of others we tend to form generalizations about the way people behave and things work. We attribute causes to events, and as long as these cause-and-effect pairings make sense, we accept them and use them for understanding future events. Yet these causal attributions are often erroneous. Sometimes they implicate the wrong causes, and for some things that happen, there is no single cause; rather, a complex chain of events that all contribute to the result: if any one of the events would not have occurred, the result would be different. But even when there is no single causal act, that doesn't stop people from assigning one.

Conceptual models are a form of story, resulting from our predisposition to find explanations. These models are essential in helping us understand our experiences, predict the outcome of our actions, and handle unexpected occurrences. We base our models on whatever knowledge we have, real or imaginary, naive or sophisticated.

Conceptual models are often constructed from fragmentary evidence, with only a poor understanding of what is happening, and with a kind of naive psychology that postulates causes, mechanisms, and relationships even where there are none. Some faulty models lead to the frustrations of everyday life, as in the case of my unseatable refrigerator, where my conceptual model of its operation (see again Figure 1.10A) did not correspond to reality (Figure 1.10B). Far more serious are faulty models of such complex systems as an industrial plant or passenger airplane. Misunderstanding there can lead to devastating accidents.

Consider the thermostat that controls room heating and cooling systems. How does it work? The average thermostat offers almost no evidence of its operation except in a highly roundabout manner. All we know is that if the room is too cold, we set a higher temperature into the thermostat. Eventually we feel warmer. Note that the same thing applies to the temperature control for almost any device whose temperature is to be regulated. Want to bake a

cake? Set the oven thermostat and the oven goes to the desired temperature.

If you are in a cold room, in a hurry to get warm, will the room heat more quickly if you turn the thermostat to its maximum setting? Or if you want the oven to reach its working temperature faster, should you turn the temperature dial all the way to maximum, then turn it down once the desired temperature is reached? Or to cool a room most quickly, should you set the air conditioner thermostat to its lowest temperature setting?

If you think that the room or oven will cool or heat faster if the thermostat is turned all the way to the maximum setting, you are wrong—you hold an erroneous folk theory of the heating and cooling system. One commonly held folk theory of the working of a thermostat is that it is like a valve: the thermostat controls how much heat (or cold) comes out of the device. Hence, to heat or cool something most quickly, set the thermostat so that the device is on maximum. The theory is reasonable, and there exist devices that operate like this, but neither the heating or cooling equipment for a home nor the heating element of a traditional oven is one of them.

In most homes, the thermostat is just an on-off switch. Moreover, most heating and cooling devices are either fully on or fully off: all or nothing, with no in-between states. As a result, the thermostat turns the heater, oven, or air conditioner completely on, at full power, until the temperature setting on the thermostat is reached. Then it turns the unit completely off. Setting the thermostat at one extreme cannot affect how long it takes to reach the desired temperature. Worse, because this bypasses the automatic shutoff when the desired temperature is reached, setting it at the extremes invariably means that the temperature overshoots the target. If people were uncomfortably cold or hot before, they will become uncomfortable in the other direction, wasting considerable energy in the process.

But how are you to know? What information helps you understand how the thermostat works? The design problem with the refrigerator is that there are no aids to understanding, no way of

forming the correct conceptual model. In fact, the information provided misleads people into forming the wrong, quite inappropriate model.

The real point of these examples is not that some people have erroneous beliefs; it is that everyone forms stories (conceptual models) to explain what they have observed. In the absence of external information, people can let their imagination run free as long as the conceptual models they develop account for the facts as they perceive them. As a result, people use their thermostats inappropriately, causing themselves unnecessary effort, and often resulting in large temperature swings, thus wasting energy, which is both a needless expense and bad for the environment. (Later in this chapter, page 69, I provide an example of a thermostat that does provide a useful conceptual model.)

Blaming the Wrong Things

People try to find causes for events. They tend to assign a causal relation whenever two things occur in succession. If some unexpected event happens in my home just after I have taken some action, I am apt to conclude that it was caused by that action, even if there really was no relationship between the two. Similarly, if I do something expecting a result and nothing happens, I am apt to interpret this lack of informative feedback as an indication that I didn't do the action correctly: the most likely thing to do, therefore, is to repeat the action, only with more force. Push a door and it fails to open? Push again, harder. With electronic devices, if the feedback is delayed sufficiently, people often are led to conclude that the press wasn't recorded, so they do the same action again, sometimes repeatedly, unaware that all of their presses were recorded. This can lead to unintended results. Repeated presses might intensify the response much more than was intended. Alternatively, a second request might cancel the previous one, so that an odd number of pushes produces the desired result, whereas an even number leads to no result.

The tendency to repeat an action when the first attempt fails can be disastrous. This has led to numerous deaths when people

tried to escape a burning building by attempting to push open exit doors that opened inward, doors that should have been pulled. As a result, in many countries, the law requires doors in public places to open outward, and moreover to be operated by so-called panic bars, so that they automatically open when people, in a panic to escape a fire, push their bodies against them. This is a great application of appropriate affordances: see the door in Figure 2.5.

Modern systems try hard to provide feedback within 0.1 second of any operation, to reassure the user that the request was received. This is especially important if the operation will take considerable time. The presence of a filling hourglass or rotating clock hands is a reassuring sign that work is in progress. When the delay can be predicted, some systems provide time estimates as well as progress bars to indicate how far along the task has gone. More systems should adopt these sensible displays to provide timely and meaningful feedback of results.

Some studies show it is wise to underpredict—that is, to say an operation will take longer than it actually will. When the system computes the amount of time, it can compute the range of possible



FIGURE 2.5. Panic Bars on Doors. People fleeing a fire would die if they encountered exit doors that opened inward, because they would keep trying to push them outward, and when that failed, they would push harder. The proper design, now required by law in many places, is to change the design of doors so that they open when pushed. Here is one example: an excellent design strategy for dealing with real behavior by the use of the proper affordances coupled with a graceful signifier, the black bar, which indicates where to push. (Photograph by author at the Ford Design Center, Northwestern University.)

times. In that case it ought to display the range, or if only a single value is desirable, show the slowest, longest value. That way, the expectations are liable to be exceeded, leading to a happy result.

When it is difficult to determine the cause of a difficulty, where do people put the blame? Often people will use their own conceptual models of the world to determine the perceived causal relationship between the thing being blamed and the result. The word *perceived* is critical: the causal relationship does not have to exist; the person simply has to think it is there. Sometimes the result is to attribute cause to things that had nothing to do with the action.

Suppose I try to use an everyday thing, but I can't. Who is at fault: me or the thing? We are apt to blame ourselves, especially if others are able to use it. Suppose the fault really lies in the device, so that lots of people have the same problems. Because everyone perceives the fault to be his or her own, nobody wants to admit to having trouble. This creates a conspiracy of silence, where the feelings of guilt and helplessness among people are kept hidden.

Interestingly enough, the common tendency to blame ourselves for failures with everyday objects goes against the normal attributions we make about ourselves and others. Everyone sometimes acts in a way that seems strange, bizarre, or simply wrong and inappropriate. When we do this, we tend to attribute our behavior to the environment. When we see others do it, we tend to attribute it to their personalities.

Here is a made-up example. Consider Tom, the office terror. Today, Tom got to work late, yelled at his colleagues because the office coffee machine was empty, then ran to his office and slammed the door shut. “Ah,” his colleagues and staff say to one another, “there he goes again.”

Now consider Tom’s point of view. “I really had a hard day,” Tom explains. “I woke up late because my alarm clock failed to go off: I didn’t even have time for my morning coffee. Then I couldn’t find a parking spot because I was late. And there wasn’t any coffee in the office machine; it was all out. None of this was my fault—I had a run of really bad events. Yes, I was a bit curt, but who wouldn’t be under the same circumstances?”

Tom's colleagues don't have access to his inner thoughts or to his morning's activities. All they see is that Tom yelled at them simply because the office coffee machine was empty. This reminds them of another similar event. "He does that all the time," they conclude, "always blowing up over the most minor things." Who is correct? Tom or his colleagues? The events can be seen from two different points of view with two different interpretations: common responses to the trials of life or the result of an explosive, irascible personality.

It seems natural for people to blame their own misfortunes on the environment. It seems equally natural to blame other people's misfortunes on their personalities. Just the opposite attribution, by the way, is made when things go well. When things go right, people credit their own abilities and intelligence. The onlookers do the reverse. When they see things go well for someone else, they sometimes credit the environment, or luck.

In all such cases, whether a person is inappropriately accepting blame for the inability to work simple objects or attributing behavior to environment or personality, a faulty conceptual model is at work.

LEARNED HELPLESSNESS

The phenomenon called *learned helplessness* might help explain the self-blame. It refers to the situation in which people experience repeated failure at a task. As a result, they decide that the task cannot be done, at least not by them: they are helpless. They stop trying. If this feeling covers a group of tasks, the result can be severe difficulties coping with life. In the extreme case, such learned helplessness leads to depression and to a belief that the individuals cannot cope with everyday life at all. Sometimes all it takes to get such a feeling of helplessness are a few experiences that accidentally turn out bad. The phenomenon has been most frequently studied as a precursor to the clinical problem of depression, but I have seen it happen after a few bad experiences with everyday objects.

Do common technology and mathematics phobias result from a kind of learned helplessness? Could a few instances of failure

in what appear to be straightforward situations generalize to every technological object, every mathematics problem? Perhaps. In fact, the design of everyday things (and the design of mathematics courses) seems almost guaranteed to cause this. We could call this phenomenon taught helplessness.

When people have trouble using technology, especially when they perceive (usually incorrectly) that nobody else is having the same problems, they tend to blame themselves. Worse, the more they have trouble, the more helpless they may feel, believing that they must be technically or mechanically inept. This is just the opposite of the more normal situation where people blame their own difficulties on the environment. This false blame is especially ironic because the culprit here is usually the poor design of the technology, so blaming the environment (the technology) would be completely appropriate.

Consider the normal mathematics curriculum, which continues relentlessly on its way, each new lesson assuming full knowledge and understanding of all that has passed before. Even though each point may be simple, once you fall behind it is hard to catch up. The result: mathematics phobia—not because the material is difficult, but because it is taught so that difficulty in one stage hinders further progress. The problem is that once failure starts, it is soon generalized by self-blame to all of mathematics. Similar processes are at work with technology. The vicious cycle starts: if you fail at something, you think it is your fault. Therefore you think you can't do that task. As a result, next time you have to do the task, you believe you can't, so you don't even try. The result is that you can't, just as you thought.

You're trapped in a self-fulfilling prophecy.

POSITIVE PSYCHOLOGY

Just as we learn to give up after repeated failure, we can learn optimistic, positive responses to life. For years, psychologists focused upon the gloomy story of how people failed, on the limits of human abilities, and on psychopathologies—depression, mania, paranoia, and so on. But the twenty-first century sees a new approach:

to focus upon a positive psychology, a culture of positive thinking, of feeling good about oneself. In fact, the normal emotional state of most people is positive. When something doesn't work, it can be considered an interesting challenge, or perhaps just a positive learning experience.

We need to remove the word *failure* from our vocabulary, replacing it instead with *learning experience*. To fail is to learn: we learn more from our failures than from our successes. With success, sure, we are pleased, but we often have no idea why we succeeded. With failure, it is often possible to figure out why, to ensure that it will never happen again.

Scientists know this. Scientists do experiments to learn how the world works. Sometimes their experiments work as expected, but often they don't. Are these failures? No, they are learning experiences. Many of the most important scientific discoveries have come from these so-called failures.

Failure can be such a powerful learning tool that many designers take pride in their failures that happen while a product is still in development. One design firm, IDEO, has it as a creed: "Fail often, fail fast," they say, for they know that each failure teaches them a lot about what to do right. Designers need to fail, as do researchers. I have long held the belief—and encouraged it in my students and employees—that failures are an essential part of exploration and creativity. If designers and researchers do not sometimes fail, it is a sign that they are not trying hard enough—they are not thinking the great creative thoughts that will provide breakthroughs in how we do things. It is possible to avoid failure, to always be safe. But that is also the route to a dull, uninteresting life.

The designs of our products and services must also follow this philosophy. So, to the designers who are reading this, let me give some advice:

- Do not blame people when they fail to use your products properly.
- Take people's difficulties as signifiers of where the product can be improved.

- Eliminate all error messages from electronic or computer systems. Instead, provide help and guidance.
- Make it possible to correct problems directly from help and guidance messages. Allow people to continue with their task: Don't impede progress—help make it smooth and continuous. Never make people start over.
- Assume that what people have done is partially correct, so if it is inappropriate, provide the guidance that allows them to correct the problem and be on their way.
- Think positively, for yourself and for the people you interact with.

Falsely Blaming Yourself

I have studied people making errors—sometimes serious ones—with mechanical devices, light switches and fuses, computer operating systems and word processors, even airplanes and nuclear power plants. Invariably people feel guilty and either try to hide the error or blame themselves for “stupidity” or “clumsiness.” I often have difficulty getting permission to watch: nobody likes to be observed performing badly. I point out that the design is faulty and that others make the same errors, yet if the task appears simple or trivial, people still blame themselves. It is almost as if they take perverse pride in thinking of themselves as mechanically incompetent.

I once was asked by a large computer company to evaluate a brand-new product. I spent a day learning to use it and trying it out on various problems. In using the keyboard to enter data, it was necessary to differentiate between the Return key and the Enter key. If the wrong key was pressed, the last few minutes' work was irrevocably lost.

I pointed out this problem to the designer, explaining that I, myself, had made the error frequently and that my analyses indicated that this was very likely to be a frequent error among users. The designer's first response was: “Why did you make that error? Didn't you read the manual?” He proceeded to explain the different functions of the two keys.

"Yes, yes," I explained, "I understand the two keys, I simply confuse them. They have similar functions, are located in similar locations on the keyboard, and as a skilled typist, I often hit Return automatically, without thought. Certainly others have had similar problems."

"Nope," said the designer. He claimed that I was the only person who had ever complained, and the company's employees had been using the system for many months. I was skeptical, so we went together to some of the employees and asked them whether they had ever hit the Return key when they should have hit Enter. And did they ever lose their work as a result?

"Oh, yes," they said, "we do that a lot."

Well, how come nobody ever said anything about it? After all, they were encouraged to report all problems with the system. The reason was simple: when the system stopped working or did something strange, they dutifully reported it as a problem. But when they made the Return versus Enter error, they blamed themselves. After all, they had been told what to do. They had simply erred.

The idea that a person is at fault when something goes wrong is deeply entrenched in society. That's why we blame others and even ourselves. Unfortunately, the idea that a person is at fault is imbedded in the legal system. When major accidents occur, official courts of inquiry are set up to assess the blame. More and more often the blame is attributed to "human error." The person involved can be fined, punished, or fired. Maybe training procedures are revised. The law rests comfortably. But in my experience, human error usually is a result of poor design: it should be called system error. Humans err continually; it is an intrinsic part of our nature. System design should take this into account. Pinning the blame on the person may be a comfortable way to proceed, but why was the system ever designed so that a single act by a single person could cause calamity? Worse, blaming the person without fixing the root, underlying cause does not fix the problem: the same error is likely to be repeated by someone else. I return to the topic of human error in Chapter 5.

Of course, people do make errors. Complex devices will always require some instruction, and someone using them without instruction should expect to make errors and to be confused. But

designers should take special pains to make errors as cost-free as possible. Here is my credo about errors:

Eliminate the term *human error*. Instead, talk about communication and interaction: what we call an error is usually bad communication or interaction. When people collaborate with one another, the word error is never used to characterize another person's utterance. That's because each person is trying to understand and respond to the other, and when something is not understood or seems inappropriate, it is questioned, clarified, and the collaboration continues. Why can't the interaction between a person and a machine be thought of as collaboration?

Machines are not people. They can't communicate and understand the same way we do. This means that their designers have a special obligation to ensure that the behavior of machines is understandable to the people who interact with them. True collaboration requires each party to make some effort to accommodate and understand the other. When we collaborate with machines, it is people who must do all the accommodation. Why shouldn't the machine be more friendly? The machine should accept normal human behavior, but just as people often subconsciously assess the accuracy of things being said, machines should judge the quality of information given it, in this case to help its operators avoid grievous errors because of simple slips (discussed in Chapter 5). Today, we insist that people perform abnormally, to adapt themselves to the peculiar demands of machines, which includes always giving precise, accurate information. Humans are particularly bad at this, yet when they fail to meet the arbitrary, inhuman requirements of machines, we call it human error. No, it is design error.

Designers should strive to minimize the chance of inappropriate actions in the first place by using affordances, signifiers, good mapping, and constraints to guide the actions. If a person performs an inappropriate action, the design should maximize the chance that this can be discovered and then rectified. This requires good, intelligible feedback coupled with a simple, clear conceptual model. When people understand what has happened, what state the system is in, and what the most appropriate set of actions is, they can perform their activities more effectively.

People are not machines. Machines don't have to deal with continual interruptions. People are subjected to continual interruptions. As a result, we are often bouncing back and forth between tasks, having to recover our place, what we were doing, and what we were thinking when we return to a previous task. No wonder we sometimes forget our place when we return to the original task, either skipping or repeating a step, or imprecisely retaining the information we were about to enter.

Our strengths are in our flexibility and creativity, in coming up with solutions to novel problems. We are creative and imaginative, not mechanical and precise. Machines require precision and accuracy; people don't. And we are particularly bad at providing precise and accurate inputs. So why are we always required to do so? Why do we put the requirements of machines above those of people?

When people interact with machines, things will not always go smoothly. This is to be expected. So designers should anticipate this. It is easy to design devices that work well when everything goes as planned. The hard and necessary part of design is to make things work well even when things do not go as planned.

HOW TECHNOLOGY CAN ACCOMMODATE HUMAN BEHAVIOR

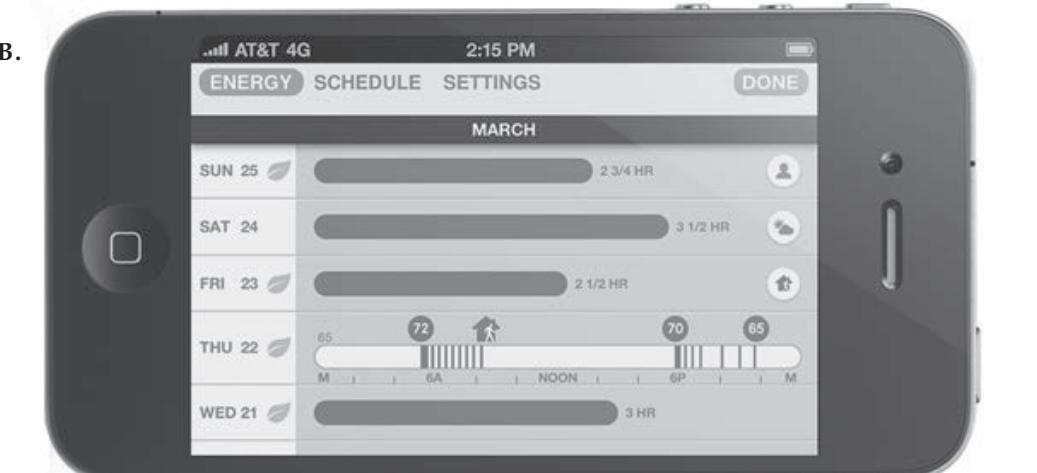
In the past, cost prevented many manufacturers from providing useful feedback that would assist people in forming accurate conceptual models. The cost of color displays large and flexible enough to provide the required information was prohibitive for small, inexpensive devices. But as the cost of sensors and displays has dropped, it is now possible to do a lot more.

Thanks to display screens, telephones are much easier to use than ever before, so my extensive criticisms of phones found in the earlier edition of this book have been removed. I look forward to great improvements in all our devices now that the importance of these design principles are becoming recognized and the enhanced quality and lower costs of displays make it possible to implement the ideas.

PROVIDING A CONCEPTUAL MODEL FOR A HOME THERMOSTAT

My thermostat, for example (designed by Nest Labs), has a colorful display that is normally off, turning on only when it senses that I

FIGURE 2.6. A Thermostat with an Explicit Conceptual Model. This thermostat, manufactured by Nest Labs, helps people form a good conceptual model of its operation. Photo A shows the thermostat. The background, blue, indicates that it is now cooling the home. The current temperature is 75°F (24°C) and the target temperature is 72°F (22°C), which it expects to reach in 20 minutes. Photo B shows its use of a smart phone to deliver a summary of its settings and the home's energy use. Both A and B combine to help the home dweller develop conceptual models of the thermostat and the home's energy consumption. (Photographs courtesy of Nest Labs, Inc.)



am nearby. Then it provides me with the current temperature of the room, the temperature to which it is set, and whether it is heating or cooling the room (the background color changes from black when it is neither heating nor cooling, to orange while heating, or to blue while cooling). It learns my daily patterns, so it changes temperature automatically, lowering it at bedtime, raising it again in the morning, and going into “away” mode when it detects that nobody is in the house. All the time, it explains what it is doing. Thus, when it has to change the room temperature substantially (either because someone has entered a manual change or because it has decided that it is now time to switch), it gives a prediction: “Now 75°, will be 72° in 20 minutes.” In addition, Nest can be connected wirelessly to smart devices that allow for remote operation of the thermostat and also for larger screens to provide a detailed analysis of its performance, aiding the home occupant’s development of a conceptual model both of Nest and also of the home’s energy consumption. Is Nest perfect? No, but it marks improvement in the collaborative interaction of people and everyday things.

ENTERING DATES, TIMES, AND TELEPHONE NUMBERS

Many machines are programmed to be very fussy about the form of input they require, where the fussiness is not a requirement of the machine but due to the lack of consideration for people in the design of the software. In other words: inappropriate programming. Consider these examples.

Many of us spend hours filling out forms on computers—forms that require names, dates, addresses, telephone numbers, monetary sums, and other information in a fixed, rigid format. Worse, often we are not even told the correct format until we get it wrong. Why not figure out the variety of ways a person might fill out a form and accommodate all of them? Some companies have done excellent jobs at this, so let us celebrate their actions.

Consider Microsoft's calendar program. Here, it is possible to specify dates any way you like: "November 23, 2015," "23 Nov. 15," or "11.23.15." It even accepts phrases such as "a week from Thursday," "tomorrow," "a week from tomorrow," or "yesterday." Same with time. You can enter the time any way you want: "3:45 PM," "15.35," "an hour," "two and one-half hours." Same with telephone numbers: Want to start with a + sign (to indicate the code for international dialing)? No problem. Like to separate the number fields with spaces, dashes, parentheses, slashes, periods? No problem. As long as the program can decipher the date, time, or telephone number into a legal format, it is accepted. I hope the team that worked on this got bonuses and promotions.

Although I single out Microsoft for being the pioneer in accepting a wide variety of formats, it is now becoming standard practice. By the time you read this, I would hope that every program would permit any intelligible format for names, dates, phone numbers, street addresses, and so on, transforming whatever is entered into whatever form the internal programming needs. But I predict that even in the twenty-second century, there will still be forms that require precise accurate (but arbitrary) formats for no reason except the laziness of the programming team. Perhaps in the years that pass between this book's publication and when you are read-

ing this, great improvements will have been made. If we are all lucky, this section will be badly out of date. I hope so.

The Seven Stages of Action: Seven Fundamental Design Principles

The seven-stage model of the action cycle can be a valuable design tool, for it provides a basic checklist of questions to ask. In general, each stage of action requires its own special design strategies and, in turn, provides its own opportunity for disaster. Figure 2.7 summarizes the questions:

1. What do I want to accomplish?
2. What are the alternative action sequences?
3. What action can I do now?
4. How do I do it?
5. What happened?
6. What does it mean?
7. Is this okay? Have I accomplished my goal?

Anyone using a product should always be able to determine the answers to all seven questions. This puts the burden on the designer



FIGURE 2.7. The Seven Stages of Action as Design Aids. Each of the seven stages indicates a place where the person using the system has a question. The seven questions pose seven design themes. How should the design convey the information required to answer the user's question? Through appropriate constraint and mappings, signifiers and conceptual models, feedback and visibility. The information that helps answer questions of execution (doing) is *feedforward*. The information that aids in understanding what has happened is *feedback*.

to ensure that at each stage, the product provides the information required to answer the question.

The information that helps answer questions of execution (doing) is *feedforward*. The information that aids in understanding what has happened is *feedback*. Everyone knows what feedback is. It helps you know what happened. But how do you know what you can do? That's the role of feedforward, a term borrowed from control theory.

Feedforward is accomplished through appropriate use of signifiers, constraints, and mappings. The conceptual model plays an important role. Feedback is accomplished through explicit information about the impact of the action. Once again, the conceptual model plays an important role.

Both feedback and feedforward need to be presented in a form that is readily interpreted by the people using the system. The presentation has to match how people view the goal they are trying to achieve and their expectations. Information must match human needs.

The insights from the seven stages of action lead us to seven fundamental principles of design:

1. **Discoverability.** It is possible to determine what actions are possible and the current state of the device.
2. **Feedback.** There is full and continuous information about the results of actions and the current state of the product or service. After an action has been executed, it is easy to determine the new state.
3. **Conceptual model.** The design projects all the information needed to create a good conceptual model of the system, leading to understanding and a feeling of control. The conceptual model enhances both discoverability and evaluation of results.
4. **Affordances.** The proper affordances exist to make the desired actions possible.
5. **Signifiers.** Effective use of signifiers ensures discoverability and that the feedback is well communicated and intelligible.
6. **Mappings.** The relationship between controls and their actions follows the principles of good mapping, enhanced as much as possible through spatial layout and temporal contiguity.

7. **Constraints.** Providing physical, logical, semantic, and cultural constraints guides actions and eases interpretation.

The next time you can't immediately figure out the shower control in a hotel room or have trouble using an unfamiliar television set or kitchen appliance, remember that the problem is in the design. Ask yourself where the problem lies. At which of the seven stages of action does it fail? Which design principles are deficient?

But it is easy to find fault: the key is to be able to do things better. Ask yourself how the difficulty came about. Realize that many different groups of people might have been involved, each of which might have had intelligent, sensible reasons for their actions. For example, a troublesome bathroom shower was designed by people who were unable to know how it would be installed, then the shower controls might have been selected by a building contractor to fit the home plans provided by yet another person. Finally, a plumber, who may not have had contact with any of the other people, did the installation. Where did the problems arise? It could have been at any one (or several) of these stages. The result may appear to be poor design, but it may actually arise from poor communication.

One of my self-imposed rules is, "Don't criticize unless you can do better." Try to understand how the faulty design might have occurred: try to determine how it could have been done otherwise. Thinking about the causes and possible fixes to bad design should make you better appreciate good design. So, the next time you come across a well-designed object, one that you can use smoothly and effortlessly on the first try, stop and examine it. Consider how well it masters the seven stages of action and the principles of design. Recognize that most of our interactions with products are actually interactions with a complex system: good design requires consideration of the entire system to ensure that the requirements, intentions, and desires at each stage are faithfully understood and respected at all the other stages.

Direct Manipulation Interfaces

**Edwin L. Hutchins, James D. Hollan, and
Donald A. Norman**

University of California, San Diego

ABSTRACT

Direct manipulation has been lauded as a good form of interface design, and some interfaces that have this property have been well received by users. In this article we seek a cognitive account of both the advantages and disadvantages of direct manipulation interfaces. We identify two underlying phenomena that give rise to the feeling of directness. One deals with the information processing distance between the user's intentions and the facilities provided by the machine. Reduction of this distance makes the interface feel direct by reducing the effort required of the user to accomplish goals. The second phenomenon concerns the relation between the input and output vocabularies of the interface language. In particular, direct manipulation requires that the system provide representations of objects that behave as if they are the objects themselves. This provides the feeling of directness of manipulation.

A version of this paper also appears as a chapter in the book, *User Centered System Design: New Perspectives on Human-Computer Interaction* (Norman & Draper, 1986).

Authors' present address: Edwin L. Hutchins, James D. Hollan, and Donald A. Norman, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA 92093.

CONTENTS

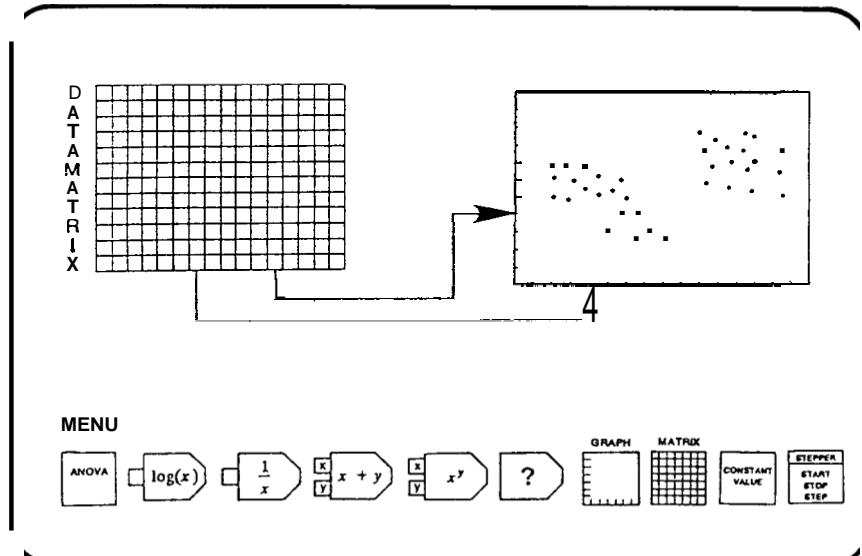
- 1. DIRECT MANIPULATION**
 - 1.1. Early Examples of Direct Manipulation**
 - 1.2. The Goal: A Cognitive Account of Direct Manipulation**
 - 2. TWO ASPECTS OF DIRECTNESS: DISTANCE AND ENGAGEMENT**
 - 2.1. Distance**
 - 2.2. Direct Engagement**
 - 3. TWO FORMS OF DISTANCE: SEMANTIC AND ARTICULATORY**
 - 3.1. Semantic Distance**
 - Semantic Distance in the Gulfs of Execution and Evaluation**
 - The Gulf of Execution
 - The Gulf of Evaluation
 - Reducing the Semantic Distance That Must Be Spanned**
 - Higher-Level Languages
 - Make the Output Show Semantic Concepts Directly
 - Automated Behavior Does Not Reduce Semantic Distance
 - The User Can Adapt to the System Representation
 - Virtuosity and Semantic Distance
 - 3.4. Articulatory Distance**
 - 3.5. Articulatory Distance in the Gulfs of Execution and Evaluation**
 - 4. DIRECT ENGAGEMENT**
 - 5. A SPACE OF INTERFACES**
 - 6. PROBLEMS WITH DIRECT MANIPULATION**
-

1. DIRECT MANIPULATION

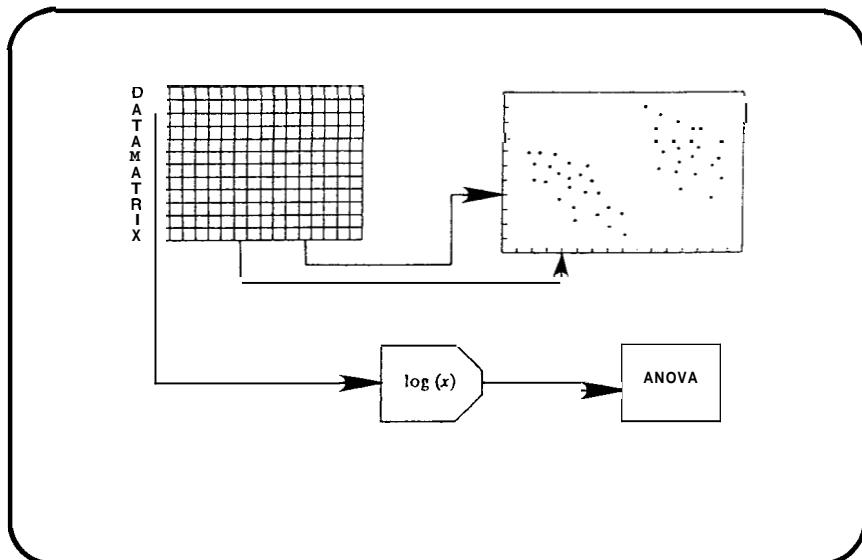
The best way to describe a direct manipulation interface is by example. Suppose we have a set of data to be analyzed with the numbers stored in matrix form. Their source and meaning are not important for this example. The numbers could be the output of a spreadsheet, a matrix of numerical values from the computations of a conventional programming language, or the results of an experiment. Our goal is to analyze the numbers, to see what relations exist among the rows and columns of the matrix. The matrix of numbers is represented on a computer display screen by an icon. To plot one column against another, simply get a copy of a graph icon, then draw a line from the output of one column to the x-axis input of the graph icon and another line from the output of the second column to the y-axis input (see Figure 1). Not what was wanted? Erase the lines and reconnect them. Want to see other graphs? Make more copies of the graph icons and connect them. Need a logarithmic transformation of one of the axes? Move up a function icon, type in the algebraic function that is desired ($y = \log x$, in this case) and connect it in the desired data stream. Want the analysis of variance of the logarithm of the data? Connect the matrix to the appropriate statistical icons. These examples are illustrated in Figure 1B.

Figure 1. An elementary example of doing simple statistical computations by direct manipulation. (A) The basic components: The data are contained in the matrix, represented by the icon in the upper left corner of the screen. At the bottom of the screen are basic icons that represent possible functions. To use one, a copy of the desired icon is moved to the screen and connected up, as is shown for the graph. (B) More complex interconnections, including the use of a logarithmic transformation of the data, a basic statistical package (for means and standard deviations), and an Analysis of Variance Package (ANOVA).

A



B



Now consider how we could partition the data. Suppose one result of our analysis was the scatter diagram shown in Figure 2. The straight line that has been fitted through the points is clearly inappropriate. The data fall into two quite different clusters and it would best to analyze each cluster separately. In the actual data matrix, the points that form the two clusters might be scattered randomly throughout the data set. The regularities are apparent only when we plot them. How do we pull out the clusters? Suppose we could simply circle the points of interest in the scatter plot and use each circled set as if it were a new matrix of values, each of which could be analyzed in standard ways, as shown in Figure 2B.

The examples of Figures 1 and 2 illustrate a powerful manipulation medium for computation. The promise of direct manipulation is that instead of an abstract computational medium, all the "programming" is done graphically, in a form that matches the way one thinks about the problem. The desired operations are performed simply by moving the appropriate icons onto the screen and connecting them together. Connecting the icons is the equivalent of writing a program or calling on a set of statistical subroutines, but with the advantage of being able to directly manipulate and interact with the data and the connections. There are no hidden operations, no syntax or command names to learn. What you see is what you get. Some classes of syntax errors are eliminated. For example, you can't point at a nonexistent object. The system requires expertise in the task domain, but only minimal knowledge of the computer or of computing.

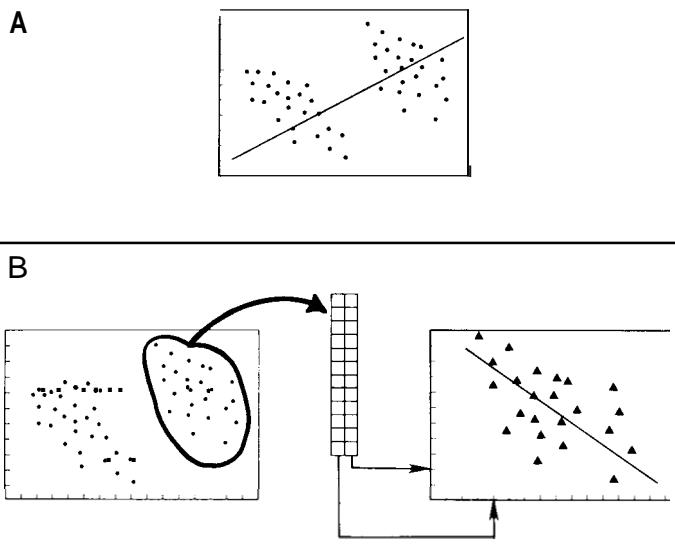
The term *direct manipulation* was coined by Shneiderman (1974, 1982, 1983) to refer to systems having the following properties:

1. Continuous representation of the object of interest.
2. Physical actions or labeled button presses instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible. (Shneiderman, 1982, p. 251)

Direct manipulation interfaces seem remarkably powerful. Shneiderman (1982) has suggested that direct manipulation systems have the following virtues:

1. Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
2. Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.
3. Knowledgeable intermittent users can retain operational concepts.
4. Error messages are rarely needed.
5. Users can see immediately if their actions are furthering their goals, and if not, they can simply change the direction of their activity.

Figure 2. (A) The scatter plot formed in Figure 1, along with the best fitting regression line to the data. It is clear that the data really fall into two quite distinct clusters and that it would be best to look at each independently. (B) The clusters are analyzed by circling the desired data, then treating the group of circled data as if they were a new matrix of values, which can be treated as a data source and analyzed in standard ways.



6. Users have reduced anxiety because the system is comprehensible and because actions are so easily reversible. (Shneiderman, 1982, p. 251)

Can this really be true? Certainly there must be problems as well as benefits. It turns out that the concept of direct manipulation is complex. Moreover, although there are important benefits there are also costs. Like everything else, direct manipulation systems trade off one set of virtues and vices against another. It is important that we understand these trade-offs. A checklist of surface features is unlikely to capture the real sources of power in direct manipulation interfaces.

1.1. Early Examples of Direct Manipulation

Hints of direct manipulation programming environments have been around for quite some time. The first major landmark is Sutherland's *Sketchpad*, a graphical design program (Sutherland, 1963). Sutherland's goal was to devise a program that would make it possible for a person and a computer "to converse rapidly through the medium of line drawings." Sutherland's work is a land-

mark not only because of historical priority but because of the ideas that he helped develop: He was one of the first to discuss the power of graphical interfaces, the conception of a display as "sheets of paper," the use of pointing devices, the virtues of constraint representations, and the importance of depicting abstractions graphically.

Sutherland's ideas took 20 years to have widespread impact. The lag is perhaps due more to hardware limitations than anything else. Highly interactive, graphical programming requires the ready availability of considerable computational power, and it is only recently that machines capable of supporting this type of computational environment have become inexpensive enough to be generally available. Now we see these ideas in many of the computer-aided design and manufacturing systems, many of which can trace their heritage directly to Sutherland's work. Borning's *ThingLab* program (1979) explored a general programming environment, building upon many of Sutherland's ideas within the Smalltalk programming environment. More recently direct manipulation systems have been appearing with reasonable frequency. For example, Bill Budge's *Pinball Construction Set* (Budge, 1983) permits a user to construct an infinite variety of electronic pinball games by directly manipulating graphical objects that represent the components of the game surface. Other examples exist in the area of intelligent training systems (e.g., the Steamer system of Hollan, Hutchins, & Weitzman, 1984; Hollan, Stevens, & Williams, 1980). Steamer makes use of similar techniques and also provides tools for the construction of interactive graphical interfaces. Finally, spreadsheet programs incorporate many of the essential features of direct manipulation. In the lead article of *Scientific American's* special issue on computer software, Kay (1984) claims that the development of dynamic spreadsheet systems gives strong hints that programming styles are in the offing that will make programming as it has been done for the past 40 years — that is, by composing text that represents instructions — obsolete.

1.2. The Goal: A Cognitive Account of Direct Manipulation

We see promise in the notion of direct manipulation, but as yet we see no explanation of it. There are systems with attractive features, and claims for the benefits of systems that give the user a certain sort of feeling, and even lists of properties that seem to be shared by systems that provide that feeling, but no account of how particular properties might produce the feeling of directness. The purpose of this article is to examine the underlying basis for direct manipulation systems. On the one hand, what is it that provides the feeling of "directness?" Why do direct manipulation systems feel so natural? What is so compelling about the notion? On the other hand, why can using such systems sometimes seem so tedious?

For us, the notion of "direct manipulation" is not a unitary concept, nor even something that can be quantified in itself. It is an orienting notion. "Directness" is an impression or a feeling about an interface. What we seek to do here is to characterize the space of interfaces and see where within that picture the range of phenomena that contribute to the feeling of directness might reside. The goal is to give cognitive accounts of these phenomena. At the root of our approach is the assumption that the feeling of directness results from the commitment of fewer cognitive resources. Or, put the other way around, the need to commit additional cognitive resources in the use of an interface leads to the feeling of indirectness. As we shall see, some of the production of the feeling of directness is due to adaptation by the user, so that the designer can neither completely control the process, nor take full credit for the feeling of directness that may be experienced by the user.

We will not attempt to set down hard and fast criteria under which an interface can be classified as direct or not direct. The sensation of directness is always relative; it is often due to the interaction of a number of factors. There are costs associated with every factor that increases the sensation of directness. At present we know of no way to measure the trade-off values, but we will attempt to provide a framework within which one can say what is being traded off against what.

2. TWO ASPECTS OF DIRECTNESS: DISTANCE AND ENGAGEMENT

There are two distinct aspects of the feeling of directness. One involves a notion of the distance between one's thoughts and the physical requirements of the system under use. A short distance means that the translation is simple and straightforward, that thoughts are readily translated into the physical actions required by the system and that the system output is in a form readily interpreted in terms of the goals of interest to the user. We will use the term *directness* to refer to the feeling that results from interaction with an interface. The term *distance* will be used to describe factors which underlie the generation of the feeling of directness.

The second aspect of directness concerns the qualitative feeling of engagement, the feeling that one is directly manipulating the objects of interest. There are two major metaphors for the nature of human-computer interaction, a conversation metaphor and a model-world metaphor. In a system built on the conversation metaphor, the interface is a language medium in which the user and system have a conversation about an assumed, but not explicitly represented world. In this case, the interface is an implied intermediary between the user and the world about which things are said. In a system built on the model-world metaphor, the interface is itself a world where the user can act,

and which changes state in response to user actions. The world of interest is explicitly represented and there is no intermediary between user and world. Appropriate use of the model-world metaphor can create the sensation in the user of acting upon the objects of the task domain themselves. We call this aspect of directness direct engagement.

2.1. Distance

We call one underlying aspect of directness distance to emphasize the fact that directness is never a property of the interface alone, but involves a relationship between the task the user has in mind and the way that task can be accomplished via the interface. Here the critical issues involve minimizing the effort required to bridge the gulf between the user's goals and the way they must be specified to the system.

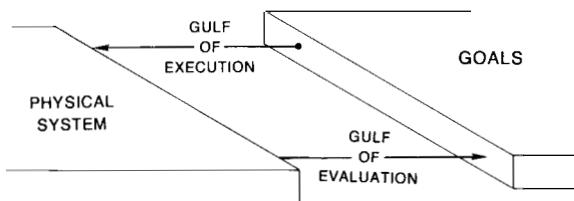
An interface introduces distance to the extent there are gulfs between a person's goals and knowledge and the level of description provided by the systems with which the person must deal. These are referred to as the *gulf of execution* and the *gulf of evaluation* (Figure 3). The gulf of execution is bridged by making the commands and mechanisms of the system match the thoughts and goals of the user. The gulf of evaluation is bridged by making the output displays present a good conceptual model of the system that is readily perceived, interpreted, and evaluated. The goal in both cases is to minimize cognitive effort.

We suggest that the feeling of directness is inversely proportional to the amount of cognitive effort it takes to manipulate and evaluate a system and, moreover, that cognitive effort is a direct result of the gulfs of execution and evaluation. The better the interface to a system helps bridge the gulfs, the less cognitive effort needed and the more direct the resulting feeling of interaction.

2.2. Direct Engagement

The description of the nature of interaction to this point begins to suggest how to make a system less difficult to use, but it misses an important point, a point that is the essence of direct manipulation. The analysis of the execution and evaluation process explains why there is difficulty in using a system, and it says something about what must be done to minimize the mental effort required to use a system. But there is more to it than that. The systems that best exemplify direct manipulation all give the qualitative feeling that one is directly engaged with control of the objects—not with the programs, not with the computer, but with the semantic objects of our goals and intentions. This is the feeling that Laurel (1986) discusses: a feeling of first-personness, of direct engagement with the objects that concern us. Are we analyzing data? Then we should be manipulating the data themselves; or if we are designing an analysis of data, we should be manipulating the analytic structures themselves. Are we

Figure 3. The gulfs of execution and evaluation. Each gulf is unidirectional: The gulf of execution goes from goals to system state; the gulf of evaluation goes from system state to goals.



playing a game? Then we should be manipulating directly the game world, touching and controlling the objects in that world, with the output of the system responding directly to our actions, and in a form compatible with them.

Historically, most interfaces have been built on the conversation metaphor. There is power in the abstractions that language provides (we discuss some of this later), but the implicit role of interface as an intermediary to a hidden world denies the user direct engagement with the objects of interest. Instead, the user is in direct contact with linguistic structures, structures that can be interpreted as referring to the objects of interest, but that are not those objects themselves. Making the central metaphor of the interface that of the model world supports the feeling of directness. Instead of describing the actions of interest, the user performs those actions. In a conventional interface, the system describes the results of the actions. In a model world the system directly presents the actions taken upon the objects. This change in central metaphor is made possible by relatively recent advances in technology. One of the exciting prospects for the study of direct manipulation is the exploration of the properties of systems that provide for direct engagement.

Building interfaces based on the model-world metaphor requires a special sort of relationship between the input interface language and the output interface language. In particular, the output language must represent its subject of discourse in a way that natural language does not normally do. The expressions of a direct manipulation output language must behave in such a way that the user can assume that they, in some sense, *are* the things they refer to. DiSessa (1985) calls this “naïve realism.” Furthermore, the nature of the relationship between input and output language must be such that an output expression can serve as a component of an input expression. Draper (1986) has coined the term *inter-referential I/O* to refer to relationships between input and output in which an expression in one can refer to an expression in the other. When these conditions are met, it is as if we are directly manipulating the things that the system represents.

Thus, consider a system in which a file is represented by an image on the screen and actions are done by pointing to and manipulating the image. In this

case, if we can specify a file by pointing at the screen representation, we have met the goal that an expression in the output language (in this case, an image) be allowed as a component of the input expression (in this case, by pointing at the screen representation). If we ask for a listing of files, we would want the result to be a representation that can, in turn, be used directly to specify the further operations to be done. Notice that this is not how a conversation works. In conversation, one may refer to what has been said previously, but one cannot operate upon what has been said. This requirement does not necessarily imply an interface of pictures, diagrams, or icons. It can be done with words and descriptions. The key properties are that the objects, whatever their form, have behaviors and can be referred to by other objects, and that referring to an object causes it to behave. In the file-listing example, we must be able to use the output expression that represents the file in question as a part of the input expression calling for whatever operation we desire upon that file, and the output expression that represents the file must change as a result of being referred to in this way. The goal is to permit the user to act as if the representation is the thing itself.

These conditions are met in many screen editors when the task is the arrangement of strings of characters. The characters appear as they are typed. They are then available for further operations. We treat them as though they are the things we are manipulating. These conditions are also met in the statistics example with which we opened this article (Figure 1), and in Steamer. The special conditions are not met in file-listing commands on most systems, the commands that allow one to display the names and attributes of file structure. The issue is that the outputs of these commands are simply "names" of the objects, and operating on the names does nothing to the objects to which the names refer. In a direct manipulation situation, we would feel that we had the files in front of us, that the program that "listed" the files actually placed the files before us. Any further operation on the files would take place upon the very objects delivered by the directory-listing command. This would provide the feeling of directly manipulating the objects that were returned.

The point is that when an interface presents a world of behaving objects rather than a language of description, manipulating a representation can have the same effects and the same feel as manipulating the thing being represented. The members of the audience of a well-staged play willfully suspend their beliefs that the players are actors and become directly engaged in the content of the drama. In a similar way, the user of a well-designed model-world interface can willfully suspend belief that the objects depicted are artifacts of some program and can thereby directly engage the world of the objects. This is the essence of the "first-personness" feeling of direct engagement. Let us now return to the issue of distance and explore the ways that an interface can be direct or indirect with respect to a particular task.

3. TWO FORMS OF DISTANCE: SEMANTIC AND ARTICULATORY

Whenever we interact with a device, we are using an interface language. That is, we must use a language to describe to the device the nature of the actions we wish to have performed. This is true regardless of whether we are dealing with an interface based on the conversation metaphor or on the model-world metaphor, although the properties of the language in the two cases are different. A description of desired actions is an expression in the interface language.

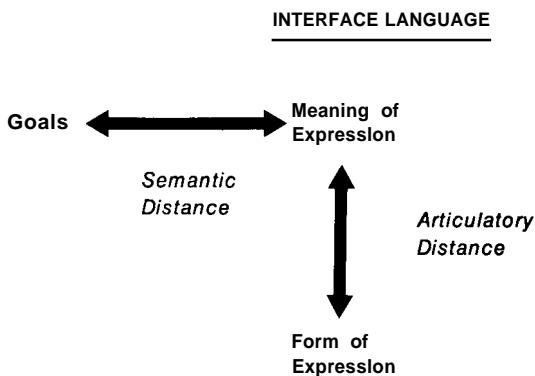
The notion of an interface language is not confined to the everyday meaning of language. Setting a switch or turning a steering wheel can be expressions in an interface language if switch setting or wheel turning are how one specifies the operations that are to be done. After an action has been performed, evaluation of the outcome requires that the device make available some indication of what has happened: that output is an expression in the output interface language. Output interface languages are often impoverished. Frequently the output interface language does not share vocabulary with the input interface language. Two forms of interface language—two dialects, if you will—must exist to span the gulfs between user and device: the input interface language and the output interface language.

Both the languages people speak and computer programming languages are almost entirely symbolic in the sense that there is an arbitrary relationship between the form of a vocabulary item and its meaning. The reference relationship is established by convention and must be learned. There is no way to infer meaning from form for most vocabulary items. Because of the relative independence of meaning and form we describe separately two properties of interface languages: semantic distance and articulatory distance. Figure 4 summarizes the relationship between semantic and articulatory distance. In the following sections we treat each of these distances separately and discuss them in relation to the gulfs of execution and evaluation.

3.1. Semantic Distance

Semantic distance concerns the relation of the meaning of an expression in the interface language to what the user wants to say. Two important questions about semantic distance are (1) *Is it possible to say what one wants to say in this language?* That is, does the language support the user's conception of the task domain? Does it encode the concepts and distinctions in the domain in the same way that the user thinks about them? (2) *Can the things of interest be said concisely?* Can the user say what is wanted in a straightforward fashion, or must the user

Figure 4. Every expression in the interface language has a meaning and a form. Semantic distance reflects the relationship between the user intentions and the meaning of expressions in the interface languages both for input and output. Articulatory distance reflects the relationship between the physical form of an expression in the interaction language and its meaning, again, both for input and output. The easier it is to go from the form or appearance of the input or output to meaning, the smaller the articulatory distance.



construct a complicated expression to do what appears in the user's thoughts as a conceptually simple piece of work?

Semantic distance is an issue with all languages. Natural languages generally evolve such that they have rich vocabularies for domains that are of importance to their speakers. When a person learns a new language—especially when the language is from a different culture—the new language may seem indirect, requiring complicated constructs to describe things the learner thinks should be easy to say. But the differences in apparent directness reflect differences in what things are thought important in the two cultures. Natural languages can and do change as the need arises. This occurs through the introduction of new vocabulary or by changing the meaning of existing terms. The result is to make the language semantically more direct with respect to the topic of interest.

3.2. Semantic Distance in the Gulfs of Execution and Evaluation

Beware the Turing tar-pit in which everything is possible but nothing of interest is easy (Perlis, 1982, p. 10).

The Gulf of Execution

At the highest level of description, a task may be described by the user's intention: "compose this piece" or "format this paper." At the lowest level of description, the performance of the task consists of the shuffling of bits inside the machine. Between the interface and the low-level operations of the machine is

the system-provided task-support structure that implements the expressions in the interface language. The situation that Perlis (1982) called the “Turing tar-pit” is one in which the interface language lies near or at the level of bit shuffling of a very simple abstract machine. In this case, the entire burden of spanning the gulf from user intention to bit manipulation is carried by the user. The relationship between the user’s intention and the organization of the instructions given to the machine is distant, complicated, and hard to follow. Where the machine is of minimal complexity, as is the case with the Turing machine example, the wide gulf between user intention and machine instructions must be filled by the user’s extensive planning and translation activities. These activities are difficult and rife with opportunities for error.

Semantic directness requires matching the level of description required by the interface language to the level at which the person thinks of the task. It is always the case that the user must generate some information-processing structure to span the gulf. Semantic distance in the gulf of execution reflects how much of the required structure is provided by the system and how much by the user. The more that the user must provide, the greater the distance to be bridged.

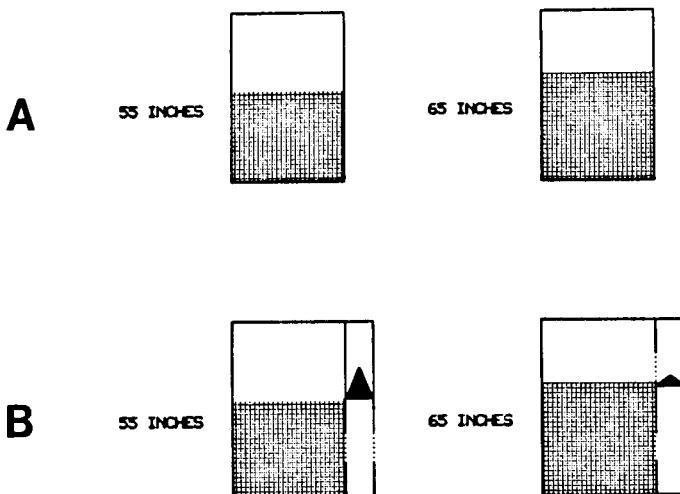
The Gulf of Evaluation

On the evaluation side, semantic distance refers to the amount of processing structure that is required for the user to determine whether the goal has been achieved. If the terms of the output are not those of the user’s intention, the user will be required to translate the output into terms that are compatible with the intention in order to make the evaluation. For example, suppose a user’s intent is to control how fast the water level in a tank rises. The user does some controlling action and observes the output. But if the output only shows the current value, the user has to observe the value over time and mentally compare the values at different times to see what the rate of change is (see Figure 5). The information needed for the evaluation is in the output, but it is not there in a form that directly fits the terms of the evaluation. The burden is on the user to perform the required transformations, and that requires effort. Suppose the rate of change were directly displayed, as in Figure 5B. This indication reduces the mental workload, making the semantic distance between intentions and output language much shorter.

3.3. Reducing the Semantic Distance That Must Be Spanned

Figure 5 provides one illustration of how semantic distance can be changed. In general, there are only two basic ways to reduce the distance, one from the system side (requiring effort on the part of the system designer), the other from the user side (requiring effort on the part of the user). Each direction of bridge building has several components. Here let us consider the following possibili-

Figure 5. Matching user's intentions by appropriate output language. The user attempts to control the rate at which the water level in the tank is rising. In (A), the only indication is a meter that shows the current level. This requires the user to observe the meter over time and to do a mental computation on the observations. (B) shows a display that is more semantically direct: The rate of change is graphically indicated. (These illustrations are from the working Steamer system of Hollan, Hutchins, & Weitzman, 1984.)



ties: (1) The designer can construct higher-level and specialized languages that move toward the user, making the semantics of the input and output languages match that of the user. (2) The user can develop competence by building new mental structures to bridge the gulfs. In particular, this requires the user to automate the response sequence and to learn to think in the same language as that required by the system.

Higher-Level Languages

One way to bridge the gulf between the intentions of the user and the specifications required by the computer is well known: Provide the user with a higher-level language, one that directly expresses frequently encountered structures of problem decomposition. Instead of requiring the complete decomposition of the task to low-level operations, let the task be described in the same language used within the task domain itself. Although the computer still requires low-level specification, the job of translating from the domain language to the programming language can be taken over by the machine itself.

This implies that designers of higher-level languages should consider how to develop interface languages for which it will be easy for the user to create the mediating structure between intentions and expressions in the language. One way to facilitate this process is to provide consistency across the interface sur-

face. That is, if the user builds a structure to make contact with some part of the interface surface, a savings in effort can be realized if it is possible to use all or part of that same structure to make contact with other areas.

The result of matching a language to the task domain brings both good news and bad news. The good news is that tasks are easier to specify. Even if considerable planning is still required to express a task in a high-level language, the amount of planning and translation that can be avoided by the user and passed off to the machine can be enormous. The bad news is that the language has lost generality. Tasks that do not easily decompose into the terms of the language may be difficult or impossible to represent. In the extreme case, what can be done is easy to do, but outside that specialized domain, nothing can be done.

The power of a specialized language system derives from carefully specified primitive operations, selected to match the predicted needs of the user, thus capturing frequently occurring structures of problem decomposition. The trouble is that there is a conflict between generality and matching to any specific problem domain. Some high-level languages and operating systems have attempted to close the gap between user intention and the interaction language while preserving freedom and ease of general expression by allowing for extensibility of the language or operating system. Such systems allow the users to move the interface closer to their conception of the task.

The Lisp language and the UNIX operating system serve as examples of this phenomenon. Lisp is a general-purpose language, but one that has extended itself to match a number of special high-level domains. As a result, Lisp can be thought of as having numerous levels on top of the underlying language kernel. There is a cost to this method. As more and more specialized domain levels get added, the language system gets larger and larger, becoming more clumsy to use, more expensive to support, and more difficult to learn. Just look at any of the manuals for the large Lisp systems (Interlisp, Zetalisp) to get a feel for the complexity involved. The same is true for the UNIX operating system, which started out with a number of low-level, general primitive operations. Users were allowed (and encouraged) to add their own, more specialized operations, or to package the primitives into higher-level operations. The results in all these cases are massive systems that are hard to learn and that require a large amount of support facilities. The documentation becomes huge, and not even system experts know all that is present. Moreover, the difficulty of maintaining such a large system increases the burden on everyone, and the possibility of having standard interfaces to each specialized function has long been given up.

The point is that as the interface approaches the user's intention end of the gulf, functions become more complicated and more specialized in purpose. Because of the incredible variety of human intentions, the lexicon of a language that aspires to both generality of coverage and domain-specific functions can grow very large. In any of the modern dialects of Lisp one sees a microcosm

of the argument about high-level languages in general. The fundamentals of the language are simple, but a great deal of effort is required to do anything useful at the low level of the language itself. Higher-level functions written in terms of lower-level ones make the system easier to use when the functions match intentions, but in doing so they may restrict possibilities, proliferate vocabulary, and require that a user know an increasing amount about the language of interaction rather than the domain of action.

Make the Output Show Semantic Concepts Directly

An example of reducing semantic distance on the output side is provided by the scenario of controlling the rate of filling a water tank, described in Figure 5. In that situation, the output display was modified to show rate of flow directly, something normally not displayed but instead left *to* the user to compute mentally.

In similar fashion, the change from line-oriented text editors to screen-oriented text editors, where the effects of editing commands can be seen instantly, is another example of matching the display to the user's semantics. In general, the development of WYSIWYG ("What You See Is What You Get") systems provides other examples. And finally, spreadsheet programs have been valuable, in part because their output format continually shows the state of the system as values are changed.

The attempt to develop good semantic matches with the system output confronts the same conflict between generality and power faced in the design of input languages. If the system is too specific and specialized, the output displays lack generality. If the system is too rich, the user has trouble learning and selecting among the possibilities. One solution for both the output and input problem is to abandon hope of maintaining general computing and output ability and to develop special-purpose systems for particular domains or tasks. In such a world, the location of the interface in semantic space is pushed closer to the domain language description. Here, things of interest are made simple because the lexicon of the interface language maps well into the lexicon of domain description. Considerable planning may still go on in the conception of the domain itself, but little or no planning or translation is required to get from the language of domain description to the language of the interface. The price paid for these advantages is a loss of generality: Many things are unnatural or even impossible.

Automated Behavior Does Not Reduce Semantic Distance

Cognitive effort is required to plan a sequence of actions to satisfy some intent. Generally, the more structure required of the user, the more effort use of the system will entail. However, this gap can be overcome if the users become familiar enough with the system. Structures that are used frequently need not

be rebuilt every time they are needed if they have been remembered. Thus, a user may remember how to do something rather than having to rederive how to do it. It is well known that when tasks are practiced sufficiently often, they become automated, requiring little or no conscious attention. As a result, over time the use of an interface to solve a particular set of problems will feel less difficult and more direct. Experienced users will sometimes argue that the interface they use directly satisfies their intentions, even when less skilled users complain of the complexity of the structures. To skilled users, the interface feels direct because the invocation of mediating structure has been automated. They have learned how to transform frequently arising intentions into action specifications. The result is a feeling of directness as compelling as that which results from semantic directness. As far as such users are concerned, the intention comes to mind and the action gets executed. There are no conscious intervening stages. (For example, a user of the vi text editor expressed this as follows: "I am an expert user of vi, and when I wish to delete a word, all I do is think 'delete that word,' my fingers automatically type 'dw,' and the word disappears from the screen. How could anything be more direct?")

The frequent use of even a poorly designed interface can sometimes result in a feeling of directness like that produced by a semantically direct interface. A user can compensate for the deficiencies of the interface through continual use and practice so that the ability to use it becomes automatic, requiring little conscious activity. While automatism is one factor which can contribute to a feeling of directness, it is essential for an interface designer to distinguish it from semantic distance. Automatization does not reduce the semantic distance that must be spanned; the gulfs between a user's intentions and the interface must still be bridged by the user. Although practice and the resulting expertise can make the crossing less difficult, it does not reduce the magnitude of the gulfs. Planning activity may be replaced by a single memory retrieval so that instead of figuring out what to do, the user remembers what to do. Automatization may feel like direct control, but it comes about for completely different reasons than semantic directness. Automatization is useful, for it improves the interaction of the user with the system, but the feeling of directness it produces depends only on how much practice a particular user has with the system and thus gives the system credit for the work the user has done. Although we need to remember that this happens, that users may adjust themselves to the interface and, with sufficient practice, may view it as directly supporting their intentions, we need to distinguish between the cases in which the feeling of directness originates from a close semantic coupling between intentions and the interface language and that which originates from practice. The resultant feeling of directness might be the same in the two cases, but there are crucial differences between how the feeling is acquired and what one needs to do as an interface designer to generate it.

The User Can Adapt to the System Representation

Another way to span the gulf is for the users to change their own conceptualization of the problem so that they come to think of it in the same terms as the system. In some sense, this means that the gulf is bridged by moving the user closer to the system. Because of their experience with the system, the users change both their understanding of the task and the language with which they think about issues. This is related to the notion of linguistic determinism. If it is true that the way we think about something is shaped by the vocabulary we have for talking about it, then it is important for the designer of a system to provide the user with a good representation of the task domain in question. The interface language should provide a powerful, productive way of thinking about the domain.

This form of the users adapting to the system representation takes place at a more fundamental level than the other ways of reducing semantic distance. While moving the interface closer to the users' intentions may make it difficult to realize some intentions, changing the users' conception of the domain may prevent some intentions from arising at all. So while a well-designed special-purpose language may give the users a powerful way of thinking about the domain, it may also restrict the users' flexibility to think about the domain in different ways.

The assumption that a user may change conceptual structure to match the interface language follows from the notion that every interface language implies a representation of the tasks it is applied to. The representation implied by an interface is not always a coherent one. Some interfaces provide a collection of partially overlapping views of a task domain. If a user is to move toward the model implied by the interface, and thus reduce the semantic distance, that model should be coherent and consistent over some conception of the domain. There is, of course, a trade-off here between the costs to the user of learning a new way to think about a domain and the potential added power of thinking about it in the new way.

Virtuosity and Semantic Distance

Sometimes users have a conception of a task and of a system that is broader and more powerful than that provided by an interface. The structures they build to make contact with the interface go beyond it. This is how we characterize virtuoso performances in which the user may "misuse" limited interface tools to satisfy intentions that even the system designer never anticipated. In such cases of virtuosity the notion of semantic distance becomes more complicated and we need to look very carefully at the task that is being accomplished. Semantic directness always involves the relationship between the task one wishes to accomplish and the ways the interface provides for accomplishing it.

If the task changes, then the semantic directness of the interface may also change.

Consider a musical example: Take the task of producing a middle-C note on two musical instruments, a piano and a violin. For this simple task, the piano provides the more direct interface because all one need do is find the key for middle-C and depress it, whereas on the violin, one must place the bow on the G string, place a choice of fingers in precisely the right location on that string, and draw the bow. A piano's keyboard is more semantically direct than the violin's strings and bow for the simple task of producing notes. The piano has a single well-defined vocabulary item for each of the notes within its range, while the violin has an infinity of vocabulary items, many of which do not produce proper notes at all. However, when the task is playing a musical piece well rather than simply producing notes, the directness of the interfaces can change. In this case, one might complain that a piano has a very indirect interface because it is a machine with which the performer "throws hammers at strings." The performer has no direct contact with the components that actually produce the sound, and so the production of desired nuances in sound is more difficult. Here, as musical virtuosity develops, the task that is to be accomplished also changes from just the production of notes to concern for how to control more subtle characteristics of the sounds like vibrato, the slight changes in pitch used to add expressiveness. For this task the violin provides a semantically more direct interface than the piano. Thus, as we have argued earlier, an analysis of the nature of the task being performed is essential in determining the semantic directness of an interface.

3.4. Articulatory Distance

In addition to its meaning, every vocabulary item in every language has a physical form and that form has an internal structure. Words in natural languages, for example, have phonetic structure when spoken and typographic structure when printed. Similarly, the vocabulary items that constitute an interface language have a physical structure. Where *semantic distance* has to do with the relationship between user's intentions and meanings of expressions, *articulatory distance* has to do with the relationship between the meanings of expressions and their physical form. On the input side, the form may be a sequence of character-selecting key presses for a command language interface, the movement of a mouse and the associated "mouseclicks" in a pointingdevice interface, or a phonetic string in a speech interface. On the output side, the form might be a string of characters, a change in an iconic representation, or variation in an auditory signal.

There are ways to design languages such that the relationships between the forms of the vocabulary items and their meanings are not arbitrary. One tech-

nique is to make the physical form of the vocabulary items structurally similar to their meanings. In spoken language this relationship is called onomatopoeia. Onomatopoetic words in spoken language refer to their meanings by imitating the sound they refer to. Thus we talk about the "boom" of explosions or the "cock-a-doodle-doo" of roosters. There is an economy here in that the user's knowledge of the structure of the surface acoustical form has a non-arbitrary relation to meaning. There is a directness of reference in this imitation; an intervening level of arbitrary symbolic relations is eliminated. Other uses of language exploit this effect partially. Thus, although the word "long" is arbitrarily associated with its meaning, sentences like "She stayed a looooooooooong time" exploit a structural similarity between the surface form of "long" (whether written or spoken) and the intended meaning. The same sorts of things can be done in the design of interface languages.

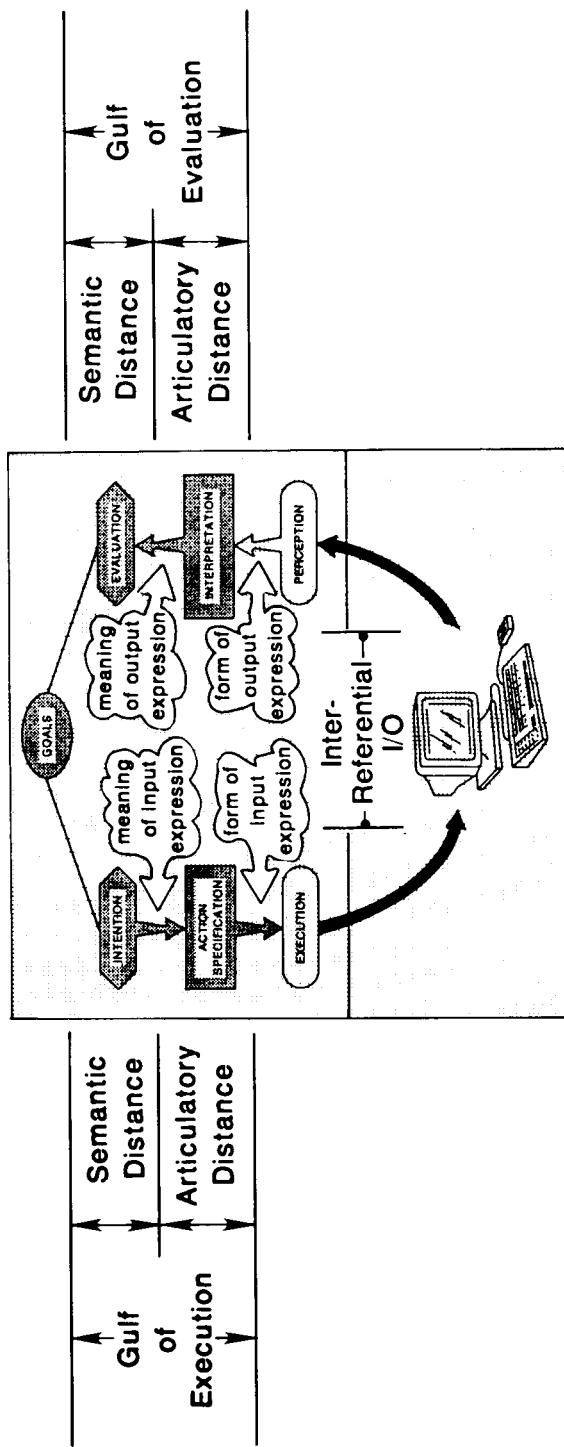
In many ways, the interface languages should have an easier time of exploiting articulatory similarity than do natural languages because of the rich technological base available to them. Thus, if the intent is to draw a diagram, the interface might accept as input drawing motions. In turn, it could present as output diagrams, graphs, and images. If one is talking about sound patterns in the input interface language, the output could be the sounds themselves. The computer has the potential to exploit articulatory similarities through technological innovation in the varieties of dimensions upon which it can operate. This potential has not been exploited, in part because of economic constraints. The restriction to simple keyboard input limits the form and structure of the input languages and the restriction to simple, alphanumeric terminals with small, low-resolution screens, limits the form and structure of the output languages.

3.5. Articulatory Distance in the Gulfs of Execution and Evaluation

The relationships among semantic distance, articulatory distance, and the gulfs of execution and evaluation are illustrated in Figure 6.

Take the simple, commonplace activity of moving a cursor on the screen. If we do this by moving a mouse, pointing with a finger or a light pen at the screen, or otherwise mimicking the desired motion, then at the level of action execution, these interactions all exhibit articulatory directness. The meaning of the intention is cursor movement and the action is specified by means of a similar movement. One way to achieve articulatory directness at the input side is to provide an interface that permits specification of an action by mimicking it, thus supporting an articulatory similarity between the vocabulary item and its meaning. Any nonarbitrary relationship between the form of an item and its meaning can be a basis for articulatory directness. While structural relationships of form to meaning may be desirable, it is sometimes necessary to re-

Figure 6. Forming an intention is the activity that spans semantic distance in the gulf of execution. The intention specifies the meaning of the input expression that is to satisfy the user's goal. Forming an action specification is the activity that spans articulatory distance in the gulf of execution. The action specification prescribes the form of an input expression having the desired meaning. The form of the input expression is executed by the user on the machine interface and the form of the output expression appears on the machine interface, to be perceived by the user. When some part of the form of a previous output expression is incorporated in the form of a new input expression, the input and output are said to be inter-referential. Interpretation is the activity that spans articulatory distance in the gulf of evaluation. Interpretation determines the meaning of the output expression from the form of the output expression. Evaluation is the activity that spans semantic distance in the gulf of evaluation. Evaluation assesses the relationship between the meaning of the output expression and the user's goal.



sort to an arbitrary relationship of form to meaning. Still, some arbitrary relationships are easier to learn than others. It may be possible to exploit previous user knowledge in creating this relationship. Much of the work on command names in command language interfaces is an instance of trying to develop memorable and discriminable relationships between the forms and the meanings of command names (Black & Moran, 1982; Black & Sebrechts, 1981; Carroll, 1985).

Articulatory directness on the output side is similar. If the user is following the changes in some variable, a moving graphical display can provide articulatory directness. A table of numbers, although containing the same semantic information, does not provide articulatory directness. Thus, the graphical display and the table of numbers might be equal in semantic directness, but unequal in articulatory directness. The goal of designing for articulatory directness is to couple the perceived form of action and meaning so naturally that the relationships between intentions and actions and between actions and output seem straightforward and obvious.

In general, articulatory directness is highly dependent upon I/O technology. Increasing the articulatory directness of actions and displays requires a much richer set of input/output devices than most systems currently have. In addition to keyboards and bit-mapped screens, we see the need for various forms of pointing devices. Such pointing devices have important *spatio-mimetic* properties and thus support the articulatory directness of input for tasks that can be represented spatially. The mouse is useful for a wide variety of tasks not because of any properties inherent in itself, but because we map so many kinds of relationships (even ones that are not intrinsically spatial) on to spatial metaphors. In addition, there are often needs for sound and speech, certainly as outputs, and possibly as inputs. Precise control of timing will be necessary for those applications where the domain of interest is time sensitive. Perhaps it is stretching the imagination beyond its willing limits, but Galton (1894) suggested and carried out a set of experiments on doing arithmetic by sense of smell. Less fancifully conceived, input might be sensitive not only to touch, place, and timing, but also to pressure or to torque (see Buxton, 1986; Minsky, 1984).

4. DIRECT ENGAGEMENT

Direct engagement occurs when a user experiences direct interaction with the objects in a domain. Here there is a feeling of involvement directly with a world of objects rather than of communication with an intermediary. The interactions are much like interacting with objects in the physical world. Actions apply to the objects, observations are made directly upon those objects, and the interface and the computer become invisible. Although we believe this feeling of direct engagement to be of critical importance, in fact, we know little about

the actual requirements for producing it. Laurel (1986) discusses some of the requirements. At a minimum, to allow a feeling of direct engagement the system requires the following:

Execution and evaluation should exhibit both semantic and articulatory directness.

Input and output languages of the interface should be inter-referential, allowing an input expression to incorporate or make use of a previous output expression. This is crucial for creating the illusion that one is directly manipulating the objects of concern.

The system should be responsive, with no delays between execution and the results, except where those delays are appropriate for the knowledge domain itself.

The interface should be unobtrusive, not interfering or intruding. If the interface itself is noticed, then it stands in a third-person relationship to the objects of interest, and detracts from the directness of the engagement.

In order to have a feeling of direct engagement, the interface must provide the user with a world in which to interact. The objects of that world must feel like they are the objects of interest, that one is doing things with them and watching how they react. In order for this to be the case, the output language must present representations of objects in forms that behave in the way that the user thinks of the objects behaving. Whatever changes are caused in the objects by the set of operations must be depicted in the representation of the objects. This use of the same object as both an input and output entity is essential to providing objects that behave as if they are the real thing. It is because an input expression can contain a previous output expression that the user feels the output expression is the thing itself and that the operation is applied directly to the thing itself.

In addition, all of the discussions of semantic and articulatory directness apply here too, because the designer of the interface must be concerned with what is to be done and how one articulates that in the languages of interaction. But the designer must also be concerned with creating and supporting an illusion. The specification of what needs to be done and evidence that it has been done must not violate the illusion, else the feeling of direct engagement will be lost.

One factor that seems especially relevant to maintaining this illusion is the form and speed of feedback. Rapid feedback in terms of changes in the behavior of objects not only allows for the modification of actions even as they are being executed, but also supports the feeling of acting directly on the objects

themselves. It removes the perception of the computer as an intermediary by providing continual representation of system state. In addition, rapidity of feedback and continual representation of state allows one to make use of perceptual faculties in evaluating the outcome of actions. We can watch the actions take place, monitoring them much like we monitor our interactions with the physical world. The reduction in the cognitive load of mentally maintaining relevant information and the form of the interaction contribute to the feeling of engagement.

5. A SPACE OF INTERFACES

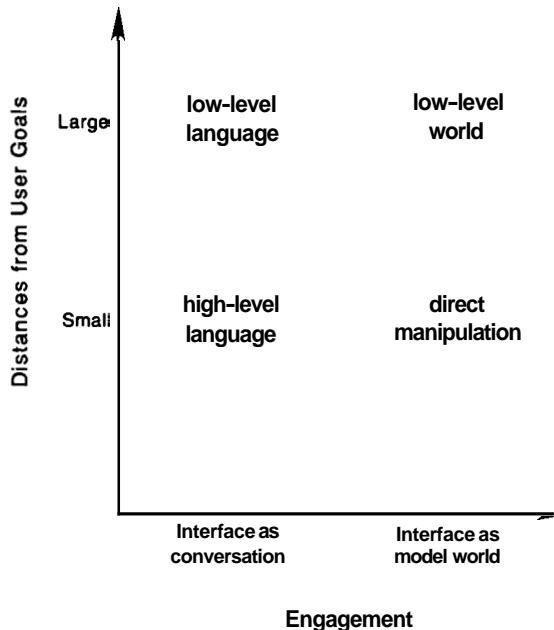
Distance and engagement are depicted in Figure 7 as two major dimensions in a space of interface designs. The dimension of engagement has two landmark values: One is the metaphor of interface as conversation; the other is the metaphor of interface as model world. The dimension of distance actually contains two distances to be spanned: semantic and articulatory distances, the two kinds of gulfs that lie between the user's conception of the task and the interface language.

The least direct interface is often one that provides a low-level language interface, for this is apt to provide the weakest semantic match between intentions and the language of the interface. In this case, the interface is an intermediary between the user and the task. Even worse, it is an intermediary that does not understand actions at the level of description in which the user likes to think of them. Here the user must translate intentions into complex or lengthy expressions in the language that the interface intermediary can understand.

A more direct situation arises when the central metaphor of the interface is a world. Then the user can be directly engaged with the objects in a world; but still, if the actions in that world do not match those that the user wishes to perform within the task domain, getting the task done may be a difficult process. The user may believe that things are getting done and may even experience a sense of engagement with the world, yet still be doing things at too low a level. This is the state of some of the recently introduced direct manipulation systems: They produce an immediate sense of engagement, but as the user develops experience with the system, the interface appears clumsy, to interfere too much, and to demand too many actions and decisions at the wrong level of specification. These interfaces appear on the surface to be direct manipulation interfaces, but they fail to produce the proper feelings of direct engagement with the task world.

Closing the distance between the user's intentions and the level of specification of the interface language allows the user to make efficient specifications of intentions. Where this is done with a high-level language, quite efficient interfaces can be designed. This is the situation in most modern integrated pro-

Figure 7. A space of interfaces. The dimensions of distance from user goals and degree of engagement form a space of interfaces within which we can locate some familiar types of interfaces. Direct manipulation interfaces are those that minimize the distances and maximize engagement. As always, the distance between user intentions and the interface language depends on the nature of the task the user is performing.



gramming environments. For some classes of tasks, such interfaces may be superior to direct manipulation interfaces.

Finally, the most direct of the interfaces will lie where engagement is maximized, where just the right semantic and articulatory matches are provided, and where all distances are minimized.

6. PROBLEMS WITH DIRECT MANIPULATION

Direct manipulation systems have both virtues and vices. For instance, the immediacy of feedback and the natural translation of intentions to actions make some tasks easy. The matching of levels of thought to the interface language — semantic directness — increases the ease and power of performing some activities at a potential cost of generality and flexibility. But not all things should be done directly. For example, a repetitive operation is probably best done via a script, that is, through a symbolic description of the tasks that

are to be accomplished. Direct manipulation interfaces have difficulty handling variables, or distinguishing the depiction of an individual element from a representation of a set or class of elements. Direct manipulation interfaces have problems with accuracy, for the notion of mimetic action puts the responsibility on the user to control actions with precision, a responsibility that is sometimes best handled through the intelligence of the system and sometimes best communicated symbolically.

A more fundamental problem with direct manipulation interfaces arises from the fact that much of the appeal and power of this form of interface comes from its ability to directly support the way we normally think about a domain. A direct manipulation interface amplifies our knowledge of the domain and allows us to think in the familiar terms of the application domain rather than in those of the medium of computation. But if we restrict ourselves to only building an interface that allows us to do things we can already do and to think in ways we already think, we will miss the most exciting potential of new technology: to provide new ways to think of and to interact with a domain. Providing these new ways and creating conditions that will make them feel direct and natural is an important challenge to the interface designer.

Direct manipulation interfaces are not a panacea. Although with sufficient practice by the user many interfaces can come to feel direct, a properly designed interface, one which exploits semantic and articulatory directness, should decrease the amount of learning required and provide a natural mapping to the task. But interface design is subject to many tradeoffs. There are surely instances when one might wisely trade off directness for generality, or for more facile ways of saying abstract things. The articulatory directness involved in pointing at objects might need to be traded off against the difficulties of moving the hands between input devices or of problems in pointing with great precision.

It is important not to equate directness with ease of use. Indeed, if the interface is really invisible, then the difficulties within the task domain get transferred directly into difficulties for the user. Suppose the user struggles to formulate an intention because of lack of knowledge of the task domain. The user may complain that the system is difficult to use. But the difficulty is in the task domain, not in the interface language. Direct manipulation interfaces do not pretend to assist in overcoming problems that result from poor understanding of the task domain.

What about the claims for direct manipulation? We believe that direct manipulation systems carry gains in ease of learning and ease of use. If the mapping is done correctly, then both the form and the meaning of commands should be easier to acquire and retain. Interpretation of the output should be immediate and straightforward. If the interface is a model of the task domain, then one could have the feeling of directly engaging the problem of interest itself. It is sometimes said that in such situations the interface disappears. It is

probably more revealing to say that the interface is no longer recognized as an interface.

But are these desirable features? Are the trade-offs too costly? As always, we are sure that the answer will depend on the tasks to be accomplished. Certain kinds of abstraction that are easy to deal with in language seem difficult in a concrete model of a task domain. When we give up the conversation metaphor, we also give up dealing in descriptions, and in some contexts, there is great power in descriptions. As an interface to a programming task, direct manipulation interfaces are problematic. We know of no really useful direct manipulation programming environments. Issues such as controlling the scope of variable bindings promise to be quite tricky in the direct manipulation environments. Will direct manipulation systems live up to their promise? Yes and no. Basically, the systems will be good and powerful for some purposes, poor and weak for others. In the end, many things done today will be replaced by direct manipulation systems. But we will still have conventional programming languages.

On the surface, the fundamental idea of a direct manipulation interface to a task flies in the face of two thousand years of development of abstract formalisms as a means of understanding and controlling the world. Until very recently, the use of computers has been an activity squarely in that tradition. So the exterior of direct manipulation, providing as it does for the direct control of a specific task world, seems somehow atavistic, a return to concrete thinking. On the inside, of course, the implementation of direct manipulation systems is yet another step in that long, formal tradition. The illusion of the absolutely manipulable concrete world is made possible by the technology of abstraction.

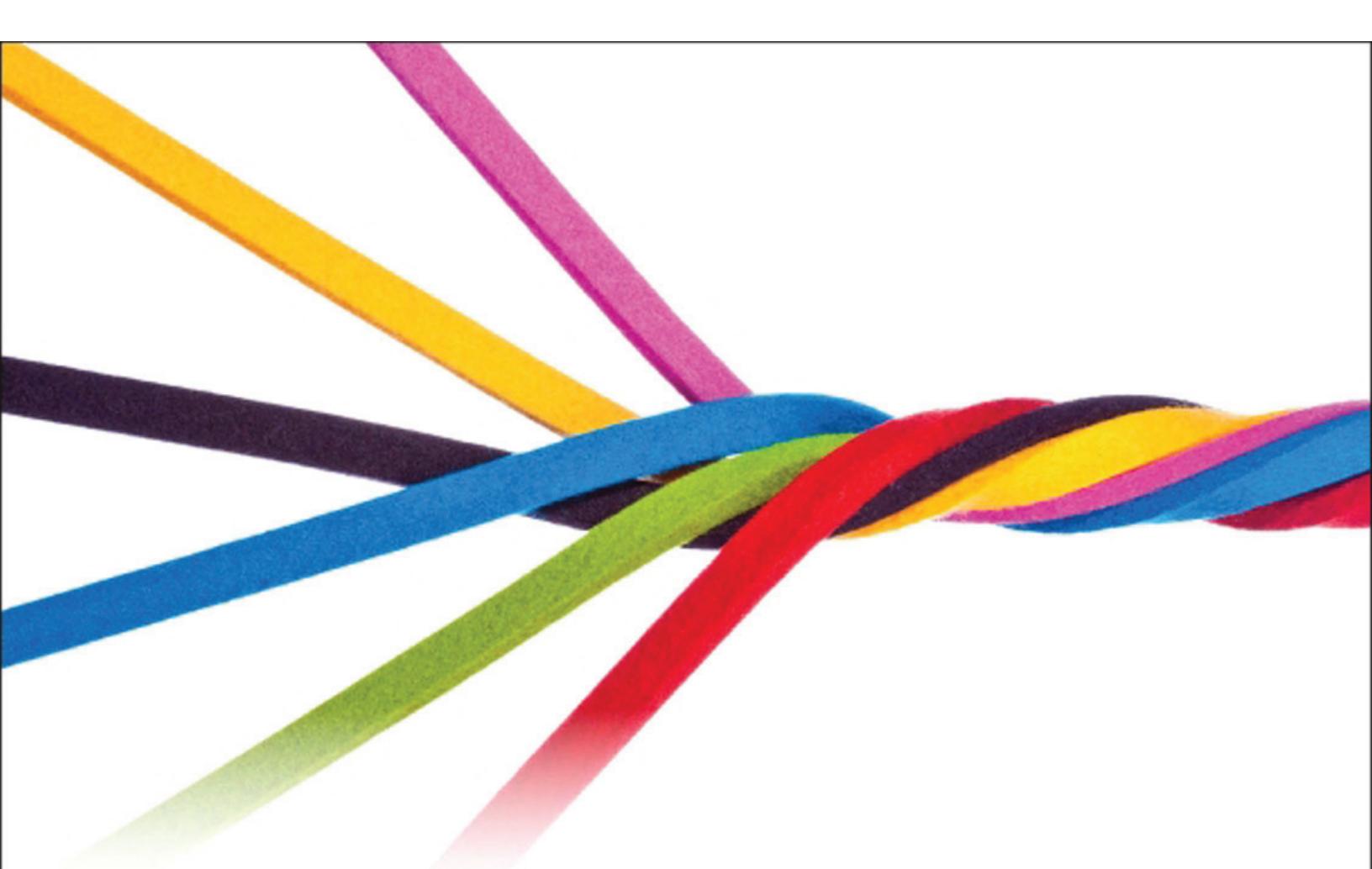
Acknowledgments. We thank Ben Shneiderman for his helpful comments on an earlier draft of the chapter, Eileen Conway for her aid with the illustrations, and Julie Norman and Sondra Buffett for extensive editorial comments.

Support. The research reported here was conducted under Contract N00014-85-C-0133, NR 667-541 with the Personnel and Training Research Programs of the Office of Naval Research and with the support of the Navy Personnel Research and Development Center. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsoring agency.

REFERENCES

- Black, J. B., & Moran, T. P. (1982). Learning and remembering command names. *Proceedings of the Human Factors in Computer Systems Conference*, 8-11. New York: ACM.
- Black, J. B., & Sebrechts, M. M. (1981). Facilitating human-computer communication. *Applied Psycholinguistics*, 2, 149-177.
- Borning, A. (1979). *ThingLab: A constraint-oriented simulation laboratory* (Tech. Rep. No. SSL-79-3). Palo Alto, CA: Xerox Palo Alto Research Center.

- Budge, B. (1983). *Pinball construction set* [Computer program]. San Mateo, CA: Electronic Arts.
- Buxton, W. (1986). There's more to interaction than meets the eye: Some issues in manual input. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
- Carrol, J. M. (1985). *What's in a name? An essay in the psychology of reference*. New York: Freeman.
- diSessa, A. A. (1985). A principles design for an integrated computational environment. *Human-Computer Interaction*, 1, 1-47.
- Draper, S. W. (1986). Display managers as the basis for user-machine communication. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
- Galton, F. (1894). Arithmetic by smell. *Psychological Review*, 1, 61-62.
- Hollan, J. D., Hutchins, E., & Weitzman, L. (1984). Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, 5, 15-27.
- Hollan, J. D., Stevens, A., & Williams, M. D. (1980). Steamer: An advanced computer-assisted instruction system for propulsion engineering. *Proceedings @ Summer Computer Simulation Conference*, 400-404. Arlington, VA: AFIPS Press.
- Kay, A. (1984, September). Computer software. *Scientific American*, 52-59.
- Laurel, B. K. (1986). Interface as mimesis. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
- Minksy, M. R. (1984, July). Manipulating simulated objects with real-world gestures using a force and position sensitive screen. *Computer Graphics*, 195-203.
- Norman, D. A., & Draper, S. W. (Eds.). (1986). *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
- Perlis, A. J. (1982). Epigrams on programming. *SIGPLAN Notices*, 17(9), 7-13.
- Shneiderman, B. (1974). A computer graphics system for polynomials. *The Mathematics Teacher*, 67(2), 111-113.
- Shneiderman, B. (1982). The future of interactive systems and the emergence of direct manipulation. *Behavior and Information Technology*, 1, 237-256.
- Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8), 57-69.
- Sutherland, I. E. (1963). Sketchpad: A man-machine graphical communication system. *Proceedings of the Spring Joint Computer Conference*, 329-346. Baltimore, MD: Spartan Books.



Human-Computer Interaction

An Empirical Research Perspective



I. Scott MacKenzie

Human-Computer Interaction

An Empirical Research Perspective

I. Scott MacKenzie



ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier



The Human Factor

2

The deepest challenges in human-computer interaction (HCI) lie in the human factor. Humans are complicated. Computers, by comparison, are simple. Computers are designed and built and they function in rather strict terms according to their programmed capabilities. There is no parallel with humans. Human scientists (including those in HCI) confront something computer scientists rarely think about: variability. Humans differ. We're young, old, female, male, experts, novices, left-handed, right-handed, English-speaking, Chinese-speaking, from the north, from the south, tall, short, strong, weak, fast, slow, able-bodied, disabled, sighted, blind, motivated, lazy, creative, bland, tired, alert, and on and on. The variability humans bring to the table means our work is never precise. It is always approximate. Designing systems that work well, period, is a lofty goal, but unfortunately, it is not possible to the degree we would like to achieve. A system might work well for a subset of people, but venture to the edges along any dimension (see list above), and the system might work poorly, or not at all. It is for this reason that HCI designers have precepts like "know thy user" (Shneiderman and Plaisant, 2005, p. 66).

Researchers in HCI have questions—lots of them. We are good at the small ones, but the big ones are difficult to answer: Why do humans make mistakes? Why do humans forget how to do things? Why do humans get confused while installing apps on their computers? Why do humans have trouble driving while talking on a mobile phone? Why do humans enjoy *Facebook* so much? Obviously, the human part is hugely important and intriguing. The more we understand humans, the better are our chances of designing interactive systems—interactions—that work as intended. So in this chapter I examine the human, but the computer and the interaction are never far away.

The questions in the preceding paragraph begin with *why*. They are big questions. Unfortunately, they do not lend themselves to empirical enquiry, which is the focus of this book. Take the first question: *Why do humans make mistakes?* From an empirical research perspective, the question is too broad. It cannot be answered with any precision. Our best bet is to narrow in on a defined group of humans (*a population*) and ask them to do a particular task on a particular system in a particular environment. We observe the interaction and measure the behavior. Along the way, we log the mistakes, classify them, count them, and take note of where and how the mistakes occurred. If our methodology is sound, we might assimilate enough information to put forth an answer to the *why* question—in a narrow sense.

If we do enough research like this, we might develop an answer in a broad sense. But a grounded and rigorous approach to empirical research requires small and narrowly focused questions.

Descriptive models, which will be discussed in Chapter 7, seek to delineate and categorize a problem space. They are tools for thinking, rather than tools for predicting. A descriptive model for “the human” would be useful indeed. It would help us get started in understanding the human, to delineate and categorize aspects of the human that are relevant to HCI. In fact there are many such models, and I will introduce several in this chapter.

2.1 Time scale of human action

Newell's *Time Scale of Human Action* is a descriptive model of the human (Newell, 1990, p. 122). It delineates the problem space by positioning different types of human actions in timeframes within which the actions occur. (See [Figure 2.1](#).) The model has four bands, a *biological band*, a *cognitive band*, a *rational band*, and a *social band*. Each band is divided into three levels. Time is ordered by seconds and appears on a logarithmic scale, with each level a factor of ten longer than the level below it. The units are microseconds at the bottom and months at the top. For nine levels, Newell ascribes a label for the human system at work (e.g., *operations* or *task*). Within these labels we see a connection with HCI. The labels for the bands suggest a worldview or theory of human action.

The most common dependent variable in experimental research in HCI is time—the time for a user to do a task. In this sense, Newell's time-scale model is relevant to HCI. The model is also appropriate because it reflects the multidisciplinary nature of the field. HCI research is both *high level* and *low level*, and we see this in the model. If desired, we could select a paper at random from an HCI conference proceedings or journal, study it, then position the work somewhere in [Figure 2.1](#). For example, research on selection techniques, menu design, force or auditory feedback, text entry, gestural input, and so on, is within the cognitive band. The tasks for these interactions typically last on the order of a few hundred milliseconds (ms) to a few dozen seconds. Newell characterizes these as deliberate acts, operations, and unit tasks.

Up in the rational band, users are engaged in tasks that span minutes, tens of minutes, or hours. Research topics here include web navigation, user search strategies, user-centered design, collaborative computing, ubiquitous computing, social navigation, and situated awareness. Tasks related to these research areas occupy users for minutes or hours.

Tasks lasting days, weeks, or months are in the social band. HCI topics here might include workplace habits, groupware usage patterns, social networking, online dating, privacy, media spaces, user styles and preferences, design theory, and so on.

Another insight Newell's model provides pertains to research methodology. Research at the bottom of the scale is highly quantitative in nature. Work in the biological band, for example, is likely experimental and empirical—at the level of neural

Scale (sec)	Time Units	System	World (theory)
10^7	Months		SOCIAL BAND
10^6	Weeks		
10^5	Days		
10^4	Hours	Task	RATIONAL BAND
10^3	10 min	Task	
10^2	Minutes	Task	
10^1	10 sec	Unit task	COGNITIVE BAND
10^0	1 sec	Operations	
10^{-1}	100 ms	Deliberate act	
10^{-2}	10 ms	Neural circuit	BIOLOGICAL BAND
10^{-3}	1 ms	Neuron	
10^{-4}	100 μ s	Organelle	

FIGURE 2.1

Newell's time scale of human action.

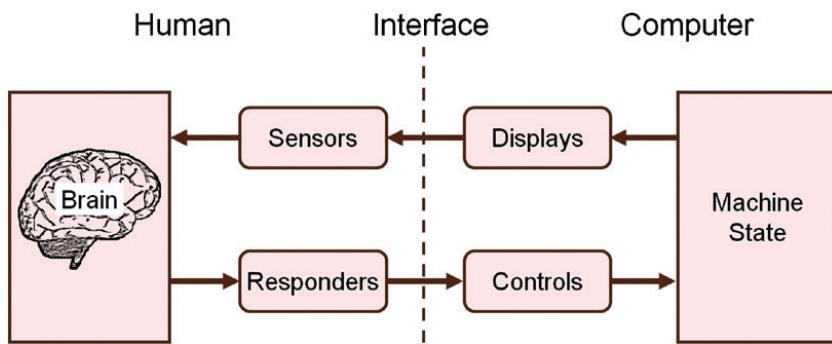
(From Newell, 1990, p. 122)

impulses. At the top of the scale, the reverse is true. In the social band, research methods tend to be qualitative and non-experimental. Techniques researchers employ here include interviews, observation, case studies, scenarios, and so on. Furthermore, the transition between qualitative and quantitative methods moving from top to bottom in the figure is gradual. As one methodology becomes more prominent, the other becomes less prominent. Researchers in the social band primarily use qualitative methods, but often include some quantitative methods. For example, research on workplace habits, while primarily qualitative, might include some quantitative methods (e.g., counting the number of personal e-mails sent each day while at work). Thus, qualitative research in the social band also includes some quantitative assessment. Conversely, researchers in the cognitive band primarily use quantitative methods but typically include some qualitative methods. For example, an experiment on human performance with pointing devices, while primarily quantitative, might include an interview at the end to gather comments and suggestions on the interactions. Thus, quantitative, experimental work in the cognitive band includes some qualitative assessment as well.

Newell speculates further on bands above the social band: a *historical band* operating at the level of years to thousands of years, and an *evolutionary band* operating at the level of tens of thousands to millions of years (Newell, 1990, p. 152). We will forgo interpreting these in terms of human-computer interaction.

2.2 Human factors

There are many ways to characterize the human in interactive systems. One is the model human processor of Card et al. (1983), which was introduced in Chapter 1.

**FIGURE 2.2**

Human factors view of the human operator in a work environment.

(After Kantowitz and Sorkin, 1983, p. 4)

Other characterizations exist as well. Human factors researchers often use a model showing a human operator confronting a machine, like the image in Figure 2.2. The human monitors the state of the computer through sensors and displays and controls the state of the computer through responders and controls. The dashed vertical line is important since it is at the interface where interaction takes place. This is the location where researchers observe and measure the behavioral events that form the interaction.

Figure 2.2 is a convenient way to organize this section, since it simplifies the human to three components: sensors, responders, and a brain.

2.3 Sensors

Rosa: You deny everything except what you want to believe. That's the sort of man you are.

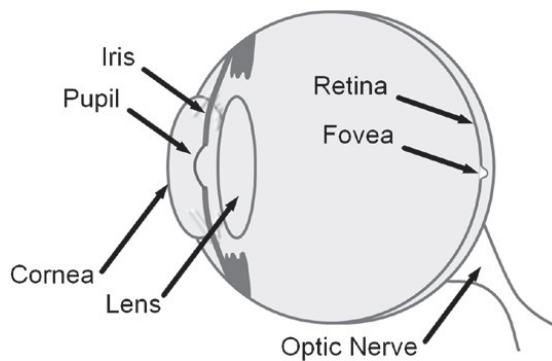
Bjartur: I have my five senses, and don't see what need there is for more.

(Halldór Laxness, *Independent People*)

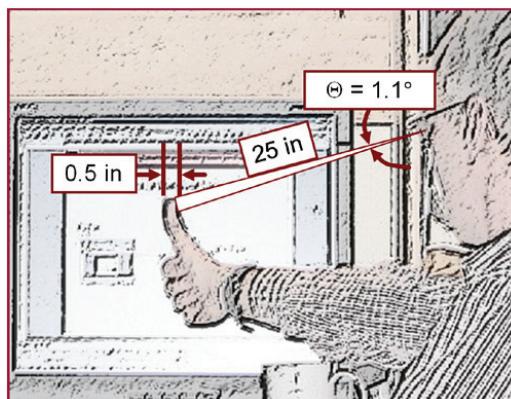
The five classical human senses are vision, hearing, taste, smell, and touch. Each brings distinctly different physical properties of the environment to the human. One feature the senses share is the reception and conversion into electrical nerve signals of physical phenomena such as sound waves, light rays, flavors, odors, and physical contact. The signals are transmitted to the brain for processing. Sensory stimuli and sense organs are purely physiological. Perception, discussed later, includes both the sensing of stimuli and use of the brain to develop identification, awareness, and understanding of what is being sensed. We begin with the first of the five senses just noted: vision.

2.3.1 Vision (Sight)

Vision, or sight, is the human ability to receive information from the environment in the form of visible light perceived by the eye. The visual sensory channel

**FIGURE 2.3**

The eye.

**FIGURE 2.4**

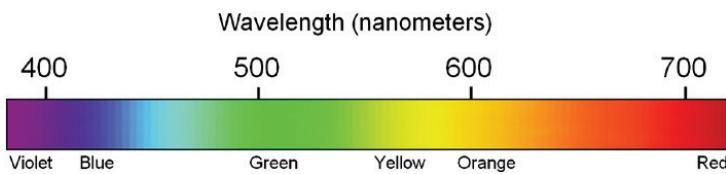
The fovea image spans a region a little more than one degree of visual angle.

is hugely important, as most people obtain about 80 percent of their information through the sense of light (Asakawa and Takagi, 2007). The act of seeing begins with the reception of light through the eye's lens. The lens focuses the light into an image projected on to the retina at the back of the eye. (See [Figure 2.3](#).) The retina is a transducer, converting visible light into neurological signals sent to the brain via the optic nerve.

Near the center of the retina is the fovea, which is responsible for sharp central vision, such as reading or watching television. The fovea image in the environment encompasses a little more than one degree of visual angle, approximately equivalent to the width of one's thumb at arm's length (see [Figure 2.4](#)). Although the fovea is only about 1 percent of the retina in size, the neural processing associated with the fovea image engages about 50 percent of the visual cortex in the brain.

As with other sensory stimuli, light has properties such as intensity and frequency.

Frequency. Frequency is the property of light leading to the perception of color. Visible light is a small band in the electromagnetic spectrum, which ranges from

**FIGURE 2.5**

The visible spectrum of electromagnetic waves.

radio waves to x-rays and gamma rays. Different colors are positioned within the visible spectrum of electromagnetic waves, with violet at one end (390 nanometers) and red at the other (750 nm). (See Figure 2.5; colors not apparent in grayscale print).

Intensity. Although the frequency of light is a relatively simple concept, the same cannot be said for the intensity of light. Quantifying light intensity, from the human perspective, is complicated because the eye's light sensitivity varies by the wavelength of the light and also by the complexity of the source (e.g., a single frequency versus a mixture of frequencies). Related to intensity is *luminance*, which refers to the amount of light passing through a given area. With luminance comes *brightness*, a subjective property of the eye that includes perception by the brain. The unit for luminance is candela per square meter (cd/m^2).

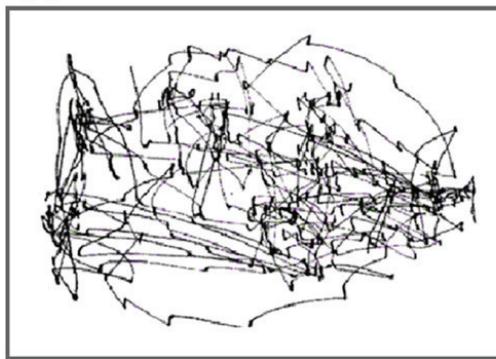
Fixations and saccades. Vision is more than the human reception of electromagnetic waves having frequency and intensity. Through the eyes, humans look at and perceive the environment. In doing so, the eyes engage in two primitive actions: *fixations* and *saccades*. During a fixation, the eyes are stationary, taking in visual detail from the environment. Fixations can be long or short, but typically last at least 200 ms. Changing the point of fixation to a new location requires a saccade—a rapid repositioning of the eyes to a new position. Saccades are inherently quick, taking only 30–120 ms. Early and influential research on fixations and saccades was presented in a 1965 publication in Russian by Yarbus, translated as *Eye Movements and Vision* (reviewed in Tatler, Wade, Kwan, Findlay, and Velichkovsky, 2010). Yarbus demonstrated a variety of inspection patterns for people viewing scenes. One example used *The Unexpected Visitor* by painter Ilya Repin (1844–1930). Participants were given instructions and asked to view the scene, shown in Figure 2.6a. Eye movements (fixations and saccades) were recorded and plotted for a variety of tasks. The results for one participant are shown in Figure 2.6b for the task “remember the position of people and objects in the room” and in Figure 2.6c for the task “estimate the ages of the people.” Yarbus provided many diagrams like this, with analyses demonstrating differences within and between participants, as well as changes in viewing patterns over time and for subsequent viewings. He noted, for example, that the similarity of inspection patterns for a single viewer was greater than the patterns between viewers.

HCI research in eye movements has several themes. One is analyzing how people read and view content on web pages. Figure 2.7 shows an example of a

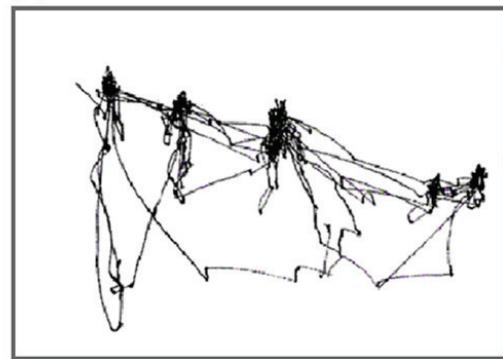
(a)



(b)

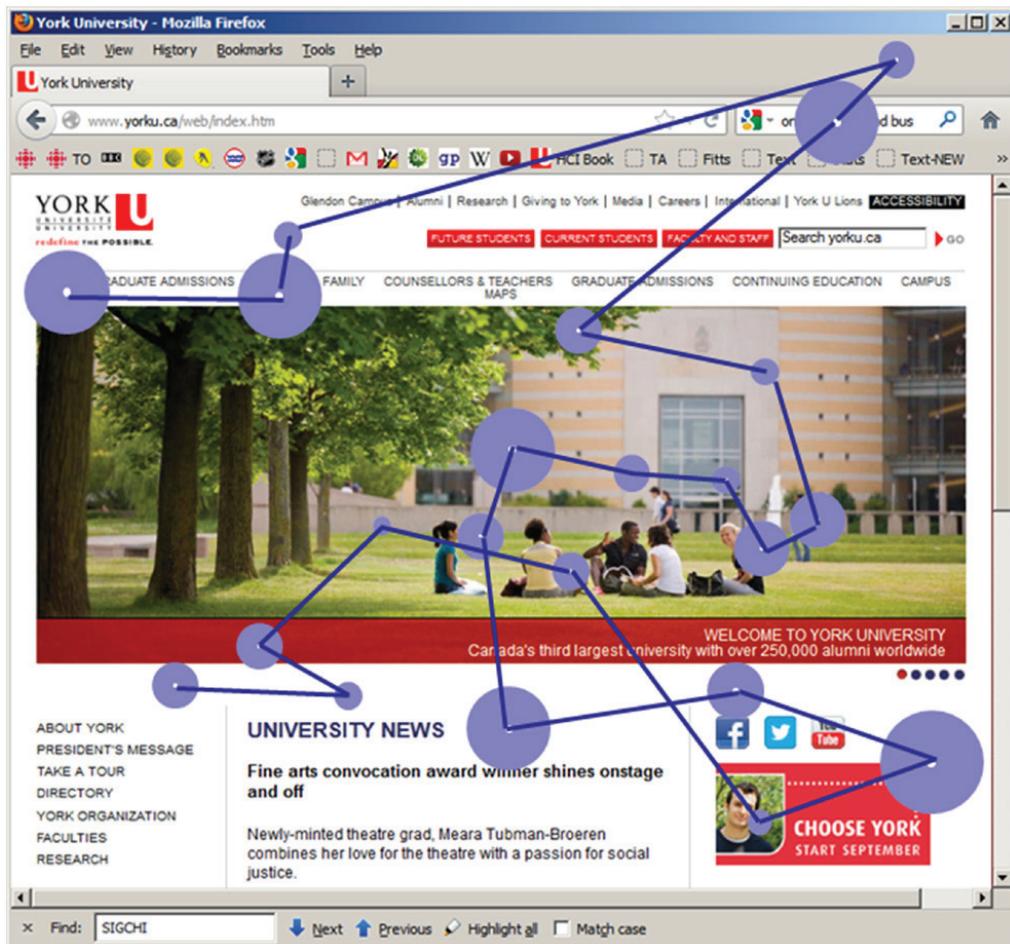


(c)

**FIGURE 2.6**

Yarbus' research on eye movements and vision (Tatler et al., 2010). (a) Scene. (b) Task: *Remember the position of the people and objects in the room.* (c) Task: *Estimate the ages of the people.*

scanpath (a sequence of fixations and saccades) for a user viewing content at different places on a page. (See also J. H. Goldberg and Helfman, 2010, Figure 2.) The results of the analyses offer implications for page design. For example, advertisers might want to know about viewing patterns and, for example, how males and females differ in viewing content. There are gender differences in eye movements

**FIGURE 2.7**

Scanpath for a user locating content on a web page.

(Pan et al., 2004), but it remains to be demonstrated how low-level experimental results can inform and guide design.

2.3.2 Hearing (Audition)

Hearing, or audition, is the detection of sound by humans. Sound is transmitted through the environment as sound waves—cyclic fluctuations of pressure in a medium such as air. Sound waves are created when physical objects are moved or vibrated, thus creating fluctuations in air pressure. Examples include plucking a string on a guitar, slamming a door, shuffling cards, or a human speaking. In the latter case, the physical object creating the sound is the larynx, or vocal cords, in the throat.

Hearing occurs when sound waves reach a human's ear and stimulate the eardrum to create nerve impulses that are sent to the brain. A single sound from a single source has at least four physical properties: intensity (loudness), frequency

(pitch), timbre, and envelope. As a simple example, consider a musical note played from an instrument such as a trumpet. The note may be loud or soft (intensity); high or low (frequency). We hear and recognize the note as coming from a trumpet, as opposed to a flute, because of the note's timbre and envelope. Let's examine each of these properties.

Loudness. Loudness is the subjective analog to the physical property of intensity. It is quantified by *sound pressure level*, which expresses the pressure in a sound wave relative to the average pressure in the medium. The unit of sound pressure level is the decibel (dB). Human hearing begins with sounds of 0–10 dB. Conversational speech is about 50–70 dB in volume. Pain sets in when humans are exposed to sounds of approximately 120–140 dB.

Pitch. Pitch is the subjective analog of frequency, which is the reciprocal of the time between peaks in a sound wave's pressure pattern. The units of pitch are cycles per second, or Hertz (Hz). Humans can perceive sounds in the frequency range of about 20 Hz–20,000 Hz (20 kHz), although the upper limit tends to decrease with age.

Timbre. Timbre (aka richness or brightness) results from the harmonic structure of sounds. Returning to the example of a musical note, harmonics are integer multiples of a note's base frequency. For example, a musical note with base frequency of 400 Hz includes harmonics at 800 Hz, 1200 Hz, 1600 Hz, and so on. The relative amplitudes of the harmonics create the subjective sense of timbre, or richness, in the sound. While the human hears the note as 400 Hz, it is the timbre that distinguishes the tone as being from a particular musical instrument. For example, if notes of the same frequency and loudness are played from a trumpet and an oboe, the two notes sound different, in part, because of the unique pattern of harmonics created by each instrument. The purest form of a note is a sine wave, which includes the base frequency but no harmonics above the base frequency. The musical notes created by a flute are close to sine waves.

Envelope. Envelope is the way a note and its harmonics build up and transition in time—from silent to audible to silent. There is considerable information in the onset envelope, or attack, of musical notes. In the example above of the trumpet and oboe playing notes of the same frequency and same loudness, the attack also assists in distinguishing the source. If the trumpet note and oboe note were recorded and played back with the attack removed, it would be surprisingly difficult to distinguish the instruments. The attack results partly from inherent properties of instruments (e.g., brass versus woodwind), but also from the way notes are articulated (e.g., staccato versus legato).

Besides physical properties, sound has other properties. These have to do with human hearing and perception. Sounds, complex sounds, can be described as being harmonious (pleasant) or discordant (unpleasant). This property has to do with how different frequencies mix together in a complex sound, such as a musical chord. Sounds may also convey a sense of urgency or speed.

Humans have two ears, but each sound has a single source. The slight difference in the physical properties of the sound as it arrives at each ear helps humans

in identifying a sound's location (direction and distance). When multiple sounds from multiple sources are heard through two ears, perceptual effects such as stereo emerge.

Sounds provide a surprisingly rich array of cues to humans, whether walking about while shopping or sitting in front of a computer typing an e-mail message. Not surprisingly, sound is crucial for blind users, for example, in conveying information about the location and distance of environmental phenomena (Talbot and Cowan, 2009).

2.3.3 Touch (Tactition)

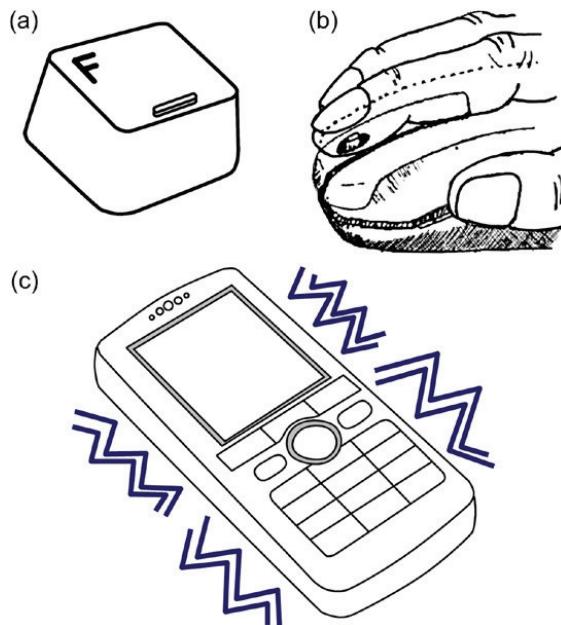
Although touch, or tactition, is considered one of the five traditional human senses, touch is just one component of the somatosensory system. This system includes sensory receptors in the skin, muscles, bones, joints, and organs that provide information on a variety of physical or environmental phenomena, including touch, temperature, pain, and body and limb position. Tactile feedback, in HCI, refers to information provided through the somatosensory system from a body part, such as a finger, when it is in contact with (touching) a physical object. Additional information, such as the temperature, shape, texture, or position of the object, or the amount of resistance, is also conveyed.

All user interfaces that involve physical contact with the user's hands (or other body parts) include tactile feedback. Simply grasping a mouse and moving it brings considerable information to the human operator: the smooth or rubbery feel of the mouse chassis, slippery or sticky movement on the desktop. Interaction with a desktop keyboard is also guided by tactile feedback. The user senses the edges and shapes of keys and experiences resistance as a key is pressed. Tactile identifiers on key tops facilitate eyes-free touch typing. Identifiers are found on the 5 key for numeric keypads and on the F and J keys for alphanumeric keyboards. Sensing the identifier informs the user that the home position is acquired. (See [Figure 2.8a](#).)

Augmenting the user experience through active tactile feedback is a common research topic. [Figure 2.8b](#) shows a mouse instrumented with a solenoid-driven pin below the index finger (Akamatsu et al., 1995). The pin is actuated (pulsed) when the mouse cursor crosses a boundary, such as the edge of a soft button or window. The added tactile feedback helps inform and guide the interaction and potentially reduces the demand on the visual channel. A common use of tactile feedback in mobile phones is vibration, signaling an incoming call or message. (See [Figure 2.8c](#).)

2.3.4 Smell and taste

Smell (olfaction) is the ability to perceive odors. For humans, this occurs through sensory cells in the nasal cavity. Taste (gustation) is a direct chemical reception of sweet, salty, bitter, and sour sensations through taste buds in the tongue and oral cavity. Flavor is a perceptual process in the brain that occurs through a partnering of the

**FIGURE 2.8**

Tactile feedback: (a) Identifier on key top. (b) Solenoid-driven pin under the index finger. (c) Vibration signals an in-coming call.

(Adapted from Akamatsu, MacKenzie, and Hasbrouq, 1995)

smell and taste senses. Although smell and taste are known intuitively by virtually all humans—and with expert-like finesse—they are less understood than the visual and auditory senses. Complex smells and tastes can be built up from simpler elements, but the perceptual processes for this remain a topic of research. For example, classification schemes have been developed for specific industries (e.g., perfume, wine) but these do not generalize to human experiences with other smells and tastes.

While humans use smell and taste all the time without effort, these senses are not generally “designed in” to systems. There are a few examples in HCI. Brewster et al. (2006) studied smell as an aid in searching digital photo albums. Users employed two tagging methods, text and smell, and then later used the tags to answer questions about the photos. Since smell has links to memory, it was conjectured that smell cues might aid in recall. In the end, recall with smell tags was poorer than with word tags. Related work is reported by Bodnar et al. (2004) who compared smell, auditory, and visual modalities for notifying users of an interruption by an incoming message. They also found poorer performance with smell. Notable in both examples, though, is the use of an empirical research methodology to explore the potential of smell in a user interface. Both studies included all the hallmarks of experimental research, including an independent variable, dependent variables, statistical significance testing, and counterbalancing of the independent variable.

2.3.5 Other senses

The word *sense* appears in many contexts apart from the five senses discussed above. We often hear of a sense of urgency, a sense of direction, musical sense, intuitive sense, moral sense, or even common sense. The value of these and related senses to HCI cannot be overstated. Although clearly operating at a higher level than the five primary senses, these additional senses encapsulate how humans feel about their interactions with computers. Satisfaction, confidence, frustration, and so on, are clearly steeped in how users feel about computing experiences. Are there receptors that pick up these senses, like cells in the nasal cavity? Perhaps. It has been argued and supported with experimental evidence that humans may have a moral sense that is like our sense of taste (Greene and Haidt, 2002). We have natural receptors that help us pick up sweetness and saltiness. In the same way, we may have natural receptors that help us recognize fairness and cruelty. Just as a few universal tastes can grow into many different cuisines, a few moral senses can grow into different moral cultures.

2.4 Responders

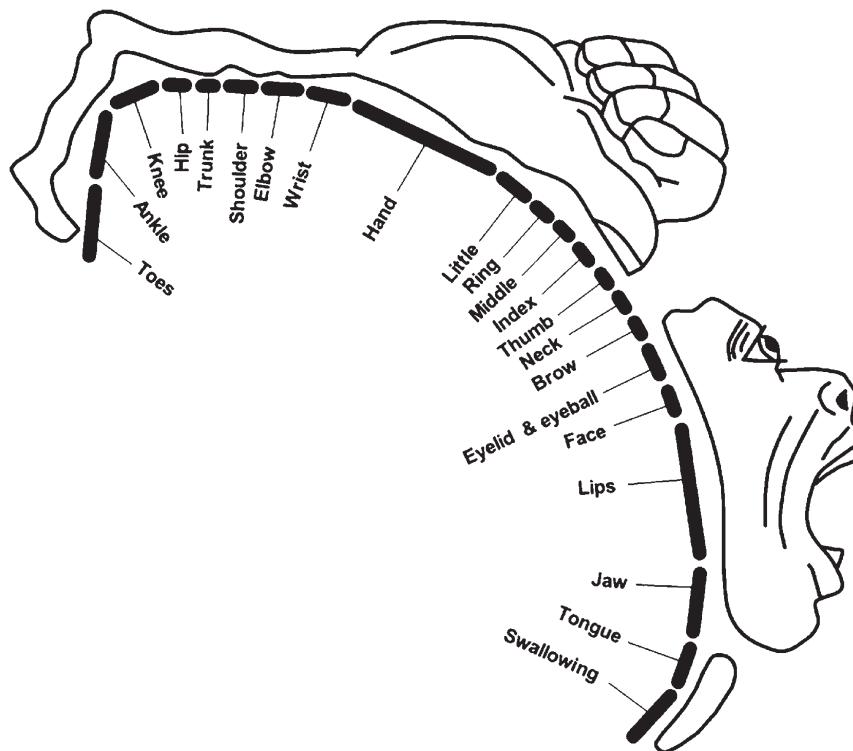
Through movement, or motor control, humans are empowered to affect the environment around them. Control occurs through *responders*. Whether using a finger to text¹ or point, the feet to walk or run, the eyebrows to frown, the vocal chords to speak, or the torso to lean, movement provides humans with the power to engage and affect the world around them. Penfield's *motor homunculus* is a classic illustration of human responders (Penfield and Rasmussen, 1990). (See Figure 2.9.) The illustration maps areas in the cerebral motor cortex to human responders. The lengths of the underlying solid bars show the relative amount of cortical area devoted to each muscle group. As the bars reveal, the muscles controlling the hand and fingers are highly represented compared to the muscles responsible for the wrist, elbow, and shoulders. Based partially on this information, Card et al. (1991) hypothesized that "those groups of muscles having a large area devoted to them are heuristically promising places to connect with input device transducers if we desire high performance," although they rightly caution that "the determinants of muscle performance are more complex than just simple cortical area" (Card et al., 1991, p. 111). (See also Balakrishnan and MacKenzie, 1997).

See also student exercise 2-1 at the end of this chapter.

2.4.1 Limbs

Human control over machines is usually associated with the limbs, particularly the upper body limbs. The same is true in HCI. With fingers, hands, and arms we

¹"Text" is now an accepted verb in English. "I'll text you after work," although strange in the 1980s, is understood today as sending a text message (SMS) on a mobile phone.

**FIGURE 2.9**

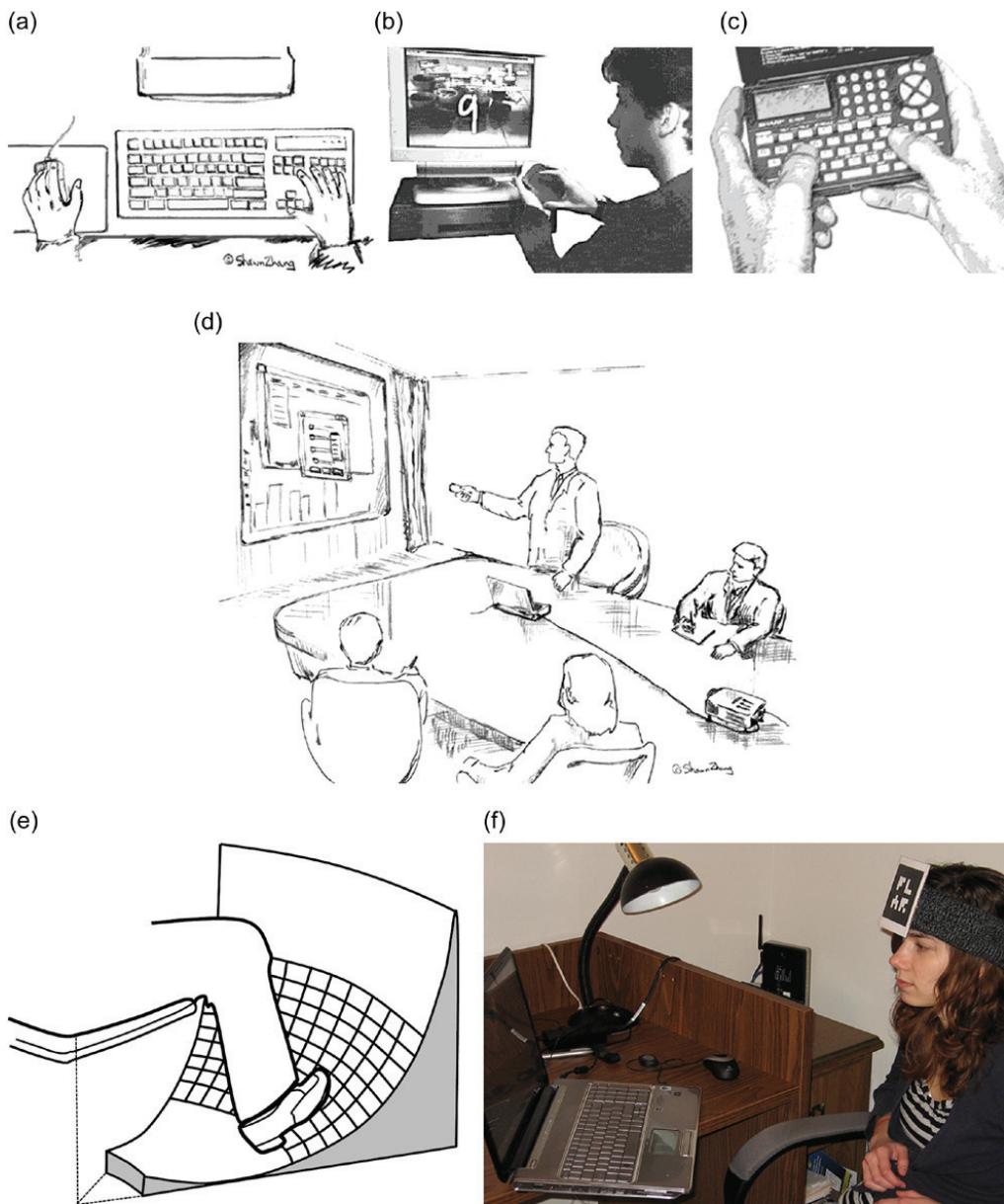
Motor homunculus showing human responders and the corresponding cortical area.

(Adapted from Penfield and Rasmussen, 1990)

type on keyboards, maneuver mice and press buttons, hold mobile phones and press keys, touch and swipe the surfaces of touchscreen phones and tablets, and wave game controllers in front of displays. Of course, legs and feet can also act as responders and provide input to a computer. For users with limited or no use of their arms, movement of the head can control an on-screen cursor. Some example scenarios are seen in [Figure 2.10](#).

Movement of the limbs is tightly coupled to the somatosensory system, particularly proprioception (Proprioception is the coordination of limb movement and position through the perception of stimuli within muscles and tendons.), to achieve accuracy and finesse as body parts move relative to the body as a whole. Grasping a mouse without looking at it and typing without looking at the keyboard are examples.

In [Figure 2.10a](#), the user's left hand grips the mouse. Presumably this user is left-handed. In [Figure 2.10b](#), the user's right index finger engages the surface of the touchpad. Presumably, this user is right-handed. Interestingly enough, handedness, or hand dominance, is not an either-or condition. Although 8 to 15 percent of people are deemed left-handed, handedness exists along a continuum, with people considered, by degree, left-handed or right-handed. Ambidextrous people are substantially indifferent in hand preference.

**FIGURE 2.10**

Use of the limbs in HCI: (a) Hands. (b) Fingers. (c) Thumbs. (d) Arms. (e) Feet. (f) Head.

(sketches a and d courtesy of Shawn Zhang; e, adapted from Pearson and Weiser, 1986)

A widely used tool to assess handedness is the Edinburgh Handedness Inventory, dating to 1971 (Oldfield, 1971). The inventory is a series of self-assessments of the degree of preference one feels toward the left or right hand in doing common tasks, such as throwing a ball. The inventory is shown in Figure 2.11

	Left	Right			
1. Writing	<input type="checkbox"/>	<input type="checkbox"/>			
2. Drawing	<input type="checkbox"/>	<input type="checkbox"/>			
3. Throwing	<input type="checkbox"/>	<input type="checkbox"/>			
4. Scissors	<input type="checkbox"/>	<input type="checkbox"/>			
5. Toothbrush	<input type="checkbox"/>	<input type="checkbox"/>			
6. Knife (without fork)	<input type="checkbox"/>	<input type="checkbox"/>			
7. Spoon	<input type="checkbox"/>	<input type="checkbox"/>			
8. Broom (upper hand)	<input type="checkbox"/>	<input type="checkbox"/>			
9. Striking a match	<input type="checkbox"/>	<input type="checkbox"/>			
10. Opening box (lid)	<input type="checkbox"/>	<input type="checkbox"/>			
Total (count checks)	<input type="text"/>	<input type="text"/>			
Difference	<input type="text"/>	<input type="text"/>	Cumulative Total	<input type="text"/>	RESULT

Instructions
Mark boxes as follows:
 preference
 strong preference
 blank no preference

Scoring
Add up the number of checks in the “Left” and “Right” columns and enter in the “Total” row for each column. Add the left total and the right total and enter in the “Cumulative Total” cell. Subtract the left total from the right total and enter in the “Difference” cell. Divide the “Difference” cell by the “Cumulative Total” cell (round to 2 digits if necessary) and multiply by 100. Enter the result in the “RESULT” cell.

Interpretation of RESULT
-100 to -40 left-handed
-40 to +40 ambidextrous
+40 to 100 right-handed

FIGURE 2.11

Edinburgh Handedness Inventory for hand dominance assessment (Oldfield, 1971).

along with the instructions, scoring, and interpretation of results.² People scoring -100 to -40 are considered left-handed, whereas those scoring $+40$ to $+100$ are considered right-handed. People scoring -40 to $+40$ are considered ambidextrous.

There are several examples in HCI where the Edinburgh Handedness Inventory was administered to participants in experiments (Hancock and Booth, 2004; Hegel, Krach, Kircher, Wrede, and Sagerer, 2008; Kabbash, MacKenzie, and Buxton, 1993; Mappus, Venkatesh, Shastry, Israeli, and Jackson, 2009; Masliah and Milgram, 2000; Matias, MacKenzie, and Buxton, 1996). In some cases, the degree of handedness is reported. For example, Hinckley et al. (1997) reported that all participants in their study were “strongly right-handed,” with a mean score of 71.7 on the inventory.

Handedness is often relevant in situations involving touch- or pressure-sensing displays. If interaction requires a stylus or finger on a display, then the user’s hand may occlude a portion of the display. Occlusion may lead to poorer performance (Forlines and Balakrishnan, 2008) or to a “hook posture” where users contort the arm position to facilitate interaction (Vogel and Balakrishnan, 2010). This can be avoided by positioning UI elements in a different region on the display (Hancock and Booth, 2004; Vogel and Baudisch, 2007). Of course, this requires sensing or determining the handedness of the user, since the occlusion is different for a left-handed user than for a right-handed user.

²Dragovic (2004) presents an updated version of the Edinburgh Inventory, using more contemporary and widely-understood tasks.

2.4.2 Voice

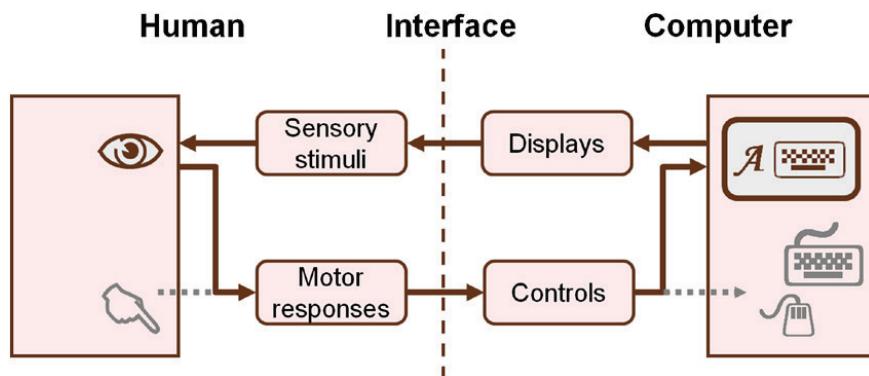
The human vocal cords are responders. Through the combination of movement in the larynx, or voice box, and pulmonary pressure in the lungs, humans can create a great variety of sounds. The most obvious form of vocalized sound—speech—is the primary channel for human communication. As an input modality, the speech must be *recognized* by algorithms implemented in software running on the host computer. With this modality, the computer interprets spoken words as though the same words were typed on the system’s keyboard. Vertanen and Kristensson (2009) describe a system for mobile text entry using automatic speech recognition. They report entry rates of 18 words per minute while seated and 13 words per minute while walking.

Computer input is also possible using non-speech vocalized sounds, a modality known as *non-verbal voice interaction* (NVVI). In this case, various acoustic parameters of the sound signal, such as pitch, volume, or timbre, are measured over time and the data stream is interpreted as an input channel. The technique is particularly useful to specify analog parameters. For example, a user could produce an utterance, such as “volume up, aaah.” In response, the system increases the volume of the television set for as long as the user sustains “aaah” (Igarashi and Hughes, 2001). Harada et al. (2006) describe the *vocal joystick*—a system using NVVI to simulate a joystick and control an on-screen cursor (e.g., “eee” = move cursor left). Applications are useful primarily for creating accessible computing for users without a manual alternative.

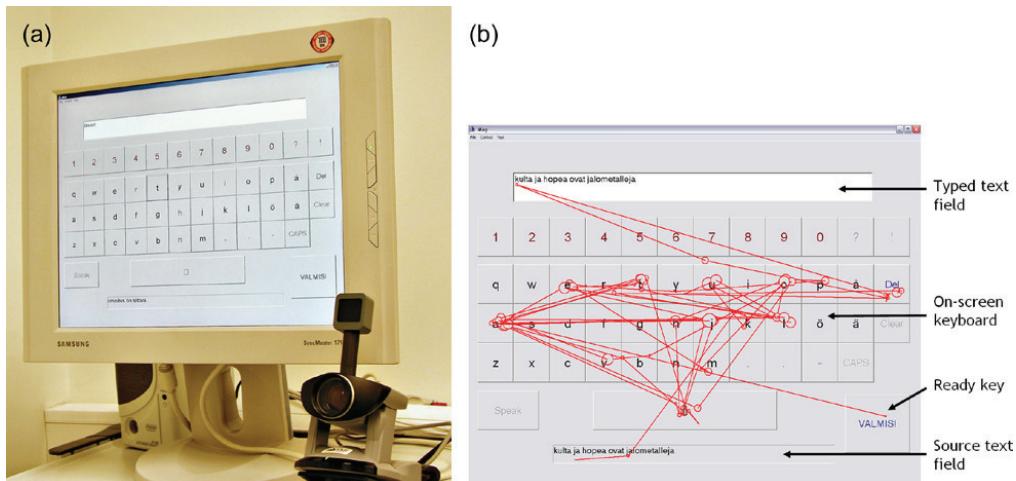
2.4.3 Eyes

In the normal course of events, the human eye receives sensory stimuli in the form of light from the environment. In viewing a scene, the eyes combine fixations, to view particular locations, and saccades, to move to different locations. This was noted earlier in considering the eye as a sensory organ. However, the eye is also capable of acting as a responder—controlling a computer through fixations and saccades. In this capacity, the eye is called upon to do double duty since it acts both as a sensor and as a responder. The idea is illustrated in [Figure 2.12](#), which shows a modified view of the human-computer interface (see [Figure 2.2](#) for comparison). The normal path from the human to the computer is altered. Instead of the hand providing motor responses to control the computer through physical devices (set in grey), the eye provides motor responses that control the computer through *soft controls*—virtual or graphical controls that appear on the system’s display.

For computer input control using the eyes, an eye tracking apparatus is required to sense and digitize the gaze location and the movement of the eyes. The eye tracker is usually configured to emulate a computer mouse. Much like point-select operations with a mouse, the eye can *look-select*, and thereby activate soft controls such as buttons, icons, links, or text (e.g., Zhang and MacKenzie, 2007). The most common method for selecting with the eye is by fixating, or dwelling, on a selectable target for a predetermined period of time, such as 750 ms.

**FIGURE 2.12**

The human-computer interface with an eye tracker. The eye serves double duty, processing sensory stimuli from computer displays and providing motor responses to control the system.

**FIGURE 2.13**

Eye typing: (a) Apparatus. (b) Example sequence of fixations and saccades (Majaranta et al., 2006).

Text entry is one application of eye tracking for input control. So-called *eye typing* uses an on-screen keyboard. The user looks at soft keys, fixating for a prescribed dwell time to make a selection. An example setup using an *iView X RED-III* eye tracking device by SensoMotoric Instruments (www.smivision.com) is shown in Figure 2.13a. Figure 2.13b shows a sequence of fixations and saccades (a scan-path) for one user while entering a phrase of text (Majaranta, MacKenzie, Aula, and Räihä, 2006). Straight lines indicate saccades. Circles indicate fixations, with the diameter indicating the duration of the fixation. Bear in mind that the fixations here are conscious, deliberate acts for controlling a computer interface. This is different

from the fixations shown in Figure 2.7, where the user was simply viewing content on a web page. In Figure 2.13b, the interaction includes numerous fixations meeting the required dwell time criterion to select soft keys. There is also a fixation (with two corresponding saccades) to view the typed text.

2.5 The brain

The brain is the most complex biological structure known. With billions of neurons, the brain provides humans with a multitude of capacities and resources, including pondering, remembering, recalling, reasoning, deciding, and communicating. While sensors (human inputs) and responders (human outputs) are nicely mirrored, it is the brain that connects them. Without sensing or experiencing the environment, the brain would have little to do. However, upon experiencing the environment through sensors, the brain's task begins.

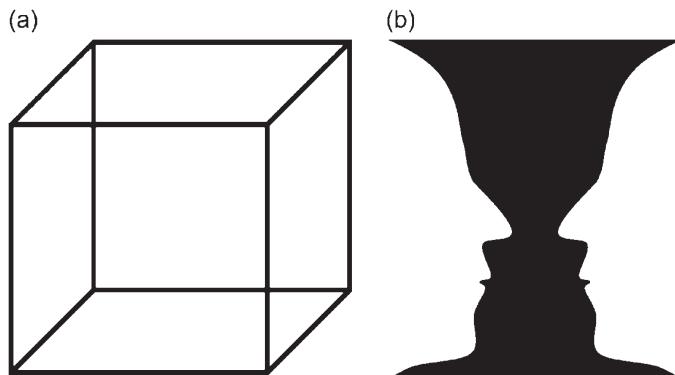
2.5.1 Perception

Perception, the first stage of processing in the brain, occurs when sensory signals are received as input from the environment. It is at the perceptual stage that associations and meanings take shape. An auditory stimulus is perceived as harmonious or discordant. A smell is pleasurable or abhorrent. A visual scene is familiar or strange. Touch something and the surface is smooth or rough, hot or cold. With associations and meaning attached to sensory input, humans are vastly superior to the machines they interact with:

People excel at perception, at creativity, at the ability to go beyond the information given, making sense of otherwise chaotic events. We often have to interpret events far beyond the information available, and our ability to do this efficiently and effortlessly, usually without even being aware that we are doing so, greatly adds to our ability to function.

(Norman, 1988, p. 136)

Since the late 19th century, perception has been studied in a specialized area of experimental psychology known as *psychophysics*. Psychophysics examines the relationship between human perception and physical phenomena. In a psychophysics experiment, a human is presented with a physical stimulus and is then asked about the sensation that was felt or perceived. The link is between a measurable property of a real-world phenomenon that stimulates a human sense and the human's subjective interpretation of the phenomenon. A common experimental goal is to measure the *just noticeable difference* (JND) in a stimulus. A human subject is presented with two stimuli, one after the other. The stimuli differ in a physical property, such as frequency or intensity, and the subject is asked if the stimuli are the same or different. The task is repeated over a series of trials with random variations in the magnitude of the difference in the physical property manipulated. Below a

**FIGURE 2.14**

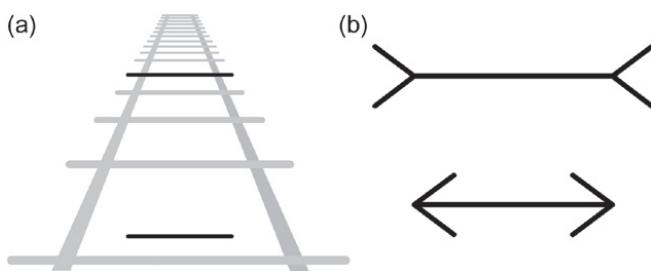
Ambiguous images: (a) Necker cube. (b) Rubin vase.

certain threshold, the difference between the two stimuli is so small that it is not perceived by the subject. This threshold is the JND. JND has been highly researched for all the human senses and in a variety of contexts. Does the JND depend on the absolute magnitude of the stimuli (e.g., high intensity stimuli versus low intensity stimuli)? Does the JND on one property (e.g., intensity) depend on the absolute value of a second property (e.g., frequency)? Does the JND depend on age, gender, or other property of the human? These are basic research questions that, on the surface, seem far afield from the sort of research likely to bear on human-computer interfaces. But over time and with new research extending results from previous research, there is indeed an application to HCI. For example, basic research in psychophysics is used in algorithms for audio compression in MP3 audio encoding.

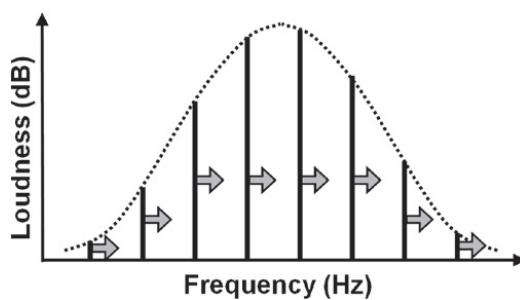
Another property of perception is ambiguity—the human ability to develop multiple interpretations of a sensory input. Ambiguous images provide a demonstration of this ability for the visual sense. Figure 2.14a shows the Necker wireframe cube. Is the top-right corner on the front surface or the back surface? Figure 2.14b shows the Rubin vase. Is the image a vase or two faces? The very fact that we sense ambiguity in these images reveals our perceptual ability to go beyond the information given.

Related to ambiguity is illusion, the deception of common sense. Figure 2.15a shows Ponzo lines. The two black lines are the same length; however, the black line near the bottom of the illustration appears shorter because of the three-dimensional perspective. Müller-Lyer arrows are shown in Figure 2.15b. In comparing the straight-line segments in the two arrows, the one in the top arrow appears longer when in fact both are the same length. Our intuition has betrayed us.

If illusions are possible in visual stimuli, it is reasonable to expect illusions in the other senses. An example of an auditory illusion is the Shepard musical scale. It is perceived by humans to rise or fall continuously, yet it somehow stays the same. A variation is a continuous musical tone known as the Shepard-Risset glissando—a tone that continually rises in pitch while also continuing to stay at the same pitch. Figure 2.16 illustrates this illusion. Each vertical line represents a sine

**FIGURE 2.15**

Visual illusion: (a) Ponzo lines. (b) Müller-Lyer arrows.

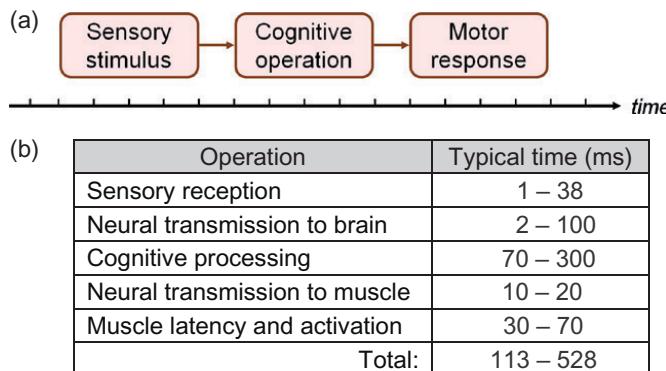
**FIGURE 2.16**

Auditory illusion. A collection of equally spaced sine waves rise in frequency. The human hears a tone that rises but stays the same.

wave. The height of each line is the perceived loudness of the sine wave. Each wave is displaced from its neighbor by the same frequency; thus, the waves are harmonics of a musical note with a base frequency equal to the displacement. This is the frequency of the single tone that a human perceives. If the sine waves collectively rise in frequency (block arrows in the figure), there is a sense that the tone is rising. Yet because the sine waves are equally spaced, there is a competing sense that the tone remains the same (because the frequency perceived is the distance between harmonics). Sine waves at the high end of the frequency distribution fade out, while new sine waves enter at the low end. Examples of the Shepard scale and the Shepard-Risset glissando can be heard on *YouTube*.

Tactile or *haptic* illusions also exist. A well-documented example is the “phantom limb.” Humans who have lost a limb through amputation often continue to sense that the limb is present and that it moves along with other body parts as it did before amputation (Halligan, Zemen, and Berger, 1999).

Beyond perception, sensory stimuli are integrated into a myriad of other experiences to yield ideas, decisions, strategies, actions, and so on. The ability to excel at these higher-level capabilities is what propels humans to the top tier in classification schemes for living organisms. By and large it is the human ability to think and reason that affords this special position.

**FIGURE 2.17**

Cognitive operation in a reaction time task: (a) Problem schematic. (b) Sequence of operations (Bailey, 1996, p. 41).

2.5.2 Cognition

Among the brain's vital faculties is cognition—the human process of conscious intellectual activity, such as thinking, reasoning, and deciding. Cognition spans many fields—from neurology to linguistics to anthropology—and, not surprisingly, there are competing views on the scope of cognition. Does cognition include social processes, or is it more narrowly concerned with deliberate goal-driven acts such as problem solving? It is beyond the reach of this book to unravel the many views of cognition. The task is altogether too great and in any case is aptly done in other references, many of them in human factors (e.g., B. H. Kantowitz and Sorkin, 1983; Salvendy, 1987; Wickens, 1987).

Sensory phenomena such as sound and light are easy to study because they exist in the physical world. Instruments abound for recording and measuring the presence and magnitude of sensory signals. Cognition occurs within the human brain, so studying cognition presents special challenges. For example, it is not possible to directly measure the time it takes for a human to make a decision. When does the measurement begin and end? Where is it measured? On what input is the human deciding? Through what output is the decision conveyed? The latter two questions speak to a sensory stimulus and a motor response that bracket the cognitive operation. Figure 2.17a illustrates this. Since sensory stimuli and motor responses are observable and measurable, the figure conveys, in a rough sense, how to measure a cognitive operation. Still, there are challenges. If the sensory stimulus is visual, the retina converts the light to neural impulses that are transmitted to the brain for perceptual processing. This takes time. So the beginning of the cognitive operation is not precisely known. Similarly, if the motor response involves a finger pressing a button, neural associations for the response are developed in the brain with nerve signals transmitted to the hand before movement begins. So the precise ending of the cognitive operation is also unknown. This sequence of events is shown in Figure 2.17b, noting the operations and the typical time for each step. The most remarkable

observation here is the wide range of values—an indication of the difficulty in pin-pointing where and how the measurements are made. Despite these challenges, techniques exist for measuring the duration of cognitive operations. These are discussed shortly.

The range of cognitive operations applicable to Figure 2.17 is substantial. While driving a car, the decision to depress a brake pedal in response to a changing signal light is simple enough. Similar scenarios abound in HCI. While using a mobile phone, one might decide to press the REJECT CALL key in response to an incoming call. While reading the morning news online, one might decide to click the CLOSE button on a popup ad. While editing a document, one might switch to e-mail in response to an audio alert of a new message. These examples involve a sensory stimulus, a cognitive operation, and a motor response, respectively.

Other decisions are more complicated. While playing the card game 21 (aka Blackjack), perhaps online³, if a card is drawn and the hand then totals 16, the decision to draw another card is likely to produce a cognitive pause. What is the chance the next card will bring the hand above 21? Which cards 6 to KING are already dealt? Clearly, the decision in this scenario goes beyond the information in the sensory stimulus. There are strategies to consider, as well as the human ability to remember and recall past events—cards previously dealt. This ability leads us to another major function of the brain—memory.

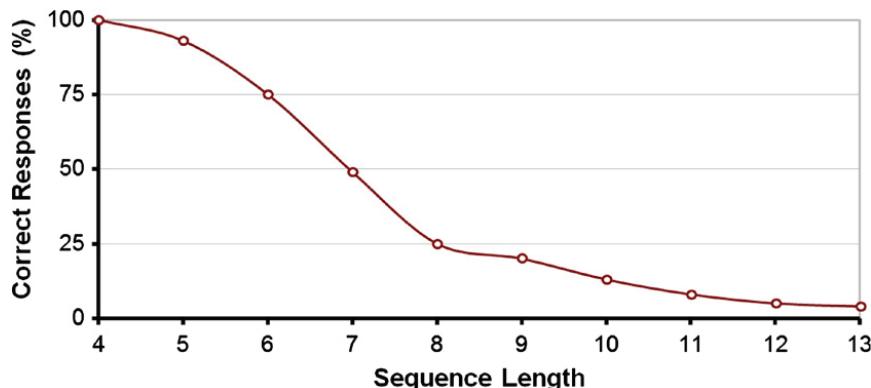
2.5.3 Memory

Memory is the human ability to store, retain, and recall information. The capacity of our memory is remarkable. Experiences, whether from a few days ago or from decades past, are collected together in the brain's vast repository known as *long-term memory*. Interestingly enough, there are similarities between memory in the brain and memory in a computer. Computer memory often includes separate areas for data and code. In the brain, memory is similarly organized. A declarative/explicit area stores information about events in time and objects in the external world. This is similar to a data space. An implicit/procedural area in the brain's memory stores information about how to use objects or how to do things. This is similar to a code space.⁴

Within long-term memory is an active area for *short-term memory* or *working memory*. The contents of working memory are active and readily available for access. The amount of such memory is small, about seven units, depending on the task and the methodology for measurement. A study of short-term memory was

³The parenthetic “perhaps online” is included as a reminder that many activities humans do in the physical world have a counterpart in computing, often on the Internet.

⁴The reader is asked to take a cautious and loose view of the analogy between human memory and computer memory. Attempts to formulate analogies from computers to humans are fraught with problems. Cognitive scientists, for example, frequently speak of human cognition in terms of operators, operands, cycles, registers, and the like, and build and test models that fit their analogies. Such *reverse anthropomorphism*, while tempting and convenient, is unlikely to reflect the true inner workings of human biology.

**FIGURE 2.18**

Results of a test of short-term memory.

published in 1956 in a classic essay by Miller, aptly titled “The Magic Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information” (G. A. Miller, 1956).⁵ Miller reviewed a large number of studies on the absolute judgment of stimuli, such as pitch in an auditory stimulus or salt concentration in water in a taste stimulus. Humans are typically able to distinguish about seven levels of a uni-dimensional stimulus.⁶

Miller extended this work to human memory, describing an experiment where participants were presented with a sequence of items and then asked to recall the items. He found that the human ability with such tasks is, similarly, about seven items (± 2). A simple demonstration of Miller’s thesis is shown in Figure 2.18. For this “mini-experiment,” log sheets were distributed to students in a class on human-computer interaction ($n \approx 60$). The instructor dictated sequences of random digits, with sequences varying in length from four digits to 13 digits. After each dictation, students copied the sequence from short-term memory onto the log sheet. The percentage of correct responses by sequence length is shown in the figure. At length seven the number of correct responses was about 50 percent. At lengths five and nine the values were about 90 percent and 20 percent, respectively.⁷ See also student exercise 2-2 at the end of this chapter.

⁵Miller’s classic work is referred to as an *essay* rather than a *research paper*. The essay is casual in style and, consequently, written in the first person; for example, “I am simply pointing to the obvious fact that...” (G. A. Miller, 1956, p. 93). Research papers, on the other hand, are generally plain in style and avoid first-person narratives (cf. “This points to the fact that...”).

⁶The human ability to distinguish levels is greater if the stimulus is multidimensional; that is, the stimulus contains two or more independent attributes, such as a sound that varies in pitch and intensity.

⁷A response was deemed correct only if all the items were correctly recalled. For the longer sequences, many responses were “mostly correct.” For example, at sequence length = 7, many of the responses had five or six items correct.

Miller extended his work by revealing and analyzing a simple but powerful process within the brain: our ability to associate multiple items as one. So-called *chunking* is a process whereby humans group a series of low-level items into a single high-level item. He described an example using binary digits. For example, a series of 16 bits, such as 1000101101110010, would be extremely difficult to commit to memory. If, however, the bits are collected into groups of four and chunked into decimal digits, the pattern is much easier to remember: 1000101101110010→1000, 1011, 0111, 0010→8, 11, 7, 2. Card et al. (1983, 36) give the example of BSCBMICRA. At nine units, the letter sequence is beyond the ability of most people to repeat back. But the sequence is similar to the following three groups of three-letter sequences: CBS IBM RCA. Shown like this, the sequence contains three chunks and is relatively easy to remember provided the person can perform the recoding rapidly enough. The process of chunking is mostly informal and unstructured. Humans intuitively build up chunked structures recursively and hierarchically, leading to complex organizations of memory in the brain.

2.6 Language

Language—the mental faculty that allows humans to communicate—is universally available to virtually all humans. Remarkably, language as speech is available without effort. Children learn to speak and understand speech without conscious effort as they grow and develop. Writing, as a codification of language, is a much more recent phenomenon. Learning to write demands effort, considerable effort, spanning years of study and practice. Daniels and Bright distinguish language and writing as follows: “Humankind is defined by language; but civilization is defined by writing.” (Daniels and Bright, 1996, p. 1). These words are a reminder that the cultural and technological status associated with civilization is enabled by systems of writing. Indeed, the term *prehistory*, as applied to humans, dates from the arrival of human-like beings, millions of years ago, to the emergence of writing. It is writing that presaged *recorded history*, beginning a mere six thousand years ago.

In HCI, our interest in language is primarily in systems of writing and in the technology that enables communication in a written form. *Text* is the written material on a page or display. How it gets there is a topic that intrigues and challenges HCI researchers, as well as the engineers and designers who create products that support text creation, or text entry. Although text entry is hugely important in HCI, our interest here is language itself in a written form.

One way to characterize and study a language in its written form is through a corpus—a large collection of text samples gathered from diverse and representative sources such as newspapers, books, e-mails, and magazines. Of course, it is not possible for a corpus to broadly yet precisely represent a language. The sampling process brings limitations: During what timeframe were the samples written? In what country? In what region of the country? On what topics are the samples focused and who wrote them? A well-known corpus is the British National Corpus

Word Rank	English	French	German	Finnish	SMS English	SMS Pinyin
1	the	de	der	ja	u	wo (我)
2	of	la	die	on	i	ni (你)
3	and	et	und	ei	to	le (了)
4	a	le	in	että	me	de (的)
5	in	à	den	oli	at	bu (不)
...
1000	top	ceci	konkurrenz	muista	ps	jiu (舅)
1001	truth	mari	stieg	paikalla	quit	tie (贴)
1002	balance	solution	notwendig	varaa	rice	ji (即)
1003	heard	expliquer	sogenannte	vie	sailing	jiao (角)
1004	speech	pluie	fahren	seuran	sale	ku (裤)
...

FIGURE 2.19

Sample words from word-frequency lists in various languages.

(BNC), which includes samples totaling 100 million words.⁸ The sources are written in British English and are from the late 20th century. So analyses gleaned from the BNC, while generally applicable to English, may not precisely apply, for example, to American English, to present day English, or to the language of teenagers sending text messages.

To facilitate study and analysis, a corpus is sometimes reduced to a word-frequency list, which tabulates unique words and their frequencies in the corpus. One such reduction of the BNC includes about 64,000 unique words with frequencies totaling 90 million (Silfverberg, MacKenzie, and Korhonen, 2000). Only words occurring three or more times in the original corpus are included. The most frequent word is *the*, representing about 6.8 percent of all words.

Figure 2.19 includes excerpts from several corpora, showing the five most frequently used words and the words ranked from 1000 to 1004. The English entries are from the British National Corpus. There are additional columns for French (New, Pallier, Brysbaert, and Ferrand, 2004), German (Sporka et al., 2011), Finnish, SMS English, and SMS Pinyin (Y. Liu and Räihä, 2010). The Finnish entries are from a database of text from a popular newspaper in Finland, *Turun Sanomat*. The SMS English entries are from a collection of about 10,000 text messages, mostly from students at the University of Singapore.⁹ SMS text messaging is a good example of the dynamic and context-sensitive nature of language. Efforts to characterize SMS English are prone to the limitations noted above. Note that there is no overlap in the entries 1–5 under English and SMS English.

The right-hand column in Figure 2.19 is for SMS Pinyin. Pinyin has been the standard coding system since 1958, using the Latin alphabet for Mandarin Chinese characters. The entries are pinyin marks, not words. Each mark maps to the Chinese

⁸See www.natcorp.ox.ac.uk.

⁹Available at www.comp.nus.edu.sg/~rpnlpir/smsCorpus.

(a)	Th std ws flld wth th rch dr f rss, nd whn th lght smmr wnd strrd mdst th trs f th grdn, thr cm thrgh th pn dr th hvy scnt f th llc, r th mr dlct prfm f th pnk-flwrng thrn.
(b)	Th std ws flld wth th rch odr of rss, and whn th lght smmr wnd strrd amdst th trs of th grdn, thr cm thrgh th opn dr th hvy scnt of th llc, or th mr dlct prfm of th pnk-flwrng thrn.
(c)	The studio was filled with the rich odour of roses, and when the light summer wind stirred amidst the trees of the garden, there came through the open door the heavy scent of the lilac, or the more delicate perfume of the pink-flowering thorn.

FIGURE 2.20

First paragraph of Oscar Wilde's *The Picture of Dorian Gray*: (a) Vowels removed.
(b) Vowels intact at beginning of words. (c) Original.

character shown in parentheses. The entries are from a corpus of 630,000 text messages containing over nine million Chinese characters.

A notable feature of some corpora is part-of-speech (POS) tagging, where words are tagged by their category, such as noun, verb, and adjective. Importantly, the part of speech is contextual, reflecting a word's use in the original text. For example, *paint* is sometimes a verb (*Children paint with passion*), sometimes a noun (*The paint is dry*). POS tagging can be important in predictive systems where knowing a word's POS limits the possibilities for the next word (Gong, Tarasewich, and MacKenzie, 2008).

2.6.1 Redundancy in language

Native speakers of a language innately possess an immense understanding of the statistics of the language. We automatically insert words that are omitted or obscured (*ham and ____ sandwich*). We anticipate words (*a picture is worth a thousand ____*), letters (*questio_*), or entire phrases (*to be or ____*). We might wonder: since humans can fill in missing letters or words, perhaps the unneeded portions can be omitted. Let's consider this further. The example in Figure 2.20 gives three variations of a paragraph of text. The original excerpt contains 243 characters. In part (a), all 71 vowels are removed, thus shortening the text by 29.2 percent. Many words are easily guessed (e.g., *smmr*→summer, *thrgh*→through) and with some effort the gist of the text is apparent. It has something to do with summer [*smmr*], gardens [*grdn*], and scent [*scnt*]. Part (b) is similar except the first letter of each word is intact, even if it is a vowel. Still, 62 vowels are missing. The meaning

(a)	My smmr hols wr CWOT. B4, we used 2go2 NY 2C my bro, his GF & thr 3 :- kids FTF. ILNY, it's a gr8 plc.	(b)	My summer holidays were a complete waste of time. Before, we used to go to New York to see my brother, his girlfriend and their three screaming kids face to face. I love New York. It's a great place.
-----	--	-----	---

FIGURE 2.21

Shortening English: (a) SMS shorthand. (b) Standard English.

is slightly easier to decipher. The original text is given in (c). It is the first paragraph from Oscar Wilde's *The Picture of Dorian Gray*.

There are other examples, as above, where portions of text are removed, yet comprehension remains. SMS text messaging is a well-documented example. In addition to removing characters, recoding is often used. There are numerous techniques employed, such as using sound (th@s→that's, gr8→great) or invented acronyms (w→with, gf→girlfriend, x→times) (Grinter and Eldridge, 2003). One anecdote tells of a 13-year-old student who submitted an entire essay written in SMS shorthand.¹⁰ Although the teacher was not impressed, the student's rationale was direct and honest: it is easier to write in shorthand than in standard English. An example from the essay is shown in Figure 2.21. Part (a) gives the shortened text. There are 26 words and 102 characters (including spaces). The expanded text in (b) contains 39 words and 199 characters. The reduction is dramatic: 48.7 percent fewer characters in the SMS shorthand. Of course, there are differences between this example and Figure 2.20. For instance, in this example, punctuation and digits are introduced for recoding. As well, the shortened message is tailored to the language of a particular community of users. It is likely the 13-year-old's teacher was not of that community.

There is, unfortunately, a more insidious side to redundancy in written text. A common fault in writing is the presence of superfluous words, with their eradication promoted in many books on writing style. Strunk and White's Rule 17 is Omit Needless Words, and advises reducing, for example, "he is a man who" to "he," or "this is a subject that" to "this subject" (Strunk and White, 2000, p. 23). Tips on writing style are given in Chapter 8.

2.6.2 Entropy in language

If redundancy in language is what we inherently know, entropy is what we don't know—the uncertainty about forthcoming letters, words, phrases, ideas, concepts, and so on. Clearly, redundancy and entropy are related: If we remove what we know, what remains is what we don't know. A demonstration of redundancy and entropy in written English was provided in the 1950s by Shannon in a letter-guessing experiment (Shannon, 1951). (See Figure 2.22.) The experiment proceeds as follows. The participant is asked to guess the letters in a phrase, starting at the

¹⁰news.bbc.co.uk/2/hi/uk_news/2814235.stm.

THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG
-----ROO-----NOT-V-----I-----SM---OB----
READING LAMP ON THE DESK SHED GLOW ON
REA-----O-----D---SHED-GLO--O-
POLISHED WOOD BUT LESS ON THE SHABBY RED CARPET
P-L-S-----O---BU--L-S--O-----SH---RE--C-----

FIGURE 2.22

Shannon's letter-guessing experiment.

(Adapted from Shannon, 1951)

beginning. As guessing proceeds, the phrase is revealed to the participant, letter by letter. The results are recorded as shown in the line below each phrase in the figure. A dash ("—") is a correct guess; a letter is an incorrect guess. Shannon called the second line the “reduced text.” In terms of redundancy and entropy, a dash represents redundancy (what is known), while a letter represents entropy (what is not known). Among the interesting observations in Figure 2.22 is that errors are more common at the beginning of words, less common as words progress. The statistical nature of the language and the participant’s inherent understanding of the language facilitate guessing within words.

The letter-guessing experiment in Figure 2.22 is more than a curiosity. Shannon was motivated to quantify the entropy of English in information-theoretic terms. He pointed out, for example, that both lines in each phrase-pair contain the same information in that it is possible, with a good statistical model, to recover the first line from the second. Because of the redundancy in printed English (viz. the dashes), a communications system need only transmit the reduced text. The original text can be recovered using the statistical model. Shannon also demonstrated how to compute the entropy of printed English. Considering letter frequencies alone, the entropy is about 4.25 bits per letter.¹¹ Considering previous letters, the entropy is reduced because there is less uncertainty about forthcoming letters. Considering long range statistical effects (up to 100 letters), Shannon estimated the entropy of printed English at about one bit per letter with a corresponding redundancy of about 75 percent.

See also student exercise 2-2 at the end of this chapter.

2.7 Human performance

Humans use their sensors, brain, and responders to do things. When the three elements work together to achieve a goal, human performance arises. Whether the

¹¹The data set and calculation are given in Chapter 7 (see Figure 7.19).

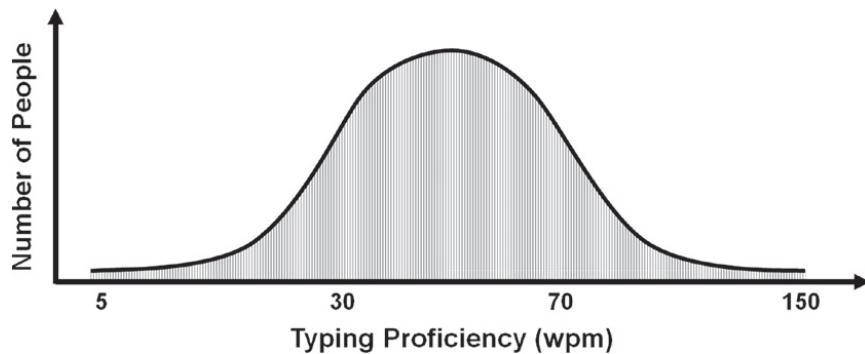


FIGURE 2.23

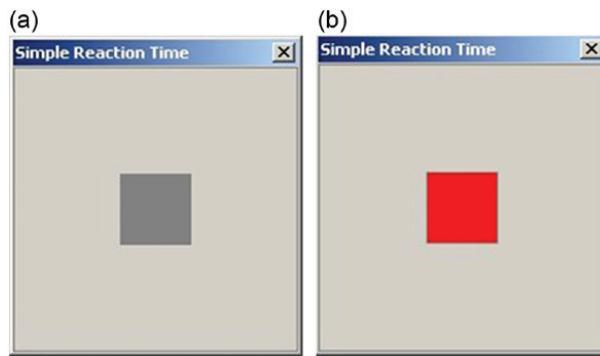
Variability of people in performing a task such as typing.

action is tying shoelaces, folding clothes, searching the Internet, or entering a text message on a mobile phone, human performance is present. Better performance is typically associated with faster or more accurate behavior, and this leads to a fundamental property of human performance—the *speed-accuracy trade-off*: go faster and errors increase; slow down and accuracy improves. Reported in academic papers dating back more than a century (see Swensson, 1972, for a review), mundane and proverbial (“Haste makes waste”), and steeped in common sense (we instinctively slow down to avoid errors), it is hard to imagine a more banal feature of human performance. Clearly, research on a new interface or interaction technique that seeks to determine the speed in doing a task must consider accuracy as well.

Humans position themselves on the speed-accuracy trade-off in a manner that is both comfortable and consistent with their goals. Sometimes we act with haste, even recklessly; at other times we act with great attention to detail. Furthermore, we may act in the presence of a secondary task, such as listening to the radio, conversing with a friend, or driving a car. Clearly, context plays an important role, as do the limits and capabilities of the sensors, the brain, and the responders.

With human performance, we begin to see complexities and challenges in HCI that are absent in traditional sciences such as physics and chemistry. Humans bring diversity and variability, and these characteristics bring imprecision and uncertainty. Some humans perform tasks better than others. As well, a particular human may perform a task better in one context and environment than when performing the same task in a different context and environment. Furthermore, if that same human performs the same task repeatedly in the same context and environment, the outcome will likely vary.

Human diversity in performing tasks is sometimes illustrated in a distribution, as in Figure 2.23. Here the distribution reveals the number of people performing a task (y-axis) versus their proficiency in doing it (x-axis). The example assumes computer users as the population and illustrates typing on a conventional computer keyboard as the task. Most people fall somewhere in the middle of the distribution.

**FIGURE 2.24**

Simple reaction time: (a) The user fixates on the grey box. (b) After a delay, the box turns red whereupon the user presses a key as quickly as possible.

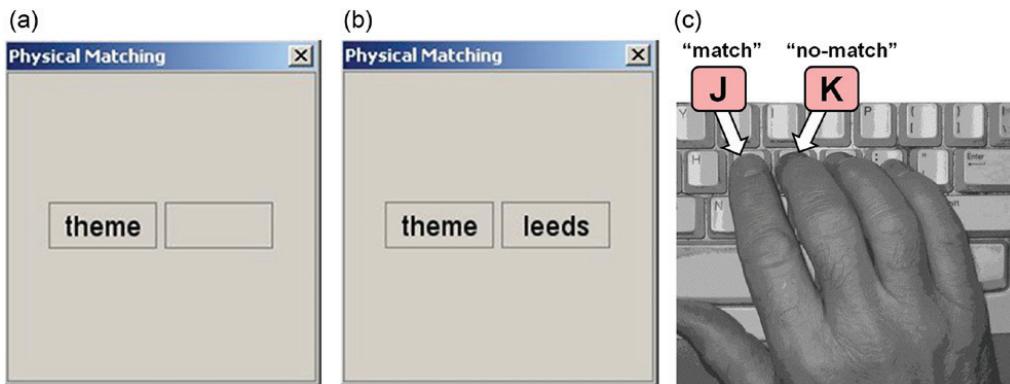
Typing speeds here are in the range of, say, 30–70 words per minute. Some people are slower, some faster. However, a small number of people will be exceedingly fast, say, 150 words per minute or faster. Yet others, also a small number, exhibit difficulty in achieving even a modest speed, such as 5 words per minute, equivalent to one word every 12 seconds.

2.7.1 Reaction time

One of the most primitive manifestations of human performance is *simple reaction time*, defined as the delay between the occurrence of a single fixed stimulus and the initiation of a response assigned to it (Fitts and Posner, 1968, p. 95). An example is pressing a button in response to the onset of a stimulus light. The task involves the three elements of the human shown in Figure 2.17. The cognitive operation is trivial, so the task is relatively easy to study. While the apparatus in experimental settings is usually simple, humans react to more complex apparatus all the time, in everyday pursuits and in a variety of contexts, such as reacting to the ring of a phone, to a traffic light, or to water in a bath (hot!). These three examples all involve a motor response. But the sensory stimuli differ. The ring of a phone is an auditory stimulus; a changing traffic light is a visual stimulus; hot water touching the skin is a tactile stimulus. It is known that simple reaction times differ according to the stimulus source, with approximate values of 150ms (auditory), 200ms (visual), 300ms (smell), and 700ms (pain) (Bailey, 1996, p. 41).

To explore reaction times further, a Java-based application was developed to experimentally test and demonstrate several reaction time tasks.¹² (See also Appendix A.) After describing each task, the results of an experiment are presented. For *simple reaction*, the interface is shown in Figure 2.24. A trial begins with the

¹²The software, a detailed API, and related files are in ReactionTimeExperiment.zip, available on this book's website.

**FIGURE 2.25**

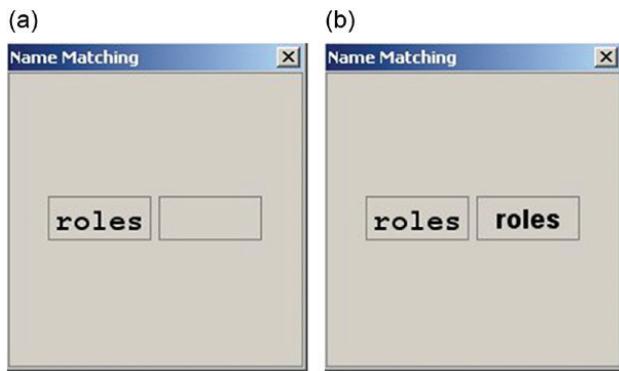
Physical matching: (a) Initial stimulus. (b) After a delay, a second stimulus appears. (c) Setup.

appearance of a grey box in a GUI window. Following a delay, the box turns red (color is not apparent in grayscale print). This is the sensory stimulus. The user’s goal is to press a key on the system keyboard as quickly as possible after the stimulus appears. The delay between the grey box appearing and the box turning red is randomized to prevent the user from anticipating the onset of the stimulus.

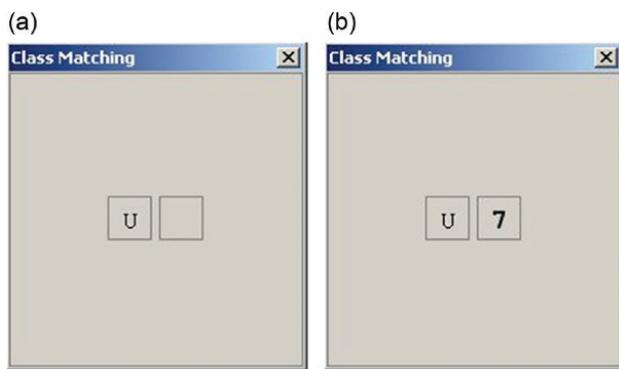
The software implements three extensions of simple reaction tasks: physical matching, name matching, and class matching. Each adds a layer of complexity to the cognitive operation. The tasks were modeled after descriptions by Card et al. (1983, 65–71). For *physical matching*, the user is presented with a five-letter word as an initial stimulus. After a delay a second stimulus appears, also a five-letter word. The user responds as quickly as possible by pressing one of two keys: a “match” key if the second stimulus matches the first stimulus, or a “no-match” key if the second stimulus differs from the first stimulus. Matches occur with 50 percent probability. An example experimental setup is shown in Figure 2.25.

Obviously, physical matching is more complicated than simple reaction, since the user must compare the stimulus to a code stored in working memory. There are many examples of similar tasks in HCI, such as entering text on a mobile phone using predictive input (T9). While entering a word, the user has in her or his mind an intended word. This is the initial stimulus. With the last keystroke, the system presents a word. This is the second stimulus. If the presented word matches the intended word, the user presses 0 to accept the word. If the presented word does not match the intended word, the user presses * to retrieve the next alternative word matching the key sequence. (Details vary depending on the phone.)

Name matching is the same as physical matching except the words vary in appearance: uppercase or lowercase, mono-spaced or sans serif, plain or bold, 18 point or 20 point. A match is deemed to occur if the words are the same, regardless of the look of the fonts. See Figure 2.26. Name matching should take longer than physical matching because “the user must now wait until the visual code has been

**FIGURE 2.26**

Name matching: (a) Initial stimulus. (b) Second stimulus.

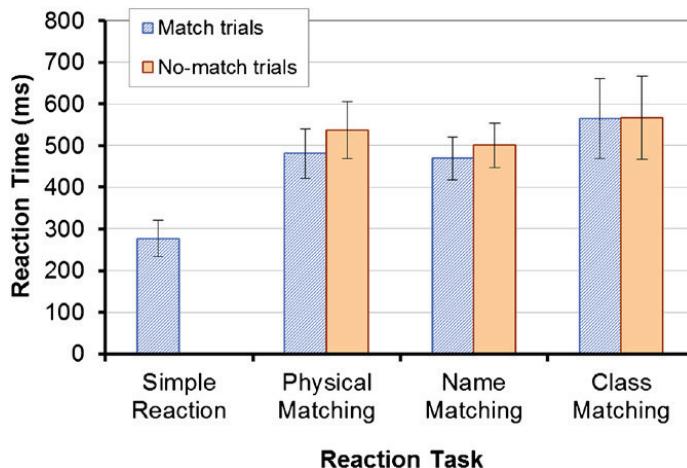
**FIGURE 2.27**

Class matching: (a) Initial stimulus. (b) Second stimulus.

recognized and an abstract code representing the name of the letter is available” (Card et al., 1983, p. 69).

For *class matching*, the initial stimulus contains a letter or digit. After a delay a second stimulus appears, also containing a letter or digit. The font is mono-spaced or sans serif, plain or italic, 18 point or 20 point. A match is deemed to occur if both symbols are of the same class; that is, both are letters or both are digits. Class matching takes longer still, because “the user has to make multiple references to long-term memory” (Card et al., 1983, p. 70). To avoid confusion, 0 (digit) and O (letter) are not included, nor are 1 (digit) and I (letter). (See Figure 2.27.)

The interfaces described above were tested in the lab component of a course on HCI. Fourteen students served as participants and performed three blocks of ten trials for each condition. The first block was considered practice and was discarded. To offset learning effects, participants were divided into two groups of equal size. One group

**FIGURE 2.28**

Results of an experiment comparing several reaction tasks. Error bars show ± 1 SD.

performed the simple reaction task first, followed in order by the physical, name, and class matching tasks. The other group performed the tasks in the reverse order.

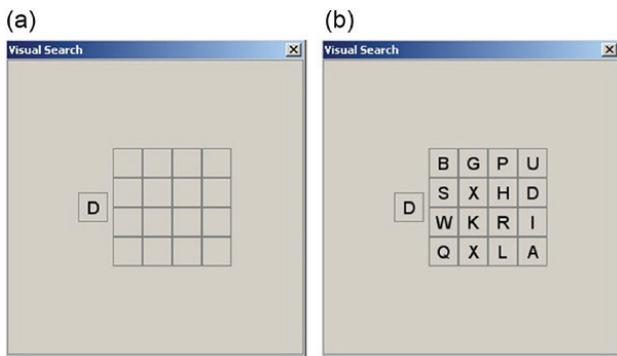
The results are shown in Figure 2.28. The mean time for simple reaction was 276 ms. This value is nicely positioned in the 113 to 528 ms range noted earlier for reaction time tasks (see Figure 2.17). Note that the time measurement began with the arrival of the second stimulus and ended with the key event registered in the software when a key was pressed; thus, the measurement includes the time for the motor response.

Physical matching took about twice as long as simple reaction, depending on whether the second stimulus was a match (482 ms) or a no-match (538 ms). Interestingly enough, name matching did not take longer than physical matching. One explanation is that the words in the name-matching task had insufficient variability in appearance to require additional cognitive processing. Class matching was the hardest of the tasks, with means of about 565 ms for both the match and no-match conditions.

Choice reaction is yet another type of reaction time task. In this case, the user has n stimuli, such as lights, and n responders, such as switches. There is a one for one correspondence between stimulus and response. Choice reaction time is discussed in Chapter 7 on modeling.

2.7.2 Visual search

A variation on reaction time is *visual search*. Here, the user scans a collection of items, searching for a desired item. Obviously, the time increases with the number of items to scan. The software described above includes a mode for visual search, with the search space configurable for 1, 2, 4, 8, 16, or 32 items. An example for $N=16$ is shown in Figure 2.29. The initial stimulus is a single letter. After a random

**FIGURE 2.29**

Visual search: (a) Initial stimulus. (b) After a delay a collection of letters appears.

delay of two to five seconds, the squares on the right are populated with letters selected at random. The initial stimulus appears on the right with 50 percent probability. The user presses a “match” or “no-match” key, as appropriate.

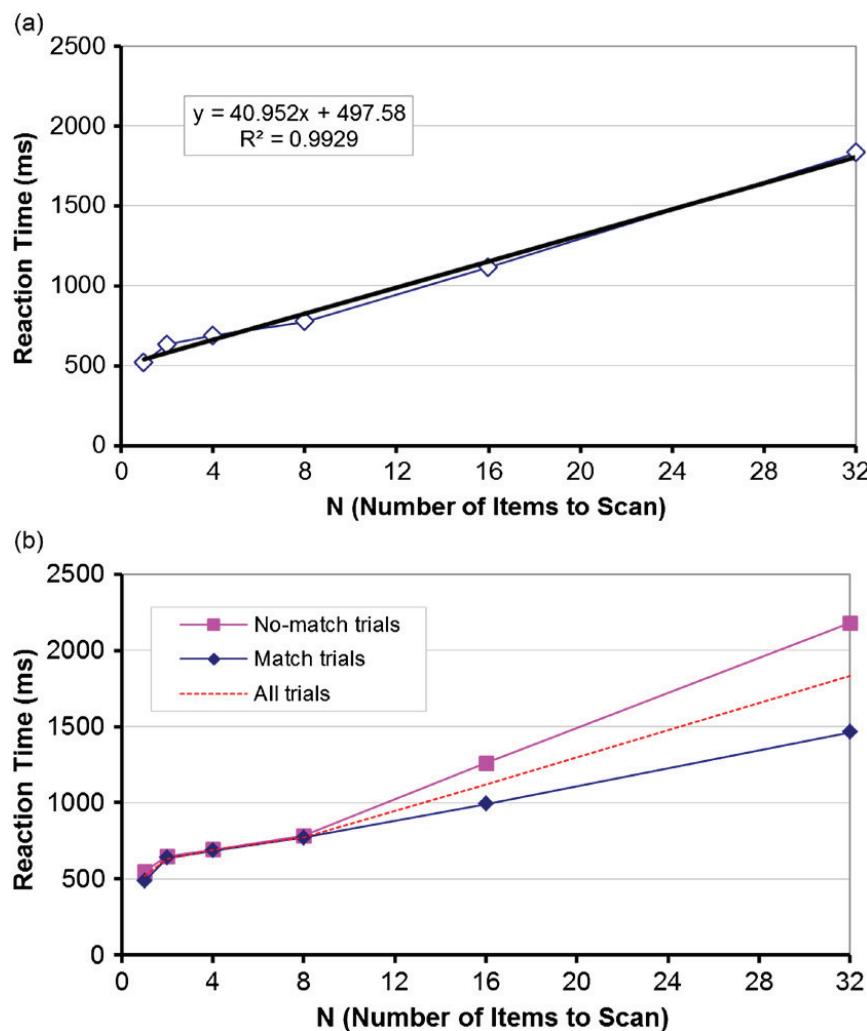
A small experiment was conducted with the same 14 students from the experiment described above, using a similar procedure. The results are shown in Figure 2.30 in two forms. In (a), reaction time (RT) versus number of items (N) is plotted. Each marker reveals the mean of $14 \times (10 + 10) = 280$ trials. The markers are connected and a linear regression line is superimposed. At $R^2 = .9929$, the regression model is an excellent fit. Clearly, there is a linear relationship between reaction time in a visual search task and the number of items to scan. This is well known in the HCI literature, particularly from research on menu selection (e.g., Cockburn, Gutwin, and Greenberg, 2007; Hornof and Kieras, 1997; Landauer and Nachbar, 1985). For this experiment,

$$RT = 498 + 41 \times N \text{ ms} \quad (1)$$

$N=1$ is a special case since there is only one item to scan. This reduces the task to physical matching. The task is slightly different than in the physical matching experiment, since the user is matching a letter rather than a word. Nevertheless, the result is consistent with the physical matching result in Figure 2.28 ($RT \approx 500$ ms).

In Figure 2.30b, the results are given separately for the match trials and the no-match trials. The no-match trials take longer. The reason is simple. If the initial stimulus is not present, an exhaustive search is required to determine such before pressing the no-match key. If the initial stimulus is present, the user presses the match key immediately when the initial stimulus is located in the right-side stimuli. The effect only surfaces at $N=16$ and $N=32$, however.

Before moving on, here is an interesting reaction time situation, and it bears directly on the title of this section, Human Performance. Consider an athlete competing in the 100 meter dash in the Olympics. Sometimes at the beginning of a race there is a “false start.” The definition of a false start is rather interesting: a false start occurs if an athlete reacts to the starter’s pistol before it is sounded *or within*

**FIGURE 2.30**

Results of visual search experiment: (a) Overall result with linear regression model. (b) Results by match and no-match trials.

*100ms after.*¹³ Clearly, an athlete who reacts before the starter's pistol sounds is anticipating, not reacting. Interesting in the definition, however, is the criterion that a false start has occurred if the athlete reacts within 100ms *after* the starter's pistol is sounded. One hundred milliseconds is precariously close to the lower bound on reaction time, which is cited in Figure 2.17 as 113ms. Card et al. peg the lower bound at 105ms (Card et al., 1983, p. 66). World records are set, and gold medals won, by humans at the extreme tails of the normal distribution. Is it possible that

¹³Rule 161.2 of the International Association of Athletics Federations (IAAF) deems a false start to occur “when the reaction time is less than 100/1000ths of a second.” See www.iaaf.org/mm/Document/imported/42192.pdf (107).

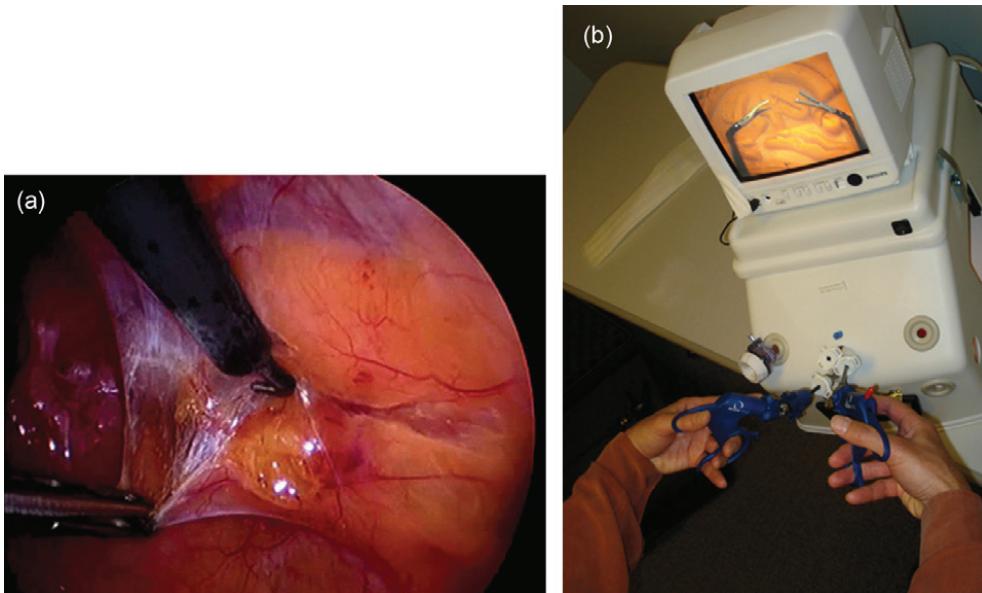
a false start is declared occasionally, very occasionally, when none occurred (e.g., honestly *reacting* 95 ms after the starter's pistol is fired)? There are slight differences between the lower-bound reaction times cited above and the false-start scenario, however. The values cited are for pressing a key with a finger in response to a visual stimulus. The motor response signals in the 100 meter dash must travel farther to reach the feet. This tends to lengthen the reaction time. Also, the stimulus in the 100 meter dash is auditory, not visual. Auditory reaction time is less than visual reaction time, so this tends to shorten the reaction time. Nevertheless, the example illustrates the application of low-level research in experimental psychology to human performance and to the design of human-machine systems.

2.7.3 Skilled behavior

The response time tasks in the previous section are simple: a sensory stimulus initiates a simple cognitive operation, which is followed by a simple motor response. It takes just a few trials to get comfortable with the task and with additional practice there is little if any improvement in performance. However, in many tasks, human performance improves considerably and continuously with practice. For such tasks, the phenomenon of learning and improving is so pronounced that the most endearing property of the task is the progression in performance and the level of performance achieved, according to a criterion such as speed, accuracy, degree of success, and so on. *Skilled behavior*, then, is a property of human behavior whereby human performance necessarily improves through practice. Examples include playing darts, playing chess and, in computing scenarios, gaming or programming. One's ability to do these tasks is likely to bear significantly on the amount of practice done.

The examples just cited were chosen for a reason. They delineate two categories of skilled behavior: sensory-motor skill and mental skill (Welford, 1968, p. 21). Proficiency in darts or gaming is likely to emphasize sensory-motor skill, while proficiency in chess or computer programming is likely to emphasize mental skill. Of course, there is no dichotomy. All skilled behavior requires mental faculties, such as perception, decision, and judgment. Similarly, even the most contemplative of skilled tasks requires coordinated, overt action by the hands or other organs.

While tasks such as gaming and computer programming may focus on sensory-motor skill or mental skill, respectively, other tasks involve considerable elements of both. Consider a physician performing minimally invasive surgery, as is common for abdominal procedures. To access the abdominal area, a camera and a light mounted at the end of a laparoscope are inserted through a small incision, with the image displayed on an overhead monitor. Tools are inserted through other incisions for convenient access to an internal organ. The surgeon views the monitor and manipulates the tools to grasp and cut tissue. In [Figure 2.31a](#), the tips of the surgeon's tools for grasping (left) and cutting (top) are shown as they appear on a monitor during a cholecystectomy, or gallbladder removal. The tools are manually operated, external to the patient. [Figure 2.31b](#) shows examples of such tools in a training simulator. The tools are complex instruments. Note, for example, that the tips of the tools

**FIGURE 2.31**

Sensory-motor skill combined with mental skill during laparoscopic surgery: (a) Tips of tools for grasping and cutting. (b) Exterior view of tools and monitor in a training simulator.

(Photos courtesy of the Centre of Excellence for Simulation Education and Innovation at Vancouver General Hospital)

articulate, or bend, thus providing an additional degree of freedom for the surgeon (Martinec, Gatta, Zheng, Denk, and Swanstrom, 2009). Clearly, the human-machine interaction involves both sensory-motor skill (operating the tools while viewing a monitor) and mental skill (knowing what to do and the strategy for doing it).

One way to study skilled behavior is to record and chart the progression of skill over a period of time. The level of skill is measured in a dependent variable, such as speed, accuracy, or some variation of these. The time element is typically a convenient procedural unit such as trial iteration, block or session number, or a temporal unit such as minutes, hours, days, months, or years. Measuring and modeling the progression of skill is common in HCI research, particularly where users confront a new interface or interaction technique. The methodology for evaluating skilled behavior is presented in Chapter 5 (see Longitudinal Studies) with the mathematical steps for modeling presented in Chapter 7 (see Skill Acquisition). See also student exercise 2-4 at the end of this chapter.

2.7.4 Attention

Texting while driving. It's hard to imagine a more provocative theme to open this discussion on attention. Although driving a car is relatively easy, even the most experienced driver is a potential killer if he or she chooses to read and send text messages while driving. The problem lies in one's inability to attend to both tasks

simultaneously. Much like the bottleneck posed by working memory (7 ± 2 items), the human ability to attend is also limited. But what is the limit? More fundamentally, what is attention? Which tasks require attention? Which do not? How is human performance impacted? According to one view, attention is a property of human behavior that occurs when a person who is attending to one thing cannot attend to another (Keele, 1973, p. 4). Typing, for example, requires attention because while typing we cannot engage in conversation. On the other hand, walking requires very little attention since we can think, converse, and do other things while walking. One way to study attention is to observe and measure humans performing two tasks separately and then to repeat the procedure with the two tasks performed simultaneously. A task with performance that degrades in the simultaneous case is said to require attention.

Attention is often studied along two themes: *divided attention* and *selected attention* (B. H. Kantowitz and Sorkin, 1983, p. 179). Divided attention is the process of concentrating on and doing more than one task at time. Texting while driving is an example, and the effect is obvious enough. In other cases, divided attention poses no problem, as in walking and talking. Selected attention (aka *focused attention*) is attending to one task to the exclusion of others. For example, we converse with a friend in a crowded noise-filled room while blocking out extraneous chatter. But there are limits. In that same conversation we are occasionally unable to recall words just spoken because our attention drifted away or was pulled away by a distraction. Selective attention, then, is the human ability to ignore extraneous events and to maintain focus on a primary task. One theory of selective attention holds that our ability to selectively attend bears on the importance of the events to the individual. A person listening to a speech is likely to stop listening if the person's name is spoken from another location (Keele, 1973, p. 140). One's own name is intrinsically important and is likely to intrude on the ability to selectively attend to the speech. Clearly, importance is subjective. Wickens gives an example of an airplane crash where the flight crew were preoccupied with a malfunction in the cockpit that had no bearing on the safety of the flight (Wickens, 1987, p. 249). The crew attended to the malfunction while failing to notice critical altimeter readings showing that the airplane was gradually descending to the ground. The malfunction was of salient importance to the flight crew.

The distinction between divided and selected attention is often explained in terms of channels (Wickens, 1987, p. 254). Events in a single channel (e.g., visual, auditory, motor) are processed in parallel, whereas events in different channels are processed in serial. When processing events in parallel (single channel) one event may intrude on the ability to focus attention on another event. When processing events in serial (different channels), we strive to focus on one event to the exclusion of others or to divide attention in a convenient manner between the channels.

Analyzing accidents is an important theme in human factors, as the aviation example above illustrates, and there is no shortage of incidents. Accidents on the road, in the air, on the seas, or in industry are numerous and in many cases the cause is at least partly attributable to the human element—to distractions or to selectively attending to inappropriate events. One such accident involving a driver

and a cyclist occurred because a Tamagotchi digital pet distracted the driver.¹⁴ Evidently, the pet developed a dire need for “food” and was distressed: *bleep, bleep, bleep, bleep, bleep*. The call of the pet was of salient importance to the driver, with a horrific and fatal outcome (Casey, 2006, pp. 255–259). More likely today, it is the call of the mobile phone that brings danger. The statistics are shocking, yet unsurprising—a 23-fold increase in the risk of collision while texting (Richtel, 2009).

Attention has relevance in HCI in for example, office environments where interruptions that demand task switching affect productivity (Czerwinski, Horvitz, and Wilhite, 2004). The mobile age has brought a milieu of issues bearing on attention. Not only are attention resources limited, these resources are engaged while users are on the move. There is a shift toward immediate, brief tasks that demand constant vigilance and user availability, with increasingly demanding expectations in response times. So-called psychosocial tasks compete for and deplete attention resources, with evidence pointing to an eventual breakdown of fluency in the interaction (Oulasvirta, Tamminen, Roto, and Kuorelahti, 2005).

2.7.5 Human error

Human error can be examined from many perspectives. In HCI experiments testing new interfaces or interaction techniques, errors are an important metric for performance. An error is a discrete event in a task, or trial, where the outcome is incorrect, having deviated from the correct and desired outcome. The events are logged and analyzed as a component of human performance, along with task completion time and other measurable properties of the interaction. Typically, errors are reported as the ratio of incorrectly completed trials to all trials, and are often reported as a percent ($\times 100$). Sometimes accuracy is reported—the ratio of correctly completed trials to all trials.

Two examples for computing tasks are shown in Figure 2.32. A GUI target selection task is shown on the left in two forms. The top image shows the goal: moving a tracking symbol from a starting position to a target and ending with a select operation. The bottom image shows an error, since the final selection was outside the target. A text entry task is shown on the right. The goal of entering the word *quickly* is shown correctly done at the top. The bottom image shows an error, since the word was entered incorrectly.

Mishaps and miscues in human performance are many. Often, a simple categorization of the outcome of a task as correct or incorrect falls short of fully capturing the behavior. We need look no further than Figure 2.32 for examples. Not only were the results of the tasks on the bottom erroneous in a discrete sense, there were additional behaviors that deviated from perfect execution of the tasks. For the target selection error, the tracking symbol veered off the direct path to the target. For the text entry error, it appears that at least part of the word was correctly entered.

Taking a broader perspective, human error is often studied by examining how and why errors occur. Once again, Figure 2.32 provides insight. In the erroneous

¹⁴Ample descriptions of the Tamagotchi are found in various online sources (search using “Tamagotchi”).

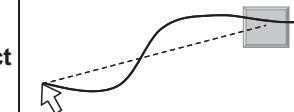
	Target Selection	Text Entry
Correct		quickly
Incorrect		qucehkly

FIGURE 2.32

Common computing tasks completed correctly (*top*) and incorrectly (*bottom*).

target selection task, was there a control problem with the input device? Was the device's gain setting too sensitive? Was the device a mouse, a touchpad, an eye tracker, a game controller, or some other input control? Note as well that the tracking symbol entered then exited the target. Was there a problem with the final target acquisition in the task? In the erroneous text entry task, if input involved a keyboard, were errors due to the user pressing keys adjacent to correct keys? Were the keys too small? If entry involved gestural input using a finger or stylus on a digitizing surface, did the user enter the wrong gesture or an ill-formed gesture? Was the digitizing surface too small, awkwardly positioned, or unstable? Clearly, there are many questions that arise in developing a full understanding of how and why errors occur. Note as well that the questions above are not simply about the human; they also question aspects of the device and the interaction.

An even broader perspective in analyzing errors may question the environmental circumstances coincident with the tasks. Were users disadvantaged due to noise, vibration, lighting, or other environmental conditions? Were users walking or performing a secondary task? Were they distracted by the presence of other people, as might occur in a social setting?

Human factors researchers often examine human error as a factor in industrial accidents where the outcome causes substantial damage or loss of life. Such events rarely occur simply because a human operator presses the wrong button, or commits an interaction error with the system or interface. Usually, the failures are systemic—the result of a confluence of events, many having little to do with the human.

To the extent that a significant accident is determined to have resulted from *human error*, a deeper analysis is often more revealing. Casey's retelling of dozens of such accidents leads to the conclusion that the failures are often *design-induced errors* (Casey, 1998, p. 2006). This point is re-cast as follows: if a human operator mistakenly flicks the wrong switch or enters an incorrect value, and the action results in a serious accident, is the failure due to human error? Partly so, perhaps, but clearly the accident is enabled by the design of whatever he or she is operating. A design that can lead to catastrophic outcomes purely on the basis of an operator's interaction error is a faulty

design. For safety-critical systems, interaction errors by an operator must be considered and accounted for. Such errors are not only possible, they are, in time, likely. Designs of safety-critical systems must accommodate such vagaries in human behavior.

STUDENT EXERCISES

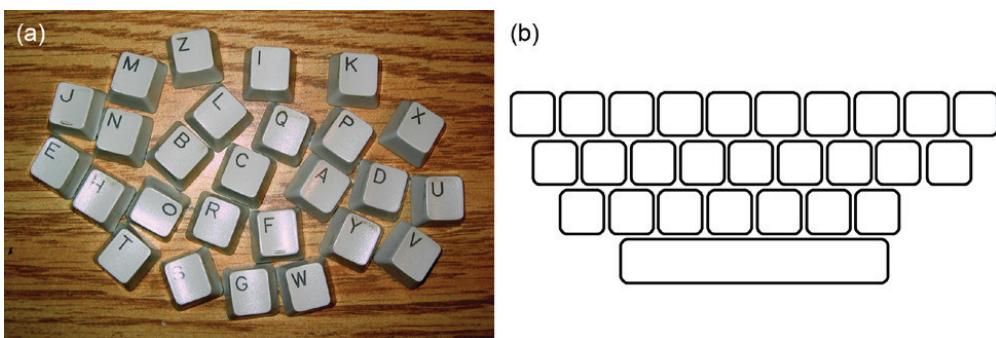
- 2-1.** Penfield's motor homunculus in Figure 2.9 illustrates the area in the cerebral cortex devoted to human responders. The sketch includes solid bars corresponding to the cortical area for each responder. The length of each bar is a quantitative indicator. Reverse engineer the motor homunculus to determine the length of each bar. The general idea is shown below for the toes and ankles.

	A	B	C	D	E	F	G	H
1	Responder	x1	y1	x2	y2	dx	dy	Length
2	Toes	52	153	55	111	-3	42	42.1
3	Ankle	56	106	64	58	-8	48	48.7

The shaded cells contain values digitized from an image processing application. The *toes* bar, for example, extends from (52, 153) to (55, 111). Using the Pythagorean theorem, the length is 42.1 pixels. Of course, the scale and units are arbitrary. Evidently, there is about 15.7 percent more cortical area devoted to the ankle than to the toes. See above. This is also evident in the figure.

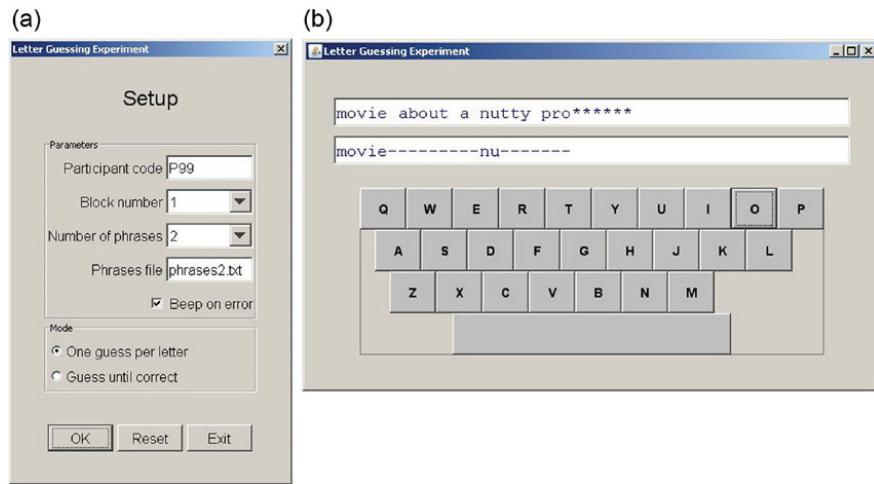
For all responders, digitize the endpoints of the corresponding bars and enter the values in a spreadsheet, as above. Create a bar chart showing the relative amounts of cortical area for each responder. It might be useful to collect together the values for the leg, arm, and head, with each shown as the sum of contributing responders. Write a brief report discussing the motor homunculus and the empirical data for the various responders.

- 2-2.** Conduct a small experiment on memory recall as follows. Find an old discarded keyboard and remove the key tops for the letters (below left). Using a drawing application, create and print an outline of the letter portion of a Qwerty keyboard (below right). Find five computer users (participants). Ask each one to position the key tops in the printout. Limit the time for the task to three minutes. Record the number of key tops correctly positioned. (Suggestion: Photograph the result and do the analysis afterward.)



Then assess each participant's typing style and typing speed as follows. Open a blank document in an editor and enter the phrase "the quick brown fox jumps over the lazy dog." On the next line, ask the participant to correctly type the same phrase. Measure and record the time in seconds. Repeat five times. For each participant, note and record whether the typing style is *touch* or *hunt-and-peck*. Enter the data into a spreadsheet. Convert the time to enter the phrase (t , in seconds) to typing speed (s , in words per minute) using $s=(43/5)/(t/60)$. Write a brief report on your findings for the *number of key tops correctly positioned*. Consider participants overall as well as *by typing speed* and *by typing style*. Discuss other relevant observations.

- 2-3.** Conduct a small experiment on redundancy and entropy in written English, similar to Shannon's letter guessing experiment described earlier (see Figure 2.22). Use 5–10 participants. For the experiment, use the LetterGuessingExperiment software provided on this book's website. Use five trials (phrases) for each participant. The setup dialog and a screen snap of the experiment procedure are shown below:



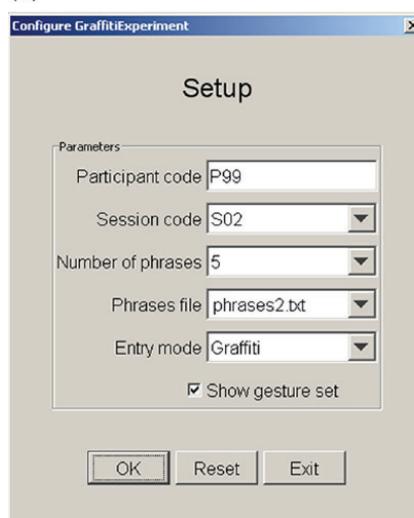
Data collection is automated in the software. Analyze the results for the number of letters correctly guessed (redundancy) and the number incorrectly guessed (entropy). Examine the results overall and by participants. Investigate, as well, whether the responses differ according to the position of letters in words and in phrases. Write a brief report on your findings.

- 2-4.** Construct a 2D chart on skilled behavior showing sensory-motor skill on one axis and mental skill on the other. For both axes, add the label *little* near the origin, and *lots* near the end. An example of a similar chart is given in Figure 3.46. Add markers in the chart showing at least five computing skills. Position the markers according to the relative emphasis on sensory-motor skill and mental skill in each task. Write a brief report, describing each skill and rationalizing the position of the marker in the chart. For guidance, see the

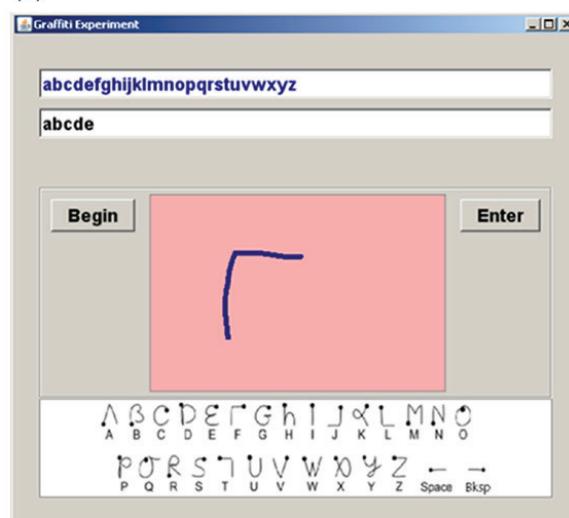
discussion in this chapter on skilled behavior. For further guidance, read the discussion in Chapter 7 on descriptive models.

- 2-5.** Conduct a small experiment on gestural input, human performance, and human error using the GraffitiExperiment software on this book's website. There is both a *Windows* version and an *Android* version. Recruit about 10 participants. Divide the participants into two groups and use a different input method for each group. Consider using a mouse and touch-pad (*Windows*) or a finger and stylus (*Android*). The software uses *Graffiti* gestures for text entry. For the experiment, the participants are to enter the alphabet 10 times. The setup dialog and a screen snap of the experimental procedure are shown below for *Windows* (top) and for *Android* (bottom).

(a)



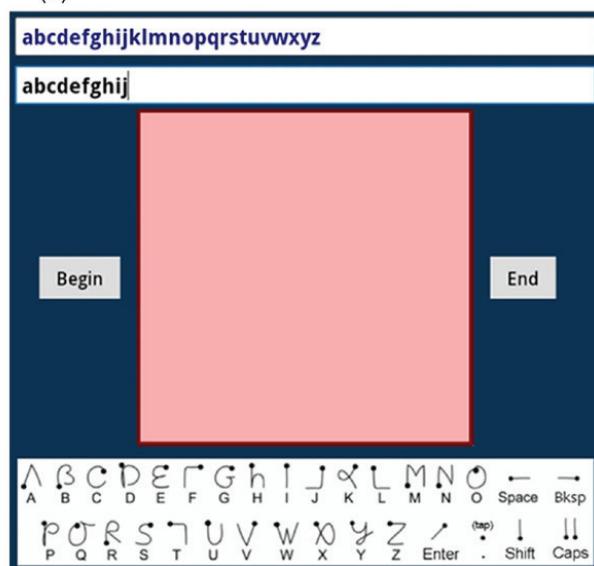
(b)



(c)



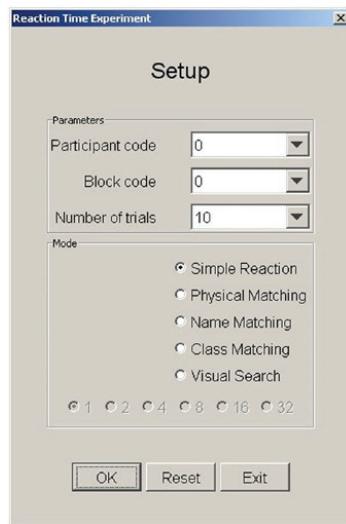
(d)



One of the options in the setup dialog is “Phrases file.” Use alphabet.txt. Set the “Number of phrases” to 10. Leave “Show gesture set” checked. The gestures are viewable in the experiment screen (see above). Participants may correct errors using the BACKSPACE stroke (\leftarrow). However, instruct participants not to attempt more than three corrections per symbol.

Data collection is automated. Consult the API for complete details. Analyze the data to reveal the progress over the 10 trials for both groups of participants. Analyze the entry speed (wpm), error rate (%), and keystrokes per char (KSPC). (A “keystroke,” here, is a gesture stroke.) Write a brief report on your findings.

- 2-6.** Conduct a small experiment on human reaction time using the ReactionTimeExperiment software provided on this book’s website. Recruit about 10 participants. The setup dialog is shown below. Examples of the experimental procedure are given above (see Reaction Time).



Consider modifying the software in some way, such as using words instead of letters for the visual search task, or using an auditory stimulus instead of a visual stimulus for the simple reaction time task. The modification can serve as a point of comparison (e.g., visual search for words versus letters, or reaction time to an auditory stimulus versus a visual stimulus). Write a brief report on your findings.

Brainstorm, Chainstorm, Cheatstorm, Tweetstorm: New Ideation Strategies for Distributed HCI Design

Haakon Faste
HCI Institute
Carnegie Mellon
hfaste@cs.cmu.edu

Nir Rachmel
HCI Institute
Carnegie Mellon
nir.rachmel@gmail.com

Russell Essary
HCI Institute
Carnegie Mellon
russell.essary@gmail.com

Evan Sheehan
HCI Institute
Carnegie Mellon
wesheehan@gmail.com

ABSTRACT

In this paper we describe the results of a design-driven study of collaborative ideation. Based on preliminary findings that identified a novel digital ideation paradigm we refer to as *chainstorming*, or online communication brainstorming, two exploratory studies were performed. First, we developed and tested a distributed method of ideation we call *cheatstorming*, in which previously generated brainstorm ideas are delivered to targeted local contexts in response to a prompt. We then performed a more rigorous case study to examine the cheatstorming method and consider its possible implementation in the context of a distributed online ideation tool. Based on observations from these studies, we conclude with the somewhat provocative suggestion that ideation need not require the generation of new ideas. Rather, we present a model of ideation suggesting that its value has less to do with the generation of novel ideas than the cultural influence exerted by unconventional ideas on the ideating team. Thus brainstorming is more than the pooling of “invented” ideas, it involves the sharing and interpretation of concepts in unintended and (ideally) unanticipated ways.

Author Keywords

Ideation; brainstorming; chainstorming; cheatstorming; tweetstormer;

ACM Classification Keywords

H.5.2. User Interfaces: Theory and Methods; H.5.3. Group and Organization Interfaces: Collaborative computing

General Terms

Design; Experimentation.

INTRODUCTION

The ability to generate new ideas as part of a creative design process is essential to research and practice in human-computer interaction. The question of how best to generate ideas is not entirely clear, however. Not only are countless design and research methodologies commonly employed by HCI teams, their ideation effectiveness depends on numerous interdependent and variable factors including the scope and objectives of the project in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI 2013, April 27–May 2, 2013, Paris, France.

Copyright © 2013 ACM 978-1-4503-1899-0/13/04...\$15.00.

question, the expertise and variety of the people involved, the strength and familiarity of their social relationships—not to mention their degree of familiarity with previous ideation and research activities—and cultural and personal factors including a person’s workplace norms and values, personal motivations and desires, confidence, degree of social collaboration, esteem, and so on. In this paper we describe the results of a design-driven study conducted with the aim of improving collaborative ideation on HCI projects using distributed software tools. Specifically we focused our research on how digital tools might be used to enhance the practice of group ideation among members of asynchronously distributed collaborative teams.

A range of different ideation techniques are used in design and HCI. In this paper, we begin with a discussion of the relative benefits and drawbacks of one such ideation method, specifically *brainstorming*, as described by Osborn [33] and evaluated by Isaksen [21], among others. We then describe a design research process that explored the creation of distributed brainstorming alternatives. Two exploratory studies were performed. First, we developed and tested an ideation method we refer to as *cheatstorming*. Using this technique, previously generated brainstorm ideas are delivered to targeted local contexts without the need for imaginative ideation. We then performed a second study of the cheatstorming method to better understand its implications and improve its efficiency. Based on observations from these studies, we conclude with the observation that ideation need not be limited to the generation of new ideas. From this perspective, the value of group ideation activities such as brainstorming has less to do with the creation of novel ideas than its cultural influence on the ideating team. Ideation, in short, is the radical redistribution of ideas to “unconventionalize” a given context.

Brainstorming Effectiveness as an Ideation Technique

The term brainstorming is best identified today with Osborn’s book on creativity titled *Applied Imagination*, first published in 1953 [33]. Osborn, who worked as an advertising executive in the 1940s and 50s, wrote a detailed examination of the creative problem solving process, and introduced brainstorming as one part of this process. Rich with current examples from that time, the book attempted to systematically define a method for deliberate creative group ideation from a very practical standpoint. Osborn divided the process into three main phases [33, p. 86]:

- (1) *Fact-finding*: Problem-definition and preparation; gathering and analyzing the relevant data.
- (2) *Idea-finding*: Idea-production and idea-development; thinking of tentative ideas and possible leads and then selecting and combining them.
- (3) *Solution finding*: Evaluation and adoption; verifying the offered solutions, and deciding on and implementing a final selected set.

In great detail, Osborn explains suggested practices for performing each of these stages, focusing in particular on the *Idea-finding* phase. He claimed that Idea-finding is “the part of problem-solving that is most likely to be neglected” by groups [33, p. 111], and offered four guidelines that should be carefully followed in order to conduct a brainstorming session effectively and yield the best results:

- 1. Criticism is ruled out: Adverse judgment of ideas must be withheld until later.**
- 2. “Free-wheeling” is welcomed: the wilder the idea the better; it is easier to tame down than to think up.**
- 3. Quantity is wanted: The greater the number of ideas, the more the likelihood of useful ideas.**
- 4. Combination and improvement are sought: In addition to contributing ideas of their own, participants should suggest how ideas of others can be turned into better ideas; or how two or more ideas can be joined into still another idea.** [33]

Since then, many have built on these rules as brainstorming has become an increasingly popular method for idea generation in business and academic contexts. For example, Osborn’s rules have been adapted to be more playful and memorable for educational purposes (e.g. “Gleefully suspend judgment,” “Leapfrog off the ideas of others” [14]), and additional rules such as “be visual,” “stay focused on the topic,” and “one conversation at a time” have been added to better guide brainstorming sessions in the context of corporate design consulting [23].

Despite its widespread adoption in collaborative innovation environments in industry, the effectiveness of brainstorming has been a hot topic of debate in the academic community since its first introduction. The first criticism was sparked by a 1958 paper published by a group from Yale University (Taylor, Berry and Block) that compared the performance of randomly assigned brainstorming groups with that of randomly assigned individuals whose work was later pooled [42]. Numerous subsequent studies (e.g. [2, 35, 8, 26]) have built on this work to critique the effectiveness of brainstorming in groups relative to individuals working independently, arguing—among other things—that fewer good ideas are generated for each hour of individual effort expended.

It is important to note, however, that the Taylor, Berry and Block study [42] did not actually test the effectiveness of the *rules* of brainstorming, since the same rules were applied to

both experimental conditions (individual and group). To be fair, Osborn recognized the necessity and advantages of working in groups for many reasons beyond the sheer quantity of ideas produced, especially when solving problems [33, p. 139]. In fact, the guidelines he suggested were specifically targeted at addressing the common inhibitory factors of group ideation. Rules such as “defer judgment” and “go wild” aimed not at individual productivity but improved social dynamics and sharing of ideas between members of a team. He also made the point to address a common misconception, stating that “group brainstorming is recommended solely as a *supplement* to individual ideation.” [33, pp. 141-142]. Still, studies critiquing the effectiveness of brainstorming on the grounds that it is inefficient were widespread through the late 1990s. More recently, the debate on productivity and collaboration has transferred to the domain of computer-mediated ideation (discussed below).

Limitations of Brainstorming

Three major explanations have been offered to account for lower purported productivity in brainstorming groups relative to ideating alone: *production blocking*, *evaluation apprehension* and *free riding* [8]. We discuss each briefly in turn.

Production blocking

Since only one person speaks at a time in a group setting, others are inhibited from expressing their ideas while another team-member is speaking, potentially slowing their ability to generate new ideas of their own. It is not the lack of speaking time in total that causes the alleged inhibition, as many times the flow of ideas ends before the end of a brainstorm session. Rather, it has been claimed that some participants’ ideas are suppressed or forgotten later in the process, as they may seem less relevant or less original than others being expressed. Furthermore, being in a situation where participants must passively listen to others’ ideas may distract and interrupt their thought processes and ability to record their own ideas. Examples of studies looking into this hypothesis can be found in [3, 24, 8, 16].

Evaluation apprehension

Creativity by definition is an unconventional act, and being creative therefore involves taking personal risks [13]. Even though one of the most important rules for successful brainstorming is to “defer judgment,” the fear of being criticized for having original ideas is often pervasive. Numerous authors have studied this phenomenon of “evaluation apprehension.” Maginn and Harris [29], for example, performed an experiment in which a brainstorming group was told that there were expert evaluators watching them through a one-way mirror. No major difference was observed between brainstorming performance in this condition relative to a control condition in which participants were not informed that they were being observed. In another study [5], groups of brainstorm participants were informed that some members of the group were “undercover” experts on the topic at hand. In this case, productivity loss was observed in groups that had been informed of their presence relative to a control group that had not been told.

Free riding

It may be the case that a brainstorming participant's motivation to work decreases if they do not perceive that they will be recognized for their participation. Since brainstorming is a group activity in which all the generated ideas are ultimately grouped together, it is often the case that the generated results are not attributed to their specific contributor. Indeed, lower identifiability of ideas may increase participants' motivation to contribute less, compared to an individual task where they know that their contribution will be recognized. Furthermore, many studies have shown that there is a lower perceived effectiveness of the individual in a group setting [8].

Structuring Ideation: Three Approaches Defined

In most of the aforementioned studies, proponents of brainstorming as an ideation technique tend to be its practitioners in the business and design communities (such as Osborn himself), while its detractors tend to be researchers interested in studying creative techniques but divorced from the nuances of its deeply embedded and culturally contextual practice [21]. Yet because the act of brainstorming incorporates numerous independent and complicated social variables—not least the makeup and experience of the team, the project objectives, the rules employed, and highly contextual success criteria—its effectiveness is difficult to study and empirically discern. Indeed, given that different ideation workplaces are likely to have differing communication patterns and communication needs depending on their cultural makeup and personnel, we find measuring the output of group ideation as a replacement for individual work to be an unsatisfactory approach. More compelling is the question of how intrinsic social and collaborative factors influence group ideation results by introducing “strangeness.” Perhaps this reflects our team’s ideological bent as design practitioners, but in today’s world, problem solving often requires experts from different fields, and new ideas are frequently sparked from novel combinations of existing concepts or the introduction of an existing concept to an unfamiliar context of use [27, 41].

Many authors have addressed the role of social factors in ideation. In this work, we ask how social factors and their resulting effects can be leveraged to develop more effective methods of group ideation online. Research has shown that social factors provide fresh sources of unexpected ideas that can help to reframe the design challenge, with design tools such as extreme characters and interaction labeling proposed as ways of dialing in the necessary “strangeness” for ideation to occur [9, 17]. Other classic ideation techniques include the use of ‘random input’ [6] and ‘oblique strategies’ [11] to generate fresh associations; by drawing on unexpected prompts and unrelated ideas to un-stick conventional thinking, such ‘trigger concepts’ bring fresh associations to the context of ideation, stimulating other associations “Like pebbles dropping in a pond.” [43] Drawing on these sources, we ask how brainstorming could be improved as a collaborative ideation technique through alternative methods of random input.

In general, we classify three common social configurations of idea generation behavior: (1) face-to-face brainstorming

in groups; (2) individual (or “nominal”) idea generation sessions; and (3) computer-mediated ideation. We discuss the unique traits of each of these approaches in turn:

Face-to-face Brainstorming Groups

The classic brainstorming session is done in face-to-face groups during a fixed period of time, usually between 15 to 45 minutes [33, p. 178], and is facilitated by a trained brainstorming expert that enforces the rules of brainstorming on the group. Participation is simultaneous and spontaneous: all participants can see each other’s ideas and are encouraged to build upon them. The ideas are recorded as they are suggested. At the end of a brainstorming session, Kelley *et al.* [23] suggest that participants vote on their favorite ideas as a way of generating closure and group consensus about which ideas are most compelling for future work. As for the optimal group’s size, in his original writings on brainstorming, Osborn suggested group sizes of up to 12 as effective [33, p. 159]. But there is no agreement in more recent literature as to optimal group-size (*e.g.* [16, 36, 4]), partly because it is difficult to define “optimal” in the context of real-world practice.

Nominal Idea Generation Sessions

Nominal idea generation is done individually. The main element that defines this method is that participants are not influenced by the variety of social factors at play in a traditional brainstorming group: they cannot build on other participants’ ideas because they are not exposed to them, they will be less influenced by perceived criticisms to their ideas in real-time (although they may be reluctant to share them afterwards), they may be highly motivated to perform their work in the anticipation that their efforts will eventually be rewarded, and so on. Extensive research has been done to study the benefits and shortcomings of classic vs. nominal brainstorming, as described above. In general, it appears that nominal brainstorming has some benefits in terms of both quality and quantity of ideas [20, 30, 10, 28] due to psychological effects defined by Diehl & Stroebe [8].

Computer Mediated Ideation

Advances in digital technology have led to the potential for a variety of computer-mediated ideation techniques. Within this category, the term “electronic brainstorming” refers to any kind of brainstorming mediated by computers (*e.g.* [40, 7, 1]). One issue attempting to define electronic brainstorming is that *any* online activity that involves people entering information into cloud-based systems can be considered the contribution of “ideas” to a digital pool. For our purposes we therefore consider an electronic brainstorm to be only that subset of software-mediated interactions in which users are asked to specifically generate creative responses to a question or prompt. This differs slightly (with regard to intent) from forums in which people are asked to contribute “best practices” or “suggestions” based on prior-knowledge simply as an act of knowledge-transfer (*e.g.*, suggestion portals wherein users can recommend local restaurants or hotels). It also differs from critique feeds and forums, such as post-blog comment streams debating the

relative merits of an advanced position and/or themed around a topic of debate—although such kinds of activities are certainly related to electronic brainstorming and can be useful tools for the evaluation of brainstorming results as well as later phases in the ideation process.

The various possible ideation approaches described above (group brainstorming, nominal idea generation, and computer-mediated ideation) are not mutually exclusive, and can be combined and mixed to make the most of each method. A brainstorming session could be performed in two parts, for example, the first in the nominal style followed by a face-to-face method to evaluate and combine ideas across participants. Electronic brainstorming can also support both nominal and group methods, or implement a diverse array of combinations between them. Indeed, it is precisely because of the flexibility of electronic methods to distribute various aspects of the brainstorming task across asynchronous distributed teams that we performed the studies described in the following section. Group ideation is an integral part of HCI research practice, and an area where the implementation of improved software interactions could greatly enhance how ideation happens in research laboratories, design firms and product companies alike.

METHODOLOGY AND DESIGN RESEARCH

Our investigation began with the simple premise that collaborative ideation could be enhanced through the use of distributed online tools, and design-driven approaches could be used to explore and investigate the potentialities of this space. Our design team consisted of four members with diverse backgrounds including design consulting, software engineering, anthropology, and management. We held regular meetings over the course of several months to conduct freeform exploratory design research. Sessions were held once or twice weekly for 1-3 hours per session. The setting was a design studio in the HCI Institute at Carnegie Mellon University. This section describes our design research process, consisting of the following phases: (1) opportunity finding; (2) electronic brainstorming; (3) concept selection and refinement; and (4) experimentation and discussion.

Opportunity finding

We began with a vision for an online space to browse and share ideas where they could be tagged, filtered, and contextualized in the cloud. This vision was founded on two beliefs: that creators are everywhere, and that they are driven by creative ideas for which they seek open outlets. Although a clear plan for how to develop such a system was not yet evident, we first created a series of exploratory concept sketches to help envision possible outcomes and establish goals. We then analyzed aspects of our concept drawings and generated a set of Post-It notes chronicling our complete list of observations and desires.

Next, we arranged these notes on a 2x2 matrix to help group them into clusters and synthesize common themes. Because our aim with this stage was to work on a meaningful project that was enjoyable and inspirational to the team, the axes of

this matrix we created, ranging from low to high in each dimension, were “Fun Impact” vs. “Social Impact.” Seven areas of opportunity emerged from this exercise: (1) Reveal hidden (personal) meanings through metaphorical leaps of imagination; (2) Facilitate the discovery of thinking patterns; (3) Track creative influence to motivate participation; (4) Associate and juxtapose unexpected ideas; (5) Help people find ideas that are important to them; (6) Invent and embody “creative movements”; and (7) Spark and inspire interest and freedom.

Electronic Brainstorming

Given our interest in exploring the possibilities of electronic brainstorming we decided to experiment with distributed ideation online. Using the identified opportunity areas as jumping-off points for generative design, we restated each of the seven opportunity statements described above as a “How could we...” question (e.g. “How could we facilitate the discovery of thinking patterns?). Each question was placed at the top of a separate new Google Docs file. We then invited some 30+ interdisciplinary undergraduate and graduate students in the HCI Institute to these seven files. All of these students had prior experience with group brainstorming, and were given the instruction to each contribute at least five ideas in response to one or more of the brainstorm questions. We performed this activity over the course of a four-day weekend, with the stated goal of achieving at least 50 ideas in response to each question. On the fourth day, five of the seven questions had more than 50 ideas. For the remaining two questions the research team made a concerted effort to generate the remaining necessary ideas. In total, 350 distinct opportunity concepts were generated. Next, seven of the most involved members of the laboratory team were asked to “vote” on their favorite ideas in each file by adding a brightly colored symbol next to the item number. In this way, a selected group of 35 “favorite” ideas were agreed upon from across all seven questions.

Concept Selection and Refinement

Favorite ideas were printed out on paper, cut into strips, and placed on an Impact/Achievability matrix [15]. We then gave each of these ideas a more concise name by applying colorful Post-it notes on top of them and drawing broad categories around them with a colorful marker. The main outcome of this phase was two key concepts, each in the “easy” and “high-impact” quadrant. The first was a group of ideas we labeled “idea factories.” Of these there was one particularly compelling idea—the concept of an idea “broken telephone” game. We refer to this concept in general as “chainstorming.” The second was a category of ideas we identified as “creative judgment tasks” involving quickly voting on pre-existing ideas, much as we had done at the end of our electronic brainstorming sessions. We refer to this concept in general as “cheatstorming,” as described in studies 1 and 2 below. Finally, while not discussed here in detail, we are currently building a working prototype system that combines chainstorming with cheatstorming, called *Tweetstormer*, also described below. To clarify, the

relationship between brainstorming, chainstorming, cheatstorming, and tweetstormer is shown in figure 1.

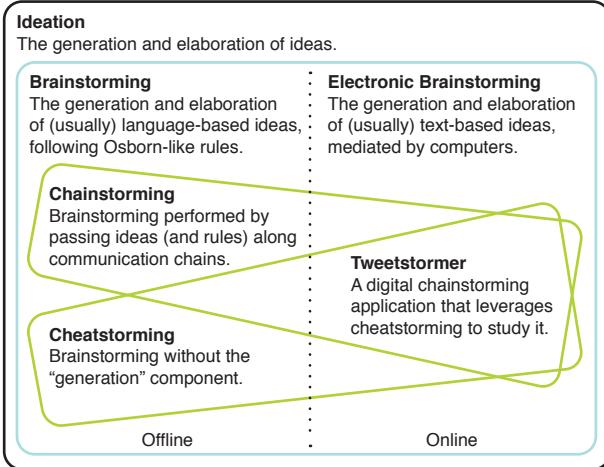


Figure 1. A taxonomy of interrelated ideation techniques.

Experimentation: Cheatstorming (Study 1)

Our main work in this paper explores the cheatstorming concept. The basic premise of this paradigm is as follows: imagine a brainstorm has been performed, resulting in 50 ideas. Participants vote on their favorite ideas, and some of them are selected for implementation. Now another brainstorm is performed on a different topic, resulting in 50 more ideas and additional voting. In time, many hundreds of brainstorm questions are asked, and thousands of ideas are generated and saved. Some have been implemented, and others have not. At this point, a wealth of valuable brainstorming has already occurred. The cheatstorming paradigm proposes that *no new ideas are necessary for further ideation to occur*. Given a new prompt question and a set of 50 random previous ideas to draw from, cheatstorming simply bypasses the concept generation phase altogether and jumps directly to voting on which ideas to advance.

To test this concept we performed a simple pilot experiment. First, each member of our team generated 3-5 “totally random” brainstorm questions on Post-It notes, not in response to any particular question or stated need (e.g. “What is the easiest way to make the most people happy cheaply?”). Next, a set of



Figure 2: Sample results from our first cheatstorming trial.

60+ solution concepts was generated equally at random (e.g. “Magnetic cellphones”, “Non-linear presentation tool”, “Magic annoying elf that re-arranges your clothing,” etc.). Finally, one of the previously generated brainstorm questions was selected at random and paired with 10 of the concept Post-Its at random. From these 10 ideas, the four concepts that most closely resonated as solutions to the given question were selected as “winners.” We repeated this process four times with four different questions. For example, one of the sample solution pairings is shown in figure 2.

We were both surprised and delighted by the results of this method. Not only did we have little difficulty identifying those ideas that best resonated with the questions being asked, the resulting set of ideas was remarkably unexpected and fresh. Most exciting, the process was fast, fun, and required low effort, and the solutions revealed unexpected combinatory patterns and juxtapositions. In the first example shown in figure 2, for instance, the question asks “How could we illuminate large cities for less money to reduce nocturnal crime?” Surprisingly, three of the selected solution concepts are screen-based ideas that all emit light. Not only was this an unanticipated means of illumination, it was also one that could provide other forms of safety from nocturnal crime—via an interactive “call for help” kiosk or informative map, for example. Furthermore, the fourth idea in this set, “airbag for walking,” suggests that perhaps solutions for reducing nocturnal crime could be built directly into a user’s clothing. Combined with the other cheatstormed ideas, this in turn sparks a train of thought that perhaps clothing should be illuminated, or—alternatively—that the city’s streets should be padded. Finally, each of the other cheatstormed questions resulted in an equally compelling set of results. In response to the question “How could we reduce global warming effectively in the next five minutes?”, for example, “biodegradable vehicles” and “micro-financing” were among the selected concepts. While neither of these ideas may enable global warming to be reduced in the next five minutes alone, when combined together they indicate a potential direction for immediate action (*i.e.*, green-vehicular crowdfunding).

Experimentation: Cheatstorming (Study 2)

There are many variables in the way that cheatstorming could be performed that we were curious to explore, such as how the variable effects of different types of “idea input” would affect cheatstorming results. We also wanted to compare cheatstorming results with results from a traditional brainstorming session. To this end, our next study leveraged the results of five previously completed brainstorming sessions from other unrelated projects as input. We chose this data from prior brainstorming sessions that had been well documented with clear questions and solutions, and which had generated more than 50 ideas apiece. These ideas had also been voted upon in the previous iteration, enabling us to track the success or failure of previously successful ideas in the new cheatstorming context. Finally, it was important for us that the brainstorming sessions had been performed by different groups of participants spanning a diverse set of HCI topics, to



Figure 3: Input data for the *cheatstorming* study.

ensure that we had a wide variety of ideas in our pool to draw from overall, and so that unanticipated biases based on the authorship of ideas was reduced.

The prompts from the five selected sets of data were as follows: (1) “*How could we summarize text-based information to make browsing it intuitive, useful, magical and fun?*”, from a project on digital mind mapping; (2) “*How could we sculpt and craft using digital tools?*”, from a project on tangible computing; (3) “*How could we encourage self-actualization and the experience of new experimental dynamics?*”, from a project on augmented reality; (4) “*How could we support the successful publication of confident high quality writing?*”, from a project on narrative fiction; and (5) “*How could we rigorously craft and curate the design of aesthetically pleasing narrative products and services?*”

”, also from the narrative fiction project.

Our study design involved four experimental conditions drawing on brainstorming results from the above-mentioned sets of data. All of the previously generated raw ideas from each set of data were printed on cards in a unique color, one color per set (Figure 3). These raw-idea cards were used as input data for each of our study conditions. In addition, those idea cards that had been originally selected within each set as the “winners” for that set were clearly marked with an asterisk; this allowed us to trace which previously successful ideas prevailed through the cheatstorming process.

The study conditions were designed to be structurally equivalent. In each case, 50 raw “input” ideas would be pared down to 10 “winning” ideas in response to the ideation prompt. We used the same ideation prompt across all conditions: question 5 (“*How could we rigorously craft and curate the design of aesthetically pleasing narrative products and services?*”). The experimental conditions, illustrated in figure 4, were as follows:

Condition A (brainstorming baseline). Previously selected brainstorming results from set 5 (those with asterisks) were chosen automatically as *de facto* winners.

Condition B (overlapping diverse input). 17 ideas were each selected at random from sets 2, 3, and 4, combining to make a total of 51 ideas. One idea was removed at random, resulting in 50 ideas. Cheatstorming then commenced using question 5 as the ideation prompt. Because set 4 was drawn from the same project as set 5, cheatstorm results were anticipated to be most similar to condition A.

Condition C (unrelated diverse input). The same diverse

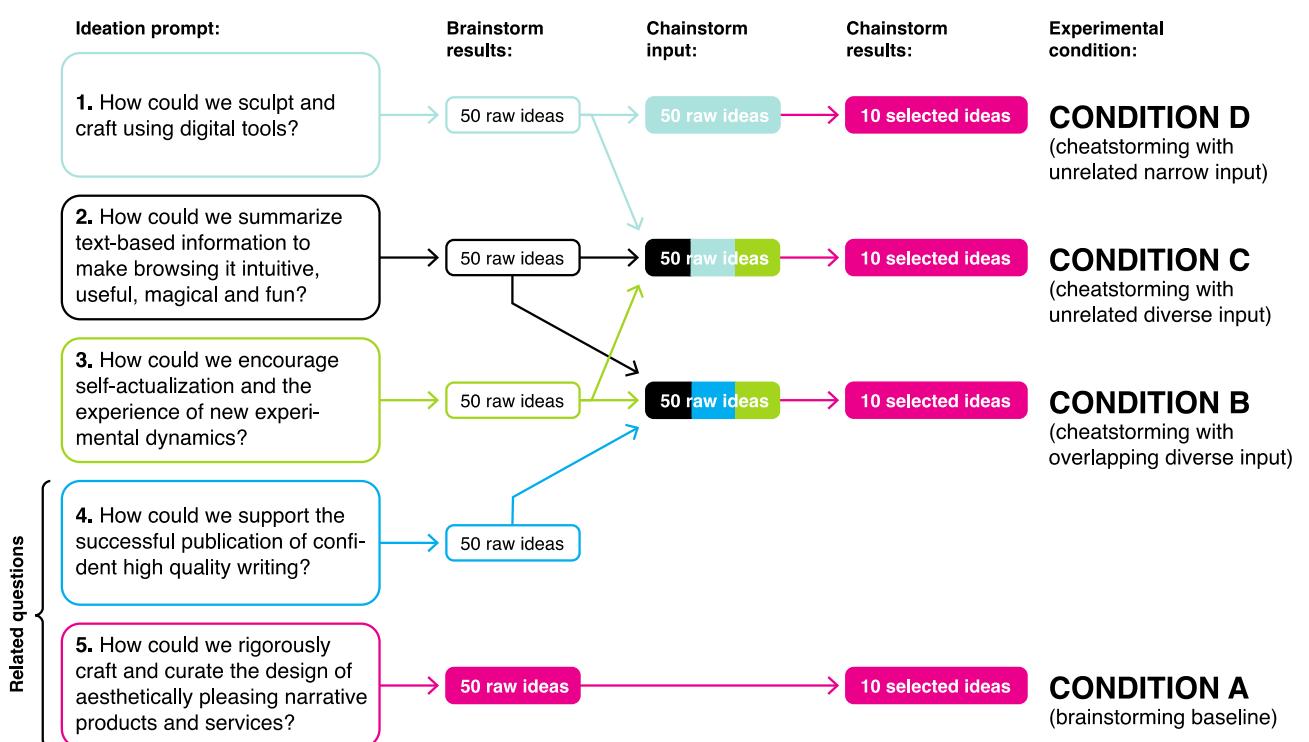


Figure 4. Experimental conditions for cheatstorming study 2.

input structure was used as in condition B, except input ideas were drawn from sets 1, 2, and 3. These ideas were not intentionally related to set 5 in any way.

Condition D (unrelated narrow input). This session used a single unrelated set of ideas as input, from set 1.

Cheatstorming proceeded by laying out all 50 input ideas for a given condition below the ideation prompt, then working through each of them one-by-one as a team, attempting to find ideas that would match with the brainstorming prompt (figure 5). Ideas that didn't seem related were put aside. Remaining ideas were grouped together into 10 "winning" clusters, such that each cluster created a meaningful concept relevant to the prompt. Each cluster was then given a more concise and meaningful title so as to relevantly depict the newly synthesized idea (figure 6).

DISCUSSION

As described in detail by Isaksen [21], evaluating the effectiveness of group ideation outcomes is fraught with methodological and practical problems. These include the necessity of identifying and isolating the different factors in the ideation tool or process likely to influence its effectiveness, being aware of the level of training (if at all) the



Figure 5: The 50 candidate cheatstorm ideas in condition C.



Figure 6: "Winning" concepts from condition B.

facilitator had gone through to run the session, determining the group's experience with creative ideation in general (and their orientation to the task at hand in particular), the preparation and presentation of the task in such a way that it promotes ideation, the effectiveness of the ideation method in highly-contextual real-world practice, and the criteria employed to evaluate the outcomes. Given these challenges, we believe it is difficult if not impossible to generalize the effectiveness of a specific culturally embedded creative activity without first recognizing the serious practical limitations of attempting to do so. For this reason, the approach taken in this study was design-oriented, in line with Fallman's characterization of design-oriented HCI as giving form to previously non-existent artifacts to uncover new knowledge that could not be arrived at otherwise [12]. We attempted to replicate a controlled methodology as precisely as possible during our cheatstorming study four times, each time varying only the set of ideas input into the selection process. Each time, 50 previously defined ideas were reduced down to 10 "winning" favorites by the same team of researchers, and each time our 10 favorite ideas were unique. Given the creative and intentionally unpredictable nature of ideation, we believe that even though we held all of these variables constant (*i.e.* same team, same brainstorming rules, same prompt question, etc.) we would likely have generated differing ideation results if we attempted to repeat this study again. This stated, some noteworthy qualitative observations can be made, and we will now reflect on the qualitative differences in both the application of the process and the outcomes it produced between differing conditions of study 2.

Findings: Process

Cheatstorming was shown to be a fast and enjoyable means of creative ideation. Especially when cheatstorming ideas that came from different and diverse input sets, we find that this method works well as a mechanism of introducing novel concepts across creative cultures, a process akin to "technology brokering" among brainstorming teams whose ideas cross-pollinate [41]. Indeed, the greatest challenge and thrill of the cheatstorming method is being faced with the task of combining what often seem to be nonsensical results from previous brainstorming sessions—in that they contain remarkably little context by which to understand them—with ideation prompts that are likely to be equally without adequate context (especially should cheatstorming be widely deployed in a distributed setting). The natural reaction of the cheatstormer—indeed, their only real option—is to force an inventive connection between ideation and prompt. In this regard we posit that the more tightly constrained the input data given to the cheatstormer when synthesizing across large sets of data, within reason, the more effective they will be at identifying such juxtapositions. We note, for example, that our first study involved the reduction of 10 input ideas to four "winners," which could be accomplished very quickly because the cheatstormer had no alternative than to pick something quickly that worked. Study 2, with its larger set of inputs and greater creative freedom, introduced a more overwhelming quantity of possible connections and, consequently, felt more tedious and less productive.

Based on this observation, we believe that setting time-oriented constraints might help to improve the cheatstorming experience. While the mandatory rigor of matching 10 winning concepts per question in study 2 was nice, it also resulted in additional room for idea comparison and judgment, leading more nuanced but ultimately (we feel) less inspired ideas. Adding a time limit or other kinds of creative constraint might encourage spontaneous connections and force weaker ideas to be eliminated more quickly on a visceral basis.

Comparing the process across experimental conditions, it seemed both easier and more immediately intuitive to group together ideas that came from the same original source. Looking at the results of our final synthesis, however, we notice a distinctly integrated mixing of source material in the creation of our final generated concepts. This also highlights one of the possible biases that became evident as a result of our process. In retrospect we wished that we had not color-coded the input ideas, as it introduced a perceptible value judgment into the study. Indeed, the simple awareness that such a bias may have existed is likely to have resulted in our intentional or unintentional effort to use equal numbers of ideas of each color, for example. As a result, it is difficult to say if the approximately even survival rate of resulting ideas across input pools within each condition resulted from this bias.

Another source of bias were ideas that repeated themselves in subsequent iterations. This influence was twofold: foremost, since the prompt remained the same for all four cheatstorming iterations, arriving at similar ideas with each round became quickly redundant. Furthermore, because each cheatstorm used a random mix of input material, about a third of the ideas from previous conditions re-appeared with each subsequent effort. In this regard, we believe that ideas that have been previously “used” by participants should be removed from the input pool in successive rounds. Using digital systems, we anticipate the implications of scaling these methods up to large crowds of users, and recommend tracking previously viewed ideas to prevent them from appearing again. Not only did ideas seem “less interesting” on the second occasion, they also became harder to associate with new outcomes and meanings. Indeed, if creativity systems are to be tasked with delivering unconventional content to users it’s essential that the content should not be familiar.

Findings: Results

In addition to the cheatstorming team’s qualitative reflections on process, we consulted with an independent judge who had worked on the narrative fiction project to which the ideation prompt had originally belonged. Together we evaluated the top 10 “winning” idea clusters from each of the 4 conditions, to see if cheatstorming results would be applicable for potential real-world use on her project.

Relative to the baseline brainstorm condition (condition A), the most noticeable quality of the winning cheatstormed ideas was that all of them were dramatically technological in nature. This is not surprising, given that the narrative fiction

project was the least technologically oriented of the prompts (the other three questions having been drawn from projects on augmented reality, tangible computing, and digital mind mapping, respectively). Furthermore, the degree to which ideas felt un-helpful to the project was directly proportional to their degree of strain. In condition A, the baseline condition, the ideas felt the most immediately useful and applicable to the project because they did not all have such a technology focus. We should also note, however, that our judge had originally been involved in selecting the baseline winners, but not the cheatstorming results, introducing a likely source of bias. Condition B, the overlapping diverse input group, was the most palatable set of the remaining ideas. It seemed to introduce fresh new ideas that were grounded in something familiar. Condition C, the unrelated diverse input group, was described as “the most random.” Ideas in this set—with names such as “tempo of experience control” and “real-time story-world generation”—were exciting but felt out-of-touch with project goals. Condition D, the unrelated diverse input group, were the most technologically immersive. Ideas such as “magic story wand” and “crowdsourced tangible narrative sculpting” were described as “nice to pursue if I had a team of designers and developers, but that would change the focus of what the project is really about.”

In summary, all of the ideas that resulted were related to the ideation prompt, but clearly reflected the spirit of the brainstorm from which they originated. This is not surprising, but it does indicate that a diverse mix of somewhat related (but also diverse and different) ideas could have a positive impact at broadening the scope and breadth of a project’s ideation.

CONCLUSIONS AND FUTURE WORK: CHAINSTORMING, TWEETSTORMING, AND CHEATSTORMING AT SCALE

This work has investigated distributed ideation from a design-driven perspective by designing and building prototypes of possible ideation mechanics and reflecting on the qualities of the outcomes. Our aim with this approach is to improve the design of HCI tools that facilitate efficient and effective group ideation.

Reflecting on our findings, we realize that we have revealed a model for group ideation with four distinct stages of progressive activity. Each stage carries with it a set of differing requirements and resulting behaviors, and we expect that the criteria leading to effective ideation outcomes at each stage will be different. These stages are: (1) *prompting*, the stage during which the ideation facilitator presents a challenge to the group that will drive ideation; (2) *sharing*, the stage in which participants suggest and communicate ideas within the context of the medium that frames the activity (*i.e.*, orally, and/or using a whiteboard, sticky-notes, database system, and so on); (3) *selecting*, the phase during which participants vote and/or otherwise determine their favorite ideas; and (4) *committing*, the stage at which a final criterion is set to evaluate and prioritize ideas, ultimately determining which ones the team moves forward with and (ideally) develops.

This framing is in contrast to previous ideation models (*e.g.* Jones' "divergence, transformation, convergence" model [22], Nijstad *et al.*'s dual pathway ideation model [32], etc.) in that, while it recognizes the cognitive distribution of ideation across social structures, *it does not view creative behavior as a "generative" activity*. Instead, ideas are simply transferred (or "shared") between people, and the act of sharing is the source of the ideation: it involves the expression and interpretation of possible conceptual meanings. Even in traditional brainstorming sessions, we propose, it is this communicative interplay between one person's conception of an idea and another's (mis)interpretation that results in the so-called "generation" of ideas. Cheatstorming demonstrates that ideas need not be created by the team for ideation to occur—they simply need to be interpreted as possibilities resulting from a collision of shared meanings. The only requirement for a successful ideation outcome is that the ideas introduced in the sharing stage are unconventional to the ideating individual, team, or culture [24] (*i.e.* "strange" [18]), and that they be interpreted as relevant (or not) to the ideation prompt.

We have introduced the concepts of *cheatstorming* as ideation without the "idea generation" component, and *chainstorming* more generally as a paradigm of communicative ideation (figure 1). Rather than conceiving of creativity as a spontaneous act of personal imagination, chainstorming is intrinsically social by nature. It is inspired by the "broken telephone" (or "Chinese whispers") social group game, in which one person (Alice) secretly tells a story to another person (Bob), such that none of the other people present can hear it. In turn, Bob tells the story as he remembers it to a third person (Carol), and so on, until all of the people in a continuous chain back to Alice are reached. The last person to hear the story shares what he or she remembers with the entire group, and that story is compared with the original story.

In chainstorming, much like this game, each participant is asked to build on the story of the previous participant in the chain. The first person in the chain generates the prompt question and one or two ideas that respond to the question before sending it off to a network of friends. Each subsequent person sees the prompt question, along with a subset of the ideas from the previous participant, and uses these ideas to build on them and generate new ideas. Using this method, which introduces a degree of randomness at each stage and which can also be controlled by the design of the communication and its rules, we propose that collective creativity can be embedded in social networks through simple interactions that reduce cognitive effort. Indeed, similar approaches have been developed in recent related work, promising the development of evolutionary creativity algorithms wherein humans pick the "fittest" ideas to result in emergent solutions to potentially complex tasks [45]. In chainstorming, where a random subset of each participant's previous ideas could be selected and passed along with each interaction, the continued juxtaposition and "constructive strain" [18] from potentially unrelated or even contradictory ideas could consistently spark unexpected new socially-generated concepts. Indeed, it is the unique ability of

cheatstorming to "dial in strangeness," as explored in our study, that makes it such a compelling example of the future of ideation online. In the case of cheatstorming, this is far more nuanced than existing methods of random input, such as future workshops [31], inspiration card workshops [19], or other similar methods for lateral thinking, in that it enables operational changes to the ideation methodology and content directly, and thus can facilitate targeted and highly contextual "leaps" from an original set of ideas to a much wider framing of the problem domain.

Clearly the success of chainstorming as paradigm depends largely on details of its implementation since, as noted in our discussion of brainstorming best practices, several factors will greatly influence the most effective outcomes. Much like offline group brainstorming, effective chainstorming is likely to depend heavily on the social constitution of the chain, the level of training (if any) that participants receive, the group's experience and orientation with the task at hand, and the criteria employed to evaluate its outcomes. Moreover, ideation of this nature introduces additional factors that will need to be addressed—especially the potential lack of context accompanying the prompt communication, which (and how many) prior concepts accompany the message as it is passed from user to user, how is their selection determined, as well as how to handle redundant concepts, dead ends, cross-posting and parallel chains, and so on. Indeed, these are complicated issues that underlie all social messaging and communication networks.

In order to investigate these questions of ideation more deeply, and identify best practices for chainstorming networks, we have begun the design and development of a new social media platform for ideation—*Tweetstormer*—which will leverage Twitter messages as the transactional medium of the chainstorming system. Using this platform, members of the online community will be able to post and respond to tweeted prompt questions to virally distribute the chainstorm. Not only will this enable Twitter users to ideate anytime from anywhere using their computer or mobile device, our plan is to implement a custom website that allows users to see other users' questions, reply to them selectively, browse other users' replies to prompts, and vote on their favorite ideas to select them. Our hope is that ideation via this and other similarly inspired platforms will enable a more nuanced empirical study of the chainstorming paradigm and how best to integrate it effectively into the social fabric of online innovation.

REFERENCES

1. Barki, H., Pinsonneault, A. (2001). Small Group Brainstorming and Idea Quality: Is Electronic Brainstorming the Most Effective Approach?, *Small Group Research*, **32**, 158.
2. Bayless, O.L. (1967). An alternative for problem solving discussion, *Journal of Communication*, **17**, 188-197.
3. Bouchard, T. J., & Hare, M. (1970). Size, performance, and potential in brainstorming groups, *Applied Psych*, **54**, 51-55.
4. Bouchard, T. J., Barsalou, J., Drauden, G. (1974). Brainstorming procedure, group size, and sex as deter-

- minants of the problem-solving effectiveness of groups and individuals, *Applied Psychology*, **59**(2), 135-138.
5. Collaros, P. a., & Anderson, L. R. (1969). Effect of perceived expertness upon creativity of members of brainstorming groups, *Applied Psychology*, **53**(2), 159-163.
 6. de Bono, E. (1970). *Lateral Thinking: Creativity Step By Step*, Harper Perennial.
 7. DeRosa, D. M., Smith, C. L., & Hantula, D. A. (2007). The Medium Matters: Mining the long-promised merit of group interaction in creative idea generation tasks in a meta-analysis of the electronic group brainstorming literature, *Computers in Human Behavior*, **23**, 1549-1581.
 8. Diehl, M., & Stroebe, W. (1987). Productivity Loss in Brainstorming Groups: Toward the Solution of a Riddle, *Personality & Social Psychology*, **53**(3), 497-509.
 9. Djajadiningrat, J.P., Gaver, W.W. & Frens, J.W. (2000). Interaction relabelling and extreme characters: Methods for exploring aesthetic interactions. *Proc. DIS 2000*, 66-71.
 10. Dunnette, M. D., Campbell, J., Jaastad, K. (1963). The effect of group participation on brainstorming effectiveness for two industrial samples, *Applied Psychology*, **47**(1), 30-37.
 11. Eno, B. (1978). *Oblique Strategies*. Opal, London.
 12. Fallman, D. (2003). Design-Oriented Human-Computer Interaction. *Proc. CHI*, 225-232
 13. Faste, R. (1993). An Improved Model for Understanding Creativity and Convention, in Cary A. Fisher (ed.), *ASME Resource Guide to Innovation in Engineering Design*, American Society of Mechanical Engineers.
 14. Faste, R. (1995). A Visual Essay on Invention and Innovation," *Design Management Journal*, **6**(2).
 15. Faste, H. & Bergamasco, M. (2009). A Strategic Map for High-Impact Virtual Experience Design, *Proc. SPIE*, **7238**
 16. Gallupe, R., Bastianutti, L. M., & Cooper, W. H. (1991). Unblocking brainstorms. *Journal of Applied Psychology*, **76**(1), 137-142.
 17. Graham, C., Rouncefield, M., Gibbs, M., Vetere, F., and Cheverst, C. (2007). How Probes Work *Proc. of OzCHI*, 29
 18. Gordon, W. J. J. (1971), *The Metaphorical Way of Learning and Knowing*, Porpoise Books, p. 20.
 19. Halskov, K., Dalsgård, P. (2006). Inspiration Card Workshops, *Proc. Designing Interactive Systems*, 2-11.
 20. Hegedus, D. M., (1986). Task Effectiveness and Interaction Process of a Modified Nominal Group Technique in Solving an Evaluation Problem, *J. of Management*, **12**(4), 545-560.
 21. Isaksen, S. G. (1998). *A Review of Brainstorming Research: Six Critical Issues for Inquiry*, Technical report, Creative Problem Solving Group, Buffalo, NY.
 22. Jones, J. C. (1970). *Design Methods*, John Wiley & Sons.
 23. Kelley, T., Littman, J. and Peters, T. (2001). *The Art of Innovation: Lessons in Creativity from IDEO, America's Leading Design Firm*, Crown Business.
 24. Koestler, Arthur, *The Act of Creation*, Dell, NY, 1964
 25. Lamm, H. and Trommsdorff, G. (1973). Group versus individual performance on tasks requiring ideational proficiency (brainstorming): A review. *European Journal of Social Psychology*, **3**, 361-388.
 26. Larry, T., Paulus, P. (1995). Social Comparison and Goal Setting in Brainstorming Groups, *Journal of Applied Social Psychology*, **25**(18), 1579-1596.
 27. Lehrer, J. (2012). Groupthink: The brainstorming myth, *The New Yorker*, January 30
 28. Madsen, D. B., & Finger, J. R. Jr., (1978). Comparison of a written feedback procedure, group brainstorming, and individual brainstorming, *Applied Psychology*, **63**(1), 120-123.
 29. Maginn, B. K., & Harris, R. J. (1980). Effects of anticipated evaluation on individual brainstorming performance, *Journal of Applied Psychology*, **65**(2), 219-225.
 30. Mullen, B., Johnson, C. (1991). Productivity Loss in Brainstorming Groups: A Meta-Analytic Integration, *Basic and Applied Social Psychology*, **12**(1), 3-23.
 31. Muller, M.J., White, E.A., and Wildman, D.M. (1993). Taxonomy of PD practices: A brief practitioner's guide. *Communications of the ACM*, **36**(6), 26-28 (June 1993).
 32. Nijstad, B. A., De Dreu, C. K. W., Rietzschel, E. F., & Baas, M. (2010). The dual pathway to creativity model: Creative ideation as a function of flexibility and persistence. *European Review of Social Psychology*, **21**, 34-77.
 33. Osborn, A. F. (1963). *Applied Imagination: Principles and procedures of creative thinking* (3rd edition), Scribner.
 34. Paulus, P. (2000). Groups, Teams and Creativity: The Creative Potential of Idea-Generating Groups, *Applied Psychology: An International Review*, **49**(2), 237-262.
 35. Price, K. (1985). Problem Solving Strategies: A Comparison by Problem-Solving Phases, *Group and Organization Studies*, **10**(3), 278-299.
 36. Renzulli, J. S., Owen, S. V., & Callahan, C. M. (1974). Fluency, flexibility, and originality as a function of group size, *Journal of Creative Behavior*, **8**(2), 107-113.
 37. Searle J., R. (1983). *Intentionality: An Essay in the Philosophy of Mind*, Cambridge University Press, 1983.
 38. Shah, H. H. & Vargas-Hernandez, N. (2002). Metrics for Measuring Ideation Effectiveness, *Design Studies*, **24**(2).
 39. Stein, M. I. (1975). *Stimulating Creativity: Group Procedures* (volume 2), Academic Press, NY
 40. Stenmark, D. (2001). The Mindpool Hybrid: Theorising a New Angle on EBS and Suggestion Systems, *Proc. Hawaii International Conference on Systems Science*.
 41. Sutton, R., & Hargadon A. (1996). Brainstorming Groups in Context: Effectiveness in a Product Design Firm, *Administrative Science Quarterly*, **41**(4), 685-718.
 42. Taylor, D. W., Berry, P. C., Block, C. H. (1958). Does group participation when using brainstorming facilitate or inhibit creative thinking? *Administrative Science Quarterly*, **3**(1), 23-47.
 43. von Oech, R. (1986). *A Kick in the Seat of the Pants*, Harper.
 44. Watson, W., Michaelsen, L. K. & Sharp, W. (1991). Member competence, group interaction, and group decision making: A longitudinal study. *Applied Psychology*, **76**, 803-809.
 45. Yu, L., & Nickerson, J. (2011). Cooks or Cobblers? Crowd Creativity through Combination, *Proc. CHI*, 1393-1402.

Observations on concept generation and sketching in engineering design

Maria C. Yang

Received: 5 September 2005 / Revised: 14 December 2006 / Accepted: 3 June 2008
© Springer-Verlag London Limited 2008

Abstract The generation of ideas is an essential element in the process of design. One suggested approach to improving the quality of ideas is through increasing their quantity. In this study, concept generation is examined via brainstorming, morphology charts and sketching. Statistically significant correlations were found between the quantity of brainstormed ideas and design outcome. In some, but not all, experiments, correlations were found between the quantity of morphological alternatives and design outcome. This discrepancy between study results hints at the role of project length and difficulty in design. The volume of dimensioned drawings generated during the early-to-middle phases of design were found to correlate with design outcome, suggesting the importance of concrete sketching, timing and milestones in the design process.

Keywords Concept generation · Sketching · Design process

1 Introduction

The generation of ideas is a key activity in the design process. This study focuses on one idea generation method known as brainstorming (Osborn 1963) which has been widely adopted in product design and development in industry (Sutton and Hargadon 1996; Kelley and Littman 2001). Brainstorming consists of a set of broad, process-

driven guidelines that can be applied in many contexts. However, Shah et al. (2000) points out that the results of brainstorming can be “unpredictable.” This study focuses on measuring a single aspect of the brainstorming process through one of its main guidelines: generate as many ideas as possible. By broadening the initial pool of ideas, the assumption is “quantity yields quality.”

Osborn’s rules have previously been examined to understand the role of social situations in creativity (Diehl and Stroebe 1991; Paulus et al. 1995; Paulus 2000). These experiments consider brainstorming through creativity exercises of brief duration and scope using subjects who generally do not have experience in design. This approach is valuable in that it limits confounding factors in the analysis of the creative process. This paper considers concept generation from another perspective, that of engineering design in multi-week long projects involving engineering design students.

This paper examines three hypotheses related to concept generation in design and builds on preliminary research by the author (Yang 2003). These hypotheses were formulated in the context of engineering design courses, and are intended to help design practitioners improve their creativity and design outcomes as well as assist educators in teaching design:

Hypothesis 1 The quantity of concepts generated at the beginning of a design project correlates with design outcome.

It has been suggested that the generation of more ideas initially leads to a higher incidence of better quality ideas. It may be further surmised that having a higher quality idea in the early stages of design process may be more conducive to a better final design outcome. Do individuals who generate more ideas also have better final design projects?

M. C. Yang (✉)
Department of Mechanical Engineering and Engineering Systems Division, Massachusetts Institute of Technology,
77 Massachusetts Avenue, Room 3-449B,
Cambridge, MA 02139, USA
e-mail: mcyang@mit.edu

This paper examines this question through the quantity of ideas generated by brainstorming. It also considers concept generation at a lower level through morphology charts which are hierarchies of alternative embodiments that can fulfill each of a device's functions (Zwicky 1969).

Hypothesis 2

The quantity of sketches generated during a project correlates with its design outcome.

Sketching is an integral part of the engineering design process. Sketches are representations of a design, but research suggests that the process itself of sketching is a fundamental element of design thinking (McKim 1980; Ullman et al. 1990; Schön and Wiggins 1992; Nagai and Noguchi 2003). In particular, the act of sketching is thought to be critical to generating concepts (Goldschmidt 1994; Goel 1995; Suwa and Tversky 1997; Purcell and Gero 1998; Verstijnen et al. 1998). Tovey et al. (2003) refer to sketching as a “language for handling design ideas.” The studies of Romer et al. (2000) posit that sketching is useful as a way to mentally offload concepts during complex design activity.

Observations of industrial practice suggest that design success is closely linked to realizing an idea through drawing and prototyping (Schrage and Peters 1999). Other research has assessed sketching to determine possible links to design outcome. Schütze et al. (2003) found that designers who were allowed to sketch during the design process produced a higher quality solution than those who were not permitted to. However, Bilda et al. (2006) suggest that sketching is not essential for design, although the more basic activity of mental imagery may be still be critical. Song and Agogino (2004) found significant correlations between sketch volume and design outcome in a product development course. Yang and Cham (2007) found no links between drawing skill and design outcome, but did show that skilled sketchers tended to draw more overall.

This paper investigates the quantity of design sketching as another way to infer concept generation volume. Sketches are divided into two categories: dimensioned drawings and non-dimensioned drawings. Dimensioned drawings are of interest because they may represent ideas that are further along in the design cycle and are therefore more developed than non-dimensioned ones.

Hypothesis 3

Increased sketching at the beginning of the project, rather than at the end, correlates with better design outcome.

Designers can, and do, form ideas throughout the design cycle. Concepts may occur under formal circumstances (for example, to meet a milestone) or more spontaneously, as a classic “eureka” moment. However, concurrent engineering holds that design decisions made in the early,

conceptual phases of design have greater impact on overall design than those made later on (Winner et al. 1988). This study observes whether sketches generated earlier or later on in the design cycle have different correlations with design outcome.

2 Related work

2.1 Measures of creativity

Idea quantity, or fluency, is but one of several measures for creativity. Other metrics for concept generation include flexibility, or the range of ideas, and originality, or novelty of idea (Guilford 1959). Shah et al. (2003) define formal metrics including novelty, variety, quality, and quantity which have been adopted by other researchers (Song and Agogino 2004; Chusilp and Jin 2006). This study focuses on primarily on quantity in part due to its relative objectivity as a measure across a range of idea representations such as sketches and textual lists of ideas.

2.2 Types of sketches

Sketches may be considered from several perspectives. Ferguson (1992) classifies three types of sketches in terms of their intended purpose: the thinking sketch serves as a reflective medium, the prescriptive sketch acts as a blueprint for design work, and the talking sketch supports design collaboration.

Sketches may be categorized by the design progression they represent. van der Lugt (2005) examines sketches as a mechanism for reinterpretation of an individual designer's ideas. Goel (1995) defines “lateral transformations” as incremental changes that build on a previous idea, while “vertical” ones result in a more detailed version of an earlier sketch. McGown et al. (1998) and Rodgers et al. (2000) label categories based on the physical elements of engineering sketches:

- Level 1: A simple monochrome line drawing that does not include shading or annotations
- Level 2: A detailed monochrome line drawing with annotations but no shading
- Level 3: A Level 2 drawing with shading to suggest 3D form
- Level 4: A Level 3 drawing with more gradations of shading and possibly color to emphasize 3D form
- Level 5: The most “realistic” type of sketch includes extensive shading and annotation

The sketches described in this paper can have aspects of all of the above categories of sketches.

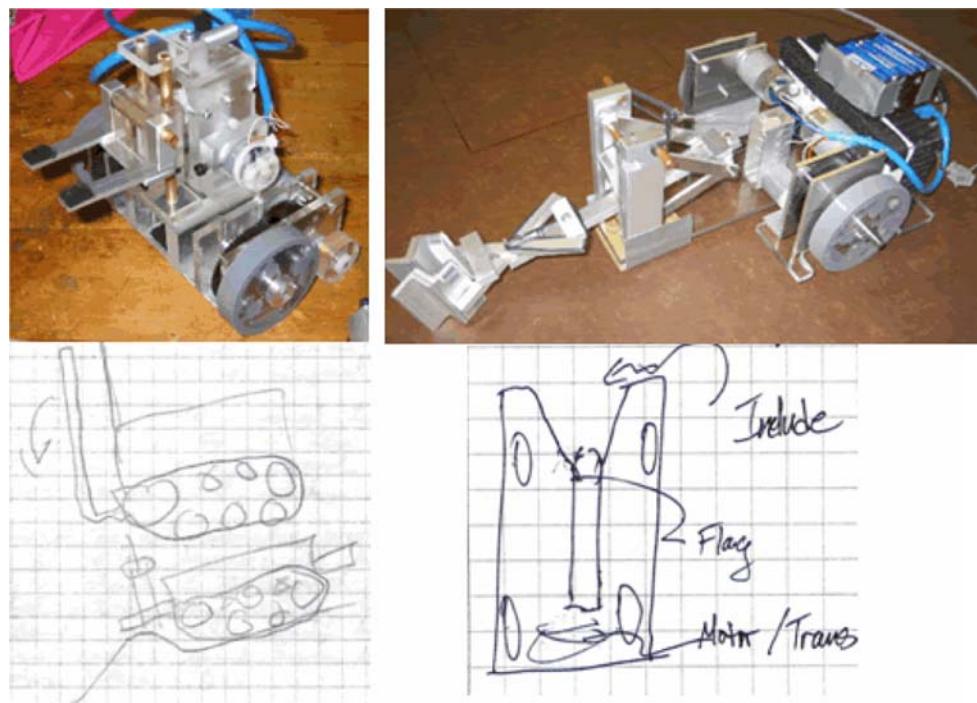
3 Methods

3.1 Test bed

The test beds for these experiments were two project-based, undergraduate mechanical engineering design courses at the California Institute of Technology. In all three courses, teams were provided with identical sets of materials from which to prototype.

The first course was in advanced engineering design. Data was collected from two separate years of this course. The advanced course had twenty-four students in one year (“Course 1”) and twenty-three in the next (“Course 2”). In both, teams of two students were presented with an open-ended, ill-defined design challenge such as a robotic capture-the-flag game. They were required to develop conceptual designs and fabricate a fully functional electro-mechanical device in the engineering machine shop. Potential design solutions could range from simple three-wheeled cars to robotic arms to combinations of custom mechanisms. The breadth of design scope provided ample opportunity for sketching many aspects of various design concepts. At the end of the ten-week course, teams competed against each other in a double elimination contest held before the entire campus. A single winner emerged. Example projects are shown in Fig. 1, along with a representative sketch from the corresponding logbooks. Note that though students worked and competed in pairs, they were expected to produce their own separate devices that were assessed independently from their teammate’s.

Fig. 1 Electromechanical design projects from the advanced design course (Courses 1 and 2). Below each photo is a sample sketch from the logbooks kept for each project



The second course was an introduction to design comprised of thirty-three students (“Course 3”) which was a pre-requisite to the advanced course. This course included a three-week long, open-ended design challenge in which students were asked to design and build a solution using “soft” materials, such as foamcore, to pop a helium-filled balloon suspended over a large water pond. Students worked in teams of three to five and were graded both as a team and as individuals. Sample projects are shown in Fig. 2, along with a drawing from the associated logbooks. As in the other course, design solutions could span a wide range of possibilities including hooking and grasping mechanisms and drivable boats. The range of potential solutions allowed a variety of sketches and ideas to be developed over the duration of the projects.

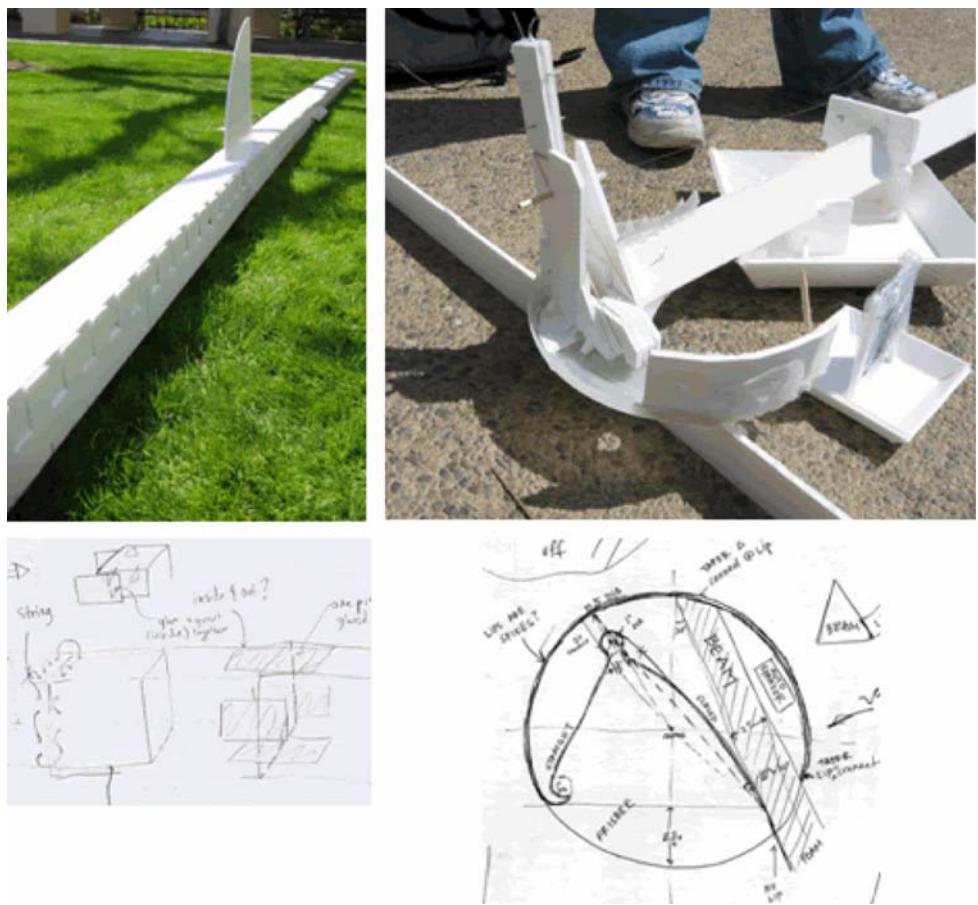
3.2 Design data

3.2.1 Brainstorming and morphology charts

Concept generation was examined through brainstorming and morphology charts. Brainstorming took place in the first third of the project for the Introductory Course. Each design team was presented with the design problem and then asked to generate and write down conceptual design alternatives in class and on their own over a span of a week. The number of brainstormed ideas generated by each student was counted.

Morphology charts were created in all three courses. Morphology charts require a more systematic, lower level

Fig. 2 Sample projects constructed of foam core from the introductory design course (Course 3). Below each photo is a sample sketch from the logbooks kept for each project



enumeration of concepts than brainstorming. Early in the projects for all three courses, each team developed a chart of the desired functions of their device, along with possible approaches to achieving those functions. For example, if the desired function was to “block an opponent,” ways to achieve that goal might be to “drive into the opponent” or “push the opponent.” Most of these embodiments were illustrated with thumbnail diagrams, and all included brief text descriptions. The total number of morphological embodiments was counted.

3.2.2 Sketching and logbooks

Paper-based design logbooks were kept by each student over the life of the project. Logbooks can be a comprehensive record of a student’s design process and thinking. The information archived in the logbooks varied widely in form, including detailed descriptions of work, plans for fabrication, engineering calculations, and sketches of many types and levels of detail.

Individual drawings in each logbook were counted and indexed by date and by whether it was dimensioned or not. Drawings were considered dimensioned if they included numeric labels for parameters such as length, width or

diameter. Such drawings may be interpreted as a step towards making a design more concrete.

As with any informal information, there is ambiguity as to what constitutes an individual drawing, and the goal was to be consistent between logbooks. In nearly all cases, however, distinct objects were obvious because of white space separation or clear annotations (such as indicating text or arrows) in the sketch. Ullman et al. (1990) uses “marks on paper” to refer to both sketches and annotation in order to capture the ambiguity inherent in defining design sketching. The quality of individual sketches was not considered, in part because other work suggests that sketch quality is not linked to design outcome (Yang and Cham 2007).

This study did not normalize for a student’s drawing ability or previous drawing experience. None of the students were given any sketch instruction. In Yang (2003), a survey was taken of one of the advanced courses (Course 1) to gauge the level of other relevant design experience that students might possess. The results of the survey showed that students generally felt they had above-average experience in engineering analysis, engineering intuition, engineering fabrication and arts and crafts. Students also self-reported below-average experience in drawing, tinkering, and construction.

Table 1 Summary of design courses and type of data captured

Design Course	Project length (weeks)	Design data collected		
		Brainstorming	Morphology	Sketch
1 (Adv)	10		×	×
2 (Adv)	10		×	×
3 (Intro)	3	×	×	

Table 1 summarizes the types of design data collected in each course.

3.3 Design outcome evaluation

Two indicators of design outcome were employed. The first was the individual final grade, counted in points, for each student. The maximum number of points possible was 100, and was based on overall performance over the 10-week period. For the introductory class (Course 3), project grade was also considered. Note that the final grade includes student effort for two other projects unrelated to the first project.

The second indicator, applied only to the advanced courses, was each team's final ranking in the contest. Contest performance was based on the number of rounds of competition that the team was able to win. The more rounds won, the higher the rank. Contest performance was decoupled from the final grade itself. It was possible for a team to perform poorly in the contest but earn a good grade, and vice versa. One reason for this is that the primary goal of the class is to teach engineering design and design process, and a student could demonstrate an excellent grasp of design process but still rank low in the contest.

4 Results and discussion

4.1 Type of sketching

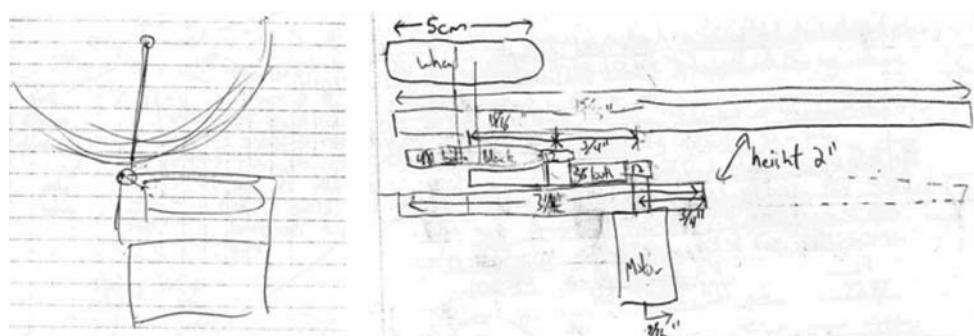
A total of 4,008 sketches were counted in the logbooks. 61.4% included dimensions and the remaining 38.6% did

not. The dimensioned sketches were generally considered “prescriptive” sketches in that many of them were likely meant as blueprints for fabrication. The remaining sketches were virtually all “thinking” sketches (Fig. 3). In these courses, logbooks were primarily a device for capturing the thinking of the individual designer, and this function was borne out in the sketches found in the logbooks. This is also consistent with the findings of Song and Agogino (2004).

Although the advanced course students were asked to document their design process in paper logbooks, students were not prohibited from using other media for visualizing their ideas, and access to CAD tools was freely available to all students. In one of the advanced courses, one out of the twenty-four students used CAD tools to create both dimensioned and non-dimensioned drawings. In the following year, six of twenty-three students employed CAD, although it is not clear why more students chose to do so. In this study, CAD drawings were treated in the same manner as hand sketches under the assumption that they are still representations of design thought. Also, at the time Ferguson wrote his book, Computer-Aided Design (CAD) tools were not as ubiquitous as they are today, and it would be reasonable to think many engineering designers create “prescriptive” sketches for themselves rather than a third party to formalize.

There are some challenges in judging sketches *post facto* for two reasons. First, without explicitly asking the designer's intended purpose with each sketch, the observer can only guess what the intention is. Second, a sketch may serve multiple purposes. For example, though many of the logbook sketches could be characterized as “thinking”

Fig. 3 Example “thinking” sketch, Level 1 detail from logbook (*left*). Example “prescriptive” sketch with Level 2 dimension details on *right*



sketches, their design work was performed in the context of the classroom, so it would be reasonable for students to expect that others (teammates or teaching staff) might refer to their sketches in the way they would a “talking” sketch. Evidence of this is seen in the way some sketches were annotated in first person narrative.

Although in most situations it was clear what mechanical component a particular drawing represented (for example, a wheel or a gear), the content of each sketch was not tracked. Since many of these logbook sketches were “thinking” sketches, it is difficult to interpret what the intention of the sketch was. A jumble of lines may be meaningless to the external observer, but hold important meaning for the sketcher. The quality of idea represented in a sketch was likewise not tracked.

Almost all drawings found in the logbooks were Levels 1 and 2, and occasionally Level 3, meaning that they were primarily line drawings with limited annotation and sometimes shading. It is important to note here that the levels of detail outlined by McGown et al. (1998) were based on observations of engineering students who had gone through art and industrial design training as part of their core curriculum. In comparison, the designers observed in this study are relative novices in sketching.

4.2 Concept quantity and design outcome

Hypothesis 1 The quantity of concepts generated at the beginning of a design project correlates with design outcome.

Table 2 shows that the quantity of ideas brainstormed had a statistically significant Spearman correlation (R_s) with both the three-week project grade and the overall final grade for the term.

Table 3 shows correlations between the quantity of morphological alternatives with both final grade and contest performance for Courses 1 and 2, and with grade only for Course 3. The number of morphological alternatives includes embodiments from all hierarchy levels. Note that for Course 1, $N = 20$ rather than 24 because of data unavailability, so for a significance level $\alpha = 0.05$, R_s must be greater than 0.377. Course 2 had even less data available for analysis ($N = 12$). Course 3 includes two design

Table 2 Correlations between brainstormed ideas and grade for Course 3 (introductory course)

	Correlation coefficient, R_s	
	Project grade	Final grade
Total ideas brainstormed	0.48	0.33
$N = 33, R_s = 0.291$ for $\alpha = 0.05$		

Table 3 Correlation of morphology alternatives for all courses

	N	R_s	Correlation coefficient, R_s	
			Grade	Contest
Course 1	20	0.377	0.19	0.160
Course 2	12	0.503	0.020	-0.07
Course 3	33	0.291	0.61 (project) 0.34 (final)	N/A N/A

outcome measures—the final grade for the course, and the grade for the three-week project only.

For Courses 1 and 2, the total morphological alternatives do not correlate positively with either grade or contest performance. In contrast, the introductory course (Course 3), correlated statistically significantly with both final and project grade.

Possible explanations for this difference between the two courses could include sample size, length of project, and level of prototyping skill required for the project. Many issues can arise during the development and fabrication that are not considered or are unforeseeable in the early, conceptual stage. In the introductory course, students need only have rudimentary soft prototyping skills, and the project is only three weeks long. The likelihood that their design will change is lower than for the advanced courses which require some level of skill and training in the machine shop and whose project lasts three times as long. The response to this hypothesis then is a “maybe.” In this case of a shorter, less involved design project such as found in the introductory course, this hypothesis holds true. However, in the more involved, longer duration project such as found in the advanced courses, this hypothesis was not found to be the case.

It should be pointed out that this finding does not necessarily mean that the number of ideas is a cause of project success, only that the two variables tend to increase and decrease together. Other factors that may also be linked to the final outcome of design will be discussed in Sect. 5.

4.3 Sketch quantity and design outcome

Hypothesis 2 The quantity of sketches generated during a project correlates with its design outcome.

No statistically significant correlations were found between the total quantity of either dimensioned or total sketches of any type and final grade or contest ranking for the three courses. Overall, prolific sketchers were no more likely to have good design outcomes than those who did not sketch as much. This diverges from the findings of Song and Agogino (2004), and it is theorized that this is due to the differing natures of the design projects. The

projects studied by Song were primarily product design projects with an emphasis on market studies. In contrast, Courses 1 and 2 focus on engineering design, with an emphasis on physical prototyping. This phenomenon is examined in closer detail in the following section.

This paper's findings may be further considered in light of the work of van der Lught (2005) who determined that idea generation was not related to better design outcomes unless designers "worked" the sketch through re-interpretive cycles. In this paper, the re-interpretation of sketches was not tracked, but may merit further examination in future work.

4.4 Sketching over time and design outcome

Hypothesis 3 Increased sketching at the beginning of the project, rather than at the end, correlates with better design outcome.

Figures 4 and 5 show the average number of sketches plotted by time in the advanced courses. The lower, darker section of each column represents the dimensioned drawings, and the entire column shows the total number of drawings (dimensioned and non-dimensioned).

The general trend in both courses is the same—fewer overall drawings in the beginning, more in the middle, and a drop off at the end of the project. There are also a proportionately greater number of total drawings during the first part of the term as compared to the remainder of the term. It is observed that the proportion of dimensioned sketches to the total number of drawings starts off low and increases towards the middle of the project. Taken together, these trends may indicate abstract conceptual design

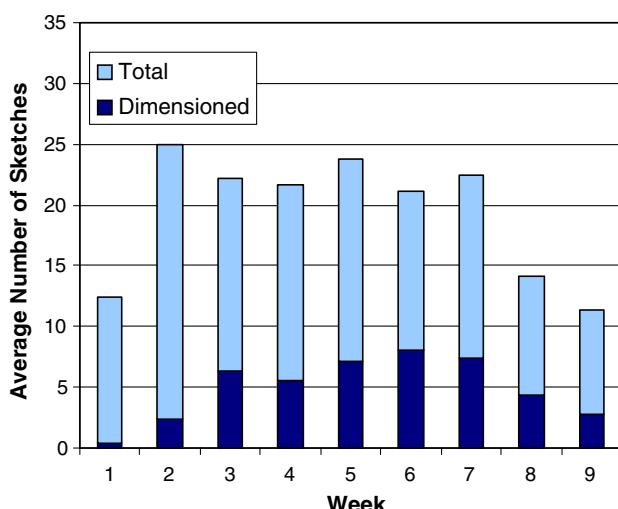


Fig. 4 Average weekly total and dimensioned sketches for Course 1 (Advanced Design)

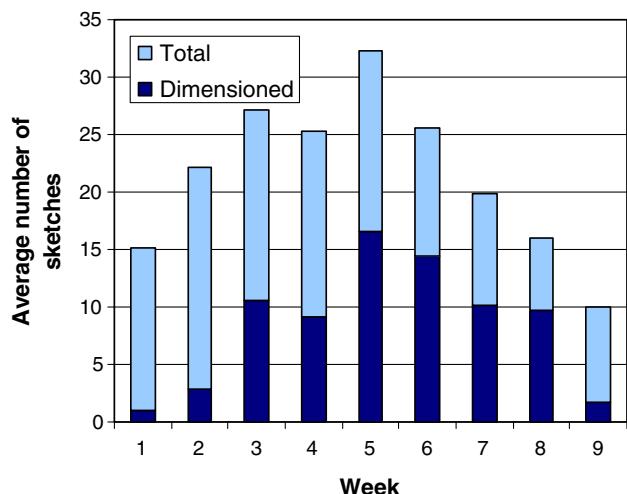


Fig. 5 Average weekly total and dimensioned sketches for Course 2 (Advanced Design)

activity early in the design cycle and more prototyping in the later stages of the design cycle.

There are differences, however, in when the peak number of drawings/sketches occur. We see that for the first advanced class (Course 1), the peak starts almost immediately in the second week. For the other advanced class (Course 2), the peak is not reached until the fifth week. A possible explanation might be due to the design milestones in which students present their work to the teaching staff, typically in the form of drawings and prototypes. Milestones serve as a project management guideline for the students, as well as a way for teaching staff to monitor progress. Interestingly, the timing of design milestones within the project is nearly identical in both courses.

Figures 6 and 7 list the Spearman correlation coefficients (R_s) for the average total non-dimensioned and dimensioned drawings per week correlated with final grades and contest performance each week of the term. Dotted horizontal lines show the threshold for statistical significance.

Figure 6 shows that dimensioned drawings and final grades are correlated in a statistically significant way during the first three weeks of the design cycle. The total number of drawings is also significantly correlated with grade during the third week, and in the first week with contest rank. This result is notable because it shows that the creation of more concrete dimensioned drawings early in the design cycle has a positive correlation with design outcome.

The trend in Fig. 7 is somewhat different. Dimensioned drawings correspond in a statistically significant way in the third and fourth weeks and total drawings correlate with contest performance only in the fifth week. In this case, dimensioned drawings are still important, but not until

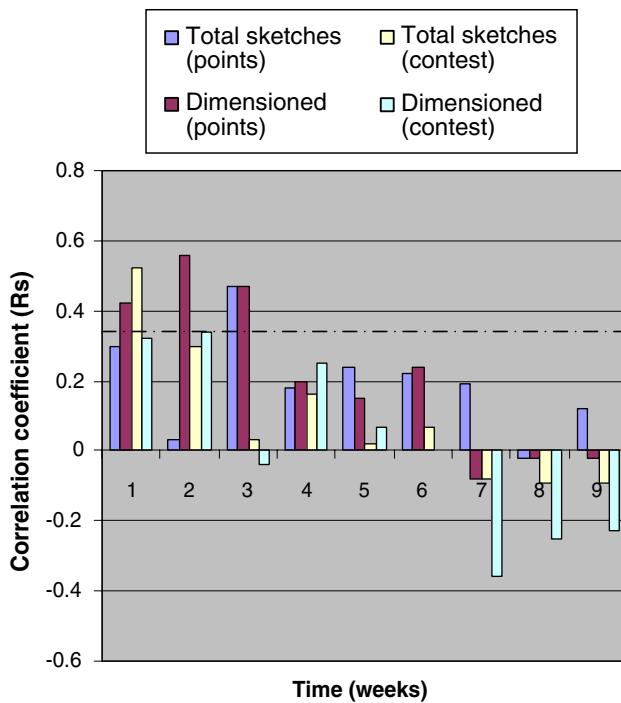


Fig. 6 Correlations between total and dimensioned sketches by week and design outcome for Advanced Course (Course 1). $N = 24$ and $R_s = 0.344$ (denoted by dotted line) for $\alpha = 0.05$

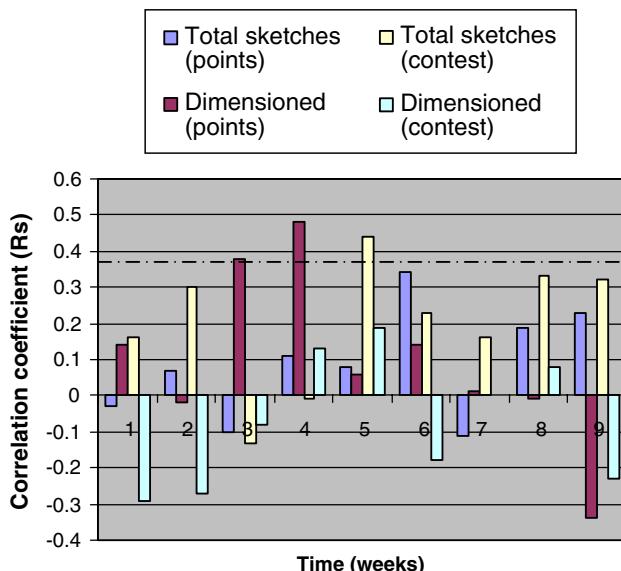


Fig. 7 Correlations between total and dimensioned sketches by week and design outcome for Advanced Course (Course 2). $N = 21$ and $R_s = 0.370$ (denoted by dotted line) for $\alpha = 0.05$

slightly later in the early-to-middle phases of the design cycle.

In both cases, the correspondences are roughly coincident with each course's peak sketch volume in Figs. 6 and 7. These peaks occur around the periodic design reviews in which students must present their project work to the

teaching staff for feedback. Average student sketching activity is at its highest when it is most strongly correlated with design grade. One possible explanation is the nature of design performance. Both final grades and contest ranking are somewhat relative measures. While it is possible that all students in a course can earn an "A," it is more likely that students will receive grades that fall along a distribution relative to each other. At the same time, the contest itself is intended to produce relative rankings, with clear standings. It makes sense, then, that the relative efforts of a student during peak design activity might be associated with design outcome.

Interestingly, the correlations with grades tend to become more negative over time. Correlations with design activity early in the design cycle are more positive, and later in the design cycle are more negative, which suggests that last minute efforts at sketching are not consistent with a good outcome.

In these cases, the proposed hypothesis is shown to be true. In addition, the role of milestones in planning design work may be important, and falling behind in design is linked to poorer outcomes.

5 Conclusions

5.1 Concept generation and sketching

Concept generation measured in the form of morphology charts showed a statistically significant correlation with both project and final term grade in the introductory course. However, morphology charts in the advanced courses did not show a statistically significant correlation. This may be due in part to the inherent differences of methods to generate concepts. Strictly speaking, brainstorming is meant to generate a wide range of concepts, while morphology charts are only intended to enumerate mechanisms for achieving specific functions. The key difference in these two courses is the timing of concept generation and design outcome. A shorter elapsed project time combined with less emphasis on detailed fabrication in the introductory class suggests that the role of idea quantity is less important the further along in the design cycle outcome is measured.

One of the unexpected findings of this study was the role of timing in concept generation and sketching. Data from the advanced courses shows that dimensioned sketch quantity is significantly correlated to design grades in the first half of the design cycle.

Total sketching volume measured at the end of the term, including both dimensioned and non-dimensioned drawings, did not correlate with graded design outcome or contest performance. This suggests that it is possible to produce copious drawings during the design process and

still have an unsuccessful graded design. Likewise, it is implied that a designer can sketch very little overall and achieve a better design grade, as long as the bulk of dimensioned drawings (and perhaps prototyping) are created early on in the design process. This suggests that starting work early on a project is beneficial.

It should be noted that there can be a range of approaches to maintaining a logbook. Design logbooks capture a wide variety of information in a flexible way. Many students detail their design activity meticulously. For others, however, logbooks may not always be a complete record of design thinking. This may be due in part to the overhead involved with keeping a logbook. As one student put it, “I do design in my head” rather than write thoughts down.

5.2 Design outcome

Two design outcomes were used in this study: final grade and contest results. Only total sketching volume correlated significantly with contest results. This is interesting because in some ways, the contest simulates the conditions of real world design in that the situations in which a product is released are unpredictable, unlike a classroom setting that is controlled and “safe.” While students were provided with as realistic a testing environment as possible throughout the term, the actual contest included technical issues that could not be replicated beforehand, such as electrical interference, as well as non-technical issues such as the stress experienced by contestants competing in front of their peers. However, unlike the real world, a contest is an artificial construct. In real-world design, there is rarely an absolute notion of winning or losing, and the definition of success is open to interpretation.

The final design grade in these cases is comprehensive and assesses the overall design process. In industry, a product can be developed following a “good” process but the result can still be thought of as unsuccessful. Contests, on the other hand, focus on end products instead of process. In either case, the development of appropriate design outcome measures that are viable for evaluation is an open area of design research.

5.3 Sketching and prototyping

Dimensioned drawings are a logical precursor to prototyping. These results suggest that prototyping earlier in the design cycle rather than later is linked to better design grades, perhaps because starting fabrication earlier leaves more time for iteration and refinement of a design.

In some ways, the correlation between early dimensioned sketching is somewhat counter-intuitive. The rules of brainstorming hold that designers should withhold judgment on a set of ideas before selecting one for further

development and manufacturing. Concurrent engineering as well as system engineering posit that more time spent up front to understand the potential implications of design decisions downstream is beneficial to design. However, the results of this study suggest that selecting a design path and prototyping earlier on is preferable. This may have to do with the fact that the course being studied has a strong emphasis on prototyping a fully functional design. In fact, Yang (2003) found that the students who self-reported having prior engineering building experience and engineering intuition that correlated significantly with design outcome. The suggestion that earlier prototyping is consistent with better design results aligns with the findings of Ward et al. (1995). Ward examined the highly successful design and manufacturing process at Toyota Motor Company. Toyota’s counter-intuitive approach at the time was to prototype many design alternatives before selecting one for full production.

5.4 Limitations

It should be emphasized that all three of the hypotheses included in this paper rely on the notion of correlation rather than causation. The products of the earliest stages of the design process, such as concepts and sketches, are not examined as the sole causes of a design outcome, but as co-occurrences with design outcome that may merit further study. One of the challenges is that numerous steps are involved in design beyond the formulation stage, including fabrication and testing, that may have little to do with the initial ideas generated. Furthermore, there may be interactions among these steps that introduce additional confounding factors.

At the same time, it should be pointed out that those designers who generate greater quantities of ideas or produce more sketches or start prototyping their projects earlier may possess characteristics that will facilitate stronger design outcomes. These designers may be more motivated, more creative, and perhaps better able to address the challenges of creating a more successful design, and thus produce a better design outcome.

6 Future work

This study draws on guidelines from design practice to correlate concept quantity and sketching with design outcome. However, this research is one of many efforts to better understand sketching and concept generation in engineering design. Future work will consider specific aspects of sketches that reflect cognitive elements of concept generation and sketching as found in Shah et al. (2000, 2001, 2003). In addition, there are a number of other types

of design activity that merit study to understand their possible links to design. Song and Agogino (2004) point out the importance of considering other modes of design thinking, in particular verbal/textual cognition. Indeed, one of the best performing designers in the advanced class did little or no drawing, but meticulously documented his design activity in textual form.

A potential impact of this work will be derived from the development of relevant design tools to facilitate design information handling in design teams, both in the classroom and in industry practice. What are ways to formulate ideation methods and sketching guidelines so that they can help students become more effective designers?

The role of the design logbook as a design tool should also be examined. This work focuses both on individual and team design efforts, and in real-world design situations, products are often the end result of collaborative team work. Paper design logbooks such as used in these courses have long been employed to capture design knowledge, but electronic versions of these hold promise as tools to facilitate design thinking and process. There is relevant research in the area of electronic design notebooks (Lakin et al. 1989; Hong et al. 1994; Viste and Cannon 1995) that specifically supports design team collaboration. There exists a related body of research in textual design information analysis (Baya and Leifer 1994; Dong and Agogino 1997; Wood et al. 1998) as well as in electronic sketch capture (Yen et al. 1999) that may help design tools to be more intelligent and designers to be better designers. However, much work remains to be done in the research and development of design tools such as these. One current stumbling block is that such computational tools do not yet provide the same level of usability and critical affordances that simple paper and pen provide (Alvarado and Davis 2001). Some ideal features include portability, ease of sketching, and seamless integration with text tools. In addition, these tools need to allow ease of annotation of design sketches with rough dimensions and notes. More formal methods for visualization, such as CAD systems, are appropriate for representing later stage designs, but at the conceptual stage, the goal is to encourage agile, unimpeded concept generation (Kavakli et al. 1998). Work in this area focuses both on software to recognize sketches symbolically (Do et al. 2000; Kurtoglu and Stahovich 2002), as well as computer input devices to allow sketches to be created more easily (Dickinson et al. 2003).

Future work should focus on integrating aspects of these research areas to produce cohesive tools to support design teams and the design process itself.

Acknowledgments The author gratefully acknowledges the thoughtful support and guidance of Prof. Erik Antonsson, Prof. Joel Burdick, and Dr. Curtis Collins, and the design efforts of the students

that are the foundation of this research. The work described in this paper was supported in part by the National Science Foundation under Award DMI-0547629. The opinions, findings, conclusions and recommendations expressed are those of the author and do not necessarily reflect the views of the sponsors. The author also acknowledges the generous sponsors of the advanced design course: Applied Materials, Amerigon, Dr. David and Mrs. Barbara Groce, Honeywell, idealab!, Mabuchi Motor, Northrop Grumman, The San Diego Foundation, and Toro.

References

- Alvarado CJ, Davis R (2001) Preserving the freedom of paper in a computer-based sketch tool. *HCI International* 2001, New Orleans, Louisiana. Lawrence Erlbaum Associates, Inc, USA
- Baya V and Leifer LJ (1994). A study of the information handling behavior of designers during conceptual design. Sixth International conference on design theory and methodology. American Society of Mechanical Engineers, Minneapolis
- Bilda Z, Gero JS, Purcell T (2006) To sketch or not to sketch? That is the question. *Des Stud* 27(5):587–613
- Chusilp P, Jin Y (2006) Impact of mental iteration on concept generation. *ASME J Mech Des* 128(1):14–25
- Dickinson JK, Pardasani A, Yu Z, Zeng Y, Antunes H, Li Z (2003) Augmenting mechanical CAD with pen and tablet. ASME design engineering technical conferences. ASME, Chicago
- Diehl M, Stroebe W (1991) Productivity loss in idea-generating groups: tracking down the blocking effect. *J Pers Soc Psychol* 61(3):392–403
- Do EY-L, Gross MD, Neiman B, Zimring C (2000) Intentions in and relations among design drawings. *Des Stud* 21(5):483–503
- Dong A, Agogino AM (1997) Text analysis for constructing design representations. *Artif Intell Eng* 11(2):65–75
- Ferguson ES (1992) Engineering and the mind's eye. The MIT Press, Cambridge
- Goel V (1995) Sketches of thought. MIT Press, Cambridge
- Goldschmidt G (1994) On visual design thinking: the vis kids of architecture. *Des Stud* 15(2):158–174
- Guilford JP (1959) Personality. McGraw-Hill, New York
- Hong J, Toye G and Leifer L (1994). Using the WWW for a Team-Based Engineering Design Class. Second WWW Conference, Chicago, IL
- Kavakli M, Scrivener SAR, Ball LJ (1998) Structure in idea sketching behaviour. *Des Stud* 19(4):485–517
- Kelley T, Littman J (2001) The Art of innovation: lessons in creativity from IDEO, America's leading design firm. Doubleday, New York
- Kurtoglu T and Stahovich TF (2002). Interpreting schematic sketches using physical reasoning. AAAI spring symposium 2002, sketch understanding
- Lakin F, Wambaugh J, Leifer L, Cannon D, Sivard C (1989) The electronic design notebook: performing medium and processing medium. *Vis Comput* 5:214–226
- McGown A, Green G, Rodgers PA (1998) Visible ideas: information patterns of conceptual sketch activity. *Des Stud* 19(4):431–453
- McKim RH (1980) Experiences in visual thinking. PWS Publishers, Boston
- Nagai Y, Noguchi H (2003) An experimental study on the design thinking process started from difficult keywords: modeling the thinking process of creative design. *J Eng Des* 14(4):429–437
- Osborn AF (1963) Applied imagination. Charles Scribner and Sons, New York
- Paulus P (2000) Groups, teams, and creativity: the creative potential of idea generating groups. *Appl Psychol* 49(2):237–262

- Paulus PB, Larey TS, Ortega AH (1995) Performance and perceptions of brainstormers in an organizational setting. *Basic Appl Soc Psych* 17(1&2):249–265
- Purcell AT, Gero JS (1998) Drawings and the design process. *Des Stud* 19(4):389–430
- Rodgers PA, Green G, McGown A (2000) Using concept sketches to track design progress. *Des Stud* 21(5):451–464
- Romer A, Leinert S, Sachse P (2000) External support of problem analysis in design problem solving. *Res Eng Design* 12(3):144–151
- Schön DA, Wiggins G (1992) Kinds of seeing and their functions in designing. *Des Stud* 13(2):135–156
- Schrage M, Peters T (1999) Serious play: how the World's best companies simulate to innovate. Harvard Business School Press, Boston
- Schütze M, Sachse P, Römer A (2003) Support value of sketching in the design process. *Res Eng Design* 14(2):89–97
- Shah JJ, Kulkarni SV, Vargas-Hernandez N (2000) Evaluation of idea generation methods for conceptual design: effectiveness metrics and design of experiments. *J Mech Des* 122(4):377–384
- Shah J, Vargas-Hernandez N, Summers J, Kulkarni S (2001) Collaborative sketching (C-Sketch): an idea generation technique for engineering design. *J Creat Behav* 35(3):168–198
- Shah JJ, Vargas-Hernandez N, Smith SM (2003) Metrics for measuring ideation effectiveness. *Des Stud* 24(2):111–134
- Song S, Agogino AM (2004) Insights on designers' sketching activities in product design teams. 2004 ASME design engineering technical conference. ASME, Salt Lake City
- Sutton RI, Hargadon A (1996) Brainstorming groups in context: effectiveness in a product design firm. *Adm Sci Q* 41(4):685–718
- Suwa M, Tversky B (1997) What do architects and students perceive in their design sketches? a protocol analysis. *Des Stud* 18(4):385–403
- Tovey M, Porter S, Newman R (2003) Sketching, concept development and automotive design. *Des Stud* 24(2):135–153
- Ullman DG, Wood S, Craig D (1990) The importance of drawing in the mechanical design process. *Comput Graph* 14(2):263–274
- van der Lugt R (2005) How sketching can affect the idea generation process in design group meetings. *Des Stud* 26(2):101–122
- Verstijnen IM, van Leeuwen C, Goldschmidt G, Hamel R, Hennessey JM (1998) Sketching and creative discovery. *Des Stud* 19(4):519–546
- Viste MJ, Cannon DM (1995) Firmware design capture. ASME design theory and methodology conference. American Society of Mechanical Engineers, Boston
- Ward A, Liker JK, Sobek D, Cristiano J (1995) The second toyota paradox: how delaying decisions can make better cars faster. *Sloan Manag Rev* 36(3):43–61
- Winner RI, Pennell JP, Bertrand HE, och Slusarczuk MMG (1988) The role of concurrent engineering in weapon systems acquisition. Institute for Defense Analysis (IDA), Boston
- Wood WH, Yang MC, Cutkosky MR, Agogino A (1998) Design information retrieval: improving access to the informal side of design. 1998 design engineering technical conferences 10th international conference on design theory and methodology. ASME, Atlanta
- Yang MC (2003) Concept generation and sketching: correlations with design outcome. 2003 ASME design engineering technical conferences. ASME, Chicago
- Yang MC, Cham JG (2007) An analysis of sketching skill and its role in early stage engineering design. *J Mech Des* 129(5):476–482
- Yen SJ, Fruchter R, Leifer L (1999) Facilitating tacit knowledge capture and reuse in conceptual design activities. ASME design engineering technical conferences, 11th international conference on design theory and methodology. ASME Press, Las Vegas
- Zwick F (1969) Discovery, invention, research through the morphological approach. MacMillan, New York

THE
DESIGN
OF EVERYDAY
THINGS

HUMAN ERROR? NO, BAD DESIGN



Most industrial accidents are caused by human error: estimates range between 75 and 95 percent. How is it that so many people are so incompetent? Answer: They aren't. It's a design problem.

If the number of accidents blamed upon human error were 1 to 5 percent, I might believe that people were at fault. But when the percentage is so high, then clearly other factors must be involved. When something happens this frequently, there must be another underlying factor.

When a bridge collapses, we analyze the incident to find the causes of the collapse and reformulate the design rules to ensure that form of accident will never happen again. When we discover that electronic equipment is malfunctioning because it is responding to unavoidable electrical noise, we redesign the circuits to be more tolerant of the noise. But when an accident is thought to be caused by people, we blame them and then continue to do things just as we have always done.

Physical limitations are well understood by designers; mental limitations are greatly misunderstood. We should treat all failures in the same way: find the fundamental causes and redesign the system so that these can no longer lead to problems. We design

equipment that requires people to be fully alert and attentive for hours, or to remember archaic, confusing procedures even if they are only used infrequently, sometimes only once in a lifetime. We put people in boring environments with nothing to do for hours on end, until suddenly they must respond quickly and accurately. Or we subject them to complex, high-workload environments, where they are continually interrupted while having to do multiple tasks simultaneously. Then we wonder why there is failure.

Even worse is that when I talk to the designers and administrators of these systems, they admit that they too have nodded off while supposedly working. Some even admit to falling asleep for an instant while driving. They admit to turning the wrong stove burners on or off in their homes, and to other small but significant errors. Yet when their workers do this, they blame them for “human error.” And when employees or customers have similar issues, they are blamed for not following the directions properly, or for not being fully alert and attentive.

Understanding Why There Is Error

Error occurs for many reasons. The most common is in the nature of the tasks and procedures that require people to behave in unnatural ways—staying alert for hours at a time, providing precise, accurate control specifications, all the while multitasking, doing several things at once, and subjected to multiple interfering activities. Interruptions are a common reason for error, not helped by designs and procedures that assume full, dedicated attention yet that do not make it easy to resume operations after an interruption. And finally, perhaps the worst culprit of all, is the attitude of people toward errors.

When an error causes a financial loss or, worse, leads to an injury or death, a special committee is convened to investigate the cause and, almost without fail, guilty people are found. The next step is to blame and punish them with a monetary fine, or by firing or jailing them. Sometimes a lesser punishment is proclaimed: make the guilty parties go through more training. Blame and punish; blame and train. The investigations and resulting punishments feel

good: “We caught the culprit.” But it doesn’t cure the problem: the same error will occur over and over again. Instead, when an error happens, we should determine why, then redesign the product or the procedures being followed so that it will never occur again or, if it does, so that it will have minimal impact.

ROOT CAUSE ANALYSIS

Root cause analysis is the name of the game: investigate the accident until the single, underlying cause is found. What this ought to mean is that when people have indeed made erroneous decisions or actions, we should determine what caused them to err. This is what root cause analysis ought to be about. Alas, all too often it stops once a person is found to have acted inappropriately.

Trying to find the cause of an accident sounds good but it is flawed for two reasons. First, most accidents do not have a single cause: there are usually multiple things that went wrong, multiple events that, had any one of them not occurred, would have prevented the accident. This is what James Reason, the noted British authority on human error, has called the “Swiss cheese model of accidents” (shown in Figure 5.3 of this chapter on page 208, and discussed in more detail there).

Second, why does the root cause analysis stop as soon as a human error is found? If a machine stops working, we don’t stop the analysis when we discover a broken part. Instead, we ask: “Why did the part break? Was it an inferior part? Were the required specifications too low? Did something apply too high a load on the part?” We keep asking questions until we are satisfied that we understand the reasons for the failure: then we set out to remedy them. We should do the same thing when we find human error: We should discover what led to the error. When root cause analysis discovers a human error in the chain, its work has just begun: now we apply the analysis to understand why the error occurred, and what can be done to prevent it.

One of the most sophisticated airplanes in the world is the US Air Force’s F-22. However, it has been involved in a number of accidents, and pilots have complained that they suffered oxygen

deprivation (hypoxia). In 2010, a crash destroyed an F-22 and killed the pilot. The Air Force investigation board studied the incident and two years later, in 2012, released a report that blamed the accident on pilot error: “failure to recognize and initiate a timely dive recovery due to channelized attention, breakdown of visual scan and unrecognized spatial distortion.”

In 2013, the Inspector General’s office of the US Department of Defense reviewed the Air Force’s findings, disagreeing with the assessment. In my opinion, this time a proper root cause analysis was done. The Inspector General asked “why sudden incapacitation or unconsciousness was not considered a contributory factor.” The Air Force, to nobody’s surprise, disagreed with the criticism. They argued that they had done a thorough review and that their conclusion “was supported by clear and convincing evidence.” Their only fault was that the report “could have been more clearly written.”

It is only slightly unfair to parody the two reports this way:

Air Force: It was pilot error—the pilot failed to take corrective action.

Inspector General: That’s because the pilot was probably unconscious.

Air Force: So you agree, the pilot failed to correct the problem.

THE FIVE WHYS

Root cause analysis is intended to determine the underlying cause of an incident, not the proximate cause. The Japanese have long followed a procedure for getting at root causes that they call the “Five Whys,” originally developed by Sakichi Toyoda and used by the Toyota Motor Company as part of the Toyota Production System for improving quality. Today it is widely deployed. Basically, it means that when searching for the reason, even after you have found one, do not stop: ask why that was the case. And then ask why again. Keep asking until you have uncovered the true underlying causes. Does it take exactly five? No, but calling the procedure “Five Whys” emphasizes the need to keep going even after a reason has been found. Consider how this might be applied to the analysis of the F-22 crash:

Five Whys

Question	Answer
Q1: Why did the plane crash?	Because it was in an uncontrolled dive.
Q2: Why didn't the pilot recover from the dive?	Because the pilot failed to initiate a timely recovery.
Q3: Why was that?	Because he might have been unconscious (or oxygen deprived).
Q4: Why was that?	We don't know. We need to find out.
Etc.	

The Five Whys of this example are only a partial analysis. For example, we need to know why the plane was in a dive (the report explains this, but it is too technical to go into here; suffice it to say that it, too, suggests that the dive was related to a possible oxygen deprivation).

The Five Whys do not guarantee success. The question *why* is ambiguous and can lead to different answers by different investigators. There is still a tendency to stop too soon, perhaps when the limit of the investigator's understanding has been reached. It also tends to emphasize the need to find a single cause for an incident, whereas most complex events have multiple, complex causal factors. Nonetheless, it is a powerful technique.

The tendency to stop seeking reasons as soon as a human error has been found is widespread. I once reviewed a number of accidents in which highly trained workers at an electric utility company had been electrocuted when they contacted or came too close to the high-voltage lines they were servicing. All the investigating committees found the workers to be at fault, something even the workers (those who had survived) did not dispute. But when the committees were investigating the complex causes of the incidents, why did they stop once they found a human error? Why didn't they keep going to find out why the error had occurred, what circumstances had led to it, and then, why those circumstances had happened? The committees never went far enough to find the deeper, root causes of the accidents. Nor did they consider redesigning the systems and procedures to make the incidents

either impossible or far less likely. When people err, change the system so that type of error will be reduced or eliminated. When complete elimination is not possible, redesign to reduce the impact.

It wasn't difficult for me to suggest simple changes to procedures that would have prevented most of the incidents at the utility company. It had never occurred to the committee to think of this. The problem is that to have followed my recommendations would have meant changing the culture from an attitude among the field workers that "We are supermen: we can solve any problem, repair the most complex outage. We do not make errors." It is not possible to eliminate human error if it is thought of as a personal failure rather than as a sign of poor design of procedures or equipment. My report to the company executives was received politely. I was even thanked. Several years later I contacted a friend at the company and asked what changes they had made. "No changes," he said. "And we are still injuring people."

One big problem is that the natural tendency to blame someone for an error is even shared by those who made the error, who often agree that it was their fault. People do tend to blame themselves when they do something that, after the fact, seems inexcusable. "I knew better," is a common comment by those who have erred. But when someone says, "It was my fault, I knew better," this is not a valid analysis of the problem. That doesn't help prevent its recurrence. When many people all have the same problem, shouldn't another cause be found? If the system lets you make the error, it is badly designed. And if the system induces you to make the error, then it is really badly designed. When I turn on the wrong stove burner, it is not due to my lack of knowledge: it is due to poor mapping between controls and burners. Teaching me the relationship will not stop the error from recurring: redesigning the stove will.

We can't fix problems unless people admit they exist. When we blame people, it is then difficult to convince organizations to restructure the design to eliminate these problems. After all, if a person is at fault, replace the person. But seldom is this the case: usually the system, the procedures, and social pressures have led

to the problems, and the problems won't be fixed without addressing all of these factors.

Why do people err? Because the designs focus upon the requirements of the system and the machines, and not upon the requirements of people. Most machines require precise commands and guidance, forcing people to enter numerical information perfectly. But people aren't very good at great precision. We frequently make errors when asked to type or write sequences of numbers or letters. This is well known: so why are machines still being designed that require such great precision, where pressing the wrong key can lead to horrendous results?

People are creative, constructive, exploratory beings. We are particularly good at novelty, at creating new ways of doing things, and at seeing new opportunities. Dull, repetitive, precise requirements fight against these traits. We are alert to changes in the environment, noticing new things, and then thinking about them and their implications. These are virtues, but they get turned into negative features when we are forced to serve machines. Then we are punished for lapses in attention, for deviating from the tightly prescribed routines.

A major cause of error is time stress. Time is often critical, especially in such places as manufacturing or chemical processing plants and hospitals. But even everyday tasks can have time pressures. Add environmental factors, such as poor weather or heavy traffic, and the time stresses increase. In commercial establishments, there is strong pressure not to slow the processes, because doing so would inconvenience many, lead to significant loss of money, and, in a hospital, possibly decrease the quality of patient care. There is a lot of pressure to push ahead with the work even when an outside observer would say it was dangerous to do so. In many industries, if the operators actually obeyed all the procedures, the work would never get done. So we push the boundaries: we stay up far longer than is natural. We try to do too many tasks at the same time. We drive faster than is safe. Most of the time we manage okay. We might even be rewarded and praised for our he-

roic efforts. But when things go wrong and we fail, then this same behavior is blamed and punished.

Deliberate Violations

Errors are not the only type of human failures. Sometimes people knowingly take risks. When the outcome is positive, they are often rewarded. When the result is negative, they might be punished. But how do we classify these deliberate violations of known, proper behavior? In the error literature, they tend to be ignored. In the accident literature, they are an important component.

Deliberate deviations play an important role in many accidents. They are defined as cases where people intentionally violate procedures and regulations. Why do they happen? Well, almost everyone of us has probably deliberately violated laws, rules, or even our own best judgment at times. Ever go faster than the speed limit? Drive too fast in the snow or rain? Agree to do some hazardous act, even while privately thinking it foolhardy to do so?

In many industries, the rules are written more with a goal toward legal compliance than with an understanding of the work requirements. As a result, if workers followed the rules, they couldn't get their jobs done. Do you sometimes prop open locked doors? Drive with too little sleep? Work with co-workers even though you are ill (and might therefore be infectious)?

Routine violations occur when noncompliance is so frequent that it is ignored. Situational violations occur when there are special circumstances (example: going through a red light “because no other cars were visible and I was late”). In some cases, the only way to complete a job might be to violate a rule or procedure.

A major cause of violations is inappropriate rules or procedures that not only invite violation but encourage it. Without the violations, the work could not be done. Worse, when employees feel it necessary to violate the rules in order to get the job done and, as a result, succeed, they will probably be congratulated and rewarded. This, of course, unwittingly rewards noncompliance. Cultures that encourage and commend violations set poor role models.

Although violations are a form of error, these are organizational and societal errors, important but outside the scope of the design of everyday things. The human error examined here is unintentional: deliberate violations, by definition, are intentional deviations that are known to be risky, with the potential of doing harm.

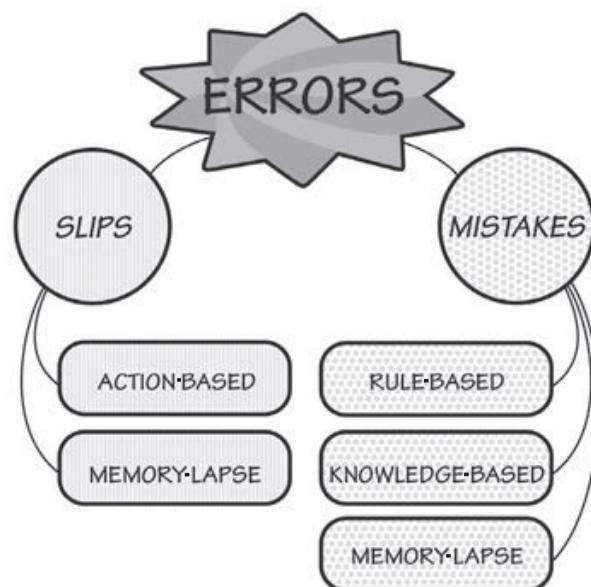
Two Types of Errors: Slips and Mistakes

Many years ago, the British psychologist James Reason and I developed a general classification of human error. We divided human error into two major categories: slips and mistakes (Figure 5.1). This classification has proved to be of value for both theory and practice. It is widely used in the study of error in such diverse areas as industrial and aviation accidents, and medical errors. The discussion gets a little technical, so I have kept technicalities to a minimum. This topic is of extreme importance to design, so stick with it.

DEFINITIONS: ERRORS, SLIPS, AND MISTAKES

Human error is defined as any deviance from “appropriate” behavior. The word *appropriate* is in quotes because in many circumstances, the appropriate behavior is not known or is only deter-

FIGURE 5.1. Classification of Errors. Errors have two major forms. Slips occur when the goal is correct, but the required actions are not done properly: the execution is flawed. Mistakes occur when the goal or plan is wrong. Slips and mistakes can be further divided based upon their underlying causes. Memory lapses can lead to either slips or mistakes, depending upon whether the memory failure was at the highest level of cognition (mistakes) or at lower (subconscious) levels (slips). Although deliberate violations of procedures are clearly inappropriate behaviors that often lead to accidents, these are not considered as errors (see discussion in text).



mined after the fact. But still, error is defined as deviance from the generally accepted correct or appropriate behavior.

Error is the general term for all wrong actions. There are two major classes of error: *slips* and *mistakes*, as shown in Figure 5.1; slips are further divided into two major classes and mistakes into three. These categories of errors all have different implications for design. I now turn to a more detailed look at these classes of errors and their design implications.

SLIPS

A slip occurs when a person intends to do one action and ends up doing something else. With a slip, the action performed is not the same as the action that was intended.

There are two major classes of slips: *action-based* and *memory-lapse*. In action-based slips, the wrong action is performed. In lapses, memory fails, so the intended action is not done or its results not evaluated. Action-based slips and memory lapses can be further classified according to their causes.

Example of an action-based slip. I poured some milk into my coffee and then put the coffee cup into the refrigerator. This is the correct action applied to the wrong object.

Example of a memory-lapse slip. I forgot to turn off the gas burner on my stove after cooking dinner.

MISTAKES

A mistake occurs when the wrong goal is established or the wrong plan is formed. From that point on, even if the actions are executed properly they are part of the error, because the actions themselves are inappropriate—they are part of the wrong plan. With a mistake, the action that is performed matches the plan: it is the plan that is wrong.

Mistakes have three major classes: *rule-based*, *knowledge-based*, and *memory-lapse*. In a rule-based mistake, the person has appropriately diagnosed the situation, but then decided upon an erroneous course of action: the wrong rule is being followed. In a knowledge-based mistake, the problem is misdiagnosed because

of erroneous or incomplete knowledge. Memory-lapse mistakes take place when there is forgetting at the stages of goals, plans, or evaluation. Two of the mistakes leading to the “Gimli Glider” Boeing 767 emergency landing were:

Example of knowledge-based mistake. Weight of fuel was computed in pounds instead of kilograms.

Example of memory-lapse mistake. A mechanic failed to complete troubleshooting because of distraction.

ERROR AND THE SEVEN STAGES OF ACTION

Errors can be understood through reference to the seven stages of the action cycle of Chapter 2 (Figure 5.2). Mistakes are errors in setting the goal or plan, and in comparing results with expectations—the higher levels of cognition. Slips happen in the execution of a plan, or in the perception or interpretation of the outcome—the lower stages. Memory lapses can happen at any of the eight transitions between stages, shown by the X’s in Figure 5.2B. A memory lapse at one of these transitions stops the action cycle from proceeding, and so the desired action is not completed.

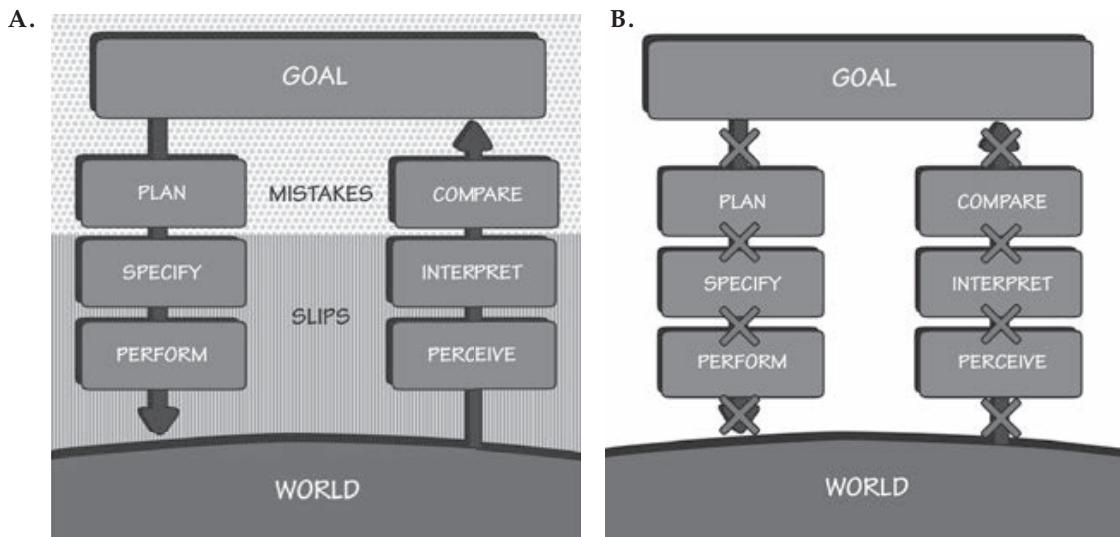


FIGURE 5.2. Where Slips and Mistakes Originate in the Action Cycle. Figure A shows that action slips come from the bottom four stages of the action cycle and mistakes from the top three stages. Memory lapses impact the transitions between stages (shown by the X's in Figure B). Memory lapses at the higher levels lead to mistakes, and lapses at the lower levels lead to slips.

Slips are the result of subconscious actions getting waylaid en route. Mistakes result from conscious deliberations. The same processes that make us creative and insightful by allowing us to see relationships between apparently unrelated things, that let us leap to correct conclusions on the basis of partial or even faulty evidence, also lead to mistakes. Our ability to generalize from small amounts of information helps tremendously in new situations; but sometimes we generalize too rapidly, classifying a new situation as similar to an old one when, in fact, there are significant discrepancies. This leads to mistakes that can be difficult to discover, let alone eliminate.

The Classification of Slips

A colleague reported that he went to his car to drive to work. As he drove away, he realized that he had forgotten his briefcase, so he turned around and went back. He stopped the car, turned off the engine, and unbuckled his wristwatch. Yes, his wristwatch, instead of his seatbelt.

The story illustrates both a memory-lapse slip and an action slip. The forgetting of the briefcase is a memory-lapse slip. The unbuckling of the wristwatch is an action slip, in this case a combination of description-similarity and capture error (described later in this chapter).

Most everyday errors are slips. Intending to do one action, you find yourself doing another. When a person says something clearly and distinctly to you, you “hear” something quite different. The study of slips is the study of the psychology of everyday errors—what Freud called “the psychopathology of everyday life.” Freud believed that slips have hidden, dark meanings, but most are accounted for by rather simple mental mechanisms.

An interesting property of slips is that, paradoxically, they tend to occur more frequently to skilled people than to novices. Why? Because slips often result from a lack of attention to the task. Skilled people—experts—tend to perform tasks automatically, under subconscious control. Novices have to pay considerable conscious attention, resulting in a relatively low occurrence of slips.

Some slips result from the similarities of actions. Or an event in the world may automatically trigger an action. Sometimes our thoughts and actions may remind us of unintended actions, which we then perform. There are numerous different kinds of action slips, categorized by the underlying mechanisms that give rise to them. The three most relevant to design are:

- capture slips
- description-similarity slips
- mode errors

CAPTURE SLIPS

I was using a copying machine, and I was counting the pages. I found myself counting, “1, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King.” I had been playing cards recently.

The capture slip is defined as the situation where, instead of the desired activity, a more frequently or recently performed one gets done instead: it captures the activity. Capture errors require that part of the action sequences involved in the two activities be identical, with one sequence being far more familiar than the other. After doing the identical part, the more frequent or more recent activity continues, and the intended one does not get done. Seldom, if ever, does the unfamiliar sequence capture the familiar one. All that is needed is a lapse of attention to the desired action at the critical junction when the identical portions of the sequences diverge into the two different activities. Capture errors are, therefore, partial memory-lapse errors. Interestingly, capture errors are more prevalent in experienced skilled people than in beginners, in part because the experienced person has automated the required actions and may not be paying conscious attention when the intended action deviates from the more frequent one.

Designers need to avoid procedures that have identical opening steps but then diverge. The more experienced the workers, the more likely they are to fall prey to capture. Whenever possible, sequences should be designed to differ from the very start.

DESCRIPTION-SIMILARITY SLIPS

A former student reported that one day he came home from jogging, took off his sweaty shirt, and rolled it up in a ball, intending to throw it in the laundry basket. Instead he threw it in the toilet. (It wasn't poor aim: the laundry basket and toilet were in different rooms.)

In the slip known as a description-similarity slip, the error is to act upon an item similar to the target. This happens when the description of the target is sufficiently vague. Much as we saw in Chapter 3, Figure 3.1, where people had difficulty distinguishing among different images of money because their internal descriptions did not have sufficient discriminating information, the same thing can happen to us, especially when we are tired, stressed, or overloaded. In the example that opened this section, both the laundry basket and the toilet bowl are containers, and if the description of the target was sufficiently ambiguous, such as "a large enough container," the slip could be triggered.

Remember the discussion in Chapter 3 that most objects don't need precise descriptions, simply enough precision to distinguish the desired target from alternatives. This means that a description that usually suffices may fail when the situation changes so that multiple similar items now match the description. Description-similarity errors result in performing the correct action on the wrong object. Obviously, the more the wrong and right objects have in common, the more likely the errors are to occur. Similarly, the more objects present at the same time, the more likely the error.

Designers need to ensure that controls and displays for different purposes are significantly different from one another. A lineup of identical-looking switches or displays is very apt to lead to description-similarity error. In the design of airplane cockpits, many controls are shape coded so that they both look and feel different from one another: the throttle levers are different from the flap levers (which might look and feel like a wing flap), which are different from the landing gear control (which might look and feel like a wheel).

MEMORY-LAPSE SLIPS

Errors caused by memory failures are common. Consider these examples:

- Making copies of a document, walking off with the copy, but leaving the original inside the machine.
- Forgetting a child. This error has numerous examples, such as leaving a child behind at a rest stop during a car trip, or in the dressing room of a department store, or a new mother forgetting her one-month-old and having to go to the police for help in finding the baby.
- Losing a pen because it was taken out to write something, then put down while doing some other task. The pen is forgotten in the activities of putting away a checkbook, picking up goods, talking to a salesperson or friends, and so on. Or the reverse: borrowing a pen, using it, and then putting it away in your pocket or purse, even though it is someone else's (this is also a capture error).
- Using a bank or credit card to withdraw money from an automatic teller machine, then walking off without the card, is such a frequent error that many machines now have a forcing function: the card must be removed before the money will be delivered. Of course, it is then possible to walk off without the money, but this is less likely than forgetting the card because money is the goal of using the machine.

Memory lapses are common causes of error. They can lead to several kinds of errors: failing to do all of the steps of a procedure; repeating steps; forgetting the outcome of an action; or forgetting the goal or plan, thereby causing the action to be stopped.

The immediate cause of most memory-lapse failures is interruptions, events that intervene between the time an action is decided upon and the time it is completed. Quite often the interference comes from the machines we are using: the many steps required between the start and finish of the operations can overload the capacity of short-term or working memory.

There are several ways to combat memory-lapse errors. One is to minimize the number of steps; another, to provide vivid reminders of steps that need to be completed. A superior method is to use the

forcing function of Chapter 4. For example, automated teller machines often require removal of the bank card before delivering the requested money: this prevents forgetting the bank card, capitalizing on the fact that people seldom forget the goal of the activity, in this case the money. With pens, the solution is simply to prevent their removal, perhaps by chaining public pens to the counter. Not all memory-lapse errors lend themselves to simple solutions. In many cases the interruptions come from outside the system, where the designer has no control.

MODE-ERROR SLIPS

A mode error occurs when a device has different states in which the same controls have different meanings: we call these states *modes*. Mode errors are inevitable in anything that has more possible actions than it has controls or displays; that is, the controls mean different things in the different modes. This is unavoidable as we add more and more functions to our devices.

Ever turn off the wrong device in your home entertainment system? This happens when one control is used for multiple purposes. In the home, this is simply frustrating. In industry, the confusion that results when operators believe the system to be in one mode, when in reality it is in another, has resulted in serious accidents and loss of life.

It is tempting to save money and space by having a single control serve multiple purposes. Suppose there are ten different functions on a device. Instead of using ten separate knobs or switches—which would take considerable space, add extra cost, and appear intimidatingly complex, why not use just two controls, one to select the function, the other to set the function to the desired condition? Although the resulting design appears quite simple and easy to use, this apparent simplicity masks the underlying complexity of use. The operator must always be completely aware of the mode, of what function is active. Alas, the prevalence of mode errors shows this assumption to be false. Yes, if I select a mode and then immediately adjust the parameters, I am not apt to be confused about the state. But what if I select the mode and then get interrupted

by other events? Or if the mode is maintained for considerable periods? Or, as in the case of the Airbus accident discussed below, the two modes being selected are very similar in control and function, but have different operating characteristics, which means that the resulting mode error is difficult to discover? Sometimes the use of modes is justifiable, such as the need to put many controls and displays in a small, restricted space, but whatever the reason, modes are a common cause of confusion and error.

Alarm clocks often use the same controls and display for setting the time of day and the time the alarm should go off, and many of us have thereby set one when we meant the other. Similarly, when time is displayed on a twelve-hour scale, it is easy to set the alarm to go off at seven A.M. only later to discover that the alarm had been set for seven P.M. The use of "A.M." and "P.M." to distinguish times before and after noon is a common source of confusion and error, hence the common use of 24-hour time specification throughout most of the world (the major exceptions being North America, Australia, India, and the Philippines). Watches with multiple functions have similar problems, in this case required because of the small amount of space available for controls and displays. Modes exist in most computer programs, in our cell phones, and in the automatic controls of commercial aircraft. A number of serious accidents in commercial aviation can be attributed to mode errors, especially in aircraft that use automatic systems (which have a large number of complex modes). As automobiles become more complex, with the dashboard controls for driving, heating and air-conditioning, entertainment, and navigation, modes are increasingly common.

An accident with an Airbus airplane illustrates the problem. The flight control equipment (often referred to as the automatic pilot) had two modes, one for controlling vertical speed, the other for controlling the flight path's angle of descent. In one case, when the pilots were attempting to land, the pilots thought that they were controlling the angle of descent, whereas they had accidentally

selected the mode that controlled speed of descent. The number (-3.3) that was entered into the system to represent an appropriate angle (-3.3°) was too steep a rate of descent when interpreted as vertical speed ($-3,300$ feet/minute: -3.3° would only be -800 feet/minute). This mode confusion contributed to the resulting fatal accident. After a detailed study of the accident, Airbus changed the display on the instrument so that vertical speed would always be displayed with a four-digit number and angle with two digits, thus reducing the chance of confusion.

Mode error is really design error. Mode errors are especially likely where the equipment does not make the mode visible, so the user is expected to remember what mode has been established, sometimes hours earlier, during which time many intervening events might have occurred. Designers must try to avoid modes, but if they are necessary, the equipment must make it obvious which mode is invoked. Once again, designers must always compensate for interfering activities.

The Classification of Mistakes

Mistakes result from the choice of inappropriate goals and plans or from faulty comparison of the outcome with the goals during evaluation. In mistakes, a person makes a poor decision, misclassifies a situation, or fails to take all the relevant factors into account. Many mistakes arise from the vagaries of human thought, often because people tend to rely upon remembered experiences rather than on more systematic analysis. We make decisions based upon what is in our memory. But as discussed in Chapter 3, retrieval from long-term memory is actually a reconstruction rather than an accurate record. As a result, it is subject to numerous biases. Among other things, our memories tend to be biased toward overgeneralization of the commonplace and overemphasis of the discrepant.

The Danish engineer Jens Rasmussen distinguished among three modes of behavior: skill-based, rule-based, and knowledge-based. This three-level classification scheme provides a practical tool that has found wide acceptance in applied areas, such as the design of

many industrial systems. Skill-based behavior occurs when workers are extremely expert at their jobs, so they can do the everyday, routine tasks with little or no thought or conscious attention. The most common form of errors in skill-based behavior is slips.

Rule-based behavior occurs when the normal routine is no longer applicable but the new situation is one that is known, so there is already a well-prescribed course of action: a rule. Rules simply might be learned behaviors from previous experiences, but includes formal procedures prescribed in courses and manuals, usually in the form of “if-then” statements, such as, “*If* the engine will not start, *then* do [the appropriate action].” Errors with rule-based behavior can be either a mistake or a slip. If the wrong rule is selected, this would be a mistake. If the error occurs during the execution of the rule, it is most likely a slip.

Knowledge-based procedures occur when unfamiliar events occur, where neither existing skills nor rules apply. In this case, there must be considerable reasoning and problem-solving. Plans might be developed, tested, and then used or modified. Here, conceptual models are essential in guiding development of the plan and interpretation of the situation.

In both rule-based and knowledge-based situations, the most serious mistakes occur when the situation is misdiagnosed. As a result, an inappropriate rule is executed, or in the case of knowledge-based problems, the effort is addressed to solving the wrong problem. In addition, with misdiagnosis of the problem comes misinterpretation of the environment, as well as faulty comparisons of the current state with expectations. These kinds of mistakes can be very difficult to detect and correct.

RULE-BASED MISTAKES

When new procedures have to be invoked or when simple problems arise, we can characterize the actions of skilled people as rule-based. Some rules come from experience; others are formal procedures in manuals or rulebooks, or even less formal guides, such as cookbooks for food preparation. In either case, all we must do is identify the situation, select the proper rule, and then follow it.

When driving, behavior follows well-learned rules. Is the light red? If so, stop the car. Wish to turn left? Signal the intention to turn and move as far left as legally permitted: slow the vehicle and wait for a safe break in traffic, all the while following the traffic rules and relevant signs and lights.

Rule-based mistakes occur in multiple ways:

- The situation is mistakenly interpreted, thereby invoking the wrong goal or plan, leading to following an inappropriate rule.
- The correct rule is invoked, but the rule itself is faulty, either because it was formulated improperly or because conditions are different than assumed by the rule or through incomplete knowledge used to determine the rule. All of these lead to knowledge-based mistakes.
- The correct rule is invoked, but the outcome is incorrectly evaluated. This error in evaluation, usually rule- or knowledge-based itself, can lead to further problems as the action cycle continues.

Example 1: In 2013, at the Kiss nightclub in Santa Maria, Brazil, pyrotechnics used by the band ignited a fire that killed over 230 people. The tragedy illustrates several mistakes. The band made a knowledge-based mistake when they used outdoor flares, which ignited the ceiling's acoustic tiles. The band thought the flares were safe. Many people rushed into the rest rooms, mistakenly thinking they were exits: they died. Early reports suggested that the guards, unaware of the fire, at first mistakenly blocked people from leaving the building. Why? Because nightclub attendees would sometimes leave without paying for their drinks.

The mistake was in devising a rule that did not take account of emergencies. A root cause analysis would reveal that the goal was to prevent inappropriate exit but still allow the doors to be used in an emergency. One solution is doors that trigger alarms when used, deterring people trying to sneak out, but allowing exit when needed.

Example 2: Turning the thermostat of an oven to its maximum temperature to get it to the proper cooking temperature faster is a mistake based upon a false conceptual model of the way the oven works. If the person wanders off and forgets to come back and check the oven

temperature after a reasonable period (a memory-lapse slip), the improper high setting of the oven temperature can lead to an accident, possibly a fire.

Example 3: A driver, unaccustomed to anti-lock brakes, encounters an unexpected object in the road on a wet, rainy day. The driver applies full force to the brakes but the car skids, triggering the anti-lock brakes to rapidly turn the brakes on and off, as they are designed to do. The driver, feeling the vibrations, believes that it indicates malfunction and therefore lifts his foot off the brake pedal. In fact, the vibration is a signal that anti-lock brakes are working properly. The driver's misevaluation leads to the wrong behavior.

Rule-based mistakes are difficult to avoid and then difficult to detect. Once the situation has been classified, the selection of the appropriate rule is often straightforward. But what if the classification of the situation is wrong? This is difficult to discover because there is usually considerable evidence to support the erroneous classification of the situation and the choice of rule. In complex situations, the problem is too much information: information that both supports the decision and also contradicts it. In the face of time pressures to make a decision, it is difficult to know which evidence to consider, which to reject. People usually decide by taking the current situation and matching it with something that happened earlier. Although human memory is quite good at matching examples from the past with the present situation, this doesn't mean that the matching is accurate or appropriate. The matching is biased by recency, regularity, and uniqueness. Recent events are remembered far better than less recent ones. Frequent events are remembered through their regularities, and unique events are remembered because of their uniqueness. But suppose the current event is different from all that has been experienced before: people are still apt to find some match in memory to use as a guide. The same powers that make us so good at dealing with the common and the unique lead to severe error with novel events.

What is a designer to do? Provide as much guidance as possible to ensure that the current state of things is displayed in a coherent

and easily interpreted format—ideally graphical. This is a difficult problem. All major decision makers worry about the complexity of real-world events, where the problem is often too much information, much of it contradictory. Often, decisions must be made quickly. Sometimes it isn't even clear that there is an incident or that a decision is actually being made.

Think of it like this. In your home, there are probably a number of broken or misbehaving items. There might be some burnt-out lights, or (in my home) a reading light that works fine for a little while, then goes out: we have to walk over and wiggle the fluorescent bulb. There might be a leaky faucet or other minor faults that you know about but are postponing action to remedy. Now consider a major process-control manufacturing plant (an oil refinery, a chemical plant, or a nuclear power plant). These have thousands, perhaps tens of thousands, of valves and gauges, displays and controls, and so on. Even the best of plants always has some faulty parts. The maintenance crews always have a list of items to take care of. With all the alarms that trigger when a problem arises, even though it might be minor, and all the everyday failures, how does one know which might be a significant indicator of a major problem? Every single one usually has a simple, rational explanation, so not making it an urgent item is a sensible decision. In fact, the maintenance crew simply adds it to a list. Most of the time, this is the correct decision. The one time in a thousand (or even, one time in a million) that the decision is wrong makes it the one they will be blamed for: how could they have missed such obvious signals?

Hindsight is always superior to foresight. When the accident investigation committee reviews the event that contributed to the problem, they know what actually happened, so it is easy for them to pick out which information was relevant, which was not. This is retrospective decision making. But when the incident was taking place, the people were probably overwhelmed with far too much irrelevant information and probably not a lot of relevant information. How were they to know which to attend to and which to ignore? Most of the time, experienced operators get things right. The one time they fail, the retrospective analysis is apt to condemn

them for missing the obvious. Well, during the event, nothing may be obvious. I return to this topic later in the chapter.

You will face this while driving, while handling your finances, and while just going through your daily life. Most of the unusual incidents you read about are not relevant to you, so you can safely ignore them. Which things should be paid attention to, which should be ignored? Industry faces this problem all the time, as do governments. The intelligence communities are swamped with data. How do they decide which cases are serious? The public hears about their mistakes, but not about the far more frequent cases that they got right or about the times they ignored data as not being meaningful—and were correct to do so.

If every decision had to be questioned, nothing would ever get done. But if decisions are not questioned, there will be major mistakes—rarely, but often of substantial penalty.

The design challenge is to present the information about the state of the system (a device, vehicle, plant, or activities being monitored) in a way that is easy to assimilate and interpret, as well as to provide alternative explanations and interpretations. It is useful to question decisions, but impossible to do so if every action—or failure to act—requires close attention.

This is a difficult problem with no obvious solution.

KNOWLEDGE-BASED MISTAKES

Knowledge-based behavior takes place when the situation is novel enough that there are no skills or rules to cover it. In this case, a new procedure must be devised. Whereas skills and rules are controlled at the behavioral level of human processing and are therefore subconscious and automatic, knowledge-based behavior is controlled at the reflective level and is slow and conscious.

With knowledge-based behavior, people are consciously problem solving. They are in an unknown situation and do not have any available skills or rules that apply directly. Knowledge-based behavior is required either when a person encounters an unknown situation, perhaps being asked to use some novel equipment, or

even when doing a familiar task and things go wrong, leading to a novel, uninterpretable state.

The best solution to knowledge-based situations is to be found in a good understanding of the situation, which in most cases also translates into an appropriate conceptual model. In complex cases, help is needed, and here is where good cooperative problem-solving skills and tools are required. Sometimes, good procedural manuals (paper or electronic) will do the job, especially if critical observations can be used to arrive at the relevant procedures to follow. A more powerful approach is to develop intelligent computer systems, using good search and appropriate reasoning techniques (artificial-intelligence decision-making and problem-solving). The difficulties here are in establishing the interaction of the people with the automation: human teams and automated systems have to be thought of as collaborative, cooperative systems. Instead, they are often built by assigning the tasks that machines can do to the machines and leaving the humans to do the rest. This usually means that machines do the parts that are easy for people, but when the problems become complex, which is precisely when people could use assistance, that is when the machines usually fail. (I discuss this problem extensively in *The Design of Future Things*.)

MEMORY-LAPSE MISTAKES

Memory lapses can lead to mistakes if the memory failure leads to forgetting the goal or plan of action. A common cause of the lapse is an interruption that leads to forgetting the evaluation of the current state of the environment. These lead to mistakes, not slips, because the goals and plans become wrong. Forgetting earlier evaluations often means remaking the decision, sometimes erroneously.

The design cures for memory-lapse mistakes are the same as for memory-lapse slips: ensure that all the relevant information is continuously available. The goals, plans, and current evaluation of the system are of particular importance and should be continually available. Far too many designs eliminate all signs of these items once they have been made or acted upon. Once again, the designer

should assume that people will be interrupted during their activities and that they may need assistance in resuming their operations.

Social and Institutional Pressures

A subtle issue that seems to figure in many accidents is social pressure. Although at first it may not seem relevant to design, it has strong influence on everyday behavior. In industrial settings, social pressures can lead to misinterpretation, mistakes, and accidents. To understand human error, it is essential to understand social pressure.

Complex problem-solving is required when one is faced with knowledge-based problems. In some cases, it can take teams of people days to understand what is wrong and the best ways to respond. This is especially true of situations where mistakes have been made in the diagnosis of the problem. Once the mistaken diagnosis is made, all information from then on is interpreted from the wrong point of view. Appropriate reconsiderations might only take place during team turnover, when new people come into the situation with a fresh viewpoint, allowing them to form different interpretations of the events. Sometimes just asking one or more of the team members to take a few hours' break can lead to the same fresh analysis (although it is understandably difficult to convince someone who is battling an emergency situation to stop for a few hours).

In commercial installations, the pressure to keep systems running is immense. Considerable money might be lost if an expensive system is shut down. Operators are often under pressure not to do this. The result has at times been tragic. Nuclear power plants are kept running longer than is safe. Airplanes have taken off before everything was ready and before the pilots had received permission. One such incident led to the largest accident in aviation history. Although the incident happened in 1977, a long time ago, the lessons learned are still very relevant today.

In Tenerife, in the Canary Islands, a KLM Boeing 747 crashed during takeoff into a Pan American 747 that was taxiing on the same runway, killing 583 people. The KLM plane had not received clearance to take off, but the weather was starting to get bad and the crew had already been delayed for too long (even being on the

Canary Islands was a diversion from the scheduled flight—bad weather had prevented their landing at their scheduled destination). And the Pan American flight should not have been on the runway, but there was considerable misunderstanding between the pilots and the air traffic controllers. Furthermore, the fog was coming in so thickly that neither plane's crew could see the other.

In the Tenerife disaster, time and economic pressures were acting together with cultural and weather conditions. The Pan American pilots questioned their orders to taxi on the runway, but they continued anyway. The first officer of the KLM flight voiced minor objections to the captain, trying to explain that they were not yet cleared for takeoff (but the first officer was very junior to the captain, who was one of KLM's most respected pilots). All in all, a major tragedy occurred due to a complex mixture of social pressures and logical explaining away of discrepant observations.

You may have experienced similar pressure, putting off refueling or recharging your car until it was too late and you ran out, sometimes in a truly inconvenient place (this has happened to me). What are the social pressures to cheat on school examinations, or to help others cheat? Or to not report cheating by others? Never underestimate the power of social pressures on behavior, causing otherwise sensible people to do things they know are wrong and possibly dangerous.

When I was in training to do underwater (scuba) diving, our instructor was so concerned about this that he said he would reward anyone who stopped a dive early in favor of safety. People are normally buoyant, so they need weights to get them beneath the surface. When the water is cold, the problem is intensified because divers must then wear either wet or dry suits to keep warm, and these suits add buoyancy. Adjusting buoyancy is an important part of the dive, so along with the weights, divers also wear air vests into which they continually add or remove air so that the body is close to neutral buoyancy. (As divers go deeper, increased water pressure compresses the air in their protective suits and lungs, so they become heavier: the divers need to add air to their vests to compensate.)

When divers have gotten into difficulties and needed to get to the surface quickly, or when they were at the surface close to shore but being tossed around by waves, some drowned because they were still being encumbered by their heavy weights. Because the weights are expensive, the divers didn't want to release them. In addition, if the divers released the weights and then made it back safely, they could never prove that the release of the weights was necessary, so they would feel embarrassed, creating self-induced social pressure. Our instructor was very aware of the resulting reluctance of people to take the critical step of releasing their weights when they weren't entirely positive it was necessary. To counteract this tendency, he announced that if anyone dropped the weights for safety reasons, he would publicly praise the diver and replace the weights at no cost to the person. This was a very persuasive attempt to overcome social pressures.

Social pressures show up continually. They are usually difficult to document because most people and organizations are reluctant to admit these factors, so even if they are discovered in the process of the accident investigation, the results are often kept hidden from public scrutiny. A major exception is in the study of transportation accidents, where the review boards across the world tend to hold open investigations. The US National Transportation Safety Board (NTSB) is an excellent example of this, and its reports are widely used by many accident investigators and researchers of human error (including me).

Another good example of social pressures comes from yet another airplane incident. In 1982 an Air Florida flight from National Airport, Washington, DC, crashed during takeoff into the Fourteenth Street Bridge over the Potomac River, killing seventy-eight people, including four who were on the bridge. The plane should not have taken off because there was ice on the wings, but it had already been delayed for over an hour and a half; this and other factors, the NTSB reported, "may have predisposed the crew to hurry." The accident occurred despite the first officer's attempt to warn the captain, who was flying the airplane (the captain and first officer—sometimes called the copilot—usually alternate flying

roles on different legs of a trip). The NTSB report quotes the flight deck recorder's documenting that "although the first officer expressed concern that something 'was not right' to the captain four times during the takeoff, the captain took no action to reject the takeoff." NTSB summarized the causes this way:

The National Transportation Safety Board determines that the probable cause of this accident was the flight crew's failure to use engine anti-ice during ground operation and takeoff, their decision to take off with snow/ice on the airfoil surfaces of the aircraft, and the captain's failure to reject the takeoff during the early stage when his attention was called to anomalous engine instrument readings. (NTSB, 1982.)

Again we see social pressures coupled with time and economic forces.

Social pressures can be overcome, but they are powerful and pervasive. We drive when drowsy or after drinking, knowing full well the dangers, but talking ourselves into believing that we are exempt. How can we overcome these kinds of social problems? Good design alone is not sufficient. We need different training; we need to reward safety and put it above economic pressures. It helps if the equipment can make the potential dangers visible and explicit, but this is not always possible. To adequately address social, economic, and cultural pressures and to improve upon company policies are the hardest parts of ensuring safe operation and behavior.

CHECKLISTS

Checklists are powerful tools, proven to increase the accuracy of behavior and to reduce error, particularly slips and memory lapses. They are especially important in situations with multiple, complex requirements, and even more so where there are interruptions. With multiple people involved in a task, it is essential that the lines of responsibility be clearly spelled out. It is always better to have two people do checklists together as a team: one to read the instruction, the other to execute it. If, instead, a single person executes the checklist and then, later, a second person checks the items, the

results are not as robust. The person following the checklist, feeling confident that any errors would be caught, might do the steps too quickly. But the same bias affects the checker. Confident in the ability of the first person, the checker often does a quick, less than thorough job.

One paradox of groups is that quite often, adding more people to check a task makes it less likely that it will be done right. Why? Well, if you were responsible for checking the correct readings on a row of fifty gauges and displays, but you know that two people before you had checked them and that one or two people who come after you will check your work, you might relax, thinking that you don't have to be extra careful. After all, with so many people looking, it would be impossible for a problem to exist without detection. But if everyone thinks the same way, adding more checks can actually increase the chance of error. A collaboratively followed checklist is an effective way to counteract these natural human tendencies.

In commercial aviation, collaboratively followed checklists are widely accepted as essential tools for safety. The checklist is done by two people, usually the two pilots of the airplane (the captain and first officer). In aviation, checklists have proven their worth and are now required in all US commercial flights. But despite the strong evidence confirming their usefulness, many industries still fiercely resist them. It makes people feel that their competence is being questioned. Moreover, when two people are involved, a junior person (in aviation, the first officer) is being asked to watch over the action of the senior person. This is a strong violation of the lines of authority in many cultures.

Physicians and other medical professionals have strongly resisted the use of checklists. It is seen as an insult to their professional competence. "Other people might need checklists," they complain, "but not me." Too bad. Too err is human: we all are subject to slips and mistakes when under stress, or under time or social pressure, or after being subjected to multiple interruptions, each essential in its own right. It is not a threat to professional competence to be

human. Legitimate criticisms of particular checklists are used as an indictment against the concept of checklists. Fortunately, checklists are slowly starting to gain acceptance in medical situations. When senior personnel insist on the use of checklists, it actually enhances their authority and professional status. It took decades for checklists to be accepted in commercial aviation: let us hope that medicine and other professions will change more rapidly.

Designing an effective checklist is difficult. The design needs to be iterative, always being refined, ideally using the human-centered design principles of Chapter 6, continually adjusting the list until it covers the essential items yet is not burdensome to perform. Many people who object to checklists are actually objecting to badly designed lists: designing a checklist for a complex task is best done by professional designers in conjunction with subject matter experts.

Printed checklists have one major flaw: they force the steps to follow a sequential ordering, even where this is not necessary or even possible. With complex tasks, the order in which many operations are performed may not matter, as long as they are all completed. Sometimes items early in the list cannot be done at the time they are encountered in the checklist. For example, in aviation one of the steps is to check the amount of fuel in the plane. But what if the fueling operation has not yet been completed when this checklist item is encountered? Pilots will skip over it, intending to come back to it after the plane has been refueled. This is a clear opportunity for a memory-lapse error.

In general, it is bad design to impose a sequential structure to task execution unless the task itself requires it. This is one of the major benefits of electronic checklists: they can keep track of skipped items and can ensure that the list will not be marked as complete until all items have been done.

Reporting Error

If errors can be caught, then many of the problems they might lead to can often be avoided. But not all errors are easy to detect. Moreover, social pressures often make it difficult for people to admit to

their own errors (or to report the errors of others). If people report their own errors, they might be fined or punished. Moreover, their friends may make fun of them. If a person reports that someone else made an error, this may lead to severe personal repercussions. Finally, most institutions do not wish to reveal errors made by their staff. Hospitals, courts, police systems, utility companies—all are reluctant to admit to the public that their workers are capable of error. These are all unfortunate attitudes.

The only way to reduce the incidence of errors is to admit their existence, to gather together information about them, and thereby to be able to make the appropriate changes to reduce their occurrence. In the absence of data, it is difficult or impossible to make improvements. Rather than stigmatize those who admit to error, we should thank those who do so and encourage the reporting. We need to make it easier to report errors, for the goal is not to punish, but to determine how it occurred and change things so that it will not happen again.

CASE STUDY: JIDOKA—HOW TOYOTA HANDLES ERROR

The Toyota automobile company has developed an extremely efficient error-reduction process for manufacturing, widely known as the Toyota Production System. Among its many key principles is a philosophy called *Jidoka*, which Toyota says is “roughly translated as ‘automation with a human touch.’” If a worker notices something wrong, the worker is supposed to report it, sometimes even stopping the entire assembly line if a faulty part is about to proceed to the next station. (A special cord, called an *andon*, stops the assembly line and alerts the expert crew.) Experts converge upon the problem area to determine the cause. “Why did it happen?” “Why was that?” “Why is that the reason?” The philosophy is to ask “Why?” as many times as may be necessary to get to the root cause of the problem and then fix it so it can never occur again.

As you might imagine, this can be rather discomforting for the person who found the error. But the report is expected, and when it is discovered that people have failed to report errors, they are punished, all in an attempt to get the workers to be honest.

POKA-YOKE: ERROR PROOFING

Poka-yoke is another Japanese method, this one invented by Shigeo Shingo, one of the Japanese engineers who played a major role in the development of the Toyota Production System. *Poka-yoke* translates as “error proofing” or “avoiding error.” One of the techniques of poka-yoke is to add simple fixtures, jigs, or devices to constrain the operations so that they are correct. I practice this myself in my home. One trivial example is a device to help me remember which way to turn the key on the many doors in the apartment complex where I live. I went around with a pile of small, circular, green stick-on dots and put them on each door beside its keyhole, with the green dot indicating the direction in which the key needed to be turned: I added signifiers to the doors. Is this a major error? No. But eliminating it has proven to be convenient. (Neighbors have commented on their utility, wondering who put them there.)

In manufacturing facilities, poka-yoke might be a piece of wood to help align a part properly, or perhaps plates designed with asymmetrical screw holes so that the plate could fit in only one position. Covering emergency or critical switches with a cover to prevent accidental triggering is another poka-yoke technique: this is obviously a forcing function. All the poka-yoke techniques involve a combination of the principles discussed in this book: affordances, signifiers, mapping, and constraints, and perhaps most important of all, forcing functions.

NASA'S AVIATION SAFETY REPORTING SYSTEM

US commercial aviation has long had an extremely effective system for encouraging pilots to submit reports of errors. The program has resulted in numerous improvements to aviation safety. It wasn’t easy to establish: pilots had severe self-induced social pressures against admitting to errors. Moreover, to whom would they report them? Certainly not to their employers. Not even to the Federal Aviation Authority (FAA), for then they would probably be punished. The solution was to let the National Aeronautics and Space Administration (NASA) set up a voluntary accident reporting system whereby pilots could submit semi-anonymous reports

of errors they had made or observed in others (semi-anonymous because pilots put their name and contact information on the reports so that NASA could call to request more information). Once NASA personnel had acquired the necessary information, they would detach the contact information from the report and mail it back to the pilot. This meant that NASA no longer knew who had reported the error, which made it impossible for the airline companies or the FAA (which enforced penalties against errors) to find out who had submitted the report. If the FAA had independently noticed the error and tried to invoke a civil penalty or certificate suspension, the receipt of self-report automatically exempted the pilot from punishment (for minor infractions).

When a sufficient number of similar errors had been collected, NASA would analyze them and issue reports and recommendations to the airlines and to the FAA. These reports also helped the pilots realize that their error reports were valuable tools for increasing safety. As with checklists, we need similar systems in the field of medicine, but it has not been easy to set up. NASA is a neutral body, charged with enhancing aviation safety, but has no oversight authority, which helped gain the trust of pilots. There is no comparable institution in medicine: physicians are afraid that self-reported errors might lead them to lose their license or be subjected to lawsuits. But we can't eliminate errors unless we know what they are. The medical field is starting to make progress, but it is a difficult technical, political, legal, and social problem.

Detecting Error

Errors do not necessarily lead to harm if they are discovered quickly. The different categories of errors have differing ease of discovery. In general, action slips are relatively easy to discover; mistakes, much more difficult. Action slips are relatively easy to detect because it is usually easy to notice a discrepancy between the intended act and the one that got performed. But this detection can only take place if there is feedback. If the result of the action is not visible, how can the error be detected?

Memory-lapse slips are difficult to detect precisely because there is nothing to see. With a memory slip, the required action is not performed. When no action is done, there is nothing to detect. It is only when the lack of action allows some unwanted event to occur that there is hope of detecting a memory-lapse slip.

Mistakes are difficult to detect because there is seldom anything that can signal an inappropriate goal. And once the wrong goal or plan is decided upon, the resulting actions are consistent with that wrong goal, so careful monitoring of the actions not only fails to detect the erroneous goal, but, because the actions are done correctly, can inappropriately provide added confidence to the decision.

Faulty diagnoses of a situation can be surprisingly difficult to detect. You might expect that if the diagnosis was wrong, the actions would turn out to be ineffective, so the fault would be discovered quickly. But misdiagnoses are not random. Usually they are based on considerable knowledge and logic. The misdiagnosis is usually both reasonable and relevant to eliminating the symptoms being observed. As a result, the initial actions are apt to appear appropriate and helpful. This makes the problem of discovery even more difficult. The actual error might not be discovered for hours or days.

Memory-lapse mistakes are especially difficult to detect. Just as with a memory-lapse slip the absence of something that should have been done is always more difficult to detect than the presence of something that should not have been done. The difference between memory-lapse slips and mistakes is that, in the first case, a single component of a plan is skipped, whereas in the second, the entire plan is forgotten. Which is easier to discover? At this point I must retreat to the standard answer science likes to give to questions of this sort: “It all depends.”

EXPLAINING AWAY MISTAKES

Mistakes can take a long time to be discovered. Hear a noise that sounds like a pistol shot and think: “Must be a car’s exhaust backfiring.” Hear someone yell outside and think: “Why can’t my

neighbors be quiet?" Are we correct in dismissing these incidents? Most of the time we are, but when we're not, our explanations can be difficult to justify.

Explaining away errors is a common problem in commercial accidents. Most major accidents are preceded by warning signs: equipment malfunctions or unusual events. Often, there is a series of apparently unrelated breakdowns and errors that culminate in major disaster. Why didn't anyone notice? Because no single incident appeared to be serious. Often, the people involved noted each problem but discounted it, finding a logical explanation for the otherwise deviant observation.

THE CASE OF THE WRONG TURN ON A HIGHWAY

I've misinterpreted highway signs, as I'm sure most drivers have. My family was traveling from San Diego to Mammoth Lakes, California, a ski area about 400 miles north. As we drove, we noticed more and more signs advertising the hotels and gambling casinos of Las Vegas, Nevada. "Strange," we said, "Las Vegas always did advertise a long way off—there is even a billboard in San Diego—but this seems excessive, advertising on the road to Mammoth." We stopped for gasoline and continued on our journey. Only later, when we tried to find a place to eat supper, did we discover that we had missed a turn nearly two hours earlier, before we had stopped for gasoline, and that we were actually on the road to Las Vegas, not the road to Mammoth. We had to backtrack the entire two-hour segment, wasting four hours of driving. It's humorous now; it wasn't then.

Once people find an explanation for an apparent anomaly, they tend to believe they can now discount it. But explanations are based on analogy with past experiences, experiences that may not apply to the current situation. In the driving story, the prevalence of billboards for Las Vegas was a signal we should have heeded, but it seemed easily explained. Our experience is typical: some major industrial incidents have resulted from false explanations of anomalous events. But do note: usually these apparent anomalies should be ignored. Most of the time, the explanation for their pres-

ence is correct. Distinguishing a true anomaly from an apparent one is difficult.

IN HINDSIGHT, EVENTS SEEM LOGICAL

The contrast in our understanding before and after an event can be dramatic. The psychologist Baruch Fischhoff has studied explanations given in hindsight, where events seem completely obvious and predictable after the fact but completely unpredictable beforehand.

Fischhoff presented people with a number of situations and asked them to predict what would happen: they were correct only at the chance level. When the actual outcome was not known by the people being studied, few predicted the actual outcome. He then presented the same situations along with the actual outcomes to another group of people, asking them to state how likely each outcome was: when the actual outcome was known, it appeared to be plausible and likely and other outcomes appeared unlikely.

Hindsight makes events seem obvious and predictable. Foresight is difficult. During an incident, there are never clear clues. Many things are happening at once: workload is high, emotions and stress levels are high. Many things that are happening will turn out to be irrelevant. Things that appear irrelevant will turn out to be critical. The accident investigators, working with hindsight, knowing what really happened, will focus on the relevant information and ignore the irrelevant. But at the time the events were happening, the operators did not have information that allowed them to distinguish one from the other.

This is why the best accident analyses can take a long time to do. The investigators have to imagine themselves in the shoes of the people who were involved and consider all the information, all the training, and what the history of similar past events would have taught the operators. So, the next time a major accident occurs, ignore the initial reports from journalists, politicians, and executives who don't have any substantive information but feel compelled to provide statements anyway. Wait until the official reports come from trusted sources. Unfortunately, this could be months or years after the accident, and the public usually wants

answers immediately, even if those answers are wrong. Moreover, when the full story finally appears, newspapers will no longer consider it news, so they won't report it. You will have to search for the official report. In the United States, the National Transportation Safety Board (NTSB) can be trusted. NTSB conducts careful investigations of all major aviation, automobile and truck, train, ship, and pipeline incidents. (Pipelines? Sure: pipelines transport coal, gas, and oil.)

Designing for Error

It is relatively easy to design for the situation where everything goes well, where people use the device in the way that was intended, and no unforeseen events occur. The tricky part is to design for when things go wrong.

Consider a conversation between two people. Are errors made? Yes, but they are not treated as such. If a person says something that is not understandable, we ask for clarification. If a person says something that we believe to be false, we question and debate. We don't issue a warning signal. We don't beep. We don't give error messages. We ask for more information and engage in mutual dialogue to reach an understanding. In normal conversations between two friends, misstatements are taken as normal, as approximations to what was really meant. Grammatical errors, self-corrections, and restarted phrases are ignored. In fact, they are usually not even detected because we concentrate upon the intended meaning, not the surface features.

Machines are not intelligent enough to determine the meaning of our actions, but even so, they are far less intelligent than they could be. With our products, if we do something inappropriate, if the action fits the proper format for a command, the product does it, even if it is outrageously dangerous. This has led to tragic accidents, especially in health care, where inappropriate design of infusion pumps and X-ray machines allowed extreme overdoses of medication or radiation to be administered to patients, leading to their deaths. In financial institutions, simple keyboard errors have led to huge financial transactions, far beyond normal limits.

Even simple checks for reasonableness would have stopped all of these errors. (This is discussed at the end of the chapter under the heading “Sensibility Checks.”)

Many systems compound the problem by making it easy to err but difficult or impossible to discover error or to recover from it. It should not be possible for one simple error to cause widespread damage. Here is what should be done:

- Understand the causes of error and design to minimize those causes.
- Do sensibility checks. Does the action pass the “common sense” test?
- Make it possible to reverse actions—to “undo” them—or make it harder to do what cannot be reversed.
- Make it easier for people to discover the errors that do occur, and make them easier to correct.
- Don’t treat the action as an error; rather, try to help the person complete the action properly. Think of the action as an approximation to what is desired.

As this chapter demonstrates, we know a lot about errors. Thus, novices are more likely to make mistakes than slips, whereas experts are more likely to make slips. Mistakes often arise from ambiguous or unclear information about the current state of a system, the lack of a good conceptual model, and inappropriate procedures. Recall that most mistakes result from erroneous choice of goal or plan or erroneous evaluation and interpretation. All of these come about through poor information provided by the system about the choice of goals and the means to accomplish them (plans), and poor-quality feedback about what has actually happened.

A major source of error, especially memory-lapse errors, is interruption. When an activity is interrupted by some other event, the cost of the interruption is far greater than the loss of the time required to deal with the interruption: it is also the cost of resuming the interrupted activity. To resume, it is necessary to remember precisely the previous state of the activity: what the goal was, where one was in the action cycle, and the relevant state of the system. Most systems make it difficult to resume after an interruption.

Most discard critical information that is needed by the user to remember the numerous small decisions that had been made, the things that were in the person's short-term memory, to say nothing of the current state of the system. What still needs to be done? Maybe I was finished? It is no wonder that many slips and mistakes are the result of interruptions.

Multitasking, whereby we deliberately do several tasks simultaneously, erroneously appears to be an efficient way of getting a lot done. It is much beloved by teenagers and busy workers, but in fact, all the evidence points to severe degradation of performance, increased errors, and a general lack of both quality and efficiency. Doing two tasks at once takes longer than the sum of the times it would take to do each alone. Even as simple and common a task as talking on a hands-free cell phone while driving leads to serious degradation of driving skills. One study even showed that cell phone usage during walking led to serious deficits: "Cell phone users walked more slowly, changed directions more frequently, and were less likely to acknowledge other people than individuals in the other conditions. In the second study, we found that cell phone users were less likely to notice an unusual activity along their walking route (a unicycling clown)" (Hyman, Boss, Wise, McKenzie, & Caggiano, 2010).

A large percentage of medical errors are due to interruptions. In aviation, where interruptions were also determined to be a major problem during the critical phases of flying—landing and takeoff—the US Federal Aviation Authority (FAA) requires what it calls a "Sterile Cockpit Configuration," whereby pilots are not allowed to discuss any topic not directly related to the control of the airplane during these critical periods. In addition, the flight attendants are not permitted to talk to the pilots during these phases (which has at times led to the opposite error—failure to inform the pilots of emergency situations).

Establishing similar sterile periods would be of great benefit to many professions, including medicine and other safety-critical operations. My wife and I follow this convention in driving: when the driver is entering or leaving a high-speed highway, conversa-

tion ceases until the transition has been completed. Interruptions and distractions lead to errors, both mistakes and slips.

Warning signals are usually not the answer. Consider the control room of a nuclear power plant, the cockpit of a commercial aircraft, or the operating room of a hospital. Each has a large number of different instruments, gauges, and controls, all with signals that tend to sound similar because they all use simple tone generators to beep their warnings. There is no coordination among the instruments, which means that in major emergencies, they all sound at once. Most can be ignored anyway because they tell the operator about something that is already known. Each competes with the others to be heard, interfering with efforts to address the problem.

Unnecessary, annoying alarms occur in numerous situations. How do people cope? By disconnecting warning signals, taping over warning lights (or removing the bulbs), silencing bells, and basically getting rid of all the safety warnings. The problem comes after such alarms are disabled, either when people forget to restore the warning systems (there are those memory-lapse slips again), or if a different incident happens while the alarms are disconnected. At that point, nobody notices. Warnings and safety methods must be used with care and intelligence, taking into account the tradeoffs for the people who are affected.

The design of warning signals is surprisingly complex. They have to be loud or bright enough to be noticed, but not so loud or bright that they become annoying distractions. The signal has to both attract attention (act as a signifier of critical information) and also deliver information about the nature of the event that is being signified. The various instruments need to have a coordinated response, which means that there must be international standards and collaboration among the many design teams from different, often competing, companies. Although considerable research has been directed toward this problem, including the development of national standards for alarm management systems, the problem still remains in many situations.

More and more of our machines present information through speech. But like all approaches, this has both strengths and

weaknesses. It allows for precise information to be conveyed, especially when the person's visual attention is directed elsewhere. But if several speech warnings operate at the same time, or if the environment is noisy, speech warnings may not be understood. Or if conversations among the users or operators are necessary, speech warnings will interfere. Speech warning signals can be effective, but only if used intelligently.

DESIGN LESSONS FROM THE STUDY OF ERRORS

Several design lessons can be drawn from the study of errors, one for preventing errors before they occur and one for detecting and correcting them when they do occur. In general, the solutions follow directly from the preceding analyses.

ADDING CONSTRAINTS TO BLOCK ERRORS

Prevention often involves adding specific constraints to actions. In the physical world, this can be done through clever use of shape and size. For example, in automobiles, a variety of fluids are required for safe operation and maintenance: engine oil, transmission oil, brake fluid, windshield washer solution, radiator coolant, battery water, and gasoline. Putting the wrong fluid into a reservoir could lead to serious damage or even an accident. Automobile manufacturers try to minimize these errors by segregating the filling points, thereby reducing description-similarity errors. When the filling points for fluids that should be added only occasionally or by qualified mechanics are located separately from those for fluids used more frequently, the average motorist is unlikely to use the incorrect filling points. Errors in adding fluids to the wrong container can be minimized by making the openings have different sizes and shapes, providing physical constraints against inappropriate filling. Different fluids often have different colors so that they can be distinguished. All these are excellent ways to minimize errors. Similar techniques are in widespread use in hospitals and industry. All of these are intelligent applications of constraints, forcing functions, and poka-yoke.

Electronic systems have a wide range of methods that could be used to reduce error. One is to segregate controls, so that easily confused controls are located far from one another. Another is to use separate modules, so that any control not directly relevant to the current operation is not visible on the screen, but requires extra effort to get to.

UNDO

Perhaps the most powerful tool to minimize the impact of errors is the Undo command in modern electronic systems, reversing the operations performed by the previous command, wherever possible. The best systems have multiple levels of undoing, so it is possible to undo an entire sequence of actions.

Obviously, undoing is not always possible. Sometimes, it is only effective if done immediately after the action. Still, it is a powerful tool to minimize the impact of error. It is still amazing to me that many electronic and computer-based systems fail to provide a means to undo even where it is clearly possible and desirable.



CONFIRMATION AND ERROR MESSAGES



Many systems try to prevent errors by requiring confirmation before a command will be executed, especially when the action will destroy something of importance. But these requests are usually ill-timed because after requesting an operation, people are usually certain they want it done. Hence the standard joke about such warnings:

Person: Delete "my most important file."

System: Do you want to delete "my most important file"?

Person: Yes.

System: Are you certain?

Person: Yes!

System "My most favorite file" has been deleted.

Person: Oh. Damn.

The request for confirmation seems like an irritant rather than an essential safety check because the person tends to focus upon the action rather than the object that is being acted upon. A better check would be a prominent display of both the action to be taken and the object, perhaps with the choice of “cancel” or “do it.” The important point is making salient what the implications of the action are. Of course, it is because of errors of this sort that the Undo command is so important. With traditional graphical user interfaces on computers, not only is Undo a standard command, but when files are “deleted,” they are actually simply moved from sight and stored in the file folder named “Trash,” so that in the above example, the person could open the Trash and retrieve the erroneously deleted file.

Confirmations have different implications for slips and mistakes. When I am writing, I use two very large displays and a powerful computer. I might have seven to ten applications running simultaneously. I have sometimes had as many as forty open windows. Suppose I activate the command that closes one of the windows, which triggers a confirmatory message: did I wish to close the window? How I deal with this depends upon why I requested that the window be closed. If it was a slip, the confirmation required will be useful. If it was by mistake, I am apt to ignore it. Consider these two examples:

A slip leads me to close the wrong window.

Suppose I intended to type the word *We*, but instead of typing Shift + W for the first character, I typed Command + W (or Control + W), the keyboard command for closing a window. Because I expected the screen to display an uppercase W, when a dialog box appeared, asking whether I really wanted to delete the file, I would be surprised, which would immediately alert me to the slip. I would cancel the action (an alternative thoughtfully provided by the dialog box) and retype the Shift + W, carefully this time.

A mistake leads me to close the wrong window.

Now suppose I really intended to close a window. I often use a temporary file in a window to keep notes about the chapter I am working on. When I am finished with it, I close it without saving its contents—after all, I am finished. But because I usually have multiple windows open, it is very easy to close the wrong one. The computer assumes that all commands apply to the active window—the one where the last actions had been performed (and which contains the text cursor). But if I reviewed the temporary window prior to closing it, my visual attention is focused upon that window, and when I decide to close it, I forget that it is not the active window from the computer’s point of view. So I issue the command to shut the window, the computer presents me with a dialog box, asking for confirmation, and I accept it, choosing the option not to save my work. Because the dialog box was expected, I didn’t bother to read it. As a result, I closed the wrong window and worse, did not save any of the typing, possibly losing considerable work. Warning messages are surprisingly ineffective against mistakes (even nice requests, such as the one shown in Chapter 4, Figure 4.6, page 143).

Was this a mistake or a slip? Both. Issuing the “close” command while the wrong window was active is a memory-lapse slip. But deciding not to read the dialog box and accepting it without saving the contents is a mistake (two mistakes, actually).

What can a designer do? Several things:

- **Make the item being acted upon more prominent.** That is, change the appearance of the actual object being acted upon to be more visible: enlarge it, or perhaps change its color.
- **Make the operation reversible.** If the person saves the content, no harm is done except the annoyance of having to reopen the file. If the person elects Don’t Save, the system could secretly save the contents, and the next time the person opened the file, it could ask whether it should restore it to the latest condition.

SENSIBILITY CHECKS

Electronic systems have another advantage over mechanical ones: they can check to make sure that the requested operation is sensible.

It is amazing that in today's world, medical personnel can accidentally request a radiation dose a thousand times larger than normal and have the equipment meekly comply. In some cases, it isn't even possible for the operator to notice the error.

Similarly, errors in stating monetary sums can lead to disastrous results, even though a quick glance at the amount would indicate that something was badly off. For example, there are roughly 1,000 Korean won to the US dollar. Suppose I wanted to transfer \$1,000 into a Korean bank account in *won* (\$1,000 is roughly ₩1,000,000). But suppose I enter the Korean number into the dollar field. Oops—I'm trying to transfer a million dollars. Intelligent systems would take note of the normal size of my transactions, querying if the amount was considerably larger than normal. For me, it would query the million-dollar request. Less intelligent systems would blindly follow instructions, even though I did not have a million dollars in my account (in fact, I would probably be charged a fee for overdrawing my account).

Sensibility checks, of course, are also the answer to the serious errors caused when inappropriate values are entered into hospital medication and X-ray systems or in financial transactions, as discussed earlier in this chapter.

MINIMIZING SLIPS

Slips most frequently occur when the conscious mind is distracted, either by some other event or simply because the action being performed is so well learned that it can be done automatically, without conscious attention. As a result, the person does not pay sufficient attention to the action or its consequences. It might therefore seem that one way to minimize slips is to ensure that people always pay close, conscious attention to the acts being done.

Bad idea. Skilled behavior is subconscious, which means it is fast, effortless, and usually accurate. Because it is so automatic, we can type at high speeds even while the conscious mind is occupied composing the words. This is why we can walk and talk while navigating traffic and obstacles. If we had to pay conscious attention to every little thing we did, we would accomplish far less in our

lives. The information processing structures of the brain automatically regulate how much conscious attention is being paid to a task: conversations automatically pause when crossing the street amid busy traffic. Don't count on it, though: if too much attention is focused on something else, the fact that the traffic is getting dangerous might not be noted.

Many slips can be minimized by ensuring that the actions and their controls are as dissimilar as possible, or at least, as physically far apart as possible. Mode errors can be eliminated by the simple expedient of eliminating most modes and, if this is not possible, by making the modes very visible and distinct from one another.

The best way of mitigating slips is to provide perceptible feedback about the nature of the action being performed, then very perceptible feedback describing the new resulting state, coupled with a mechanism that allows the error to be undone. For example, the use of machine-readable codes has led to a dramatic reduction in the delivery of wrong medications to patients. Prescriptions sent to the pharmacy are given electronic codes, so the pharmacist can scan both the prescription and the resulting medication to ensure they are the same. Then, the nursing staff at the hospital scans both the label of the medication and the tag worn around the patient's wrist to ensure that the medication is being given to the correct individual. Moreover, the computer system can flag repeated administration of the same medication. These scans do increase the workload, but only slightly. Other kinds of errors are still possible, but these simple steps have already been proven worthwhile.

Common engineering and design practices seem as if they are deliberately intended to cause slips. Rows of identical controls or meters is a sure recipe for description-similarity errors. Internal modes that are not very conspicuously marked are a clear driver of mode errors. Situations with numerous interruptions, yet where the design assumes undivided attention, are a clear enabler of memory lapses—and almost no equipment today is designed to support the numerous interruptions that so many situations entail. And failure to provide assistance and visible reminders for performing infrequent procedures that are similar to much more

frequent ones leads to capture errors, where the more frequent actions are performed rather than the correct ones for the situation. Procedures should be designed so that the initial steps are as dissimilar as possible.

The important message is that good design can prevent slips and mistakes. Design can save lives.

THE SWISS CHEESE MODEL OF HOW ERRORS LEAD TO ACCIDENTS

Fortunately, most errors do not lead to accidents. Accidents often have numerous contributing causes, no single one of which is the root cause of the incident.

James Reason likes to explain this by invoking the metaphor of multiple slices of Swiss cheese, the cheese famous for being riddled with holes (Figure 5.3). If each slice of cheese represents a condition in the task being done, an accident can happen only if holes in all four slices of cheese are lined up just right. In well-designed systems, there can be many equipment failures, many errors, but they will not lead to an accident unless they all line up precisely. Any leakage—passageway through a hole—is most likely blocked at the next level. Well-designed systems are resilient against failure.

This is why the attempt to find “the” cause of an accident is usually doomed to fail. Accident investigators, the press, government officials, and the everyday citizen like to find simple explanations for the cause of an accident. “See, if the hole in slice A

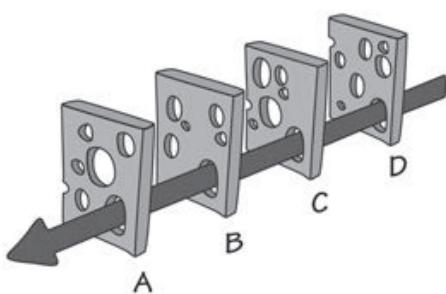


FIGURE 5.3. Reason’s Swiss Cheese Model of Accidents. Accidents usually have multiple causes, whereby had any single one of those causes not happened, the accident would not have occurred. The British accident researcher James Reason describes this through the metaphor of slices of Swiss cheese: unless the holes all line up perfectly, there will be no accident. This metaphor provides two lessons: First, do not try to find “the” cause of an accident; Second, we can decrease accidents and make systems more resilient by designing them to have extra precautions against error (more slices of cheese), less opportunities for slips, mistakes, or equipment failure (less holes), and very different mechanisms in the different subparts of the system (trying to ensure that the holes do not line up). (Drawing based upon one by Reason, 1990.)

had been slightly higher, we would not have had the accident. So throw away slice A and replace it." Of course, the same can be said for slices B, C, and D (and in real accidents, the number of cheese slices would sometimes measure in the tens or hundreds). It is relatively easy to find some action or decision that, had it been different, would have prevented the accident. But that does not mean that this was the cause of the accident. It is only one of the many causes: all the items have to line up.

You can see this in most accidents by the "if only" statements. "If only I hadn't decided to take a shortcut, I wouldn't have had the accident." "If only it hadn't been raining, my brakes would have worked." "If only I had looked to the left, I would have seen the car sooner." Yes, all those statements are true, but none of them is "the" cause of the accident. Usually, there is no single cause. Yes, journalists and lawyers, as well as the public, like to know the cause so someone can be blamed and punished. But reputable investigating agencies know that there is not a single cause, which is why their investigations take so long. Their responsibility is to understand the system and make changes that would reduce the chance of the same sequence of events leading to a future accident.

The Swiss cheese metaphor suggests several ways to reduce accidents:

- Add more slices of cheese.
- Reduce the number of holes (or make the existing holes smaller).
- Alert the human operators when several holes have lined up.

Each of these has operational implications. More slices of cheese means more lines of defense, such as the requirement in aviation and other industries for checklists, where one person reads the items, another does the operation, and the first person checks the operation to confirm it was done appropriately.

Reducing the number of critical safety points where error can occur is like reducing the number or size of the holes in the Swiss cheese. Properly designed equipment will reduce the opportunity for slips and mistakes, which is like reducing the number of holes

and making the ones that remain smaller. This is precisely how the safety level of commercial aviation has been dramatically improved. Deborah Hersman, chair of the National Transportation Safety Board, described the design philosophy as:

U.S. airlines carry about two million people through the skies safely every day, which has been achieved in large part through design redundancy and layers of defense.

Design redundancy and layers of defense: that's Swiss cheese. The metaphor illustrates the futility of trying to find the one underlying cause of an accident (usually some person) and punishing the culprit. Instead, we need to think about systems, about all the interacting factors that lead to human error and then to accidents, and devise ways to make the systems, as a whole, more reliable.

When Good Design Isn't Enough

WHEN PEOPLE REALLY ARE AT FAULT

I am sometimes asked whether it is really right to say that people are never at fault, that it is always bad design. That's a sensible question. And yes, of course, sometimes it is the person who is at fault.

Even competent people can lose competency if sleep deprived, fatigued, or under the influence of drugs. This is why we have laws banning pilots from flying if they have been drinking within some specified period and why we limit the number of hours they can fly without rest. Most professions that involve the risk of death or injury have similar regulations about drinking, sleep, and drugs. But everyday jobs do not have these restrictions. Hospitals often require their staff to go without sleep for durations that far exceed the safety requirements of airlines. Why? Would you be happy having a sleep-deprived physician operating on you? Why is sleep deprivation considered dangerous in one situation and ignored in another?

Some activities have height, age, or strength requirements. Others require considerable skills or technical knowledge: people

not trained or not competent should not be doing them. That is why many activities require government-approved training and licensing. Some examples are automobile driving, airplane piloting, and medical practice. All require instructional courses and tests. In aviation, it isn't sufficient to be trained: pilots must also keep in practice by flying some minimum number of hours per month.

Drunk driving is still a major cause of automobile accidents: this is clearly the fault of the drinker. Lack of sleep is another major culprit in vehicle accidents. But because people occasionally are at fault does not justify the attitude that assumes they are always at fault. The far greater percentage of accidents is the result of poor design, either of equipment or, as is often the case in industrial accidents, of the procedures to be followed.

As noted in the discussion of deliberate violations earlier in this chapter (page 169), people will sometimes deliberately violate procedures and rules, perhaps because they cannot get their jobs done otherwise, perhaps because they believe there are extenuating circumstances, and sometimes because they are taking the gamble that the relatively low probability of failure does not apply to them. Unfortunately, if someone does a dangerous activity that only results in injury or death one time in a million, that can lead to hundreds of deaths annually across the world, with its 7 billion people. One of my favorite examples in aviation is of a pilot who, after experiencing low oil-pressure readings in all three of his engines, stated that it must be an instrument failure because it was a one-in-a-million chance that the readings were true. He was right in his assessment, but unfortunately, he was the one. In the United States alone there were roughly 9 million flights in 2012. So, a one-in-a-million chance could translate into nine incidents.

Sometimes, people really are at fault.

Resilience Engineering

In industrial applications, accidents in large, complex systems such as oil wells, oil refineries, chemical processing plants, electrical power systems, transportation, and medical services can have major impacts on the company and the surrounding community.

Sometimes the problems do not arise in the organization but outside it, such as when fierce storms, earthquakes, or tidal waves demolish large parts of the existing infrastructure. In either case, the question is how to design and manage these systems so that they can restore services with a minimum of disruption and damage. An important approach is *resilience engineering*, with the goal of designing systems, procedures, management, and the training of people so they are able to respond to problems as they arise. It strives to ensure that the design of all these things—the equipment, procedures, and communication both among workers and also externally to management and the public—are continually being assessed, tested, and improved.

Thus, major computer providers can deliberately cause errors in their systems to test how well the company can respond. This is done by deliberately shutting down critical facilities to ensure that the backup systems and redundancies actually work. Although it might seem dangerous to do this while the systems are online, serving real customers, the only way to test these large, complex systems is by doing so. Small tests and simulations do not carry the complexity, stress levels, and unexpected events that characterize real system failures.

As Erik Hollnagel, David Woods, and Nancy Leveson, the authors of an early influential series of books on the topic, have skillfully summarized:

Resilience engineering is a paradigm for safety management that focuses on how to help people cope with complexity under pressure to achieve success. It strongly contrasts with what is typical today—a paradigm of tabulating error as if it were a thing, followed by interventions to reduce this count. A resilient organisation treats safety as a core value, not a commodity that can be counted. Indeed, safety shows itself only by the events that do not happen! Rather than view past success as a reason to ramp down investments, such organisations continue to invest in anticipating the changing potential for failure because they appreciate that their knowledge of the gaps is imperfect and that their environment constantly changes. One measure of resilience is therefore the ability to create foresight—to anticipate the changing shape of risk,

before failure and harm occurs. (Reprinted by permission of the publishers.)

Hollnagel, Woods, & Leveson, 2006, p. 6.)

The Paradox of Automation

Machines are getting smarter. More and more tasks are becoming fully automated. As this happens, there is a tendency to believe that many of the difficulties involved with human control will go away. Across the world, automobile accidents kill and injure tens of millions of people every year. When we finally have widespread adoption of self-driving cars, the accident and casualty rate will probably be dramatically reduced, just as automation in factories and aviation have increased efficiency while lowering both error and the rate of injury.

When automation works, it is wonderful, but when it fails, the resulting impact is usually unexpected and, as a result, dangerous. Today, automation and networked electrical generation systems have dramatically reduced the amount of time that electrical power is not available to homes and businesses. But when the electrical power grid goes down, it can affect huge sections of a country and take many days to recover. With self-driving cars, I predict that we will have fewer accidents and injuries, but that when there is an accident, it will be huge.

Automation keeps getting more and more capable. Automatic systems can take over tasks that used to be done by people, whether it is maintaining the proper temperature, automatically keeping an automobile within its assigned lane at the correct distance from the car in front, enabling airplanes to fly by themselves from takeoff to landing, or allowing ships to navigate by themselves. When the automation works, the tasks are usually done as well as or better than by people. Moreover, it saves people from the dull, dreary routine tasks, allowing more useful, productive use of time, reducing fatigue and error. But when the task gets too complex, automation tends to give up. This, of course, is precisely when it is needed the most. The paradox is that automation can take over the dull, dreary tasks, but fail with the complex ones.

When automation fails, it often does so without warning. This is a situation I have documented very thoroughly in my other books and many of my papers, as have many other people in the field of safety and automation. When the failure occurs, the human is “out of the loop.” This means that the person has not been paying much attention to the operation, and it takes time for the failure to be noticed and evaluated, and then to decide how to respond.

In an airplane, when the automation fails, there is usually considerable time for the pilots to understand the situation and respond. Airplanes fly quite high: over 10 km (6 miles) above the earth, so even if the plane were to start falling, the pilots might have several minutes to respond. Moreover, pilots are extremely well trained. When automation fails in an automobile, the person might have only a fraction of a second to avoid an accident. This would be extremely difficult even for the most expert driver, and most drivers are not well trained.

In other circumstances, such as ships, there may be more time to respond, but only if the failure of the automation is noticed. In one dramatic case, the grounding of the cruise ship *Royal Majesty* in 1997, the failure lasted for several days and was only detected in the postaccident investigation, after the ship had run aground, causing several million dollars in damage. What happened? The ship’s location was normally determined by the Global Positioning System (GPS), but the cable that connected the satellite antenna to the navigation system somehow had become disconnected (nobody ever discovered how). As a result, the navigation system had switched from using GPS signals to “dead reckoning,” approximating the ship’s location by estimating speed and direction of travel, but the design of the navigation system didn’t make this apparent. As a result, as the ship traveled from Bermuda to its destination of Boston, it went too far south and went aground on Cape Cod, a peninsula jutting out of the water south of Boston. The automation had performed flawlessly for years, which increased people’s trust and reliance upon it, so the normal manual checking of location or careful perusal of the display (to see the tiny letters “dr” indicating “dead reckoning” mode) were not done. This was a huge mode error failure.

Design Principles for Dealing with Error

People are flexible, versatile, and creative. Machines are rigid, precise, and relatively fixed in their operations. There is a mismatch between the two, one that can lead to enhanced capability if used properly. Think of an electronic calculator. It doesn't do mathematics like a person, but can solve problems people can't. Moreover, calculators do not make errors. So the human plus calculator is a perfect collaboration: we humans figure out what the important problems are and how to state them. Then we use calculators to compute the solutions.

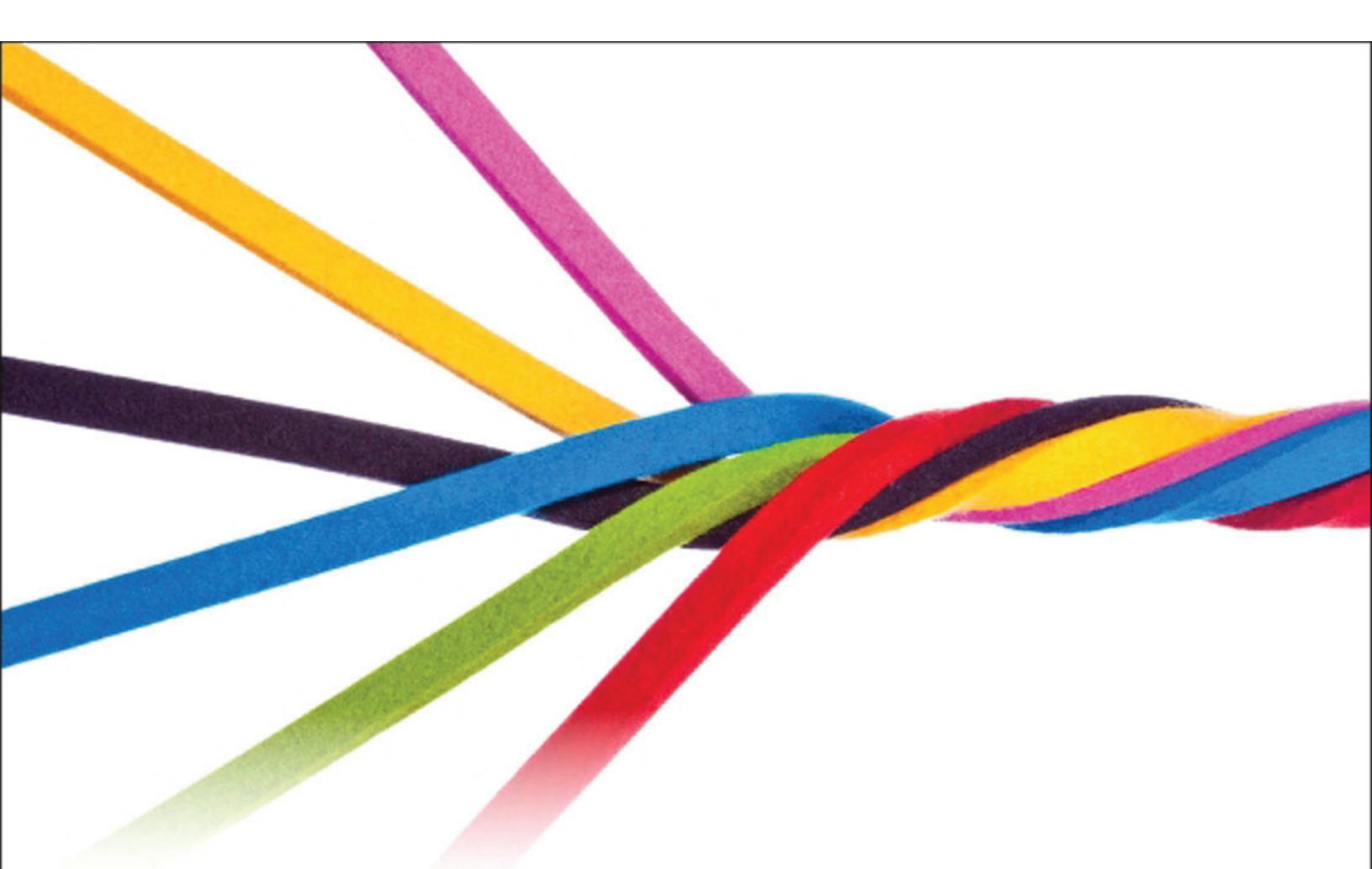
Difficulties arise when we do not think of people and machines as collaborative systems, but assign whatever tasks can be automated to the machines and leave the rest to people. This ends up requiring people to behave in machine like fashion, in ways that differ from human capabilities. We expect people to monitor machines, which means keeping alert for long periods, something we are bad at. We require people to do repeated operations with the extreme precision and accuracy required by machines, again something we are not good at. When we divide up the machine and human components of a task in this way, we fail to take advantage of human strengths and capabilities but instead rely upon areas where we are genetically, biologically unsuited. Yet, when people fail, they are blamed.

What we call "human error" is often simply a human action that is inappropriate for the needs of technology. As a result, it flags a deficit in our technology. It should not be thought of as error. We should eliminate the concept of error: instead, we should realize that people can use assistance in translating their goals and plans into the appropriate form for technology.

Given the mismatch between human competencies and technological requirements, errors are inevitable. Therefore, the best designs take that fact as given and seek to minimize the opportunities for errors while also mitigating the consequences. Assume that every possible mishap will happen, so protect against them. Make actions reversible; make errors less costly. Here are key design principles:

- Put the knowledge required to operate the technology in the world. Don't require that all the knowledge must be in the head. Allow for efficient operation when people have learned all the requirements, when they are experts who can perform without the knowledge in the world, but make it possible for non-experts to use the knowledge in the world. This will also help experts who need to perform a rare, infrequently performed operation or return to the technology after a prolonged absence.
- Use the power of natural and artificial constraints: physical, logical, semantic, and cultural. Exploit the power of forcing functions and natural mappings.
- Bridge the two gulfs, the Gulf of Execution and the Gulf of Evaluation. Make things visible, both for execution and evaluation. On the execution side, provide feedforward information: make the options readily available. On the evaluation side, provide feedback: make the results of each action apparent. Make it possible to determine the system's status readily, easily, accurately, and in a form consistent with the person's goals, plans, and expectations.

We should deal with error by embracing it, by seeking to understand the causes and ensuring they do not happen again. We need to assist rather than punish or scold.



Human-Computer Interaction

An Empirical Research Perspective



I. Scott MacKenzie

Human-Computer Interaction

An Empirical Research Perspective

I. Scott MacKenzie

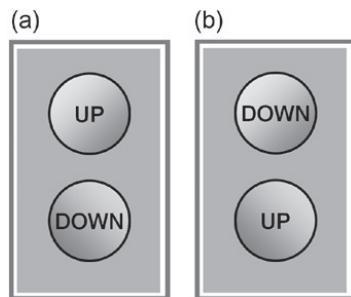


ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier



**FIGURE 3.18**

Button arrangements for an elevator control panel. (a) Correct. (b) Incorrect.

geographic regions have experienced and learned it differently. What is accepted in one region may differ from what is accepted in another.

If there is a physical contradiction, then the situation is different. Consider elevators (in buildings, not scrollbars). Early elevators didn't have buttons to specify floors; they only had UP and DOWN buttons. Consider the two arrangements for the button controls shown in Figure 3.18. Clearly the arrangement in (a) is superior. When the UP control is pressed, the display (the elevator) moves UP. The stimulus (control) and response (display) are compatible beyond doubt. In (b) the position of the controls is reversed. Clearly, there is an incompatibility between the stimulus and the response. This situation is different from the scroll pane example given in Figure 3.6 because there is no physical analogy to help the user (can you think of one?). If all elevator control panels had the arrangement in Figure 3.18b, would a population stereotype emerge, as with the light switch example? Well, sort of. People would learn the relationship, because they must. But they would make more errors than if the relationship was based on a correct physical mapping. This particular point has been the subject of considerable experimental testing, dating back to the 1950s (Fitts and Seeger, 1953). See also Newell (1990, 276–278) and Kantowitz and Sorkin (1983, 323–331). The gist of this work is that people take longer and commit more errors if there is a physical misalignment between displays and controls, or between controls and the responses they effect.

This work is important to HCI at the very least to highlight the challenges in designing human-computer systems. The physical analogies that human factors engineers seek out and exploit in designing better systems are few and far between in human-computer interfaces. Sure, there are physical relationships like “mouse right, cursor right,” but considering the diversity of people’s interactions with computers, the tasks with physical analogies are the exception. For example, what is the physical analogy for “file save”? Human-computer interfaces require a different way of thinking. Users need help—a lot of help. The use of metaphor is often helpful.

3.4 Mental models and metaphor

There is more to learning or adapting than simply experiencing. One of the most common ways to learn and adapt is through *physical analogy* (Norman, 1988, p. 23)

or *metaphor* (Carroll and Thomas, 1982). Once we latch on to a physical understanding of an interaction based on experience, it all makes sense. We've experienced it, we know it, it seems natural. With a scroll pane, moving the slider up moves the *view* up. If the relationship were reversed, moving the slider up would move the *content* up. We could easily develop a physical sense of slider up → view up or slider up → content up. The up-up in each expression demonstrates the importance of finding a spatially congruent physical understanding. These two analogies require opposite control-display relationships, but either is fine and we could work with one just as easily as with the other, provided implementations were consistent across applications and platforms.

Physical analogies and metaphors are examples of the more general concept of *mental models*, also known as *conceptual models*. Mental models are common in HCI. The idea is simple enough: “What is the user’s mental model of . . . ?” An association with human experience is required. HCI’s first mental model was perhaps that of the office or desktop. The desktop metaphor helped users understand the graphical user interface. Today it is hard to imagine the pre-GUI era, but in the late 1970s and early 1980s, the GUI was strange. It required a new way of thinking. Designers exploited the metaphor of the office or desktop to give users a jump-start on the interface (Johnson et al., 1989). And it worked. Rather than learning something new and unfamiliar, users could act out with concepts already understood: documents, folders, filing cabinets, trashcans, the top of the desk, pointing, selecting, dragging, dropping, and so on. This is the essence of mental models.

Implementation models are to be avoided. These are systems that impose on the user a set of interactions that follow the inner workings of an application. Cooper and Reimann give the example of a software-based fax product where the user is paced through a series of agonizing details and dialogs (Cooper and Riemann, 2003, p. 25). Interaction follows an implementation model, rather than the user’s mental model of how to send a fax. The user is prompted for information when it is convenient for the program to receive it, not when it makes sense to the user. Users often have pre-existing experiences with artifacts like faxes, calendars, media players, and so on. It is desirable to exploit these at every opportunity in designing a software-based product. Let’s examine a few other examples in human-computer interfaces.

Toolbars in GUIs are fertile ground for mental models. To keep the buttons small and of a consistent size, they are adorned with an icon rather than a label. An icon is a pictorial representation. In HCI, icons trigger a mental image in the user’s mind, a clue to a real-world experience that is similar to the action associated with the button or tool. Icons in drawing and painting applications provide good examples. Figure 3.19a shows the Tool Palette in Corel’s *Paint Shop Pro*, a painting and image manipulation application.¹² The palette contains 21 buttons, each displaying an icon. Each button is associated with a function and its icon is carefully chosen to elicit the association in the user’s mind. Some are clear, like the magnifying glass or the paintbrush. Some are less clear. Have a look. Can you tell what action is

¹²www.jasc.com.

**FIGURE 3.19**

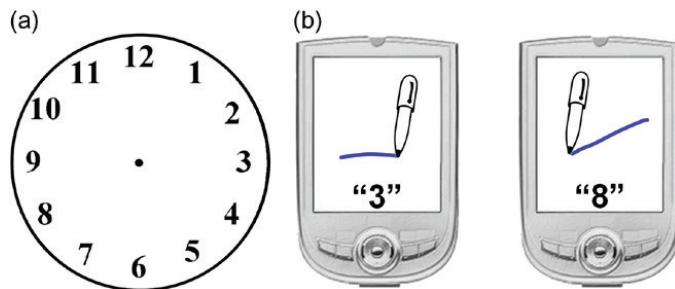
Icons create associations. (a) Array of toolbar buttons from Corel's *Paint Shop Pro*.
 (b) Tooltip help for "Picture Tube" icon.

associated with each button? Probably not. But users of this application likely know the meaning of most of these buttons.

Preparing this example gave me pause to consider my own experience with this toolbar. I use this application frequently, yet some of the buttons are entirely strange to me. In 1991 Apple introduced a method to help users like me. Hover the mouse pointer over a GUI button and a field pops up providing a terse elaboration on the button's purpose. Apple called the popups *balloons*, although today they are more commonly known as *tooltips* or *screen tips*. Figure 3.19b gives an example for a button in *Paint Shop Pro*. Apparently, the button's purpose is related to a picture tube. I'm still in the dark, but I take solace in knowing that I am just a typical user: "Each user learns the smallest set of features that he needs to get his work done, and he abandons the rest." (Cooper, 1999, p. 33)

Another example of mental models are a compass and a clock face as metaphors for direction. Most users have an ingrained understanding of a compass and a clock. The inherent labels can serve as mental models for direction. Once there is an understanding that a metaphor is present, the user has a mental model and uses it efficiently and accurately for direction: *north*, for straight ahead or up, *west* for left, and so on. As an HCI example, Lindeman et al. (2005) used the mental model of a compass to help virtual reality users navigate a building. Users wore a vibro-tactile belt with eight actuators positioned according to compass directions. They were able to navigate the virtual building using a mental model of the compass. There is also a long history in HCI of using a compass metaphor for stylus gestures with pie menus (Callahan et al., 1988) and marking menus (G. P. Kurtenbach, Sellen, and Buxton, 1993; Li, Hinckley, Guan, and Landay, 2005).

With twelve divisions, a clock provides finer granularity than a compass ("obstacle ahead at 2 o'clock!"). Examples in HCI include numeric entry (Goldstein, Chincholle, and Backström, 2000; Isokoski and Käki, 2002; McQueen, MacKenzie, and Zhang, 1995) and locating people and objects in an environment (Sáenz and Sánchez, 2009; A. Sellen, Eardley, Iazdi, and Harper, 2006). Using a clock metaphor for numeric entry with a stylus is shown in Figure 3.20. Instead of scripting numbers using Roman characters, the numbers are entered using straight-line strokes. The direction of the stroke is the number's position on a clock face. In a longitudinal study, McQueen et al. (1995) found that numeric entry was about

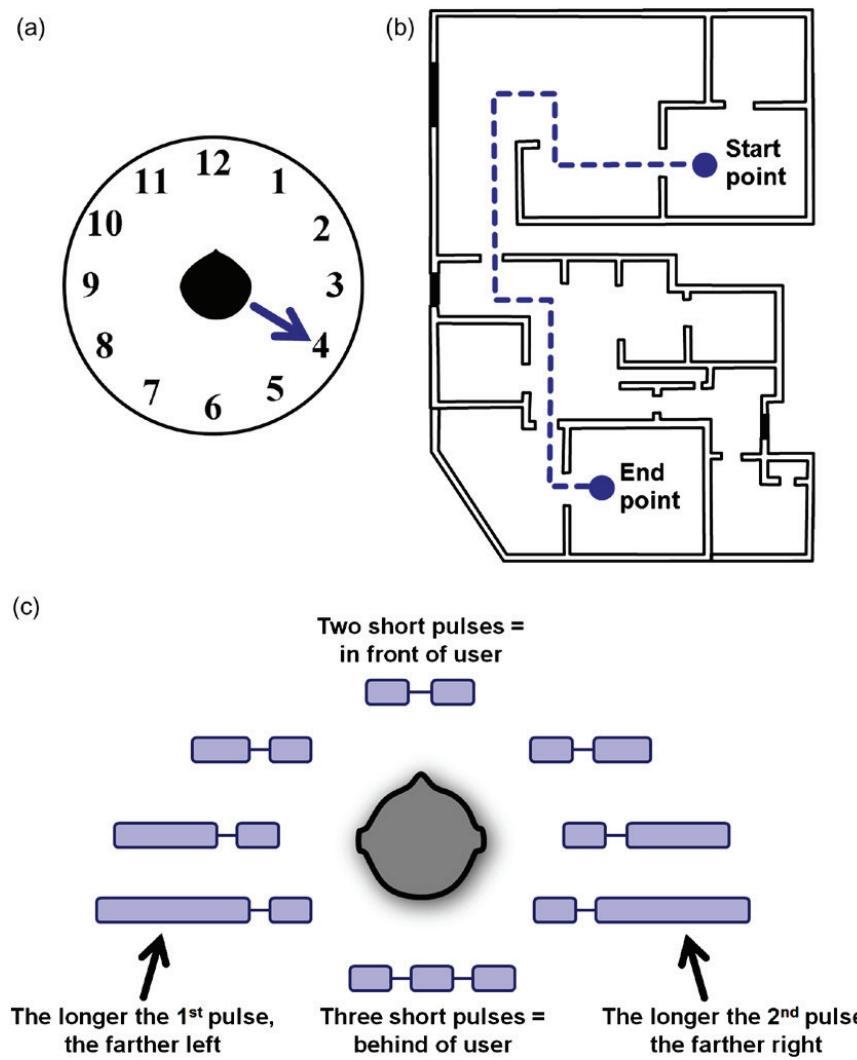
**FIGURE 3.20**

Mental model example: (a) Clock face. (b) Numeric entry with a stylus.

24 percent faster using straight-line strokes compared to handwritten digits. The 12 o'clock position was used for 0. The 10 o'clock and 11 o'clock positions were reserved for system commands.

Sáenz and Sánchez describe a system to assist the blind (Sáenz and Sánchez, 2009) using the clock metaphor. Users carried a mobile locating device that provided spoken audio information about the location of nearby objects (see Figure 3.21a). For the metaphor to work, the user is assumed to be facing the 12 o'clock position. The system allowed users to navigate a building eyes-free (Figure 3.21b). Users could request position and orientation information from the locator. Auditory responses were provided using the clock metaphor and a text-to-speech module (e.g., “door at 3 o'clock”). A similar interface is Rümelin et al.'s *NaviRadar* (Rümelin, Rukzio, and Hardy, 2012), which uses tactile feedback rather than auditory feedback. Although not specifically using the clock metaphor, *NaviRadar* leverages users' spatial sense of their surroundings to aid navigation. Users receive combinations of long and short vibratory pulses to indicate direction (Figure 3.21c). Although the patterns must be learned, the system is simple and avoids auditory feedback, which may be impractical in some situations.

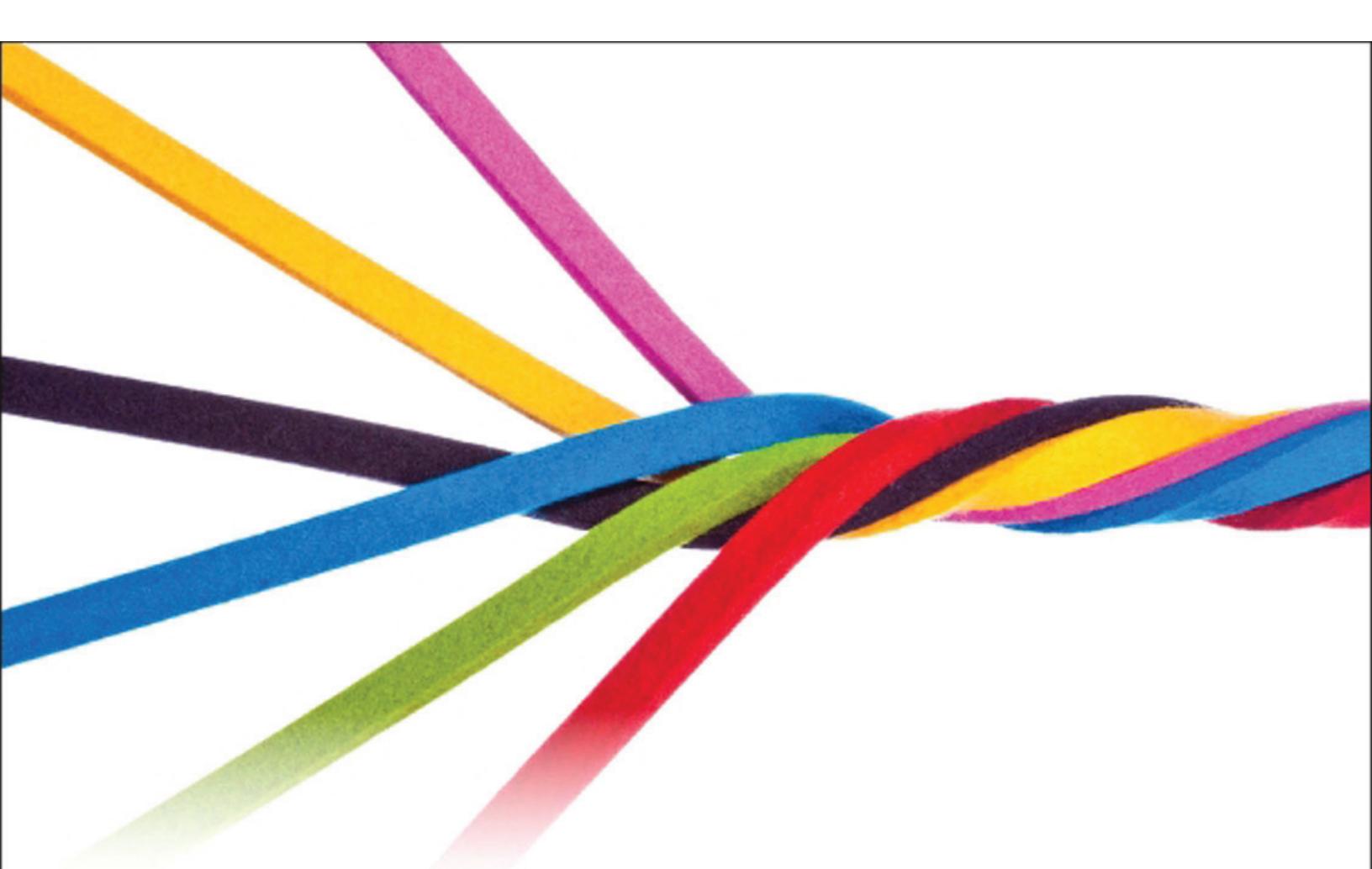
The systems described by Sáenz and Sánchez (2009) and Rümelin et al. (2012) have similar aims yet were presented and evaluated in different ways. Sáenz and Sánchez emphasized and described the system architecture in detail. Although this is of interest to some in the HCI community, from the user's perspective the system architecture is irrelevant. A user test was reported, but the evaluation was not experimental. There were no independent or dependent variables. Users performed tasks with the system and then responded to questionnaire items, expressing their level of agreement to assertions such as “The software was motivating,” or “I like the sounds in the software.” While qualitative assessments are an essential component of any evaluation, the navigation and locating aides described in this work are well suited to experimental testing. Alternative implementations, even minor modifications to the interface, are potential independent variables. Speed (e.g., the time to complete tasks) and accuracy (e.g., the number of wrong turns, retries, direction changes, wall collisions) are potential dependent variables.

**FIGURE 3.21**

Spatial metaphor: (a) Auditory feedback provides information for locating objects, such as “object at 4 o’clock.” (b) Navigation task. (c) NaviRadar.

(Source: b, adapted from Sáenz and Sánchez, 2009; c, adapted from Rümelin et al., 2012)

Rümelin et al. (2012) took an empirical approach to system tests. Their research included both the technical details of *NaviRadar* and an evaluation in a formal experiment with independent variables, dependent variables, and so on. The main independent variable included different intensities, durations, and rhythms in the tactile pulses. Since their approach was empirical, valuable analyses were possible. They reported, for example, the deviation of indicated and reported directions and how this varied according to direction and the type of tactile information given. Their approach enables other researchers to study the strengths and weaknesses in *NaviRadar* in empirical terms and consider methods of improvement.



Human-Computer Interaction

An Empirical Research Perspective



I. Scott MacKenzie

Human-Computer Interaction

An Empirical Research Perspective

I. Scott MacKenzie



ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier



3.8 Interaction errors

Most of analyses in this chapter are directed at the physical properties of the human-machine interface, such as degrees of freedom in 2D or 3D or spatial and temporal relationships between input controllers and output displays. Human performance, although elaborated in Chapter 2, has not entered into discussions here, except through secondary observations that certain interactions are better, worse, awkward, or unintuitive. At the end of the day, however, human performance is what counts. Physical properties, although instructive and essential, are secondary. Put another way, human performance is like food, while physical properties are like plates and bowls. It is good and nutritious food that we strive for.

Empirical research in HCI is largely about finding the physical properties and combinations that improve and enhance human performance. We conclude this chapter on interaction elements with comments on that nagging aspect of human performance that frustrates users: interaction errors. Although the time to complete a task can enhance or hinder by degree, errors only hinder. Absence of errors is, for the most part, invisible. As it turns out, errors—interaction errors—are germane to the HCI experience. The big errors are the easy ones—they get fixed. It is the small errors that are interesting.

As the field of HCI matures, a common view that emerges is that the difficult problems (in desktop computing) are solved, and now researchers should focus on new frontiers: mobility, surface computing, ubiquitous computing, online social networking, gaming, and so on. This view is partially correct. Yes, the emerging themes are exciting and fertile ground for HCI research, and many frustrating UI problems from the old days are gone. But desktop computing is still fraught with problems, lots of them. Let's examine a few of these. Although the examples below are from desktop computing, there are counterparts in mobile computing. See also student exercise 3-8 at the end of this chapter.

The four examples developed in the following discussion were chosen for a specific reason. There is a progression between them. In severity, they range from serious problems causing loss of information to innocuous problems that most users rarely think about and may not even notice. In frequency, they range from rarely, if ever, occurring any more, to occurring perhaps multiple times every minute while users engage in computing activities. The big, bad problems are well-traveled in the literature, with many excellent sources providing deep analyses on what went wrong and why (e.g., Casey, 1998, 2006; Cooper, 1999; Johnson, 2007;

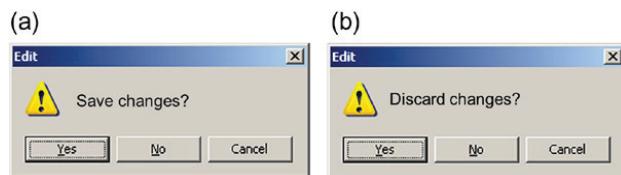


FIGURE 3.42

HCI has come a long way: (a) Today's UIs consistently use the same, predictable dialog to alert the user to a potential loss of information. (b) Legacy dialog rarely (if ever) seen today.

B. H. Kantowitz and Sorkin, 1983; Norman, 1988). While the big problems get lots of attention, and generally get fixed, the little ones tend to linger. We'll see the effect shortly. Let's begin with one of the big problems.

Most users have, at some point, lost information while working on their computers. Instead of saving new work, it was mistakenly discarded, overwritten, or lost in some way. Is there any user who has not experienced this? Of course, nearly everyone has a story of losing data in some silly way. Perhaps there was a distraction. Perhaps they just didn't know what happened. It doesn't matter. It happened. An example is shown in Figure 3.42. A dialog box pops up and the user responds a little too quickly. Press ENTER with the "Save changes?" dialog box (Figure 3.42a) and all is well, but the same response with the "Discard changes?" dialog box spells disaster (Figure 3.42b). The information is lost. This scenario, told by Cooper (1999, 14), is a clear and serious UI design flaw. The alert reader will quickly retort, "Yes, but if the 'Discard changes?' dialog box defaults to 'No,' the information is safe." But that misses the point. The point is that a user expectation is broken. Broken expectations sooner or later cause errors.

Today, systems and applications consistently use the "Save changes?" dialog box in Figure 3.42a. With time and experience, user expectations emerge and congeal. The "Save changes?" dialog box is expected, so we act without hesitating and all is well. But new users have no experiences, no expectations. They will develop them sure enough, but there will be some scars along the way. Fortunately, serious flaws like the "Discard changes?" dialog box are rare in desktop applications today.

The following is another error to consider. If prompted to enter a password, and CAPS_LOCK mode is in effect, logging on will fail and the password must be reentered. The user may not know that CAPS_LOCK is on. Perhaps a key-stroking error occurred. The password is reentered, slowly and correctly, with the CAPS_LOCK mode still in effect. Oops! Commit the same error a third time and further log-on attempts may be blocked. This is not as serious as losing information by pressing ENTER in response to a renegade dialog box, but still, this is an interaction error. Or is it a design flaw? It is completely unnecessary, it is a nuisance, it slows our interaction, and it is easy to correct. Today, many systems have corrected this problem (Figure 3.43a), while others have not (Figure 3.43b).

The CAPS_LOCK error is not so bad. But it's bad enough that it occasionally receives enough attention to be the beneficiary of the few extra lines of code necessary to pop up a CAPS_LOCK alert.

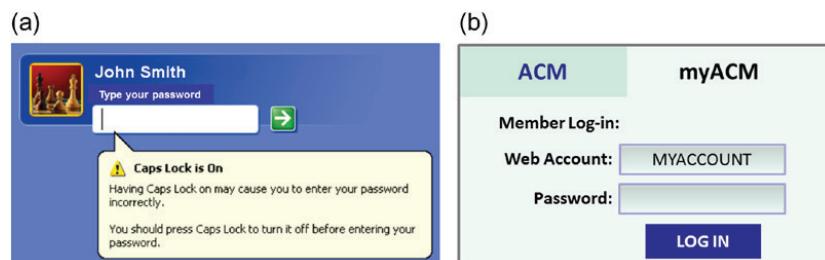


FIGURE 3.43

Entering a password: (a) Many systems alert the user if CAPS_LOCK is on. (b) Others do not.

Let's examine another small problem. In editing a document, suppose the user wishes move some text to another location in the document. The task is easy. With the pointer positioned at the beginning of the text, the user presses and holds the primary mouse button and begins dragging. But the text spans several lines and extends past the viewable region. As the dragging extent approaches the edge of the viewable region, the user is venturing into a difficult situation. The interaction is about to change dramatically. (See Figure 3.44.) Within the viewable region, the interaction is position-control—the displacement of the mouse pointer controls the *position* of the dragging extent. As soon as the mouse pointer moves outside the viewable region, scrolling begins and the interaction becomes velocity-control—the displacement of the mouse pointer now controls the *velocity* of the dragging extent. User beware!

Once in velocity-control mode, it is anyone's guess what will happen. This is a design flaw. A quick check of several applications while working on this example revealed dramatically different responses to the transition from position control to velocity control. In one case, scrolling was so fast that the dragging region extended to the end of the document in less time than the user could react (≈ 200 ms). In another case, the velocity of scrolling was controllable but frustratingly slow. Can you think of a way to improve this interaction? A two-handed approach, perhaps. Any technique that gets the job done and allows the user to develop an expectation of the interaction is an improvement. Perhaps there is some empirical research waiting in this area.

Whether the velocity-control is too sensitive or too sluggish really doesn't matter. What matters is that the user experience is broken or awkward. Any pretense to the interaction being facile, seamless, or transparent is gone. The user will recover, and no information will be lost, but the interaction has degraded to error recovery. This is a design error or, at the very least, a design-induced error. Let's move on to a very minor error.

When an application or a dialog box is active, one of the UI components has focus and receives an event from the keyboard if a key is pressed. For buttons,

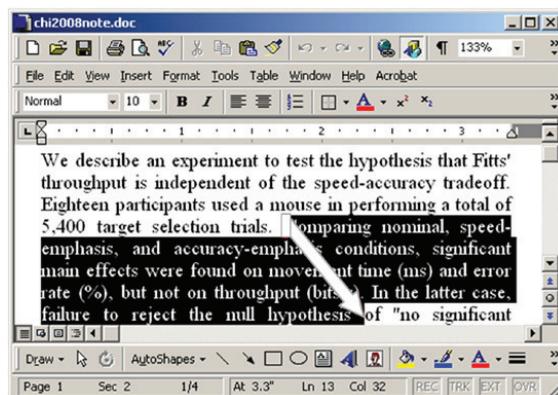


FIGURE 3.44

On the brink of hyper-speed scrolling. As the mouse pointer is dragged toward the edge of the viewable region, the user is precipitously close to losing control over the speed of dragging.

focus is usually indicated with a dashed border (see “Yes” button in [Figure 3.42](#)). For input fields, it is usually indicated with a flashing insertion bar (“|”). “Focus advancement” refers to the progression of focus from one UI component to the next. There is wide-spread inconsistency in current applications in the way UI widgets acquire and lose focus and in the way focus advances from one component to the next. The user is in trouble most of the time. Here is a quick example. When a login dialog box pops up, can you immediately begin to enter your username and password? Sometimes yes, sometimes no. In the latter case, the entry field does not have focus. The user must click in the field with the mouse pointer or press TAB to advance the focus point to the input field. [Figure 3.43](#) provides examples. Both are real interfaces. The username field in (a) appears with focus; the same field in (b) appears without focus. The point is simply that users don’t know. This is a small problem (or is it an interaction error?), but it is entirely common. Focus uncertainty is everywhere in today’s user interfaces. Here is another, more specific example:

Many online activities, such as reserving an airline ticket or booking a vacation, require a user to enter data into a form. The input fields often require very specific information, such as a two-digit month, a seven-digit account number, and so on. When the information is entered, does focus advance automatically or is a user action required? Usually, we just don’t know. So we remain “on guard.” [Figure 3.45](#) gives a real example from a typical login dialog box. The user is first requested to enter an account number. Account numbers are nine digits long, in three three-digit segments. After seeing the dialog box, the user looks at the keyboard and begins entering: 9, 8, 0, and then what? Chances are the user is looking at the keyboard while entering the numeric account number. Even though the user can enter the entire nine digits at once, interaction is halted after the first three-digit group because the user doesn’t know if the focus will automatically advance to the next field. There are no expectations here, because this example of GUI interaction has not evolved and stabilized to a consistent pattern. Data entry fields have not reached the evolutionary status of, for example, dialog boxes for saving versus discarding changes ([Figure 3.42a](#)). The user either acts, with an approximately 50 percent likelihood of committing an error, or pauses to attend to the display (*Has the focus advanced to the next field?*).

Strictly speaking, there is no gulf of evaluation here. Although not shown in the figure, the insertion point is present. After entering 980, the insertion point is either after the 0 in the first field, if focus did not advance, or at the beginning of the next field, if focus advanced. So the system does indeed “provide a physical

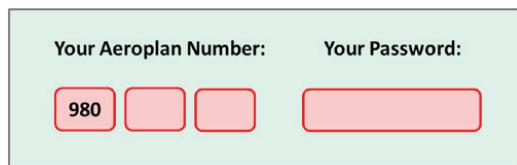


FIGURE 3.45

Inconsistent focus advancement keeps the user on guard. “What do I do next?”

representation that can be perceived and that is directly interpretable in terms of the intentions and expectations of the person” (Norman, 1988, p. 51). That’s not good enough. The user’s attention is on the keyboard while the physical presentation is on the system’s display. The disconnect is small, but nevertheless, a shift in the user’s attention is required.

The absence of expectations keeps the user on guard. The user is often never quite sure what to do or what to expect. The result is a slight increase in the attention demanded during interaction, which produces a slight decrease in transparency. Instead of engaging in the task, attention is diverted to the needs of the computer. The user is like a wood carver who sharpens tools rather than creates works of art.

Where the consequences of errors are small, such as an extra button click or a gaze shift, errors tend to linger. For the most part, these errors aren’t on anyone’s radar. The programmers who build the applications have bigger problems to focus on, like working on their checklist of new features to add to version 2.0 of the application before an impending deadline.²² The little errors persist. Often, programmers’ discretion rules the day (Cooper, 1999, p. 47). An interaction scenario that makes sense to the programmer is likely to percolate through to the final product, particularly if it is just a simple thing like focus advancement. Do programmers ever discuss the nuances of focus advancement in building a GUI? Perhaps. But was the discussion framed in terms of the impact on the attention or gaze shifts imposed on the user? Not likely.

Each time a user shifts his or her attention (e.g., from the keyboard to the display and back), the cost is two gaze shifts. Each gaze shift, or saccade, takes from 70 to 700 ms (Card et al., 1983, p. 28).²³ These little bits of interaction add up. They are the fine-grained details—the microstructures and microstrategies used by, or imposed on, the user. “Microstrategies focus on what designers would regard as the mundane aspects of interface design; the ways in which subtle features of interactive technology influence the ways in which users perform tasks” (W. D. Gray and Boehm-Davis, 2000, p. 322). Designers might view these fine-grained details as a mundane sidebar to the bigger goal, but the reality is different. Details are everything. User experiences exist as collections of microstrategies. Whether booking a vacation online or just hanging out with friends on a social networking site, big actions are collections of little actions. To the extent possible, user actions form the experience, our experience. It is unfortunate that they often exist simply to serve the needs of the computer or application.

²²The reader who detects a modicum of sarcasm here is referred to Cooper (1999, 47–48 and elsewhere) for a full frontal assault on the insidious nature of feature bloat in software applications. The reference to version 2.0 of a nameless application is in deference to Johnson’s second edition of his successful book where the same tone appears in the title: *GUI Bloopers 2.0*. For a more sober and academic look at software bloat, feature creep, and the like, see McGrenere and Moore (2000).

²³An eye movement involves both a saccade and fixation. A saccade—the actual movement of the eye—is fast, about 30 ms. Fixations takes longer as they involve perceiving the new stimulus and cognitive processing of the stimulus.

Another reason little errors tend to linger is that they are often deemed *user errors*, not design, programming, or system errors. These errors, like most, are more correctly called *design-induced errors* (Casey, 2006, p. 12). They occur “when designers of products, systems, or services fail to account for the characteristics and capabilities of people and the vagaries of human behavior” (Casey, 1998, p. 11). We should all do a little better.

Figure 3.46 illustrates a tradeoff between the cost of errors and the frequency of errors. There is no solid ground here, so it’s just a sketch. The four errors described above are shown. The claim is that high-cost errors occur with low frequency. They receive a lot of attention and they get dealt with. As systems mature and the big errors get fixed, designers shift their efforts to fixing less costly errors, like the `CAPS_LOCK` design-induced error, or consistently implementing velocity-controlled scrolling. Over time, more and more systems include reasonable and appropriate implementations of these interactions. Divergence in the implementations diminishes and, taken as a whole, there is an industry-wide coalescing toward the same consistent implementation (e.g., a popup alert for `CAPS_LOCK`). The ground is set for user expectation to take hold.

Of the errors noted in Figure 3.46, discard changes is ancient history (in computing terms), `CAPS_LOCK` is still a problem but is improving, scrolling frenzy is much more controlled in new applications, and focus uncertainty is, well, a mess. The cost is minor, but the error happens frequently.

In many ways, the little errors are the most interesting, because they slip past designers and programmers. A little self-observation and reflection goes a long way here. Observe little errors that you encounter. What were you trying to do? Did it work the first time, just as expected? Small interactions are revealing. What were your hands and eyes doing? Were your interactions quick and natural, or were there unnecessary or awkward steps? Could a slight reworking of the interaction help? Could an attention shift be averted with the judicious use of auditory or tactile feedback? Is there a “ready for input” auditory signal that could sound when an input field receives focus? Could this reduce the need for an attention shift? Would this improve user performance? Would it improve the user experience? Would users like it, or would it be annoying? The little possibilities add up. Think of them as opportunities for empirical research in HCI.

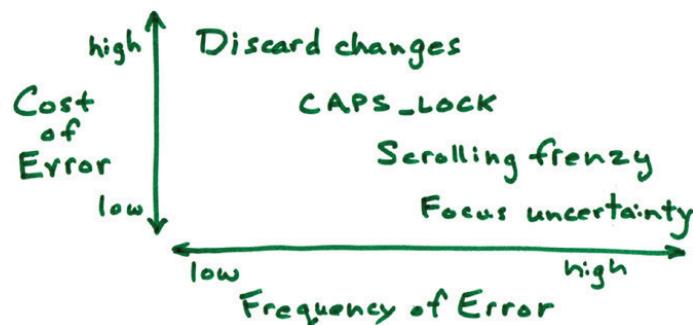


FIGURE 3.46

Trade-off between the cost of errors and the frequency of errors.

A 'Pile' Metaphor for Supporting Casual Organization of Information

Richard Mander, Gitta Salomon and Yin Yin Wong

Human Interface Group, Advanced Technology
 Apple Computer, Inc.
 20525 Mariani Ave., MS 76-3H
 Cupertino, California 95014
 (408)996-1010

ABSTRACT

A user study was conducted to investigate how people deal with the flow of information in their workspaces. Subjects reported that, in an attempt to quickly and informally manage their information, they created piles of documents. Piles were seen as complementary to the folder filing system, which was used for more formal archiving. A new desktop interface element – the pile – was developed and prototyped through an iterative process. The design includes direct manipulation techniques and support for browsing, and goes beyond physical world functionality by providing system assistance for automatic pile construction and reorganization. Preliminary user tests indicate the design is promising and raise issues that will be addressed in future work.

KEYWORDS: interface design, design process, interactive systems, user observation, desktop metaphor, interface metaphors, pile metaphor, information visualization, information organization, end-user programming.

INTRODUCTION

As the amount of information users confront on their computers increases, tools to organize and manipulate this information become increasingly important.

Today's direct manipulation computer interfaces, such as the Macintosh® desktop interface [1], offer limited means of handling information. Users can manually place files within folders, organized in a rigid hierarchy. Users are responsible for appropriately filing all items; the system offers little assistance in this often tedious task. Recent enhancements, such as "aliases" [2], allow users to overcome a frequent problem, namely that an item belongs in more than one folder. However, the folder as the sole container type presents an impoverished set of possibilities.

The real world provides a rich array of organization systems. In the past, researchers have looked at how users find items in their physical offices [9]. We conducted a study to observe how users *organize* the large amounts of information they work with in their physical offices. Our study differed from previous work in that we looked at ways in which people use and interact with filing systems.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

We were also interested in how people work with assistants when dealing with information.

By examining individuals' information management schemes, we were able to extract and extrapolate a number of interesting interface ideas for a graphical interface. Our intent was not simply to emulate physical world functionality – several investigators have argued against this procedure [3,6] – but rather to leverage users' knowledge to create an intuitive and powerful system that goes beyond physical world capabilities. Using this approach, we sought to construct a design which provides new functionality and enhances the user interface.

Like Malone [9], we found that users like to group items spatially and often prefer to deal with information by creating physical piles of paper, rather than immediately categorizing it into specific folders. Computer users are confronted with large amounts of information, but currently are only provided with a hierarchical filing system for managing it.

Therefore, we propose that incorporating 'piles' within a graphical user interface could provide a number of interesting possibilities. Users have difficulty deciding where to file a new item; piling requires less mental effort. Today's office assistants use piles as a way of suggesting categories to others; computerized agents might make use of them in the same way to convey a certain degree of imprecision in the suggested organization. Piles may also provide an appropriate representation for the results of information retrieval algorithms which are inherently inexact [13].

At least one system, BUSINESS [11], previously explored this interface metaphor as a construct within a text-based application programming language. For example, a user could initiate an action by typing an instruction such as "Empty the In Box onto the Work Pile." However, there was no graphical representation, and so the system could do little more than allow the user to issue programmatic commands using a subset of English.

This paper provides both specific design ideas and insight into our design process as it progressed from user interviews to design to testing. The first section describes findings from observing and interviewing office workers. In particular, we report why folders were not always appropriate and how and when users found piles useful. We then describe the interface designs inspired by these observations. In the third section, we report results from informal tests of

these designs. In conclusion, we describe directions for future work.

USER INTERVIEWS

As part of our design process, we undertook a user study to find out how people deal with information in their physical workspaces. Studies of this kind are important in helping us understand the user's perspective. Our aim was to identify aspects of the real world work process which could offer insight into a new, more powerful interface.

Method

The study involved interviews with thirteen men and women in Marketing, Support, Human Resources, and Technical departments within Apple Computer, Inc. The interviews lasted between 30 and 60 minutes and were conducted in the participant's work area. All interviews were videotaped.

We asked people to describe the way information arrived in their work area, what they initially did with this information, where it went next, and how it was finally stored. Participants gave us a tour of their workspace to help us understand what purpose various cabinets, shelves, and storage devices served. We also took a small pile of documents with us and asked participants to judge what these documents were on the basis of their appearance. In this way, we could find out how they worked with completely unfamiliar information. Since we are interested in developing ways for the computer to help the user, we also asked people how they worked with assistants.

Findings

Our subjects used a variety of techniques – folders, file cabinets, file racks, piles, binders, card files, and bulletin boards – for managing the information in their offices. Since our primary concern in this paper is the uses of folders and piles, we'll focus on observations relevant to these items.

Uses for folders. File folders were used in several ways. As could be expected, items were placed in folders which were in turn placed in file cabinets as a means of archiving information not currently needed. Users applied a variety of organizations to their file cabinets, ranging from totally random arrangements to strict alphabetical and color coded systems.

Users were sometimes dissatisfied with using folders in this way, because they were required to make an explicit decision about how to categorize individual items. This was often especially difficult with new information. One user said "I'm not always as good at categorizing things as I would like...it's hard to get it right and I'm sort of a perfectionist, so I think that I should know exactly how I should do it...I like things in their place, but I can't figure out exactly what place."

One solution, identified by several users, would be to file the information in several places. However, even though copiers were near at hand, people did not choose to duplicate information in order to store it in more than one folder.

Folders were also used in informal ways. Many people mounted folders in racks, which enabled the folders to stand up. These folders were used for frequently accessed information – most often action items and items requiring

regular maintenance, such as expense reports and things to read. Some users ordered or changed the orientation of the folders in their racks to make the most important or urgent information prominent. Folders were also used as a storage medium within piles, as a way to hold together a certain group of items. As one user commented, "...[I] folderize to keep things neat...there's no hierarchy in there, because building a hierarchy takes too much time."

Based on these observations, we inferred two things about the current folder-based interface offered on the Macintosh: the categorization problems are presumably amplified by the use of multiply nested folders, and support for more informal grouping techniques, such as racks, could be useful.

Piles: A less rigid categorization system. In addition to using folders, users grouped items into piles. For example, most workers kept information they needed in a specific working area. A common strategy was to create separate piles for each project and place them within the working area, at distances that reflected their urgency. Many workers also created piles for incoming information that they could not deal with immediately. The contents of users' piles was clearly not restricted to paper documents – we observed piles composed of various items such as books, folders, reports, binders, cassette tapes, video tapes, postcards, envelopes, magazines, journals, and boxes.

People used piles instead of hierarchical folders because they did not require detailed categorization and they could be more easily reordered than a folder and file system. For many workers, the pile was viewed as an entity that was subject to change. Users reported that over a period of time, items within a pile would often be reshuffled and broken down into several sub-piles, and an informal process of categorization would begin. We noted several approaches to separating material within piles: some users stacked materials at different angles, while some placed dividers within the pile.

To the outside observer, an office containing piles often appears disorganized. However, all of our participants had several piles in their workspace and in most cases, they knew what was in each pile and could tell us quite a lot about its history. Seemingly disordered piles were often sensible to the person who created them, because they developed through many interactions over a long period of time. For instance, many piles grew as newer items were added to the top, and workers could tell where things were by their date, since the stack was ordered chronologically.

Piles: self-revealing, browseable. Several users remarked that the outer appearance of their piles conveniently allowed them to recognize particular items. Our subjects were also able to make use of the appearance of previously unseen piles. We asked them to look at a small pile of unfamiliar materials which we took with us to the interview. By looking at the pile's outside form, they were able to infer quite a lot about its contents.

Consequently, we noted that piles facilitate browsing, and we observed four different browsing methods. In the *edge* browse method described above, people looked at the outside edges of the pile for clues about the items within. Information such as color, texture, and thickness was commonly used to judge the contents of a pile. In the *restack*

method, people started at the top of the pile and dealt with each item in turn by lifting it off the pile, looking at it and then placing it somewhere other than back on the pile. In the *hinge* method, the items stayed in the pile, but the pile was hinged open at different points to display a single item. The final method was to *spread out* a pile and look at its contents in parallel.

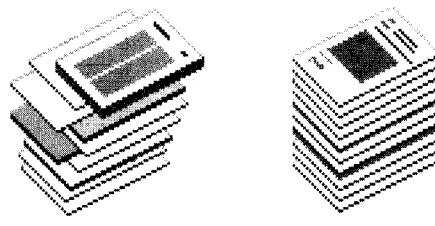
Assistance with information management. Most participants did not have an assistant, but said they would welcome one. We asked those who did have assistants to describe how they worked together.

Assistants commonly took care of routine tasks, such as sorting mail into different categories. For people who had to deal with large amounts of information, their assistant acted as a filter, passing along urgent material and removing junk mail. Some assistants would reorganize the workspace and create a filing system in which information could be more easily organized. This usually happened in collaboration with the worker. Typically the assistant would suggest categories and discuss these with the worker before actually filing the material. The assistant would often not understand the technical content, but could scan through the materials looking for keywords that might help in the categorization task. Piles were often used by assistants to indicate potential categories. As one assistant remarked, "I'll go into his office and put [labels] on piles on his floor and he'll look at it and say 'no' or he'll say 'that's pretty good'."

FROM OBSERVATION TO DESIGN SKETCH

The next step in our process was to take our observations and develop a number of design sketches using MacroMind's Director™ application [8] which supports scripted interaction and animation. These design sketches illustrated particular interaction techniques and were used to facilitate group discussion about interaction possibilities and the technology necessary to support them. They centered around the development of a new organizational element – the pile – which would support informal groupings of items on the computer desktop. In addition, we extended the metaphor to include functionality which could only be provided by the existence of a computer. The design sketches created are described below.

User-created piles. One objective was to allow users to create piles of mixed content and multiple data types. Each item within a pile would be represented by a miniature depicting its first page and extent (see Figure 1a). We wanted to maintain the informal quality of physical piles by provid-



(a)

(b)

Figure 1. Piles on the desktop. In general, piles can contain various media, such as folders and individual documents. The pile in (a) was created by the user, and is consequently disheveled in appearance. In addition, the system can create piles for the user, based on rules explicitly stated by the user or developed through user-system collaboration. These piles have a neat appearance, as shown in (b), to indicate that there is a script, or set of rules, behind them.

ing direct manipulation techniques which resemble real world interactions. For example, a pile is created by overlapping two items; items are added to an existing pile by simply placing them on top (Figure 2). These user-created piles have a disheveled look.

System-created piles. In addition, we postulated that the system could create piles for a user. As shown in Figure 1 (b), these piles would have an orderly appearance. The system would assemble these piles using a script either developed through user-system collaboration, or explicitly written by the user. By creating and maintaining piles for the user, the system could serve as an office assistant.

How might this user-system collaboration work? Potentially, the user could supply sample documents as input for pile construction. By analyzing these documents, the system could offer various criteria for script construction. For example, the system could determine a document's unique terms and let the user select the specific terms to use as piling criteria. Additionally, the system might extract structural data, such as the "Re:" line in a mail message and ask the user if similar mail messages should be collected into the pile. Malone suggested a similar tact for automatic classification [9] and reported successful results in the Information Lens system [10]. As shown in Figure 3, our design provides a way for users to gradually learn to create scripts. As in the work of MacLean et al [7], we wanted to provide a natural way for users to approach "tailorability" of piles as a part of the system.

Support for browsing. We wanted to support some of the browsing techniques users applied in their physical offices.

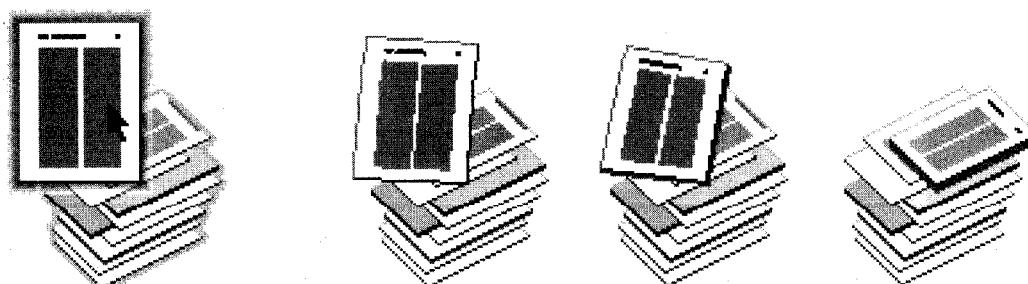


Figure 2. Adding a document to a pile. If a document is positioned over an existing pile, the pile highlights to show that it can accept the new document. When the mouse button is released the document 'drops' onto the pile.

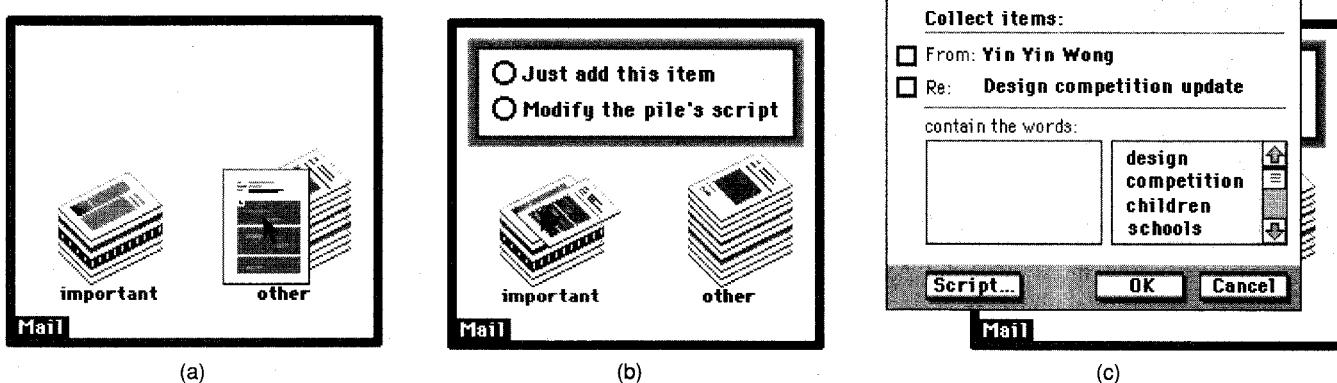


Figure 3. Scripting a pile. (a) depicts a mail area containing two scripted piles; one for important items, one for everything else. Over time, the criteria for 'important' may change. As shown, an item in the 'other' pile has been removed because that user desires that it, and items like it, now appear in the important pile. When this item is dropped onto the important pile, as shown in (b), the system queries the user to find out whether this action is a singular event or whether the pile's script should be modified. If the user chooses to modify the script, the system suggests criteria which could be used, as shown in (c). Alternatively, users can gain direct access to the scripting language and write their own criteria via the "Script..." button. Once the script is updated, items satisfying the new criteria visibly move to the 'important' pile.

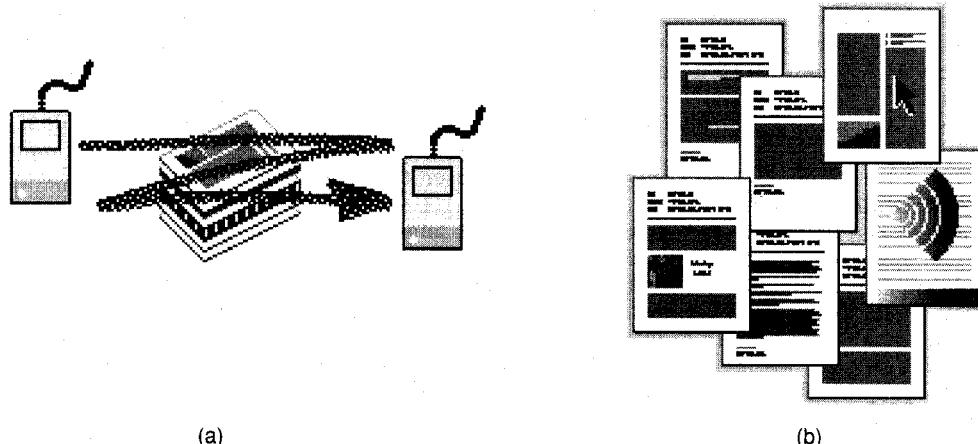


Figure 4. Browsing by spreading out a pile. Gesturing sideways with the mouse pointer, or with a finger in the case of a touch screen, causes the pile contents to spread out. Individual items can now be directly manipulated.

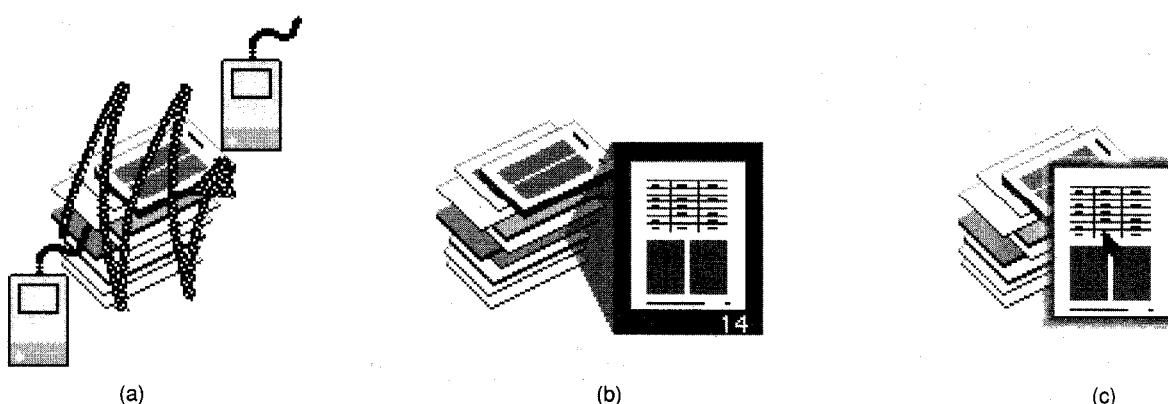


Figure 5. Browsing while maintaining the pile's structure. Gesturing vertically with the mouse pointer as shown in (a), or with a finger in the case of a touch screen, generates a 'viewing cone' (b) that contains a miniature version of the first page of the item under the pointer. This viewing cone will follow the vertical position of the pointer; the miniature changes as the pointer moves over each item. The user can move through the pages of an item in the viewing cone by using the left and right cursor keys on the keyboard. When an item is visible in the viewing cone, it can be selected by clicking the mouse button. The item then appears next to the pile on the desktop, as shown in (c).

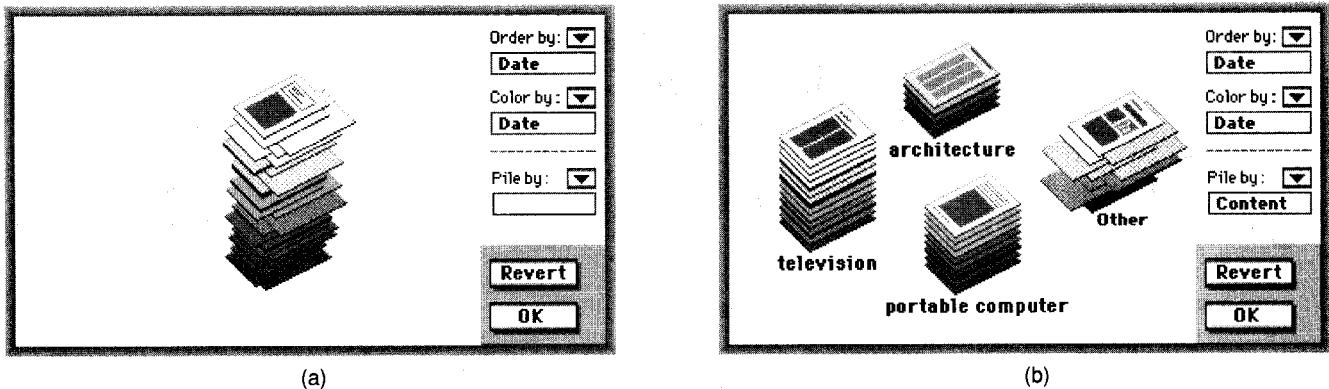


Figure 6. Visualizing a pile's contents. The pile shown is within a 'visualizing environment' that allows the user to select and visualize several criteria. Criteria can be mapped to the pile's order, the color of the items within the pile, or the way a pile is broken into sub-piles. In (a) the pile is both ordered and colored by date. In (b) the user chose to 'pile by' content. Therefore, the system separated the original pile into four content-based piles. Three are labeled with specific terms suggested by the system (e.g. "architecture"), appear neat and are now scripted to maintain similar content. The remaining disheveled pile, "other," contains items which did not fit into any of the other three piles.

By virtue of using miniatures of the actual documents, we offered edge browsing capabilities. In addition, we explored gestural inputs as a way to invoke other browsing methods. For example, a horizontal gesture would spread out a pile so that miniatures of each item's first page were visible (Figure 4). A vertical up-and-down movement over a pile would allow users to browse a pile using a 'viewing cone' (Figure 5). When an item was visible in the viewing cone, the user could move through miniature representations of its pages by using the cursor keys on the keyboard. In the design sketches, a mouse was used to create the gestures, but we thought that these interaction techniques would be particularly well-suited to a touch screen display.

Managing piles. In physical offices, the user is confronted with many pile management tasks, such as re-piling and sub-piling when a particular pile becomes unwieldy or specific information must be retrieved. We wanted the system to act as a collaborator in dealing with these issues, and therefore designed a 'visualizing environment' which would help users understand the contents of piles. As shown in Figure 6, a user might choose to emphasize certain criteria in a pile by using order, color or sub-piles. A user can elect to view combinations of criteria simultaneously. For example, the user could choose that items in a pile be ordered by date. The user might also request that the items in a pile be color coded according to their data type. Additionally, the user might have the system suggest subject-based sub-piles, by choosing the "pile by content" option. The sub-piles deemed useful could be moved out of the visualization area for use on the desktop.

TESTING USER'S EXPECTATIONS OF PILES

The design sketches raised interest amongst our colleagues and were the focal point for discussions. However, since the sketches were 'hard-wired,' we did not know if the interaction techniques were usable and of value to end-users. Consequently, we undertook a user test of the interaction techniques.

We constructed a suite of prototypes in Director that supported the interactivity we wished to test. We hoped to gauge people's expectations about the inclusion of piles on the desktop. Our method was informal, resembling the type of testing described in [4,12], in order to provide us with quick results that could be used in design iteration.

Method

Five men and five women in nontechnical positions at Apple Computer were individually tested in approximately one hour sessions. The subjects were asked to think aloud [5] while working through 5 tasks, and the sessions were videotaped. The first two tasks compared two different pile models. In the third task, users explored methods of initiating pile browsing. Task four allowed users to indicate preferences between three different viewing cone representations. In the final task, users were asked to locate items within a pile. At the conclusion of the test, users were informally asked for their comments and general impressions concerning piles.

Piling models. Two different models for a pile were compared: a "document-centered" model and a "pile-centered" model. Possible ordering effects were avoided by varying the presentation of these two models across users.

In the "document-centered" task, the pile was represented as a collection of individual items. The user was presented with a series of colored rectangles within a white screen area. These rectangles were intended to represent files on a desktop. The rectangles could be selected and moved with the mouse. When one rectangle was placed over another rectangle, both would fall back to create a disheveled pile. Additional documents could be added to an existing pile by moving them over the pile and releasing the mouse. Documents could be removed by individually selecting them via any visible region and dragging them away from the pile. The pile itself could not be moved as a unit.

In the "pile-centered" task, piles were created in the same way, except that the pile acted more like a Macintosh folder – a single entity containing a collection of documents. When one document 'rectangle' was moved over another rectangle, the latter would highlight to indicate a pile would be formed if the mouse button was released. Subsequent documents moved to the pile would automatically drop onto the top of the pile. The pile itself, as opposed to independent documents, could then be dragged around the desktop by mouse-clicking on any part of it.

Initiating browsing. In this task, participants tried out different ways of initiating pile browsing. They compared double-clicking and the horizontal gesture (shown in Figure

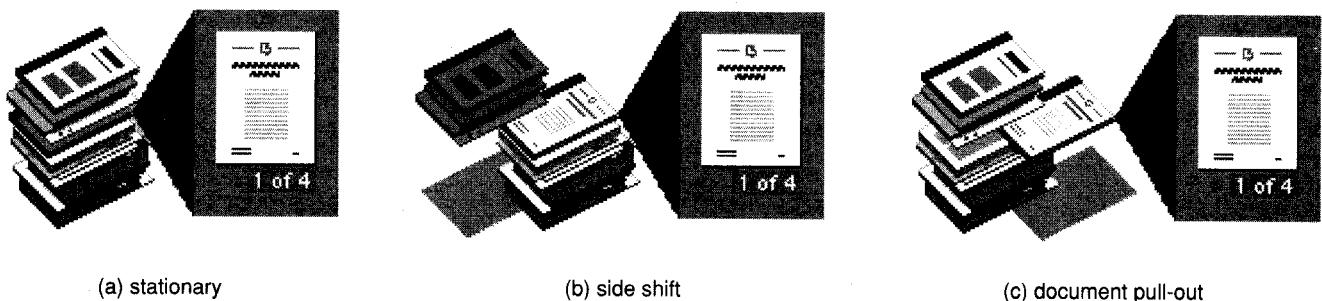


Figure 7. Test of different viewing cone representations. Users were presented with three different ways a pile could react during viewing cone browsing. In each case, the viewing cone contains a miniature of the first page of the document being examined. In style (a), the pile remains stationary. In (b), each item above the item being currently viewed is moved to the left side. In (c), the item currently being viewed temporarily moves out of the pile and to the right. Users preferred methods (b) and (c).

4) as ways to spread out a pile's contents. They also compared double-clicking and vertical gesturing (Figure 5) for initiating browsing with the view cone.

Viewing cone representations. In this task, participants were presented with three different visual representations of the viewing cone (see Figure 7). All were initiated by the same interaction – a single mouse click on the pile – but the order of presentation was varied for each user. When users settled on a preferable representation, they were shown how to use the keyboard to examine a miniature document's pages while it was within the viewing cone. Users were shown which key would move the document forward one page, and which would move it backwards.

Finding items within a pile. In this task, participants were asked to use the viewing cone and paging ability to locate specific pages within documents in the pile. First they were asked to locate a picture of a hand on a mouse, for which they were shown a real report illustration as a stimulus. Then they were asked to locate three separate items within the pile: a colored bar graph, a map of North America which was contained in an Atlas document, and a document containing bullet point text. Users were not timed – this part of the test was aimed at determining if pile browsing was, in general, qualitatively suitable to users for locating information.

Results

Piling models. Although each user had a clear preference for one of our methods of pile creation ("pile-centered" or "document-centered"), neither method was judged to be clearly superior. In the "document-centered" model, users liked the ability to grab an individual document within a pile. A problem with this model was that users were not sure how to move a pile as a unit, since selecting any part of the pile led to moving an individual item rather than the pile as a whole. In the "pile-centered" model, users liked the way the system automatically aligned the items in the pile, the ability to move a pile as a unit, and the highlighting that indicated a pile was ready to accept an item. A problem with this model was the difficulty of selecting an individual item within the pile.

Most users also expected that any desktop item could be added to a pile. This led to discussion of what would happen if a document was placed on top of an isolated folder; users were unsure whether the item would go into the folder or if a new pile would be created. Most users thought

that, based on their previous Macintosh experience, the item would go into the folder. This raises questions about how the pile metaphor fits into the current Macintosh desktop metaphor.

Most users asked for features generally available in desktop systems, but which were not present in the testing prototypes. For example, they wanted to be able to add a selected group of items to a pile, name piles, apply ordering schemes based on date, size, name, and kind, and control where a document was placed within a pile.

Since users liked and disliked certain features of each model, new design work will be undertaken to create models that both embody users' preferences and are internally consistent. Further testing of these new models will be conducted.

Initiating browsing. Subjects tried using both gestures and mouse double-clicks to spread out a pile and also to obtain the viewing cone. In both cases, 9 out of the 10 participants preferred the double-click method. They found it faster and more Macintosh-like, which was not unexpected given that the subjects were all accustomed to the Macintosh. However, users also felt that the gestures were non-intuitive and somewhat ambiguous, and that the piles might be spread out accidentally while moving the cursor around on the screen. The gestures were originally intended for use on a touch screen and most participants said that using a finger on the screen for the gesture might be more intuitive than using a mouse. This needs to be confirmed in a test with a touch screen. Note that we did not ask users *which* of the browsing methods they would want initiated by the double-click action; we only ascertained that they preferred double-clicking over gesturing.

In general, users thought they would make use of the 'spread out' view. Since all items were visible at once, it supported recognition and comparison. A few users expressed interest in viewing a grid layout rather than the overlapping one used in our testing prototype. While in this view, most users expected to both be able to act on individual items in standard ways, and move the documents as a group. In addition to the miniature representation of each item, many users requested that other information such as name, date, and kind be made available, and that the system provide representations which would specifically help the user differentiate similar items.

Viewing cone representations. Of the three viewing cone designs, the stationary pile version (Figure 7a) was rejected by all 10 users. All thought it was difficult to gauge where they were within the pile. Four users preferred the 'side shift' style (Figure 7b), 5 preferred the 'document pull-out' style (Figure 7c), and 1 user was undecided between these latter two designs. Both of the preferred designs clearly provided a view of an item's location in the pile, in addition to a representation shown within the viewing cone. Although it was not implemented in the prototype, once users had an item visible in the cone they often tried to grab it by either releasing and then quickly clicking the mouse, or by attempting to drag it from the pile.

Most users liked the viewing cone as a browsing method. It made it possible for them to identify items by their miniature representation without disturbing the pile's state. Users also liked the ability to view any page of an individual item, although not all were pleased with using the cursor keys on the keyboard to cause this action. Page numbering information (e.g. '1 of 10') was found valuable while paging through the document, because it indicated the relative size of each item, as well as position within an item. One user desired random access to any page via selection of its number from the keyboard. A few users expressed interest in being able to target the viewing cone at any item on the desktop – a single document, a folder – and not just items in piles.

During the tests we noted a potential problem with the viewing cone implementation – users might need to depress the mouse button for a long time while browsing, which could lead to repetitive stress injury. A possible solution is to invoke the cone whenever the user clicks on a pile, thereby alleviating the need for the mouse button to be continuously depressed. This would also allow the user to click the mouse button again to select an item for removal from the pile while the viewing cone was active.

Finding items within the pile. We showed the subjects a physical version of a report and identified a specific illustration which we wanted them to find within a pile on the computer desktop. Users were asked to use the viewing cone and cursor keys to examine items in the pile. All of the users were able to find the illustration within a reasonable amount of time. As mentioned earlier, we were not concerned with timing information, but rather with the feasibility of the viewing cone for this task.

Since the picture was within a report which was bound in a green-edged cover, several users recognized the document within the pile by its clearly visible green spine. A common strategy was to subsequently move through the document's miniature pages, looking for a small colored image in the top right corner of a page. Only one user took advantage of the page numbers on the miniature representations to identify the page. A few users did not expect the document on the computer to have the green spine that was present on the physical report because they perceived it to be an addition which the system could not have known about. These users' strategy was to start at the top of the pile and systematically look through every item, page by page, to find the picture.

We also asked the subjects to find a colored bar graph, a map of the North American continent within an atlas, and

some bullet point text. Only some of the users found the items, and with some difficulty. Many users felt they would do better with their own information and their difficulty was due to a lack of familiarity with the material in the pile. Several users discussed ways they would like the system to help them in such a situation. They commonly wanted the ability to search for specific data types, names, keywords, and other identifiers.

From this feedback, we inferred that it might be useful to give the user control over the information presented in the viewing cone. For instance, when searching for a graph, the user could select data type 'graph' as the search criteria, thereby causing the viewing cone to display only pages containing graph data types. This could be a powerful way to search, since it would enable the user to tailor the view according to current needs.

General discussion. At the end of the test, we asked users how they would use piles, and how the system might assist them. In general, users were receptive to the idea of having the system help them with their routine tasks, such as sorting incoming mail. Most users reported having between two and five mail systems, fax, and voice mail. They liked the idea of receiving all incoming information in a pile which could be accessed with the viewing cone. Within such a pile, they would want the system to prioritize items using characteristics such as sender, topic, content keywords, date, and urgency. We anticipate these priorities could be learned by the system over a period of time by watching the user interact with incoming information.

FUTURE WORK: FROM DESIGN SKETCH TO IMPLEMENTATION

There are many areas in which this work can proceed. A few of our current directions are described below.

Improving Designs and Working with Familiar Data

We plan to further explore the appropriate model for a pile – and how to possibly combine users' expectations about its document-centeredness vs. pile-centeredness – by iterating on our previous Director prototypes.

In addition, we intend to build prototypes that incorporate items of relevance to the individual being tested. The informal tests described above involved fabricated data that was unfamiliar to our subjects. In order to continue refining our designs, we need to construct prototypes that will allow subjects to interactively use piles for their own information over an extended period of time. An extension to the current Finder interface that would allow users to create and work with piles alongside folders would provide an excellent opportunity to further these designs. However, it may prove more feasible to undertake the next round of iteration by addressing a limited domain, such as a mail system.

Browsing by Other Criteria

The current design allows users to browse the contents of piles by viewing miniature representations of each item. While users found this feature useful in the tests, they also expressed interest in accessing other representations. We are currently exploring the types of "browse by..." criteria the system might offer. For example, users might want to selectively emphasize some data type during browsing, as in the case of 'show me all the documents containing movies within this pile.' When confronted with unfamiliar

data, users might want to browse by textual abstract, since a miniature visual representation might not provide insight into an unknown item's content.

Technology to Support Pile Interactions

The interface designs described in this paper were primarily inspired by observations with users, and not necessarily by existing technology. At the time of design, we were unsure if information retrieval techniques could adequately support some of the interactions, such as pile scripting-by-example or sub-pile creation. Consequently, we initiated a collaborative research effort with the Information Retrieval Team within Apple Computer's Advanced Technology Group.

Some preliminary work in implementing low level support for pile functionality has been undertaken. Current research is focussing on a clustering technique that would automatically create sub-piles. For example, a user could supply the system with a pile of documents, and based on the content of the documents within that pile, the system would suggest and describe suitable sub-piles.

As this work progresses, we plan to adapt our designs to reflect the technology that can be realized.

ACKNOWLEDGEMENTS

We would like to thank Dan Rose and Tim Oren for exploring information retrieval systems that will support sub-piling and other pile management operations; Stephanie Houde for creating the Director prototypes used in testing; Penny Bauersfeld and Leo Degen for their participation in early design sessions; and Tom Erickson for input on the user study design and feedback on the various drafts of this paper.

BIBLIOGRAPHY

- [1] Apple Computer, Inc. *Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1987.
- [2] Apple Computer, Inc. *Inside Macintosh, Volume VI*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1991.
- [3] Cole, I. Human aspects of office filing: Implications for the electronic office. *Proceedings of the Human Factors Society, 26th Annual Meeting*, Seattle, Washington. 1982.
- [4] Gomoll, K. Some Techniques for Observing Users. *The Art of Human-Computer Interface Design* (ed. Brenda Laurel) Addison-Wesley Publishing Company, Inc., Reading, MA. 1990. pp. 85-90.
- [5] Ericsson, K.A. and Simon, H. A. *Protocol analysis*. Cambridge, Massachusetts: MIT Press. 1984.
- [6] Lansdale, M. The psychology of personal information management. *Applied Ergonomics*, 55, (1988), pp. 55-66.
- [7] MacLean, A., Carter, K., Lovstrand, L. and Moran, T. User-Tailorable Systems: Pressing the Issues with Buttons. In *Proceedings of CHI 1990 (Seattle, Washington, April 1-5, 1990)* ACM, New York, 1990. pp. 175-182.
- [8] MacroMind, Inc. *Director™ 2.0*. April 1990.
- [9] Malone, T. W. How do People Organize Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Information Systems*, 1,1, (January 1983), pp. 99-112.
- [10] Malone, T. W., Grant, K.R., Turbak, F.A., Brobst, S.A. and Cohen, M.D. Intelligent Information-Sharing Systems. *Communications of the ACM*, 30, 5, (May 1987), pp. 390-402.
- [11] Miller, P., Tetelbaum, S. and Webb, K. BUSINESS – an end-user oriented application development language. *SIGMOD Record*, 12, 1, (October 1981), pp. 38-69.
- [12] Nielsen, J. Usability Engineering at a Discount. In G. Salvendy and M.J. Smith (Eds.), *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*. Amsterdam: Elsevier. 1989.
- [13] van Rijsbergen, C.J. *Information Retrieval. (2nd. Ed.)* Butterworths, London, England. 1983.

Chapter 52

Prototyping Tools and Techniques

Michel Beaudouin-Lafon, Université Paris-Sud, mbl@lri.fr

Wendy E. Mackay, INRIA, wendy.mackay@inria.fr

1. Introduction

“A good design is better than you think” (Rex Heftman, cited by Raskin, 2000).

Design is about making choices. In many fields that require creativity and engineering skill, such as architecture or automobile design, prototypes both inform the design process and help designers select the best solution.

This chapter describes tools and techniques for using prototypes to design interactive systems. The goal is to illustrate how they can help designers generate and share new ideas, get feedback from users or customers, choose among design alternatives, and articulate reasons for their final choices.

We begin with our definition of a prototype and then discuss prototypes as design artifacts, introducing four dimensions for analyzing them. We then discuss the role of prototyping within the design process, in particular the concept of a design space, and how it is expanded and contracted by generating and selecting design ideas. The next three sections describe specific prototyping approaches: Rapid prototyping, both off-line and on-line, for early stages of design, iterative prototyping, which uses on-line development tools, and evolutionary prototyping, which must be based on a sound software architecture.

What is a prototype?

We define a prototype as a *concrete representation* of part or all of an interactive system. A prototype is a tangible artifact, not an abstract description that requires interpretation. Designers, as well as managers, developers, customers and end-users, can use these artifacts to envision and reflect upon the final system.

Note that prototypes may be defined differently in other fields. For example, an architectural prototype is a scaled-down model of the final building. This is not possible for interactive system prototypes: the designer may limit the amount of information the prototype can handle, but the actual interface must be presented at full scale. Thus, a prototype interface to a database may handle only a small pseudo database but must still present a full-size display and interaction techniques. Full-scale, one-of-a-kind models, such as a hand-made dress sample, are another type of prototype. These usually require an additional design phase in order to mass-produce the final design. Some interactive system prototypes begin as one-of-a-kind models which are then distributed widely (since the cost of duplicating software is so low). However, most successful software prototypes evolve into the final product and then continue to evolve as new versions of the software are released.

Hardware and software engineers often create prototypes to study the feasibility of a technical process. They conduct systematic, scientific evaluations with respect to pre-defined benchmarks and, by systematically varying parameters, fine-tune the system. Designers in creative fields, such as typography or graphic design, create prototypes to express ideas and reflect on them. This approach is intuitive, oriented more to discovery and generation of new ideas than to evaluation of existing ideas.

Human-Computer Interaction is a multi-disciplinary field which combines elements of science, engineering and design (Mackay and Fayard, 1997, Djikstra-Erikson et al., 2001). Prototyping is primarily a design activity, although we use software engineering to ensure that software prototypes evolve into technically-sound working systems and we use scientific methods to study the effectiveness of particular designs.

2. Prototypes as design artifacts

We can look at prototypes as both concrete artifacts in their own right or as important components of the design process. When viewed as artifacts, successful prototypes have several characteristics: They support *creativity*, helping the developer to capture and generate ideas, facilitate the exploration of a design space and uncover relevant information about users and their work practices. They encourage *communication*, helping designers, engineers, managers, software developers, customers and users to discuss options and interact with each other. They also permit *early evaluation* since they can be tested in various ways, including traditional usability studies and informal user feedback, throughout the design process.

We can analyze prototypes and prototyping techniques along four dimensions:

- *Representation* describes the form of the prototype, e.g., sets of paper sketches or computer simulations;
- *Precision* describes the level of detail at which the prototype is to be evaluated; e.g., informal and rough or highly polished;
- *Interactivity* describes the extent to which the user can actually interact with the prototype; e.g., watch-only or fully interactive; and
- *Evolution* describes the expected life-cycle of the prototype, e.g. throw-away or iterative.

2.1 Representation

Prototypes serve different purposes and thus take different forms. A series of quick sketches on paper can be considered a prototype; so can a detailed computer simulation. Both are useful; both help the designer in different ways. We distinguish between two basic forms of representation: off-line and on-line.

Off-line prototypes (also called *paper prototypes*) do not require a computer. They include paper sketches, illustrated story-boards, cardboard mock-ups and videos. The most salient characteristics of off-line prototypes (of interactive systems) is that they are created quickly, usually in the early stages of design, and they are usually thrown away when they have served their purpose.

On-line prototypes (also called *software prototypes*) run on a computer. They include computer animations, interactive video presentations, programs written with scripting languages, and applications developed with interface builders. The cost of producing on-line prototypes is usually higher, and may require skilled programmers to implement advanced interaction and/or visualization techniques or

to meet tight performance constraints. Software prototypes are usually more effective in the later stages of design, when the basic design strategy has been decided.

In our experience, programmers often argue in favor of software prototypes even at the earliest stages of design. Because they already are already familiar with a programming language, these programmers believe it will be faster and more useful to write code than to "waste time" creating paper prototypes. In twenty years of prototyping, in both research and industrial settings, we have yet to find a situation in which this is true.

First, off-line prototypes are very inexpensive and quick. This permits a very rapid iteration cycle and helps prevent the designer from becoming overly attached to the first possible solution. Off-line prototypes make it easier to *explore the design space* (see section 3.1), examining a variety of design alternatives and choosing the most effective solution. On-line prototypes introduce an intermediary between the idea and the implementation, slowing down the design cycle.

Second, off-line prototypes are less likely to constrain how the designer thinks. Every programming language or development environment imposes constraints on the interface, limiting creativity and restricting the number of ideas considered. If a particular tool makes it easy to create scroll-bars and pull-down menus and difficult to create a zoomable interface, the designer is likely to limit the interface accordingly. Considering a wider range of alternatives, even if the developer ends up using a standard set of interface widgets, usually results in a more creative design.

Finally and perhaps most importantly, off-line prototypes can be created by a wide range of people: not just programmers. Thus all types of designers, technical or otherwise, as well as users, managers and other interested parties, can all contribute on an equal basis. Unlike programming software, modifying a storyboard or cardboard mock-up requires no particular skill. Collaborating on paper prototypes not only increases participation in the design process, but also improves communication among team members and increases the likelihood that the final design solution will be well accepted.

Although we believe strongly in off-line prototypes, they are not a panacea. In some situations, they are insufficient to fully evaluate a particular design idea. For example, interfaces requiring rapid feedback to users or complex, dynamic visualizations usually require software prototypes. However, particularly when using video and wizard-of-oz techniques, off-line prototypes can be used to create very sophisticated representations of the system.

Prototyping is an iterative process and all prototypes provide information about some aspects while ignoring others. The designer must consider the purpose of the prototype (Houde and Hill, 1997) at each stage of the design process and choose the representation that is best suited to the current design question.

2.2 Precision

Prototypes are explicit representations that help designers, engineers and users reason about the system being built. By their nature, prototypes require details. A verbal description such as "the user opens the file" or "the system displays the results" provides no information about what the user actually does. Prototypes force designers to *show* the interaction: just how does the user open the file and what are the specific results that appear on the screen?

Precision refers to the relevance of details with respect to the purpose of the prototype¹. For example, when sketching a dialog box, the designer specifies its size, the positions of each field and the titles of each label. However not all these details are relevant to the goal of the prototype: it may be necessary to show where the labels are, but too early to choose the text. The designer can convey this by writing nonsense words or drawing squiggles, which shows the need for labels without specifying their actual content.

Although it may seem contradictory, a detailed representation need not be precise. This is an important characteristic of prototypes: those parts of the prototype that are not precise are those open for future discussion or for exploration of the design space. Yet they need to be incarnated in some form so the prototype can be evaluated and iterated.

The level of precision usually increases as successive prototypes are developed and more and more details are set. The forms of the prototypes reflect their level of precision: sketches tend not to be precise, whereas computer simulations are usually very precise. Graphic designers often prefer using hand sketches for early prototypes because the drawing style can directly reflect what is precise and what is not: the wiggly shape of an object or a squiggle that represents a label are directly perceived as imprecise. This is more difficult to achieve with an on-line drawing tool or a user-interface builder.

The form of the prototype must be adapted to the desired level of precision. Precision defines the tension between what the prototype states (relevant details) and what the prototype leaves open (irrelevant details). What the prototype states is subject to evaluation; what the prototype leaves open is subject to more discussion and design space exploration.

2.3 Interactivity

An important characteristic of HCI systems is that they are *interactive*: users both respond to them and act upon them. Unfortunately, designing effective interaction is difficult: many interactive systems (including many web sites) have a good “look” but a poor “feel”. HCI designers can draw from a long tradition in visual design for the former, but have relatively little experience with how interactive software systems should be used: personal computers have only been commonplace for about a decade. Another problem is that the quality of interaction is tightly linked to the end users and a deep understanding of their work practices: a word processor designed for a professional typographer requires a different interaction design than one designed for secretaries, even though ostensibly they serve similar purposes. Designers must take the context of use into account when designing the details of the interaction.

A critical role for an interactive system prototype is to illustrate how the user will interact with the system. While this may seem more natural with on-line prototypes, in fact it is often easier to explore different interaction strategies with off-line prototypes. Note that interactivity and precision are orthogonal dimensions. One can create an imprecise prototype that is highly interactive, such as a series of paper screen images in which one person acts as the user and the other plays the system. Or, one may create a very precise but non-interactive

¹ Note that the terms *low-fidelity* and *high-fidelity* prototypes are often used in the literature. We prefer the term *precision* because it refers to the content of the prototype itself, not its relationship to the final, as-yet-undefined system.

prototype, such as a detailed animation that shows feedback from a specific action by a user.

Prototypes can support interaction in various ways. For off-line prototypes, one person (often with help from others) plays the role of the interactive system, presenting information and responding to the actions of another person playing the role of the user. For on-line prototypes, parts of the software are implemented, while others are "played" by a person. (This approach, called the *Wizard of Oz* after the character in the 1939 movie of the same name, is explained in section 4.1.) The key is that the prototype *feels* interactive to the user.

Prototypes can support different levels of interaction. *Fixed prototypes*, such as video clips or pre-computed animations, are non-interactive: the user cannot interact, or pretend to interact, with it. Fixed prototypes are often used to illustrate or test scenarios (see chapter 53). *Fixed-path prototypes* support limited interaction. The extreme case is a fixed prototype in which each step is triggered by a pre-specified user action. For example, the person controlling the prototype might present the user with a screen containing a menu. When the user points to the desired item, she presents the corresponding screen showing a dialog box. When the user points to the word "OK", she presents the screen that shows the effect of the command. Even though the position of the click is irrelevant (it is used as a trigger), the person in the role of the user can get a feel for the interaction. Of course, this type of prototype can be much more sophisticated, with multiple options at each step. Fixed-path prototypes are very effective with scenarios and can also be used for horizontal and task-based prototypes (see section 3.1).

Open prototypes support large sets of interactions. Such prototypes work like the real system, with some limitations. They usually only cover part of the system (see vertical prototypes, section 3.1), and often have limited error-handling or reduced performance relative to that of the final system.

Prototypes may thus illustrate or test different levels of interactivity. Fixed prototypes simply illustrate what the interaction might look like. Fixed-path prototypes provide designers and users with the experience of what the interaction might be like, but only in pre-specified situations. Open prototypes allow designers to test a wide range of examples of how users will interact with the system.

2.4 Evolution

Prototypes have different life spans: *rapid* prototypes are created for a specific purpose and then thrown away, *iterative* prototypes evolve, either to work out some details (increasing their precision) or to explore various alternatives, and *evolutionary* prototypes are designed to become part of the final system.

Rapid prototypes are especially important in the early stages of design. They must be inexpensive and easy to produce, since the goal is to quickly explore a wide variety of possible types of interaction and then throw them away. Note that rapid prototypes may be off-line or on-line. Creating precise software prototypes, even if they must be re-implemented in the final version of the system, is important for detecting and fixing interaction problems. Section 4 presents specific prototyping techniques, both off-line and on-line.

Iterative prototypes are developed as a reflection of a design in progress, with the explicit goal of evolving through several design iterations. Designing prototypes that support evolution is sometimes difficult. There is a tension between evolving

toward the final solution and exploring an unexpected design direction, which may be adopted or thrown away completely. Each iteration should inform some aspect of the design. Some iterations explore different variations of the same theme. Others may systematically increase precision, working out the finer details of the interaction. Section 5 describes tools and techniques for creating iterative prototypes.

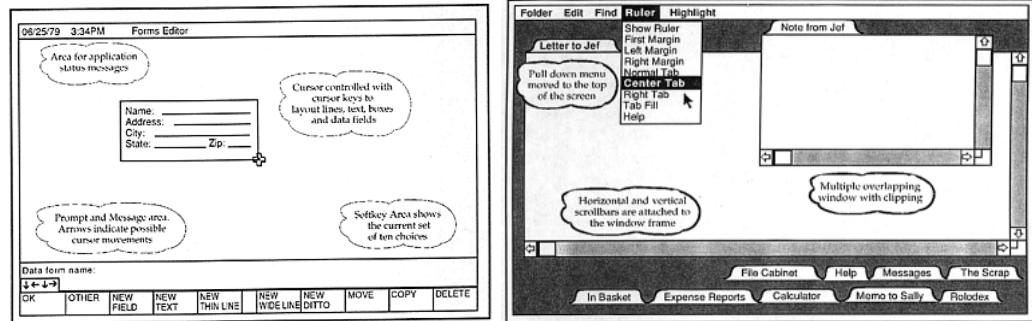


Figure 1: Evolutionary prototypes of the Apple Lisa:
July 1979 (left), October 1980 (right) (Perkins et al., 1997) [[need permission]]

Evolutionary prototypes are a special case of iterative prototypes in which the prototype evolves into part or all of the final system (Fig.1). Obviously this only applies to software prototypes. Extreme Programming (Beck, 2000), advocates this approach, tightly coupling design and implementation and building the system through constant evolution of its components. Evolutionary prototypes require more planning and practice than the approaches above because the prototypes are both representations of the final system and the final system itself, making it more difficult to explore alternative designs. We advocate a combined approach, beginning with rapid prototypes and then using iterative or evolutionary prototypes according to the needs of the project. Section 6 describes how to create evolutionary prototypes, by building upon software architectures specifically designed to support interactive systems.

3. Prototypes and the design process

In the previous section, we looked at prototypes as artifacts, i.e. the results of a design process. Prototypes can also be seen as artifacts *for* design, i.e. as an integral part of the design process. Prototyping helps designers think: prototypes are the tools they use to solve design problems. In this section we focus on prototyping as a process and its relationship to the overall design process.

User-centered design

The field of Human-Computer Interaction is both user-centered (Norman & Draper, 1986) and iterative. User-centered design places the user at the center of the design process, from the initial analysis of user requirements (see chapters 48-50 in this volume) to testing and evaluation (see chapters 56-59 in this volume). Prototypes support this goal by allowing users see and experience the final system long before it is built. Designers can identify functional requirements, usability problems and performance issues early and improve the design accordingly.

Iterative design involves multiple design-implement-test loops², enabling the designer to generate different ideas and successively improve upon them. Prototypes support this goal by allowing designers to evaluate concrete representations of design ideas and select the best.

Prototypes reveal the strengths as well as the weaknesses of a design. Unlike pure ideas, abstract models or other representations, they can be *contextualized* to help understand how the real system would be used in a real setting. Because prototypes are concrete and detailed, designers can explore different real-world scenarios and users can evaluate them with respect to their current needs. Prototypes can be compared directly with other, existing systems, and designers can learn about the context of use and the work practices of the end users. Prototypes can help designers (re)analyze the user's needs during the design process, not abstractly as with traditional requirements analysis, but in the context of the system being built.

Participatory design

Participatory (also called Cooperative) Design is a form of user-centered design that actively involves the user in all phases the design process (see Greenbaum and Kyng, 1991, and chapter 54 in this volume.) Users are not simply consulted at the beginning and called in to evaluate the system at the end; they are treated as partners throughout. This early and active involvement of users helps designers avoid unpromising design paths and develop a deeper understanding of the actual design problem. Obtaining user feedback at each phase of the process also changes the nature of the final evaluation, which is used to fine-tune the interface rather than discover major usability problems.

A common misconception about participatory design is that designers are expected to abdicate their responsibilities as designers, leaving the design to the end user. In fact, the goal is for designers and users to work together, each contributing their strengths to clarify the design problem as well as explore design solutions. Designers must understand what users can and cannot contribute. Usually, users are best at understanding the context in which the system will be used and subtle aspects of the problems that must be solved. Innovative ideas can come from both users and designers, but the designer is responsible for considering a wide range of options that might not be known to the user and balancing the trade-offs among them.

Because prototypes are shared, concrete artifacts, they serve as an effective medium for communication within the design team. We have found that collaborating on prototype design is an effective way to involve users in participatory design. Prototypes help users articulate their needs and reflect on the efficacy of design solutions proposed by designers.

3.1 Exploring the design space

Design is not a natural science: the goal is not to describe and understand existing phenomena but to create something new. Designers do, of course, benefit from scientific research findings and they may use scientific methods to evaluate interactive systems. But designers also require specific techniques for generating new ideas and balancing complex sets of trade-offs, to help them develop and refine design ideas.

² Software engineers refer to this as the Spiral model (Boehm, 1988).

Designers from fields such as architecture and graphic design have developed the concept of a *design space*, which constrains design possibilities along some dimensions, while leaving others open for creative exploration. Ideas for the design space come from many sources: existing systems, other designs, other designers, external inspiration and accidents that prompt new ideas. Designers are responsible for creating a design space specific to a particular design problem. They explore this design space, expanding and contracting it as they add and eliminate ideas. The process is iterative: more cyclic, than reductionist. That is, the designer does not begin with a rough idea and successively add more precise details until the final solution is reached. Instead, she begins with a design problem, which imposes set of constraints, and generates a set of ideas to form the initial design space. She then explores this design space, preferably with the user, and selects a particular design direction to pursue. This closes off part of the design space, but opens up new dimensions that can be explored. The designer generates additional ideas along these dimensions, explores the expanded design space, and then makes new design choices. Design principles (e.g., Beaudouin-Lafon and Mackay, 2000) help this process by guiding it both in the exploration and choice phases. The process continues, in a cyclic expansion and contraction of the design space, until a satisfying solution is reached.

All designers work with constraints: not just limited budgets and programming resources, but also design constraints. These are not necessarily bad: one cannot be creative along all dimensions at once. However, some constraints are unnecessary, derived from poor framing of the original design problem. If we consider a design space as a set of ideas and a set of constraints, the designer has two options. She can modify ideas within the specified constraints or modify the constraints to enable new sets of ideas. Unlike traditional engineering, which treats the design problem as a given, designers are encouraged to challenge, and if necessary, change the initial design problem. If she reaches an impasse, the designer can either generate new ideas or redefine the problem (and thus change the constraints). Some of the most effective design solutions derive from a more careful understanding and reframing of the design brief.

Note that all members of the design team, including users, may contribute ideas to the design space and help select design directions from within it. However, it is essential that these two activities are kept separate. Expanding the design space requires creativity and openness to new ideas. During this phase, everyone should avoid criticizing ideas and concentrate on generating as many as possible. Clever ideas, half-finished ideas, silly ideas, impractical ideas: all contribute to the richness of the design space and improve the quality of the final solution. In contrast, contracting the design space requires critical evaluation of ideas. During this phase, everyone should consider the constraints and weigh the trade-offs. Each major design decision must eliminate part of the design space: rejecting ideas is necessary in order to experiment and refine others and make progress in the design process. Choosing a particular design direction should spark new sets of ideas, and those new ideas are likely to pose new design problems. In summary, exploring a design space is the process of moving back and forth between creativity and choice.

Prototypes aid designers in both aspects of working with a design space: generating concrete representations of new ideas and clarifying specific design directions. The next two sections describe techniques that have proven most useful in our own prototyping work, both for research and product development.

Expanding the design space: Generating ideas

The most well-known idea generation technique is *brainstorming*, introduced by Osborn (1957). His goal was to create synergy within the members of a group:

ideas suggested by one participant would spark ideas in other participants. Subsequent studies (Collaros and Anderson, 1969, Diehl and Stroebe, 1987) challenged the effectiveness of group brainstorming, finding that aggregates of individuals could produce the same number of ideas as groups. They found certain effects, such as production blocking, free-riding and evaluation apprehension, were sufficient to outweigh the benefits of synergy in brainstorming groups. Since then, many researchers have explored different strategies for addressing these limitations. For our purposes, the quantity of ideas is not the only important measure: the relationships among members of the group are also important. As de Vreede et al. (2000) point out, one should also consider elaboration of ideas, as group members react to each other's ideas.

We have found that brainstorming, including a variety of variants, is an important group-building exercise and participatory design. Designers may, of course, brainstorm ideas by themselves. But brainstorming in a group is more enjoyable and, if it is a recurring part of the design process, plays an important role in helping group members share and develop ideas together.

The simplest form of brainstorming involves a small group of people. The goal is to generate as many ideas as possible on a pre-specified topic: quantity not quality, is important. Brainstorming sessions have two phases: the first for generating ideas and the second for reflecting upon them. The initial phase should last no more than an hour. One person should moderate the session, keeping time, ensuring that everyone participates and preventing people from critiquing each other's ideas. Discussion should be limited to clarifying the meaning of a particular idea. A second person records every idea, usually on a flipchart or transparency on an overhead projector. After a short break, participants are asked to reread all the ideas and each person marks their three favorite ideas.

One variation is designed to ensure that everyone contributes, not just those who are verbally dominant. Participants write their ideas on individual cards or post-it notes for a pre-specified period of time. The moderator then reads each idea aloud. Authors are encouraged to elaborate (but not justify) their ideas, which are then posted on a whiteboard or flipchart. Group members may continue to generate new ideas, inspired by the others they hear.

We use a variant of brainstorming that involves prototypes called video brainstorming (Mackay, 2000): participants not only write or draw their ideas, they act them out in front of a video camera (Fig. 2). The goal is the same as other brainstorming exercises, i.e. to create as many new ideas as possible, without critiquing them. The use of video, combined with paper or cardboard mock-ups, encourages participants to actively experience the details of the interaction and to understand each idea from the perspective of the user.

Each video brainstorming idea takes 2-5 minutes to generate and capture, allowing participants to simulate a wide variety of ideas very quickly. The resulting video clips provide illustrations of each idea that are easier to understand (and remember) than hand-written notes. (We find that raw notes from brainstorming sessions are not very useful after a few weeks because the participants no longer remember the context in which the ideas were created.)

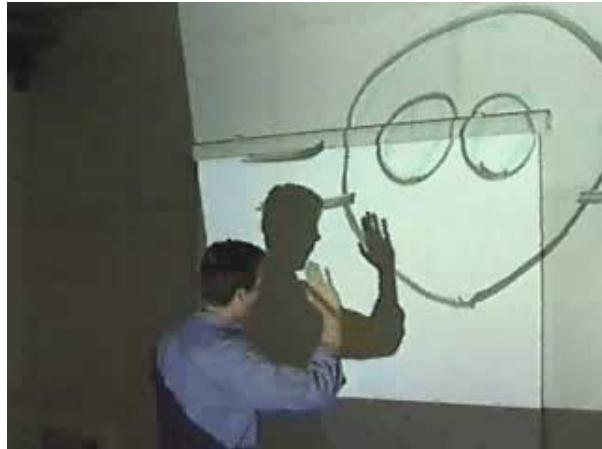


Figure 2: Video Brainstorming: One person moves the transparency, projected onto the wall, in response to the actions of the user, who explores how he might interact with an on-line animated character. Each interaction idea is recorded and videotaped.

Video brainstorming requires thinking more deeply about each idea. It is easier to stay abstract when describing an interaction in words or even with a sketch, but acting out the interaction in front of the camera forces the author of the idea (and the other participants) to seriously consider how a user would interact with the idea. It also encourages designers and users to think about new ideas in the context in which they will be used. Video clips from a video brainstorming session, even though rough, are much easier for the design team, including developers, to interpret than ideas from a standard brainstorming session.

We generally run a standard brainstorming session, either oral or with cards, prior to a video brainstorming session, to maximize the number of ideas to be explored. Participants then take their favorite ideas from the previous session and develop them further as video brainstorms. Each person is asked to "direct" at least two ideas, incorporating the hands or voices of other members of the group. We find that, unlike standard brainstorming, video brainstorming encourages even the quietest team members to participate.

Contracting the design space: Selecting alternatives

After expanding the design space by creating new ideas, designers must stop and reflect on the choices available to them. After exploring the design space, designers must evaluate their options and make concrete design decisions: choosing some ideas, specifically rejecting others, and leaving other aspects of the design open to further idea generation activities. Rejecting good, potentially effective ideas is difficult, but necessary to make progress.

Prototypes often make it easier to evaluate design ideas from the user's perspective. They provide concrete representations that can be compared. Many of the evaluation techniques described elsewhere in this handbook can be applied to prototypes, to help focus the design space. The simplest situation is when the designer must choose among several discrete, independent options. Running a simple experiment, using techniques borrowed from Psychology (see chapter 56) allows the designer to compare how users respond to each of the alternatives. The designer builds a prototype, with either fully-implemented or simulated versions of each option. The next step is to construct tasks or activities that are typical of how the system would be used, and ask people from the user population to try each of the options under controlled conditions. It is important to keep everything the same, except for the options being tested.

Designers should base their evaluations on both quantitative measures, such as speed or error rate, and qualitative measures, such as the user's subjective impressions of each option. Ideally, of course, one design alternative will be clearly faster, prone to fewer errors and preferred by the majority of users. More often, the results are ambiguous, and the designer must take other factors into account when making the design choice. (Interestingly, running small experiments often highlights other design problems and may help the designer reformulate the design problem or change the design space.)

The more difficult (and common) situation, is when the designer faces a complex, interacting set of design alternatives, in which each design decision affects a number of others. Designers can use heuristic evaluation techniques, which rely on our understanding of human cognition, memory and sensory-perception (see chapters 1-6). They can also evaluate their designs with respect to ergonomic criteria (see chapter 51) or design principles (Beaudouin-Lafon and Mackay, 2000). See chapters 56-60 for a more thorough discussion of testing and evaluation methods.

Another strategy is to create one or more scenarios (see chapter 53) that illustrate how the combined set of features will be used in a realistic setting. The scenario must identify who is involved, where the activities take place, and what the user does over a specified period of time. Good scenarios involve more than a string of independent tasks; they should incorporate real-world activities, including common or repeated tasks, successful activities and break-downs and errors, with both typical and unusual events. The designer then creates a prototype that simulates or implements the aspects of the system necessary to illustrate each set of design alternatives. Such prototypes can be tested by asking users to "walk through" the same scenario several times, once for each design alternative. As with experiments and usability studies, designers can record both quantitative and qualitative data, depending on the level of the prototypes being tested.

The previous section described an idea-generation technique called video brainstorming, which allows designers to generate a variety of ideas about how to interact with the future system. We call the corresponding technique for focusing in on a design *video prototyping*. Video prototyping can incorporate any of the rapid-prototyping techniques (off-line or on-line) described in section 4.1. They are quick to build, force designers to consider the details of how users will react to the design in the context in which it will be used, and provide an inexpensive method of comparing complex sets of design decisions. See section 4.1 for more information on how to develop scenarios, storyboard and then videotape them.

To an outsider, video brainstorming and video prototyping techniques look very similar: both involve small design groups working together, creating rapid prototypes and interacting with them in front of a video camera. Both result in video illustrations that make abstract ideas concrete and help team members communicate with each other. The critical difference is that video brainstorming expands the design space, by creating a number of unconnected collections of individual ideas, whereas video prototyping contracts the design space, by showing how a specific collection of design choices work together.

3.2 Prototyping strategies

Designers must decide what role prototypes should play with respect to the final system and in which order to create different aspects of the prototype. The next section presents four strategies: *horizontal*, *vertical*, *task-oriented* and *scenario-based*, which focus on different design concerns. These strategies can use any of the prototyping techniques covered in sections 4, 5 and 6.

Horizontal prototypes

The purpose of a horizontal prototype is to develop one entire layer of the design at the same time. This type of prototyping is most common with large software development teams, where designers with different skill sets address different layers of the software architecture. Horizontal prototypes of the user interface are useful to get an overall picture of the system from the user's perspective and address issues such as consistency (similar functions are accessible through similar user commands), coverage (all required functions are supported) and redundancy (the same function is/is not accessible through different user commands).

User interface horizontal prototypes can begin with rapid prototypes and progress through to working code. Software prototypes can be built with an interface builder (see section 5.1), without creating any of the underlying functionality making it possible to test how the user will interact with the user interface without worrying about how the rest of the architecture works. However some level of scaffolding or simulation of the rest of the application is often necessary, otherwise the prototype cannot be evaluated properly. As a consequence, software horizontal prototypes tend to be evolutionary, i.e. they are progressively transformed into the final system.

Vertical prototypes

The purpose of a vertical prototype is to ensure that the designer can implement the full, working system, from the user interface layer down to the underlying system layer. Vertical prototypes are often built to assess the feasibility of a feature described in a horizontal, task-oriented or scenario-based prototype. For example, when we developed the notion of magnetic guidelines in the CPN2000 system to facilitate the alignment of graphical objects (Beaudouin-Lafon and Mackay, 2000), we implemented a vertical prototype to test not only the interaction technique but also the layout algorithm and the performance. We knew that we could only include the particular interaction technique if we could implement a sufficiently fast response.

Vertical prototypes are generally high precision, software prototypes because their goal is to validate an idea at the system level. They are often thrown away because they are generally created early in the project, before the overall architecture has been decided, and they focus on only one design question. For example, a vertical prototype of a spelling checker for a text editor does not require text editing functions to be implemented and tested. However, the final version will need to be integrated into the rest of the system, which may involve considerable architectural or interface changes.

Task-oriented prototypes

Many user interface designers begin with a task analysis (see chapter 48), to identify the individual tasks that the user must accomplish with the system. Each task requires a corresponding set of functionality from the system. Task-based prototypes are organized as a series of tasks, which allows both designers and users to test each task independently, systematically working through the entire system.

Task-oriented prototypes include only the functions necessary to implement the specified set of tasks. They combine the breadth of horizontal prototypes, to cover the functions required by those tasks, with the depth of vertical prototypes, enabling detailed analysis of how the tasks can be supported. Depending on the

goal of the prototype, both off-line and on-line representations can be used for task-oriented prototypes.

Scenario-based prototypes

Scenario-based prototypes are similar to task-oriented ones, except that they do not stress individual, independent tasks, but rather follow a more realistic scenario of how the system would be used in a real-world setting. Scenarios are stories that describe a sequence of events and how the user reacts (see chapter 53). A good scenario includes both common and unusual situations, and should explore patterns of activity over time. Bødker (1995) has developed a checklist to ensure that no important issues have been left out.

We find it useful to begin with *use scenarios* based on observations of or interviews with real users. Ideally, some of those users should participate in the creation of the specific scenarios, and other users should critique them based on how realistic they are. Use scenarios are then turned into *design scenarios*, in which the same situations are described but with the functionality of the new system. Design scenarios are used, among other things, to create scenario-based video prototypes or software prototypes. Like task-based prototypes, the developer needs to write only the software necessary to illustrate the components of the design scenario. The goal is to create a situation in which the user can experience what the system would be like in a realistic situation, even if it addresses only a subset of the planned functionality.

Section 4 describes a variety of rapid prototyping techniques which can be used in any of these four prototyping strategies. We begin with off-line rapid prototyping techniques, followed by on-line prototyping techniques.

4. Rapid prototypes

The goal of rapid prototyping is to develop prototypes very quickly, in a fraction of the time it would take to develop a working system. By shortening the prototype-evaluation cycle, the design team can evaluate more alternatives and iterate the design several times, improving the likelihood of finding a solution that successfully meets the user's needs.

How rapid is rapid depends on the context of the particular project and the stage in the design process. Early prototypes, e.g. sketches, can be created in a few minutes. Later in the design cycle, a prototype produced in less than a week may still be considered "rapid" if the final system is expected to take months or years to build. Precision, interactivity and evolution all affect the time it takes to create a prototype. Not surprisingly, a precise and interactive prototype takes more time to build than an imprecise or fixed one.

The techniques presented in this section are organized from most rapid to least rapid, according to the representation dimension introduced in section 2. Off-line techniques are generally more rapid than on-line one. However, creating successive iterations of an on-line prototype may end up being faster than creating new off-line prototypes.

4.1 Off-line rapid prototyping techniques

Off-line prototyping techniques range from simple to very elaborate. Because they do not involve software, they are usually considered a tool for thinking through the design issues, to be thrown away when they are no longer needed. This

section describes simple paper and pencil sketches, three-dimensional mock-ups, wizard-of-oz simulations and video prototypes.

Paper & pencil

The fastest form of prototyping involves paper, transparencies and post-it notes to represent aspects of an interactive system (for an example, see Muller, 1991). By playing the roles of both the user and the system, designers can get a quick idea of a wide variety of different layout and interaction alternatives, in a very short period of time.

Designers can create a variety of low-cost "special effects". For example, a tiny triangle drawn at the end of a long strip cut from an overhead transparency makes a handy mouse pointer, which can be moved by a colleague in response to the user's actions. Post-it notesTM, with prepared lists, can provide "pop-up menus". An overhead projector pointed at a whiteboard, makes it easy to project transparencies (hand-drawn or pre-printed, overlaid on each other as necessary) to create an interactive display on the wall. The user can interact by pointing (Fig. 3) or drawing on the whiteboard. One or more people can watch the user and move the transparencies in response to her actions. Everyone in the room gets an immediate impression of how the eventual interface might look and feel.



Figure 3: Hand-drawn transparencies can be projected onto a wall, creating an interface a user can respond to.

Note that most paper prototypes begin with quick sketches on paper, then progress to more carefully-drawn screen images made with a computer (Fig. 4). In the early stages, the goal is to generate a wide range of ideas and expand the design space, not determine the final solution. Paper and pencil prototypes are an excellent starting point for horizontal, task-based and scenario-based prototyping strategies.

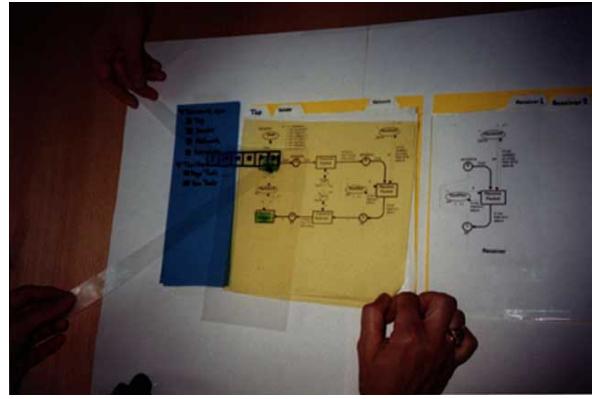


Figure 4: Several people work together to simulate interacting with this paper prototype. One person moves a transparency with a mouse pointer while another moves the diagram accordingly.

Mock-ups

Architects use mock-ups or scaled prototypes to provide three-dimensional illustrations of future buildings. Mock-ups are also useful for interactive system designers, helping them move beyond two-dimensional images drawn on paper or transparencies (see Bødker et al., 1988). Generally made of cardboard, foamcore or other found materials, mock-ups are physical prototypes of the new system. Fig. 5 shows an example of a hand-held mockup showing the interface to a new hand-held device. The mock-up provides a deeper understanding of how the interaction will work in real-world situations than possible with sets of screen images.



Figure 5: Mock-up of a hand-held display with carrying handle.

Mock-ups allow the designer to concentrate on the physical design of the device, such as the position of buttons or the screen. The designer can also create several mock-ups and compare input or output options, such as buttons vs. trackballs. Designers and users should run through different scenarios, identifying potential problems with the interface or generating ideas for new functionality. Mock-ups can also help the designer envision how an interactive system will be incorporated into a physical space (Fig. 6).



Figure 6: Scaled mock-up of an air traffic control table, connected to a wall display.

Wizard of Oz

Sometimes it is useful to give users the impression that they are working with a real system, even before it exists. Kelley (1993) dubbed this technique the *Wizard of Oz*, based on the scene in the 1939 movie of the same name. The heroine, Dorothy, and her companions ask the mysterious Wizard of Oz for help. When they enter the room, they see an enormous green human head, breathing smoke and speaking with a deep, impressive voice. When they return later, they again see the Wizard. This time, Dorothy's small dog pulls back a curtain, revealing a frail old man pulling levers and making the mechanical Wizard of Oz speak. They realize that the impressive being before them is not a wizard at all, but simply an interactive illusion created by the old man.

The software version of the Wizard of Oz operates on the same principle. A user sits a terminal and interacts with a program. Hidden elsewhere, the software designer (the wizard) watches what the user does and, by responding in different ways, creates the illusion of a working software program. In some cases, the user is unaware that a person, rather than a computer, is operating the system.

The Wizard-of-Oz technique lets users interact with partially-functional computer systems. Whenever they encounter something that has not been implemented (or there is a bug), a human developer who is watching the interaction overrides the prototype system and plays the role destined to eventually be played by the computer. A combination of video and software can work well, depending upon what needs to be simulated.

The Wizard of Oz was initially used to develop natural language interfaces (e.g. Chapanis, 1982, Wixon, Whiteside, Good and Jones, 1993). Since then, the technique has been used in a wide variety of situations, particularly those in which rapid responses from users are not critical. Wizard of Oz simulations may consist of paper prototypes, fully-implemented systems and everything in between.

Video prototyping

Video prototypes (Mackay, 1988) use video to illustrate how users will interact with the new system. As explained in section 3.1, they differ from video brainstorming in that the goal is to refine a single design, not generate new ideas.

Video prototypes may build on paper & pencil prototypes and cardboard mock-ups and can also use existing software and images of real-world settings.

We begin our video prototyping exercises by reviewing relevant data about users and their work practices, and then review ideas we video brainstormed. The next step is to create a use scenario, describing the user at work. Once the scenario is described in words, the designer develops a storyboard. Similar to a comic book, the storyboard shows a sequence of rough sketches of each action or event, with accompanying actions and/or dialog (or subtitles), with related annotations that explain what is happening in the scene or the type of shot (Fig. 7). A paragraph of text in a scenario corresponds to about a page of a storyboard.

<Insert image in file: figure.7.eps >

Figure 7: Storyboard. This storyboard is based on observations of real Coloured Petri Net users in a small company and illustrates how the CPN developer modifies a particular element of a net, the "Simple Protocol".

Storyboards help designers refine their ideas, generate 'what if' scenarios for different approaches to a story, and communicate with the other people who are involved in creating the production. Storyboards may be informal "sketches" of ideas, with only partial information. Others follow a pre-defined format and are used to direct the production and editing of a video prototype. Designers should jot down notes on storyboards as they think through the details of the interaction.

Storyboards can be used like comic books to communicate with other members of the design team. Designers and users can discuss the proposed system and alternative ideas for interacting with it (figure 8). Simple videos of each successive frame, with a voice over to explain what happens, can also be effective. However, we usually use storyboards to help us shoot video prototypes, which illustrate how a new system will look to a user in a real-world setting. We find that placing the elements of a storyboard on separate cards and arranging them (Mackay and Pagani, 1994) helps the designer experiment with different linear sequences and insert or delete video clips. However, the process of creating a video prototype, based on the storyboard, provides an even deeper understanding of the design.



Figure 8: Video Prototyping: The CPN design team reviews their observations of CPN developers and then discuss several design alternatives. They work out a scenario and storyboard it, then shoot a video prototype that reflects their design.

The storyboard guides the shooting of the video. We often use a technique called "editing-in-the-camera" (see Mackay, 2000) which allows us to create the video

directly, without editing later. We use title cards, as in a silent movie, to separate the clips and to make it easier to shoot. A narrator explains each event and several people may be necessary to illustrate the interaction. Team members enjoy playing with special effects, such as "time-lapse photography". For example, we can record a user pressing a button, stop the camera, add a new dialog box, and then restart the camera, to create the illusion of immediate system feedback.

Video is not simply a way to capture events in the real world or to capture design ideas, but can be a tool for sketching and visualizing interactions. We use a second live video camera as a Wizard-of-Oz tool. The wizard should have access to a set of prototyping materials representing screen objects. Other team members stand by, ready to help move objects as needed. The live camera is pointed at the wizard's work area, with either a paper prototype or a partially-working software simulation. The resulting image is projected onto a screen or monitor in front of the user. One or more people should be situated so that they can observe the actions of the user and manipulate the projected video image accordingly. This is most effective if the wizard is well prepared for a variety of events and can present semi-automated information. The user interacts with the objects on the screen as wizard moves the relevant materials in direct response to each user action. The other camera records the interaction between the user and the simulated software system on the screen or monitor, to create either a video brainstorm (for a quick idea) or a fully-storyboarded video prototype).

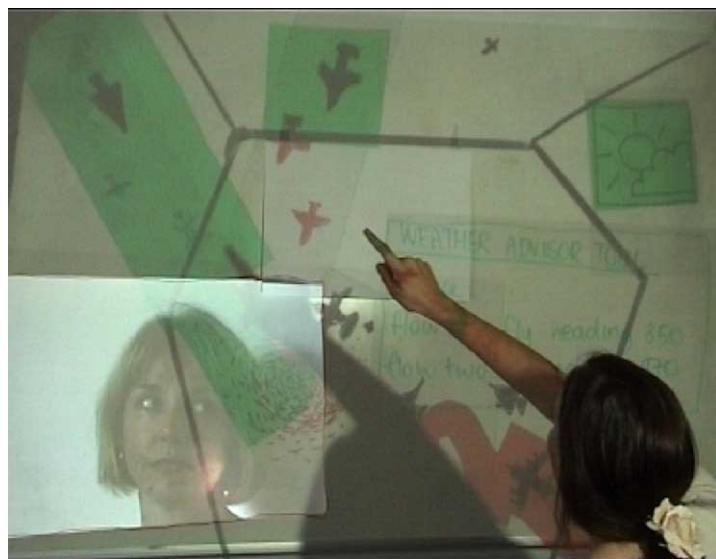


Figure 9: Complex wizard-of-oz simulation, with projected image from a live video camera and transparencies projected from an overhead projector.

Fig. 9 shows a Wizard-of-oz simulation with a live video camera, video projector, whiteboard, overhead projector and transparencies. The setup allows two people to experience how they would communicate via a new interactive communication system. One video camera films the blonde woman, who can see and talk to the brunette. Her image is projected live onto the left-side of the wall. An overhead projector displays hand-drawn transparencies, manipulated by two other people, in response to gestures made by the brunette. The entire interaction is videotaped by a second video camera.

Combining wizard-of-oz and video is a particularly powerful prototyping technique because it gives the person playing the user a real sense of what it might actually feel like to interact with the proposed tool, long before it has been implemented. Seeing a video clip of someone else interacting with a simulated tool is more effective than simply hearing about it; but interacting with it directly is

more powerful still. Video prototyping may act as a form of specification for developers, enabling them to build the precise interface, both visually and interactively, created by the design team.

4.2 On-line rapid prototyping techniques

The goal of on-line rapid prototyping is to create higher-precision prototypes than can be achieved with off-line techniques. Such prototypes may prove useful to better communicate ideas to clients, managers, developers and end users. They are also useful for the design team to fine tune the details of a layout or an interaction. They may exhibit problems in the design that were not apparent in less precise prototypes. Finally they may be used early on in the design process for low precision prototypes that would be difficult to create off-line, such as when very dynamic interactions or visualizations are needed.

The techniques presented in this section are sorted by interactivity. We start with non-interactive simulations, i.e. animations, followed by interactive simulations that provide fixed or multiple-paths interactions. We finish with scripting languages which support open interactions.

Non-interactive simulations

A non-interactive simulation is a computer-generated animation that represents what a person would see of the system if he or she were watching over the user's shoulder. Non-interactive simulations are usually created when off-line prototypes, including video, fail to capture a particular aspect of the interaction and it is important to have a quick prototype to evaluate the idea. It's usually best to start by creating a storyboard to describe the animation, especially if the developer of the prototype is not a member of the design team.

One of the most widely-used tools for non-interactive simulations is Macromedia Director™. The designer defines graphical objects called *sprites*, and defines paths along which to animate them. The succession of events, such as when sprites appear and disappear, is determined with a time-line. Sprites are usually created with drawing tools, such as Adobe Illustrator or Deneba Canvas, painting tools, such as Adobe Photoshop, or even scanned images. Director is a very powerful tool; experienced developer can create sophisticated interactive simulations. However, non-interactive simulations are much faster to create. Other similar tools exist on the market, including Abvent Katabounga, Adobe AfterEffects and Macromedia Flash (Fig. 10).

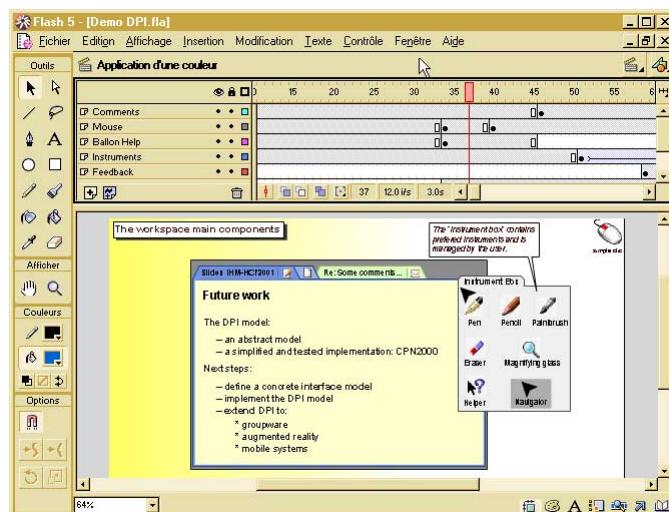


Figure 10: A non-interactive simulation of a desktop interface created with Macromedia Flash. The time-line (top) displays the active sprites while the main window (bottom) shows the animation. (O. Beaudoux, with permission)

Figure 11 shows a set of animation movies created by Dave Curbow to explore the notion of accountability in computer systems (Dourish, 1997). These prototypes explore new ways to inform the user of the progress of a file copy operation. They were created with Macromind Director by combining custom-made sprites with sprites extracted from snapshots of the Macintosh Finder. The simulation features cursor motion, icons being dragged, windows opening and closing, etc. The result is a realistic prototype that shows how the interface looks and behaves, that was created in just a few hours. Note that the simulation also features text annotations to explain each step, which helps document the prototype.

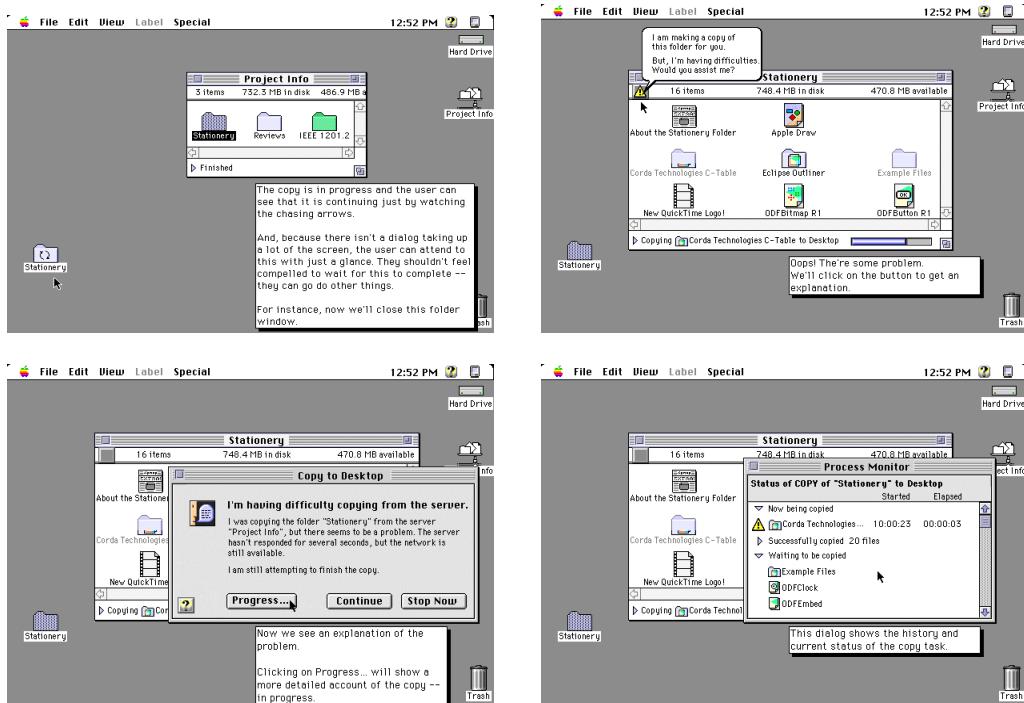


Figure 11: Frames from an animated simulation created with Macromind Director (D. Curbow, with permission)

Non-interactive animations can be created with any tool that generates images. For example, many Web designers use Adobe Photoshop to create simulations of their web sites. Photoshop images are composed of various layers that overlap like transparencies. The visibility and relative position of each layer can be controlled independently. Designers can quickly add or delete visual elements, simply by changing the characteristics of the relevant layer. This permits quick comparisons of alternative designs and helps visualize multiple pages that share a common layout or banner. Skilled Photoshop users find this approach much faster than most web authoring tools.

We used this technique in the CPN2000 project (Mackay et al., 2000) to prototype the use of transparency. After several prototyping sessions with transparencies and overhead projectors, we moved to the computer to understand the differences between the physical transparencies and the transparent effect as it would be rendered on a computer screen. We later developed an interactive prototype with OpenGL, which required an order of magnitude more time to implement than the Photoshop mock-up.

Interactive simulations

Designers can also use tools such as Adobe Photoshop to create Wizard-of-Oz simulations. For example, the effect of dragging an icon with the mouse can be obtained by placing the icon of a file in one layer and the icon of the cursor in another layer, and by moving either or both layers. The visibility of layers, as well as other attributes, can also create more complex effects. Like Wizard-of-Oz and other paper prototyping techniques, the behavior of the interface is generated by the user who is operating the Photoshop interface.

More specialized tools, such as Hypercard and Macromedia Director, can be used to create simulations that the user can directly interact with. Hypercard (Goodman, 1987) is one of the most successful early prototyping tools. It is an authoring environment based on a stack metaphor: a stack contains a set of cards that share a background, including fields and buttons. Each card can also have its own unique contents, including fields and buttons (Fig. 12). Stacks, cards, fields and buttons react to user events, e.g. clicking a button, as well as system events, e.g. when a new card is displayed or about to disappear (Fig. 13). Hypercard reacts according to events programmed with a scripting language called Hypertalk. For example, the following script is assigned to a button, which switches to the next card in the stack whenever the button is clicked. If this button is included in the stack background, the user will be able to browse through the entire stack:

```
on click
  goto next card
end click
```

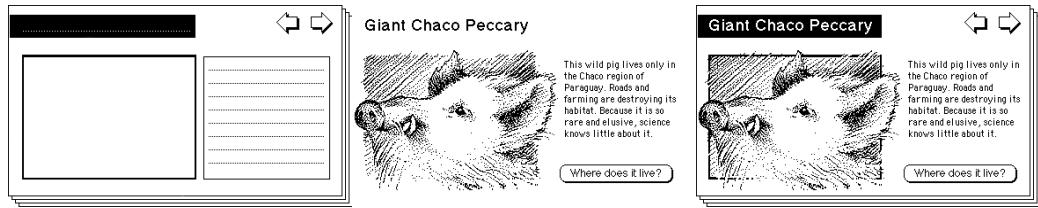


Figure 12: A Hypercard card (right) is the combination of a background (left) and the card's content (middle). (*Apple Computer, permission requested*)

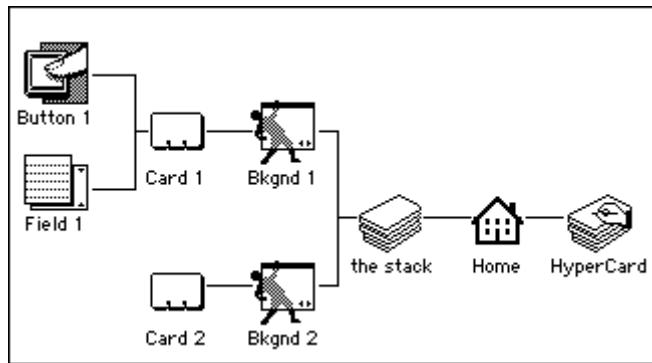


Figure 13: The hierarchy of objects in Hypercard determines the order (from left to right) in which a handler is looked up for an event (*Apple Computer, permission requested*)

Interfaces can be prototyped quickly with this approach, by drawing different states in successive cards and using buttons to switch from one card to the next. Multiple-path interactions can be programmed by using several buttons on each

card. More open interactions require more advanced use of the scripting language, but are fairly easy to master with a little practice.

Director uses a different metaphor, attaching behaviors to sprites and to frames of the animation. For example, a button can be defined by attaching a behavior to the sprite representing that button. When the sprite is clicked, the animation jumps to a different sequence. This is usually coupled with a behavior attached to the frame containing the button that loops the animation on the same frame. As a result, nothing happens until the user clicks the button, at which point the animation skips to a sequence where, for example, a dialog box opens. The same technique can be used to make the OK and Cancel buttons of the dialog box interactive. Typically, the Cancel button would skip to the original frame while the OK button would skip to a third sequence. Director comes with a large library of behaviors to describe such interactions, so that prototypes can be created completely interactively. New behaviors can also be defined with a scripting language called Lingo.

Many educational and cultural CD-ROMs are created exclusively with Director. They often feature original visual displays and interaction techniques that would be almost impossible to create with the traditional user interface development tools described in section 5. Designers should consider tools like Hypercard and Director as user interface builders or user interface development environments. In some situations, they can even be used for evolutionary prototypes (see section 6).

Scripting languages

Scripting languages are the most advanced rapid prototyping tools. As with the interactive-simulation tools described above, the distinction between rapid prototyping tools and development tools is not always clear. Scripting languages make it easy to quickly develop throw-away quickly (a few hours to a few days), which may or may not be used in the final system, for performance or other technical reasons.

A scripting language is a programming language that is both lightweight and easy to learn. Most scripting languages are interpreted or semi-compiled, i.e. the user does not need to go through a compile-link-run cycle each time the script (program) is changed. Scripting languages can be forbidding: they are not strongly typed and non fatal errors are ignored unless explicitly trapped by the programmer. Scripting languages are often used to write small applications for specific purposes and can serve as glue between pre-existing applications or software components. Tcl (Ousterhout, 1993) was inspired by the syntax of the Unix shell, it makes it very easy to interface existing applications by turning the application programming interface (API) into a set of commands that can be called directly from a Tcl script.

Tcl is particularly suitable to develop user interface prototypes (or small to medium-size applications) because of its Tk user interface toolkit. Tk features all the traditional interactive objects (called “widgets”) of a UI toolkit: buttons, menus, scrollbars, lists, dialog boxes, etc. A widget is typically only one line. For example:

```
button .dialogbox.ok -text OK -command {destroy .dialogbox}
```

This command creates a button, called “.dialogbox.ok”, whose label is “OK”. It deletes its parent window “.dialogbox” when the button pressed. A traditional programming language and toolkit would take 5-20 lines to create the same button.

Tcl also has two advanced, heavily-parameterized widgets: the text widget and the canvas widget. The text widget can be used to prototype text-based interfaces. Any character in the text can react to user input through the use of *tags*. For example, it is possible to turn a string of characters into a hypertext link. In Beaudouin-Lafon (2000), the text widget was used to prototype a new method for finding and replacing text. When entering the search string, all occurrences of the string are highlighted in the text (Fig. 14). Once a replace string has been entered, clicking an occurrence replaces it (the highlighting changes from yellow to red). Clicking a replaced occurrence returns it to its original value. This example also uses the canvas widget to create a custom scrollbar that displays the positions and status of the occurrences.

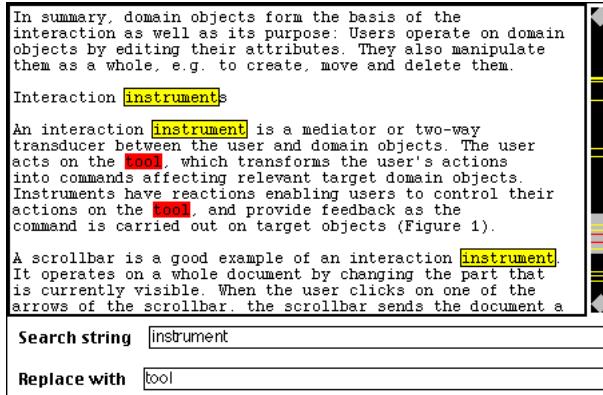


Figure 14: using the Tk text and canvas widgets to prototype a novel search and replace interaction technique (Beaudouin-Lafon, 2000).

The Tk canvas widget is a drawing surface that can contain arbitrary objects: lines, rectangles, ovals, polygons, text strings, and widgets. Tags allow behaviors (i.e. scripts) that are called when the user acts on these objects. For example, an object that can be dragged will be assigned a tag with three behaviors: button-press, mouse-move and button-up. Because of the flexibility of the canvas, advanced visualization and interaction techniques can be implemented more quickly and easily than with other tools. For example, Fig. 15 shows a prototype exploring new ideas to manage overlapping windows on the screen (Beaudouin-Lafon, 2001). Windows can be stacked and slightly rotated so that it is easier to recognize them, and they can be folded so it is possible to see what is underneath without having to move the window. Even though the prototype is not perfect (for example, folding a window that contains text is not properly supported), it was instrumental in identifying a number of problems with the interaction techniques and finding appropriate solutions through iterative design.

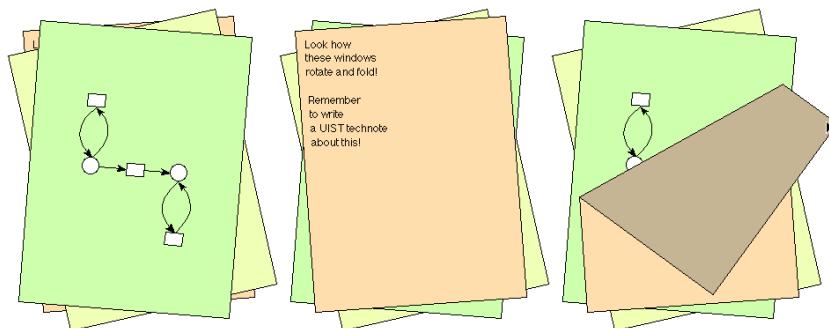


Figure 15: using the Tk canvas widget to prototype a novel window manager (Beaudouin-Lafon, 2001).

Tcl and Tk can also be used with other programming languages. For example, Pad++ (Bederson & Meyer, 1998) is implemented as an extension to Tcl/Tk: the zoomable interface is implemented in C for performance, and accessible from Tk as a new widget. This makes it easy to prototype interfaces that use zooming. It is also a way to develop evolutionary prototypes: a first prototype is implemented completely in Tcl, then parts of are re-implemented in a compiled language to performance. Ultimately, the complete system may be implemented in another language, although it is more likely that some parts will remain in Tcl.

Software prototypes can also be used in conjunction with hardware prototypes. Figure 16 shows an example of a hardware prototype that captures hand-written text from a paper flight strip (using a combination of a graphics tablet and a custom-designed system for detecting the position of the paper strip holder). We used Tk/TCL, in conjunction with C++, to present information on a RADAR screen (tied to an existing air traffic control simulator) and to provide feedback on a touch-sensitive display next to the paper flight strips (Caméléon, Mackay et al., 1998). The user can write in the ordinary way on the paper flight strip, and the system interprets the gestures according to the location of the writing on the strip. For example, a change in flight level is automatically sent to another controller for confirmation and a physical tap on the strip's ID lights up the corresponding aircraft on the RADAR screen.

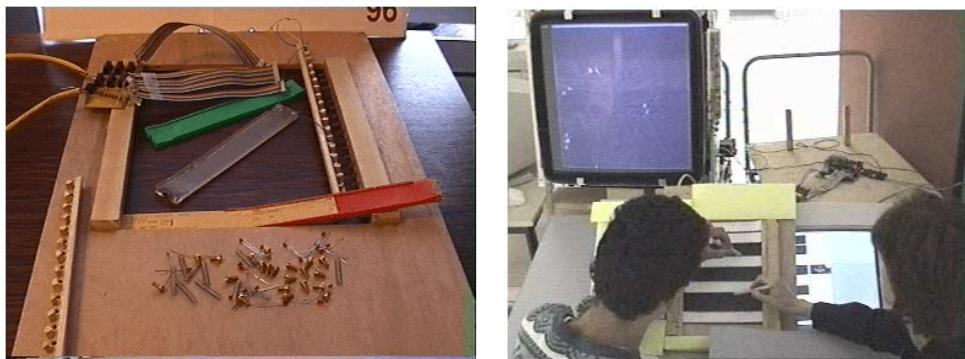


Fig. 16: Caméléon's augmented stripboard (left) is a working hardware prototype that identifies and captures hand-writing from paper flight strips. Members of the design team test the system (right), which combines both hardware and software prototypes into a single interactive simulation.

5. Iterative prototypes

Prototypes may also be developed with traditional software development tools. In particular, high-precision prototypes usually require a level of performance that cannot be achieved with the rapid on-line prototyping techniques described above. Similarly, evolutionary prototypes intended to evolve into the final product require more traditional software development tools. Finally, even shipped products are not "final", since subsequent releases can be viewed as initial designs for prototyping the next release.

Development tools for interactive systems have been in use for over twenty years and are constantly being refined. Several studies have shown that the part of the development cost of an application spent on the user interface is 50% - 80% of the total cost of the project (Myers & Rosson, 1992). The goal of development tools is to shift this balance by reducing production and maintenance costs. Another goal of development tools is to anticipate the evolution of the system over successive releases and support iterative design.

Interactive systems are inherently more powerful than non interactive ones (see Wegner, 1997, for a theoretical argument). They do not match the traditional, purely algorithmic, type of programming: an interactive system must handle user input and generate output at almost any time, whereas an algorithmic system reads input at the beginning, processes it, and displays results at the end. In addition, interactive systems must process input and output at rates that are compatible with the human perception-action loop, i.e. in time frames of 20ms to 200ms. In practice, interactive systems are both reactive and real-time systems, two active areas in computer science research.

The need to develop interactive systems more efficiently has led to two inter-related streams of work. The first involves creation of software tools, from low-level user-interface libraries and toolkits to high-level user interface development environments (UIDE). The second addresses software architectures for interactive systems: how system functions are mapped onto software modules. The rest of this section presents the most salient contributions of these two streams of work.

5.1 Software tools

Since the advent of graphical user interfaces in the eighties, a large number of tools have been developed to help with the creation of interactive software, most aimed at visual interfaces. This section presents a collection of tools, from low-level, i.e. requiring a lot of programming, to high-level.

The lowest-level tools are *graphical libraries*, that provide hardware-independence for painting pixels on a screen and handling user input, and *window systems* that provide an abstraction (the window) to structure the screen into several “virtual terminals”. *User interface toolkits* structure an interface as a tree of interactive objects called widgets, while user interface builders provide an interactive application to create and edit those widget trees. *Application frameworks* build on toolkits and UI builders to facilitate creation of typical functions such as cut/copy/paste, undo, help and interfaces based on editing multiple documents in separate windows. *Model-based tools* semi-automatically derive an interface from a specification of the domain objects and functions to be supported. Finally, *user interface development environments* or UIDEs provide an integrated collection of tools for the development of interactive software.

Before we describe each of these categories in more detail, it is important to understand how they can be used for prototyping. It is not always best to use the highest-level available tool for prototyping. High-level tools are most valuable in the long term because they make it easier to maintain the system, port it to various platforms or localize it to different languages. These issues are irrelevant for vertical and throw-away prototypes, so a high-level tool may prove less effective than a lower-level one.

The main disadvantage of higher-level tools is that they constrain or stereotype the types of interfaces they can implement. User interface toolkits usually contain a limited set of “widgets” and it is expensive to create new ones. If the design must incorporate new interaction techniques, such as bimanual interaction (Kurtenbach et al., 1997) or zoomable interfaces (Bederson & Hollan, 1994), a user interface toolkit will hinder rather than help prototype development. Similarly, application frameworks assume a stereotyped application with a menu bar, several toolbars, a set of windows holding documents, etc. Such a framework would be inappropriate for developing a game or a multimedia educational CD-ROM that requires a fluid, dynamic and original user interface.

Finally, developers need to truly master these tools, especially when prototyping in support of a design team. Success depends on the programmer's ability to quickly change the details as well as the overall structure of the prototype. A developer will be more productive when using a familiar tool than if forced to use a more powerful but unknown tool.

Graphical libraries and Window systems

Graphical libraries underlie all the other tools presented in this section. Their main purpose is to provide the developer with a hardware-independent, and sometimes cross-platform application programming interface (API) for drawing on the screen. They can be separated into two categories: direct drawing and scene-graph based. Direct drawing libraries provide functions to draw shapes on the screen, once specified their geometry and their graphical attributes. This means that every time something is to be changed on the display, the programmer has to either redraw the whole screen or figure out exactly which parts have changed. Xlib on Unix systems, Quickdraw on MacOS, Win32 GDI on Windows and OpenGL (Woo et al., 1997) on all three platforms are all direct drawing libraries. They offer the best compromise between performance and flexibility, but are difficult to program.

Scene-graph based libraries explicitly represent the contents of the display by a structure called a scene graph. It can be a simple list (called display list), a tree (as used by many user interface toolkits – see next subsection), or a direct acyclic graph (DAG). Rather than painting on the screen the developer creates and updates the scene graph, and the library is responsible for updating the screen to reflect the scene graph. Scene graphs are mostly used for 3D graphics, e.g., OpenInventor (Strass, 1993), but in recent years they have been used for 2D as well (Bederson et al., 2000, Beaudouin-Lafon & Lassen, 2000). With the advent of hardware-accelerated graphics card, scene-graph based graphics libraries can offer outstanding performance while easing the task of the developer.

Window systems provide an abstraction to allow multiple client applications to share the same screen. Applications create windows and draw into them. From the application perspective, windows are independent and behave as separate screens. All graphical libraries include or interface with a window system. Window systems also offer a user interface to manipulate windows (move, resize, close, change stacking order, etc.), called the window manager. The window manager may be a separate application (as in X-Windows), or it may be built into the window system (as in Windows), or it may be controlled of each application (as in MacOS). Each solution offers a different trade-off between flexibility and programming cost.

Graphical libraries include or are complemented by an input subsystem. The input subsystem is event driven: each time the user interacts with an input device, an event recording the interaction is added to an input event queue. The input subsystem API lets the programmer query the input queue and remove events from it. This technique is much more flexible than polling the input devices repeatedly or waiting until an input device is activated. In order to ensure that input event are handled in a timely fashion, the application has to execute an event loop that retrieves the first event in the queue and handles it as fast as possible. Every time an event sits in the queue, there is a delay between the user action and the system reaction. As a consequence, the event loop sits at the heart of almost every interactive system.

Window systems complement the input subsystem by routing events to the appropriate client application based on its focus. The focus may be specified explicitly for a device (e.g. the keyboard) or implicitly through the cursor position

(the event goes to the window under the cursor). Scene-graph based libraries usually provide a picking service to identify which objects in the scene graph are under or in the vicinity of the cursor.

Although graphical libraries and window systems are fairly low-level, they must often be used when prototyping novel interaction and/or visualization techniques. Usually, these prototypes are developed when performance is key to the success of a design. For example, a zoomable interface that cannot provide continuous zooming at interactive frame rates is unlikely to be usable. The goal of the prototype is then to measure performance in order to validate the feasibility of the design.

User interface toolkits

User interface toolkits are probably the most widely used tool nowadays to implement applications. All three major platforms (Unix/Linux, MacOS and Windows) come with at least one standard UI toolkit. The main abstraction provided by a UI toolkit is the *widget*. A widget is a software object that has three facets that closely match the MVC model: a presentation, a behavior and an application interface.

The *presentation* defines the graphical aspect of the widget. Usually, the presentation can be controlled by the application, but also externally. For example, under X-Windows, it is possible to change the appearance of widgets in any application by editing a text file specifying the colors, sizes and labels of buttons, menu entries, etc. The overall presentation of an interface is created by assembling widgets into a tree. Widgets such as buttons are the leaves of the tree. Composite widgets constitute the nodes of the tree: a composite widgets contains other widgets and controls their arrangement. For example menu widgets in a menu bar are stacked horizontally, while command widgets in a menu are stacked vertically. Widgets in a dialog box are laid out at fixed positions, or relative to each other so that the layout may be recomputed when the window is resized. Such constraint-based layout saves time because the interface does not need to be re-laid out completely when a widget is added or when its size changes as a result of, e.g., changing its label.

The *behavior* of a widget defines the interaction methods it supports: a button can be pressed, a scrollbar can be scrolled, a text field can be edited. The behavior also includes the various possible states of a widget. For example, most widgets can be active or inactive, some can be highlighted, etc. The behavior of a widget is usually hardwired and defines its class (menu, button, list, etc.). However it is sometimes parameterized, e.g. a list widget may be set to support single or multiple selection.

One limitation of widgets is that their behavior is limited to the widget itself. Interaction techniques that involve multiple widgets, such as drag-and-drop, cannot be supported by the widgets' behavior alone and require a separate support in the UI toolkit. Some interaction techniques, such as toolglasses or magic lenses (Bier et al., 1993), break the widget model both with respect to the presentation and the behavior and cannot be supported by traditional toolkits. In general, prototyping new interaction techniques requires either implementing them within new widget classes, which is not always possible, or not using a toolkit at all. Implementing a new widget class is typically more complicated than implementing the new technique outside the toolkit, e.g. with a graphical library, and is rarely justified for prototyping. Many toolkits provide a “blank” widget (Canvas in Tk, Drawing Area in Motif, JFrame in Java Swing) that can be used by the application to implement its own presentation and behavior. This is usually a good alternative to implementing a new widget class, even for production code.

The *application interface* of a widget defines how it communicate the results of the user interactions to the rest of the application. Three main techniques exist. The first and most common one is called *callback functions* or *callback* for short: when the widget is created, the application registers the name of a one or more functions with it. When the widget is activated by the user, it calls the registered functions (Fig. 17). The problem with this approach is that the logic of the application is split among many callback functions (Myers, 1991).



Fig. 17: Callback functions

The second approach is called *active variables* and consists of associating a widget with a variable of the application program (Fig. 18). A controller ensures that when the widget state changes, the variable is updated with a new value and, conversely, when the value of the variable changes, the widget state reflects the new value. This allows the application to change the state of the interface without accessing the widgets directly, therefore decoupling the functional core from the presentation. In addition, the same active variable can be used with multiple widgets, providing an easy way to support multiple views. Finally, it is easier to change the mapping between widgets and active variables than it is to change the assignment of callbacks. This is because active variables are more declarative and callbacks more procedural. Active variables work only for widgets that represent data, e.g. a list or a text field, but not for buttons or menus. Therefore they complement, rather than replace, callbacks. Few user interface toolkits implement active variables. Tcl/Tk (Ousterhout, 1994) is a notable exception.

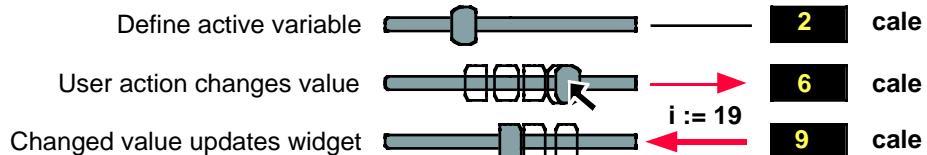


Fig. 18: Active variables

The third approach for the application interface is based on *listeners*. Rather than registering a callback function with the widget, the application registers a listener object (Fig. 19). When the widget is activated, it sends a message to its listener describing the change in state. Typically, the listener of a widget would be its model (using the MVC terminology) or Abstraction (using the PAC terminology). The first advantage of this approach therefore is to better match the most common architecture models. It is also more truthful to the object-oriented approach that underlies most user interface toolkits. The second advantage is that it reduces the “spaghetti of callbacks” described above: by attaching a single listener to several widgets, the code is more centralized. A number of recent toolkits are based on the listener model, including Java Swing (Eckstein et al., 1998).

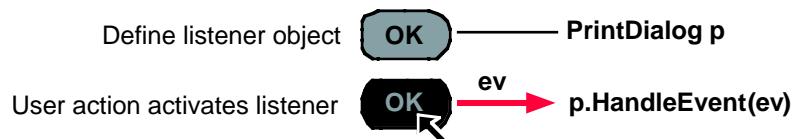


Fig. 19: Listener objects

User interface toolkits have been an active area of research over the past 15 years. InterViews (Linton et al., 1989) has inspired many modern toolkits and user interface builders. A number of toolkits have also been developed for specific applications such as groupware (Roseman and Greenberg, 1996, 1999) or visualization (Schroeder et al., 1997).

Creating an application or a prototype with a UI toolkit requires a solid knowledge of the toolkit and experience with programming interactive applications. In order to control the complexity of the inter-relations between independent pieces of code (creation of widgets, callbacks, global variables, etc.), it is important to use well-known design patterns. Otherwise the code quickly becomes unmanageable and, in the case of a prototype, unsuitable to design space exploration. Two categories of tools have been designed to ease the task of developers: user interface builders and application frameworks.

User-interface builders

A user interface builder allows the developer of an interactive system to create the presentation of the user interface, i.e. the tree of widgets, interactively with a graphical editor. The editor features a palette of widgets that the user can use to “draw” the interface in the same way as a graphical editor is used to create diagrams with lines, circles and rectangles. The presentation attributes of each widget can be edited interactively as well as the overall layout. This saves a lot of time that would otherwise be spent writing and fine-tuning rather dull code that creates widgets and specifies their attributes. It also makes it extremely easy to explore and test design alternatives.

User interface builders focus on the presentation of the interface. They also offer some facilities to describe the behavior of the interface and to test the interaction. Some systems allow the interactive specification of common behaviors such as a menu command opening a dialog box, a button closing a dialog box, a scrollbar controlling a list or text. The user interface builder can then be switched to a “test” mode where widgets are not passive objects but work for real. This may be enough to test prototypes for simple applications, even though there is no functional core nor application data.

In order to create an actual application, the part of the interface generated by the UI builder must be assembled with the missing parts, i.e. the functional core, the application interface code that could not be described from within the builder, and the run-time module of the generator. Most generators save the interface into a file that can be loaded at run-time by the generator’s run-time (Fig. 20). With this method, the application needs only be re-generated when the functional core changes, *not* when the user interface changes. This makes it easy to test alternative designs or to iteratively create the interface: each time a new version of the interface is created, it can be readily tested by re-running the application.

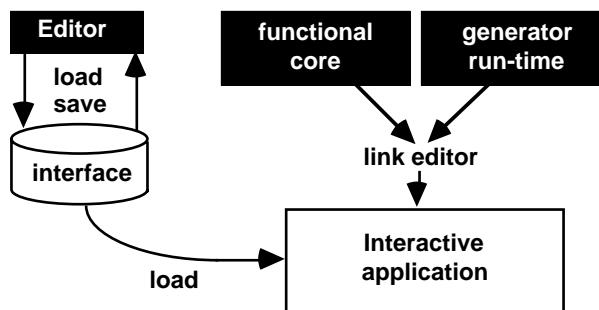


Figure 20: iterative user interface builder.

In order to make it even easier to modify the interface and test the effects with the real functional core, the interface editor can be built into the target application (Fig. 21). Changes to the interface can then be made from within the application and tested without re-running it. This situation occurs most often with interface builders based on an interpreted language (e.g. Tcl/Tk, Visual Basic).

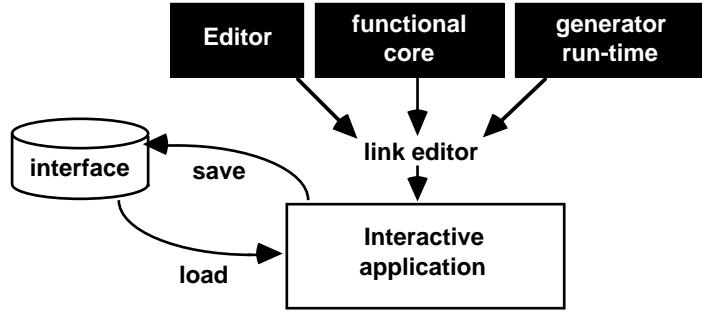


Figure 21: interactive user interface builder.

In either case, a final application can be created by compiling the interface generated by the user interface builder into actual code, linked with the functional core and a minimal run-time module. In this situation, the interface is not loaded from a file but directly created by the compiled code (Fig. 22). This is both faster and eliminates the need for a separate interface description file.

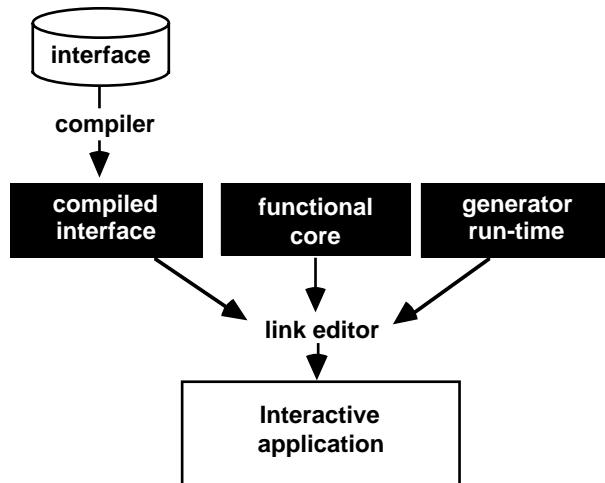


Figure 22: generation of the final application.

User interface builders are widely used to develop prototypes as well as final applications. They are easy to use, they make it easy to change the look of the interface and they hide a lot of the complexity of creating user interfaces with UI toolkits. However, despite their name, they do not cover the whole user interface, only the presentation. Therefore they still require a significant amount of programming, namely some part of the behavior and all the application interface. Systems such as NeXT's Interface Builder (Next, 1991) ease this task by supporting part of the specification of the application objects and their links with the user interface. Still, user interface builders require knowledge of the underlying toolkit and an understanding of their limits, especially when prototyping novel visualization and interaction techniques.

5.2 Software environments

Application frameworks

Application frameworks address a different problem than user interface builders and are actually complementary. Many applications have a standard form where windows represent documents that can be edited with menu commands and tools from palettes; each document may be saved into a disk file; standard functions such as copy/paste, undo, help are supported. Implementing such stereotyped applications with a UI toolkit or UI builder requires replicating a significant amount of code to implement the general logic of the application and the basics of the standard functions.

Application frameworks address this issue by providing a shell that the developer fills with the functional core and the actual presentation of the non-standard parts of the interface. Most frameworks have been inspired by MacApp, a framework developed in the eighties to develop applications for the Macintosh (Apple Computer, 1996). Typical base classes of MacApp include Document, View, Command and Application. MacApp supports multiple document windows, multiple views of a document, cut/copy/paste, undo, saving documents to files, scripting, and more.

With the advent of object-oriented technology, most application frameworks are implemented as collections of classes. Some classes provide services such as help or drag-and-drop and are used as client classes. Many classes are meant to be derived in order to add the application functionality through inheritance rather than by changing the actual code of the framework. This makes it easy to support successive versions of the framework and limits the risks of breaking existing code. Some frameworks are more specialized than MacApp. For example, Unidraw (Vlissides and Linton, 1990) is a framework for creating graphical editors in domains such as technical and artistic drawing, music composition, or circuit design. By addressing a smaller set of applications, such a framework can provide more support and significantly reduce implementation time.

Mastering an application framework takes time. It requires knowledge of the underlying toolkit and the design patterns used in the framework, and a good understanding of the design philosophy of the framework. A framework is useful because it provides a number of functions “for free”, but at the same time it constrains the design space that can be explored. Frameworks can prove effective for prototyping if their limits are well understood by the design team.

Model-based tools

User interface builders and application frameworks approach the development of interactive applications through the presentation side: first the presentation is built, then behavior, i.e., interaction, is added, finally the interface is connected to the functional core. Model-based tools take the other approach, starting with the functional core and domain objects, and working their way towards the user interface and the presentation (Szekely et al., 1992, 1993). The motivation for this approach is that the *raison d'être* of a user interface is the application data and functions that will be accessed by the user. Therefore it is important to start with the domain objects and related functions and derive the interface from them. The goal of these tools is to provide a semi-automatic generation of the user interface from the high-level specifications, including specification of the domain objects and functions, specification of user tasks, specification of presentation and interaction styles.

Despite significant efforts, the model-based approach is still in the realm of research: no commercial tool exists yet. By attempting to define an interface declaratively, model-based tools rely on a knowledge base of user interface design to be used by the generation tools that transform the specifications into an actual interface. In other words, they attempt to do what designers do when they iteratively and painstakingly create an interactive system. This approach can probably work for well-defined problems with well-known solutions, i.e. families of interfaces that address similar problems. For example, it may be the case that interfaces for Management Information Systems (MIS) could be created with model-based tools because these interfaces are fairly similar and well understood.

In their current form, model-based tools may be useful to create early horizontal or task-based prototypes. In particular they can be used to generate a “default” interface that can serve as a starting point for iterative design. Future systems may be more flexible and therefore usable for other types of prototypes.

User interface development environments

Like model-based tools, user interface development environments (UIDE) attempt to support the development of the whole interactive system. The approach is more pragmatic than the model-based approach however. It consists in assembling a number of tools into an environment where different aspects of an interactive system can be specified and generated separately.

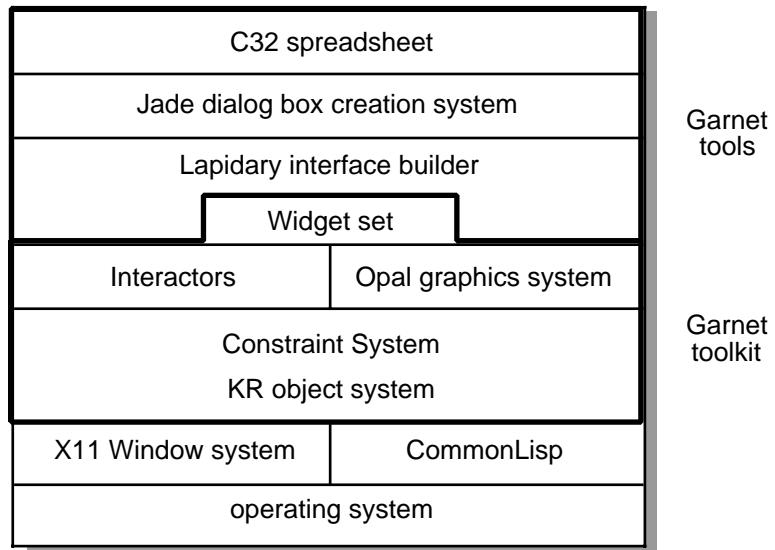


Fig. 23: The Garnet toolkit and tools (Myers et al., 1990)

Garnet (Fig. 23) and its successor Amulet (Myers et al, 1997) provide a comprehensive set of tools, including a traditional user interface builder, a semi-automatic tool for generating dialog boxes, a user interface builder based on a demonstration approach, etc. One particular tool, Silk, is aimed explicitly at prototyping user interfaces.

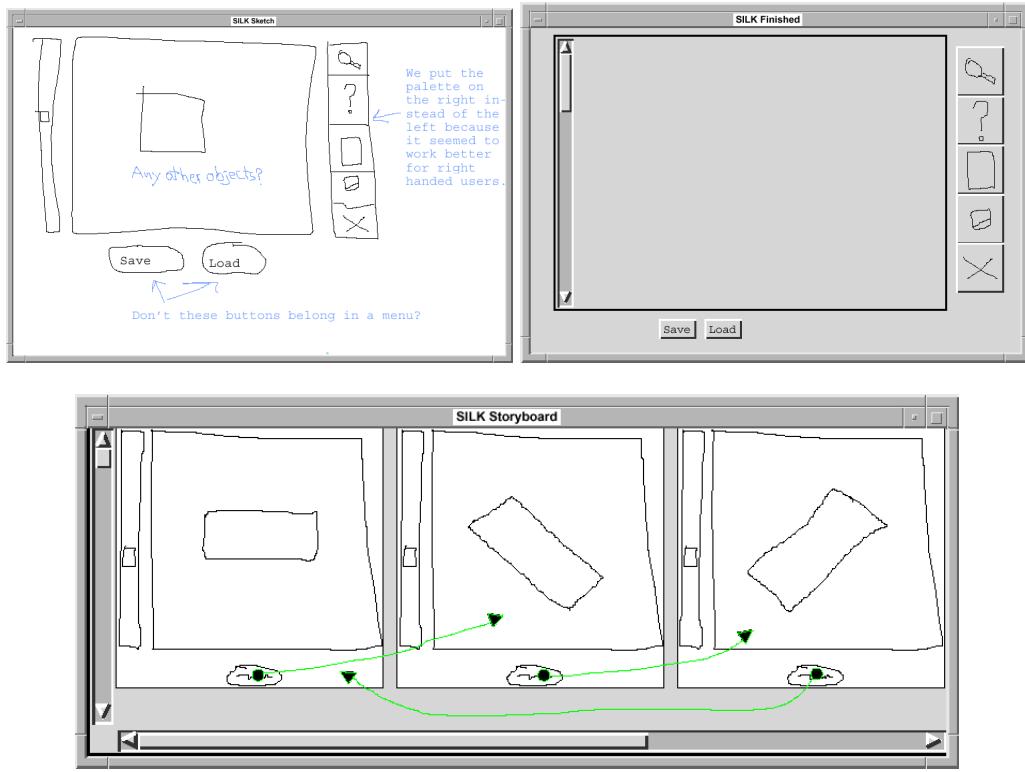


Fig. 24: A sketch created with Silk (top left) and its automatic transformation into a Motif user interface (top right). A storyboard (bottom) used to test sequences of interactions, here a button that rotates an object. (J. Landay, with permission)

Silk (Landay & Myers, 2001) is a tool aimed at the early stages of design, when interfaces are sketched rather than prototyped in software. Using Silk, a user can sketch a user interface directly on the screen (Fig. 24). Using gesture recognition, Silk interprets the marks as widgets, annotations, etc. Even in its sketched form, the user interface is functional: buttons can be pressed, tools can be selected in a toolbar, etc. The sketch can also be turned into an actual interface, e.g. using the Motif toolkit. Finally, storyboards can be created to describe and test sequences of interactions. Silk therefore combines some aspects of off-line and on-line prototyping techniques, trying to get the best of both worlds. This illustrates a current trend in research where on-line tools attempt to support not only the development of the final system, but the whole design process.

6. Evolutionary Prototypes

Evolutionary prototypes are a special case of iterative prototypes, that are intended to evolve into the final system. Methodologies such as Extreme Programming (Beck, 2000) consist mostly in developing evolutionary prototypes.

Since prototypes are rarely robust nor complete, it is often impractical and sometimes dangerous to evolve them into the final system. Designers must think carefully about the underlying *software architecture* of the prototype, and developers should use well-documented *design patterns* to implement them.

6.1 Software architectures

The definition of the software architecture is traditionally done after the functional specification is written, but before coding starts. The designers design on the structure of the application and how functions will be implemented by software

modules. The software architecture is the assignment of functions to modules. Ideally, each function should be implemented by a single module and modules should have minimal dependencies among them. Poor architectures increase development costs (coding, testing and integration), lower maintainability, and reduce performance. An architecture designed to support prototyping and evolution is crucial to ensure that design alternatives can be tested with maximum flexibility and at a reasonable cost.

Seeheim and Arch

The first generic architecture for interactive systems was devised at a workshop in Seeheim (Germany) in 1985 and is known as the Seeheim model (Pfaff, 1985). It separates the interactive application into a *user interface* and a *functional core* (then called “application”, because the user interface was seen as adding a “coat of paint” on top of an existing application). The user interface is made of three modules: the presentation, the dialogue controller, and the application interface (Fig. 25). The *presentation* deals with capturing user’s input at a low level (often called lexical level by comparison with the lexical, syntactic and semantic levels of a compiler). The presentation is also responsible for generating output to the user, usually as visual display. The *dialogue controller* assembles the user input into commands (a.k.a. syntactic level), provides some immediate feedback for the action being carried out, such as an elastic rubber line, and detects errors. Finally, the *application interface* interprets the commands into calls to the functional core (a.k.a. semantic level). It also interprets the results of these calls and turns them into output to be presented to the user.

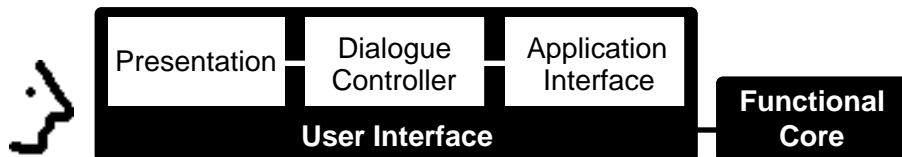


Figure 25: Seeheim model (Pfaff, 1985)

All architecture models for interactive systems are based on the Seeheim model. They all recognize that there is a part of the system devoted to capturing user actions and presenting output (the presentation) and another part devoted to the functional core (the computational part of the application). In between are one or more modules that transform user actions into functional calls, and application data (including results from functional calls) into user output.

A modern version of the Seeheim model is the Arch model (The UIMS Workshop Tool Developers, 1992). The Arch model is made of five components (Fig. 26). The *interface toolkit component* is a pre-existing library that provides low-level services such as buttons, menus, etc. The *presentation component* provides a level of abstraction over the user interface toolkit. Typically, it implements interaction and visualization techniques that are not already supported by the interface toolkit. It may also provide platform independence by supporting different toolkits. The *functional core component* implements the functionality of the system. In some cases it is already existent and cannot be changed. The *domain adapter component* provides additional services to the dialogue component that are not in the functional core. For example, if the functional core is a Unix-like file system and the user interface is a iconic interface similar to the Macintosh Finder, the domain adapter may provide the dialogue controller with a notification service so the presentation can be updated whenever a file is changed. Finally, the *dialogue component* is the keystone of the arch; it handles the translation between the user interface world and the domain world.

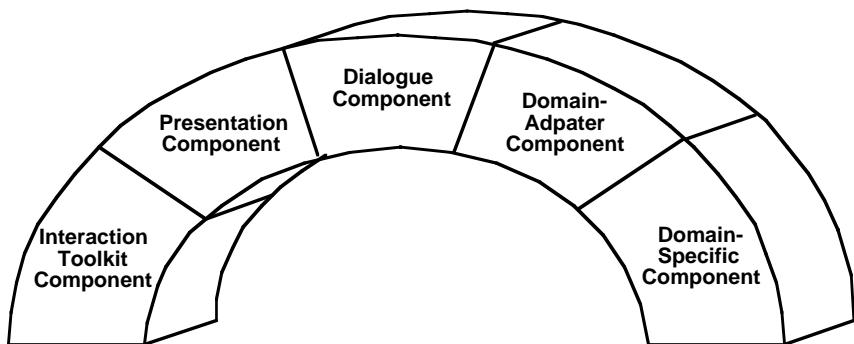


Figure 26: The Arch Model (The UIMS Workshop Developers Tool, 1992)

The Arch model is also known as the Slinky model because the relative sizes of the components may vary across applications as well as during the life of the software. For example, the presentation component may be almost empty if the interface toolkit provides all the necessary services, and be later expanded to support specific interaction or visualization techniques, or multiple platforms. Similarly, early prototypes may have a large domain adapter simulating the functional core of the final system, or interfacing to an early version of the functional core; the domain adapter may shrink to almost nothing when the final system is put together.

The separation that Seeheim, Arch and most other architecture models make between user interface and functional core is a good, pragmatic approach but it may cause problems in some cases. A typical problem is a performance penalty when the interface components (left leg) have to query the domain components (right leg) during an interaction such as drag-and-drop. For example, when dragging the icon of a file over the desktop, icons of folders and applications that can receive the file should highlight. Determining which icons to highlight is a semantic operation that depends on file types and other information and must therefore be carried out by the functional core or domain adapter. If drag-and-drop is implemented in the user interface toolkit, this means that each time the cursor goes over a new icon, up to four modules have to be traversed once by the query and once by the reply to find out whether or not to highlight the icon. This is both complicated and slow. A solution to this problem, called *semantic delegation*, involves shifting, in the architecture, some functions such as matching files for drag-and-drop from the domain leg into the dialogue or presentation component. This may solve the efficiency problem, but at the cost of an added complexity especially when maintaining or evolving the system, because it creates dependencies between modules that should otherwise be independent.

MVC and PAC

Architecture models such as Seeheim and Arch are abstract models and are thus rather imprecise. They deal with *categories* of modules such as presentation or dialogue, when in an actual architecture several modules will deal with presentation and several others with dialogue.

The Model-View-Controller or MVC model (Krasner and Pope, 1988) is much more concrete. MVC was created for the implementation of the Smalltalk-80 environment (Goldberg & Robson, 1983) and is implemented as a set of Smalltalk classes. The model describes the interface of an application as a collection of triplets of objects. Each triplet contains a model, a view and a controller. A *Model* represents information that needs to be represented and interacted with. It is controlled by applications objects. A *View* displays the information in a model in a certain way. A *Controller* interprets user input on the

view and transforms it into changes in the model. When a model changes it notifies its view so the display can be updated.

Views and controllers are tightly coupled and sometimes implemented as a single object. A model is abstract when it has no view and no controller. It is non-interactive if it has a view but no controller. The MVC triplets are usually composed into a tree, e.g. an abstract model represents the whole interface, it has several components that are themselves models such as the menu bar, the document windows, etc., all the way down to individual interface elements such as buttons and scrollbars. MVC supports multiple views fairly easily: the views share a single model; when a controller modifies the model, all the views are notified and update their presentation.

The Presentation-Abstraction-Control model, or PAC (Coutaz, 1987) is close to MVC. Like MVC, an architecture based on PAC is made of a set of objects, called PAC agents, organized in a tree. A PAC agent has three facets: the *Presentation* takes care of capturing user input and generating output; the *Abstraction* holds the application data, like a Model in MVC; the *Control* facet manages the communication between the abstraction and presentation facets of the agent, and with sub-agents and super-agents in the tree. Like MVC, multiple views are easily supported. Unlike MVC, PAC is an abstract model, i.e. there is no reference implementation.

A variant of MVC, called MVP (Model-View-Presenter), is very close to PAC and is used in ObjectArts' Dolphin Smalltalk. Other architecture models have been created for specific purposes such as groupware (Dewan, 1999) or graphical applications (Fekete and Beaudouin-Lafon, 1996).

6.2 Design patterns

Architecture models such as Arch or PAC only address the overall design of interactive software. PAC is more fine-grained than Arch, and MVC is more concrete since it is based on an implementation. Still, a user interface developer has to address many issues in order to turn an architecture into a working system.

Design patterns have emerged in recent years as a way to capture effective solutions to recurrent software design problems. In their book, Gamma et al. (1995) present 23 patterns. It is interesting to note that many of these patterns come from interactive software, and most of them can be applied to the design of interactive systems. It is beyond the scope of this chapter to describe these patterns in detail. However it is interesting that most patterns for interactive systems are behavioral patterns, i.e. patterns that describe how to implement the control structure of the system.

Indeed, there is a battle for control in interactive software. In traditional, algorithmic software, the algorithm is in control and decides when to read input and write output. In interactive software, the user interface needs to be in control because user input should drive the system's reactions. Unfortunately, more often than not, the functional core also needs to be in control. This is especially common when creating user interfaces for legacy applications. In the Seeheim and Arch models, it is often believed that control is located in the dialog controller when in fact these architecture models do not explicitly address the issue of control. In MVC, the three basic classes Model, View and Controller implement a sophisticated protocol to ensure that user input is taken into account in a timely manner and that changes to a model are properly reflected in the view (or views). Some authors actually describe MVC as a design pattern, not an architecture. In fact it is both: the inner workings of the three basic classes is a pattern, but the

decomposition of the application into a set of MVC triplets is an architectural issue.

It is now widely accepted that interactive software is event-driven, i.e. the execution is driven by the user's actions, leading to a control localized in the user interface components. Design patterns such as Command, Chain of Responsibility, Mediator, and Observer (Gamma et al., 1995) are especially useful to implement the transformation of low-level user event into higher-level commands, to find out which object in the architecture responds to the command, and to propagate the changes produced by a command from internal objects of the functional core to user interface objects.

Using design patterns to implement an interactive systems not only saves time, it also makes the system more open to changes and easier to maintain. Therefore software prototypes should be implemented by experienced developers who know their pattern language and who understand the need for flexibility and evolution.

7. Summary

Prototyping is an essential component of interactive system design. Prototypes may take many forms, from rough sketches to detailed working prototypes. They provide concrete representations of design ideas and give designers, users and developers and managers an early glimpse into how the new system will look and feel. Prototypes increase creativity, allow early evaluation of design ideas, help designers think through and solve design problems, and support communication within multi-disciplinary design teams.

Prototypes, because they are concrete and not abstract, provide a rich medium for exploring a design space. They suggest alternate design paths and reveal important details about particular design decisions. They force designers to be creative and to articulate their design decisions. Prototypes embody design ideas and encourage designers to confront their differences of opinion. The precise aspects of a prototype offer specific design solutions: designers can then decide to generate and compare alternatives. The imprecise or incomplete aspects of a prototype highlight the areas that must be refined or require additional ideas.

We begin by defining prototypes and then discuss them as design artifacts. We introduce four dimensions by which they can be analyzed: representation, precision, interactivity and evolution. We then discuss the role of prototyping within the design process and explain the concept of creating, exploring and modifying a design space. We briefly describe techniques for generating new ideas, to expand the design space, and techniques for choosing among design alternatives, to contract the design space.

We describe a variety of rapid prototyping techniques for exploring ideas quickly and inexpensively in the early stages of design, including off-line techniques (from paper&pencil to video) and on-line techniques (from fixed to interactive simulations). We then describe iterative prototyping techniques for working out the details of the on-line interaction, including software development tools and software environments. We conclude with evolutionary prototyping techniques, which are designed to evolve into the final software system, including a discussion of the underlying software architectures, design patterns and extreme programming.

This chapter has focused mostly on graphical user interfaces (GUIs) that run on traditional workstations. Such applications are dominant today, but this is changing as new devices are being introduced, from cell-phones and PDAs to

wall-size displays. New interaction styles are emerging, such as augmented reality, mixed reality and ubiquitous computing. Designing new interactive devices and the interactive software that runs on them is becoming ever more challenging: whether aimed at a wide audience or a small number of specialists, the hardware and software systems must be adapted to their contexts of use. The methods, tools and techniques presented in this chapter can easily be applied to these new applications.

We view design as an active process of working with a design space, expanding it by generating new ideas and contracting as design choices are made. Prototypes are flexible tools that help designers envision this design space, reflect upon it, and test their design decisions. Prototypes are diverse and can fit within any part of the design process, from the earliest ideas to the final details of the design. Perhaps most important, prototypes provide one of the most effective means for designers to communicate with each other, as well as with users, developers and managers, throughout the design process.

8. References

- Apple Computer (1996). Programmer's Guide to MacApp.
- Beaudouin-Lafon, M. (2000). Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. *Proceedings ACM Human Factors in Computing Systems, CHI'2000*, pp.446-453, ACM Press.
- Beaudouin-Lafon, M. (2001). Novel Interaction Techniques for Overlapping Windows. *Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2001*, CHI Letters 3(2), ACM Press. In Press.
- Beaudouin-Lafon, M. and Lassen, M. (2000) The Architecture and Implementation of a Post-WIMP Graphical Application. *Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2000*, CHI Letters 2(2):191-190, ACM Press.
- Beaudouin-Lafon, M. and Mackay, W. (2000) Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. In *Proc. Conference on Advanced Visual Interfaces, AVI 2000*, Palermo, Italy, May 2000, p.102-109.
- Beck, K. (2000). *Extreme Programming Explained*. New York: Addison-Wesley.
- Bederson, B. and Hollan, J. (1994). Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of ACM Symposium on User Interface Software and Technology, UIST'94*, pp.17-26, ACM Press.
- Bederson, B. and Meyer, J. (1998). Implementing a Zooming Interface: Experience Building Pad++. *Software Practice and Experience*, 28(10):1101-1135.
- Bederson, B.B., Meyer, J., Good, L. (2000) Jazz: An Extensible Zoomable User Interface Graphics ToolKit in Java. *Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2000*, CHI Letters 2(2):171-180, ACM Press.
- Bier, E., Stone, M., Pier, K., Buxton, W., De Rose, T. (1993) Toolglass and Magic Lenses : the See-Through Interface. *Proceedings ACM SIGGRAPH*, pp.73-80, ACM Press.

Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5):61-72.

Bødker, S., Ehn, P., Knudsen, J., Kyng, M. and Madsen, K. (1988) Computer support for cooperative design. In *Proceedings of the CSCW'88 ACM Conference on Computer-Supported Cooperative Work*. Portland, OR: ACM Press, pp. 377-393.

Chapanis, A. (1982) Man/Computer Research at Johns Hopkins, *Information Technology and Psychology: Prospects for the Future*. Kasschau, Lachman & Laughery (Eds.) Praeger Publishers, Third Houston Symposium, NY, NY.

Collaros, P.A., Anderson, L.R. (1969), Effect of perceived expertness upon creativity of members of brainstorming groups. *Journal of Applied Psychology*, 53, 159-163.

Coutaz, J. (1987). PAC, an Object Oriented Model for Dialog Design. In *Proceedings of INTERACT'87*, Bullinger, H.-J. and Shackel, B. (eds.), pp.431-436, Elsevier Science Publishers.

Dewan, P. (1999). Architectures for Collaborative Applications. In Beaudouin-Lafon, M. (ed.), *Computer-Supported Co-operative Work*, Trends in Software Series, Wiley, pp.169-193.

Dijkstra-Erikson, E., Mackay, W.E. and Arnowitz, J. (March, 2001) Trialogue on Design of. ACM/Interactions, pp. 109-117.

Dourish, P. (1997). Accounting for System Behaviour: Representation, Reflection and Resourceful Action. In Kyng and Mathiassen (eds), *Computers and Design in Context*. Cambridge: MIT Press, pp.145-170.

Eckstein, R., Loy, M. and Wood, D. (1998). *Java Swing*. Cambridge MA: O'Reilly.

Fekete, J-D. and Beaudouin-Lafon, M. (1996). Using the Multi-layer Model for Building Interactive Graphical Applications. In *Proc. ACM Symposium on User Interface Software and Technology*, UIST'96, ACM Press, p. 109-118.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). *Design Patterns, Elements of Reusable Object-Oriented Software*. Reading MA: Addison Wesley.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin.

Goldberg, A. and Robson, D. (1983). *Smalltalk--80: The language and its implementation*. Reading MA: Addison Wesley.

Goodman, D. (1987). *The Complete HyperCard Handbook*. New York: Bantam Books.

Greenbaum, J. and Kyng, M., eds (1991). *Design at Work: Cooperative Design of Computer Systems*. Hillsdale NJ: Lawrence Erlbaum Associates.

Houde, S. and Hill, C. (1997). What do Prototypes Prototype? In *Handbook of Human Computer Interaction 2ed completly revised*. North-Holland, pp.367-381.

Kelley, J.F. (1983) An empirical methodology for writing user-friendly natural language computer applications. In *Proceedings of CHI '83 Conference on Human Factors in Computing Systems*. Boston, Massachusetts.

Krasner, E.G. and Pope, S.T. (1988). A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, August/September 1988, pp.27-49.

Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, W. (1997). The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. *Proceedings of ACM Human Factors in Computing Systems, CHI'97*, pp.35-42, ACM Press.

Landay, J. and Myers, B.A. (2001). Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer*, 34(3):56-64.

Linton, M.A., Vlissides, J.M., Calder, P.R. (1989). Composing user interfaces with InterViews, *IEEE Computer*, 22(2):8-22.

Mackay, W.E. (1988) Video Prototyping: A technique for developing hypermedia systems. Demonstration in *Proceedings of CHI'88, Conference on Human Factors in Computing*, Washington, D.C.

Mackay, W.E. and Pagani, D. (1994) Video Mosaic: Laying out time in a physical space. *Proceedings of ACM Multimedia '94*. San Francisco, CA: ACM, pp.165-172.

Mackay, W.E. and Fayard, A-L. (1997) HCI, Natural Science and Design: A Framework for Triangulation Across Disciplines. *Proceedings of ACM DIS '97, Designing Interactive Systems*. Amsterdam, Pays-Bas: ACM/SIGCHI, pp.223-234.

Mackay, W.E. (2000) Video Techniques for Participatory Design: Observation, Brainstorming & Prototyping. *Tutorial Notes, CHI 2000, Human Factors in Computing Systems*. The Hague, the Netherlands. (148 pages) URL: www.lri.fr/~mackay/publications

Mackay, W., Ratzer, A. and Janecek, P. (2000) Video Artifacts for Design: Bridging the Gap between Abstraction and Detail. *Proceedings ACM Conference on Designing Interactive Systems, DIS 2000*, pp.72-82, ACM Press.

Myers, B.A., Giuse, D.A., Dannenberg, R.B., Vander Zander, B., Kosbie, D.S., Pervin, E., Mickish, A., Marchal, P. (1990). Garnet: Comprehensive Support for Graphical, Highly-Interactive User Interfaces. *IEEE Computer*, 23(11):71-85.

Myers, B.A. (1991). Separating application code from toolkits: Eliminating the spaghetti of call-backs. *Proceedings of ACM SIGGRAPH Symposium on User Interface Software and Technology, UIST '91*, pp.211-220.

Myers, B.A. and Rosson, M.B. (1992). Survey on user interface programming. In *ACM Conference on Human Factors in Computing Systems, CHI'92*, pp.195-202, ACM Press.

Myers, B.A., McDaniel, R.G., Miller, R.C., Ferrency, A.S., Faulring, A., Kyle, B.D., Mickish, A., Klimotivtski, A., Doane, P. (1997). The Amulet environment. *IEEE Transactions on Software Engineering*, 23(6):347 - 365.

NeXT Corporation (1991). *NeXT Interface Builder Reference Manual*. Redwood City, California.

Norman, D.A. and Draper S.W., eds (1986). *User Centered System Design*. Hillsdale NJ: Lawrence Erlbaum Associates.

Osborn, A. (1957), *Applied imagination: Principles and procedures of creative thinking* (rev. ed.), New York: Scribner's.

Ousterhout, J.K. (1994). *Tcl and the Tk Toolkit*. Reading MA: Addison Wesley.

Perkins, R., Keller, D.S. and Ludolph, F (1997). Inventing the Lisa User Interface. *ACM Interactions*, 4(1):40-53.

Pfaff, G.P. and P. J. W. ten Hagen, P.J.W., eds (1985). *User Interface Management Systems*. Berlin: Springer.

Raskin, J. (2000). *The Humane Interface*. New York: Addison-Wesley.

Roseman, M. and Greenberg, S. (1999). Groupware Toolkits for Synchronous Work. In Beaudouin-Lafon, M. (ed.), *Computer-Supported Co-operative Work*, Trends in Software Series, Wiley, pp.135-168.

Roseman, M. and Greenberg, S. (1996). Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1):66-106.

Schroeder, W., Martin, K., Lorensen, B. (1997). *The Visualization Toolkit*. Prentice Hall.

Strass, P. (1993) IRIS Inventor, a 3D Graphics Toolkit. *Proceedings ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, OOPSLA '93*, pp.192-200.

Szekely, P., Luo, P. and Neches, R. (1992). Facilitating the Exploration of Interface Design Alternatives: The HUMANOID. *Proceedings of ACM Conference on Human Factors in Computing Systems, CHI'92*, pp.507-515.

Szekely, P., Luo, P. and Neches, R. (1993). Beyond Interface Builders: Model-based Interface Tools. *Proceedings of ACM/IFIP Conference on Human Factors in Computing Systems, INTERCHI'93*, pp.383-390.

The UIMS Workshop Tool Developers (1992). A Metamodel for the Runtime Architecture of an Interactive System. *SIGCHI Bulletin*, 24(1):32-37.

Vlissides, J.M. and Linton, M.A. (1990). Unidraw: a framework for building domain-specific graphical editors. *ACM Transactions on Information Systems*, 8(3):237 - 268.

Wegner, P. (1997). Why Interaction is More Powerful Than Algorithms. *Communications of the ACM*, 40(5):80-91.

Woo, M., Neider, J. and Davis, T. (1997) *OpenGL Programming Guide*, Reading MA: Addison-Wesley

“Can I Not Be Suicidal on a Sunday?”: Understanding Technology-Mediated Pathways to Mental Health Support

Sachin R. Pendse
Georgia Institute of Technology
Atlanta, GA, USA
sachin.r.pendse@gatech.edu

Amit Sharma
Microsoft Research
Bangalore, India
amshar@microsoft.com

Aditya Vashistha
Cornell University
Ithaca, NY, USA
adityav@cornell.edu

Munmun De Choudhury
Georgia Institute of Technology
Atlanta, GA, USA
munmund@gatech.edu

Neha Kumar
Georgia Institute of Technology
Atlanta, GA, USA
neha.kumar@gatech.edu

ABSTRACT

Individuals in distress adopt varied pathways in pursuit of care that aligns with their individual needs. Prior work has established that the first resource an individual leverages can influence later care and recovery, but less is understood about how the *design* of a point of care might interact with subsequent pathways to care. We investigate how the design of the Indian mental health helpline system interacts with complex sociocultural factors to marginalize caller needs. We draw on interviews with 18 helpline stakeholders, including individuals who have engaged with helplines in the past, shedding light on how they navigate both technological and structural barriers in pursuit of relief. Finally, we use a design justice framework rooted in Amartya Sen’s conceptualization of realization-focused justice to discuss implications and present recommendations towards the design of technology-mediated points of mental health support.

CCS CONCEPTS

- Human-centered computing → Empirical studies in HCI;
User studies.

KEYWORDS

Mental Health; India; Pathways to Care; Realization-Focused Justice; Social Justice; Technology-Mediated Mental Health Support

ACM Reference Format:

Sachin R. Pendse, Amit Sharma, Aditya Vashistha, Munmun De Choudhury, and Neha Kumar. 2021. “Can I Not Be Suicidal on a Sunday?”: Understanding Technology-Mediated Pathways to Mental Health Support. In *CHI Conference on Human Factors in Computing Systems (CHI ’21), May 8–13, 2021, Yokohama, Japan*. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3411764.3445410>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI ’21, May 8–13, 2021, Yokohama, Japan

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-8096-6/21/05...\$15.00
<https://doi.org/10.1145/3411764.3445410>

1 INTRODUCTION

[Content Warning] “I didn’t even care to see what helpline it was—I just found the number and kept on dialing, kept on calling them. But they were not picking up, so I went back to read what was written in the description. It said it was Monday to Saturday. And I remember, it was Sunday. 9PM-ish. So I was like, what the fuck? Are you kidding me? Can I not be suicidal on a Sunday? And I didn’t know about this before—I thought all the helplines are 24/7.”—*Juhí*

In the wake of tragedy or crisis, telephone-based helplines aim to quickly connect people to information [33, 74] and care [42]. For those in immense and often unexpected distress, the accessibility [62, 119] and experience of using [96, 115] these technology-mediated crisis resources can make the difference between life and death [1]. With the intention of providing support to callers experiencing severe mental distress or suicidal ideation through volunteer-based telephone counseling [65], mental health helplines are a simple and pervasive [15, 17, 47, 107] form of technology-mediated mental health support (TMMHS) [87] system. The care that mental health helplines provide is not limited to crisis support however. Volunteers have observed that callers will also use helplines as a substitute for formal mental health care or to ask for information about mental health [65, 87]. Due to their ubiquitous nature, mental health helplines can also function as an initial point of access to care [11]. Through facilitating connections with broader mental health services, mental health helplines can potentially be the first step on a pathway to further psychiatric care [59, 77].

Our experience of mental health, including how we understand [52] and express [78] how we are feeling to others, is fundamentally tied to the societies that we belong to and the identities that we hold [86]. Social norms [57], laws [66], and our relation to institutions created to provide for or govern our well-being [73] all have an influence on what kinds of care we believe is accessible to us, and how we go about finding care. The utilization of “points of first care” [11] can thus be as diverse as helplines [18, 59], a referral by a general practitioner [21], or religious and traditional healers [14]. In these interactions, later engagements with care resources are influenced by the institutions that assess and validate distress, or what Goldberg and Huxley conceptualize as a “pathway to psychiatric care” [40]. The pathways available to individuals in

distress are influenced by societal factors, including marginalization, stigma, and oppression.

It is well understood that interactions with the “point of first contact” [11] with care can influence an individual’s later engagements with care, but little work has been done to understand how interactions with *technology-mediated* points of access to care influence an individual’s experiences with a broader pathway to care, particularly given the influence of societal factors at each point of care. Though past work has understood how helplines do satisfy caller needs [24, 36, 64], little research has been done to understand how helplines may *not* accommodate caller needs, and who is excluded from accessing care via helplines. The Mental Health Care Act of 2017 [31] affirms that every person [61] in India has the right to access (government-funded) mental healthcare—however, little work has been done to understand where existing resources, such as helplines, may not be doing justice to people’s care needs.

In this work, we ask: **how does the design of the Indian mental health helpline system interact with societal factors to marginalize callers’ individual, identity-based needs?** We focus our investigation around India given the diversity of forms of care practiced [8, 91], the immense resource constraints associated with delivering care [3, 28, 31, 50], and the role that helplines play in understanding and meeting care needs in this ecosystem [87, 109]. Past work in this space has focused on how helpline volunteers in India negotiate and meet the needs of callers [87, 108], but little work has been done to understand the experiences of callers, including their discovery and experiences with care both through and outside of the helpline system. Given the structural and individual nature of mental health [86], understanding these experiences is important when considering how existing resources might meet or marginalize the needs of people in distress, and towards facilitating more equitable and potentially technology-mediated points of access to care.

Through interviews with 18 helpline stakeholders, including individuals who have engaged with helplines in the past, we investigate how callers discover and use helplines, the barriers they face when using them, and the pathways they take to find care that works for them. We then use a design justice [23] approach to understand how callers’ needs are not met as a result of both the interface of the helpline system and overarching societal factors, such as stigma, class, and identity-based prejudices. We find that callers frequently call helplines in a state of immediate distress after trying other options for care, and are faced with an opaque system that often does not meet their individual needs at each step of their quest for care. Stemming from the stigma associated with mental health concerns, callers have little frame of reference for what the experience of calling a helpline might look like, and also little opportunity to safely hold helplines accountable when experiences are lacking or harmful.

This work makes multiple contributions. First, leveraging Costanza-Chock’s design justice framework [23], we highlight the lived experiences of marginalized individuals who have found their needs not met by institutions created to meet those needs. Second, engaging Sen’s delineation between transcendental institutional justice (*niti*) and experienced, realization-focused justice (*nyaya*) [102], we foreground the role of systemic factors that shape how people access and pursue their pathways to care. Finally, we leverage Sen’s

framework to provide recommendations for a more just helpline system, and discuss the implications of this work for the design and evaluation of TMMHS systems.

Privacy and Ethics: This study was approved by the Georgia Institute of Technology and Microsoft Research ethics and institutional review boards. Further information on approaches taken by the researchers to protect participant privacy, safety, and accurately represent the lived experiences of participants without compromising anonymity can be found in Section 4.1. In discussing the experiences that helpline callers have in seeking and engaging with care, this paper presents graphic descriptions of self-harm, suicidal ideation, and sexual harassment. Following Pendse et al.’s [87] suggested use of content warnings, we add a bolded **CW** tag before any quotes that have graphic descriptions of suicide or self-harm.

2 RELATED WORK

Understanding whether care institutions successfully provide for the care to the most marginalized necessitates a deep exploration of the lived experiences of those who engage with care, as well as a deep consideration of the structural factors that influence access to care. In this work, we engage with a design justice [23] framework to understand how structural factors and the design of the helpline system interact to prevent realized, *experienced justice* [102] from being done to the needs of those in intense distress at each point of the process of accessing care through helplines.

2.1 Pathways to Care

The pathway to psychiatric care model was first proposed by Goldberg and Huxley [40] to map out the process by which people come to be defined as mentally ill and receive escalating forms of mental health care. In their original framework, Goldberg and Huxley theorized that one could escalate the level of care received based on progressive referrals from health professionals [45]. In Goldberg and Huxley’s conceptualization, this pathway began with community health professionals and ended with inpatient admission to a hospital. Power to determine the legitimacy and mode of care for distress, including whether care necessitated specialist care from a psychiatrist, was thus put in the hands of referring health professionals [34, 45]. This power differential can be problematic, as health professionals, often trained to identify Western symptoms of mental illness, can often miss culturally bound symptoms [78]. As a result, those with the same psychiatric illness may proceed along a pathway to care in drastically different ways [11]. As a consequence of the power differential between the institution providing care and the person in distress, the potential pathway of care that an individual may have access to is dependent on the level to which institutions on that pathway validate the identity of the individual and their understanding of their distress. Systemic and institutional biases against these identity-based factors may prevent justice from being done to the care needs of those in distress, and potentially even make mental health concerns worse, such as observed in the case of systemic racism against minorities [75].

In addition, pathways to care can be non-linear and bidirectional [11, 39], and the forms of care that help people attain relief can be provided by a diverse set of institutions, including traditional healers, mental health helplines [87], community-based care [49] or

even digital mental health applications [48, 114]. In India in particular, work on pathways to care has focused around examinations of the amount of time associated with gaining psychiatric care given a more pervasive point of first contact with care, such as practitioners of Western medicine, religious healers, and specialized mental health professionals [32, 56]. Little work has been done to understand what alternative pathways to care might look like outside of a deterministic progression from informal to psychiatric care.

In this work, we build on past work to analyze the factors that influence how people in distress find care. We analyze how structural factors and the design of the helpline system have an influence on whose care needs are met and how those needs are met at each point of an individual's pathway to care. Given the influence of these structural factors, we call for a greater consideration of social justice at each point of care when using a “pathways to care” approach to understand individuals’ care needs and design new approaches to aid.

2.2 Technology-Mediated Points of Access to Care

Bhui and Bhugra [11] call the starting point of a pathway to care the “point of first contact.” These points of first contact can have an influence on people’s engagements with mental health care. For example, several studies in the UK have shown that as a result of systemic racism among law enforcement [75], Black individuals are more likely than white individuals to come into contact with mental health services via police [70], and more likely to later be involuntarily and forcibly hospitalized [69]. As noted by Morgan et al. [71], it is theorized that constant and successive negative interactions with a systemically racist mental healthcare system are the reason for what Morgan et al. dub a “vicious cycle of negative experiences, coercion, disengagement, [and] relapse” [71, 98].

The first point of contact with information about mental health and potentially avenues to care can often be technologically mediated, such as through Internet searches [13]. While past work has explored the usability of mental health technologies through user reviews for mental health apps [82], various user evaluations of their interface [114], how they might integrate within clinical workflows [76], including their impact on the therapeutic alliance [41], little work has been done to understand how the affordances and interactions associated with the design of a particular TMMHS system have an influence on later engagements with mental health care.

Rather, the success of a TMMHS system has often been associated with measurements of immediate and short-term relief [44, 116] as opposed to looking more holistically at how experiences with the technology might influence later pathways to care. Looking specifically at crisis helplines in India, Pendse et al. [87] explored the different ways that mental health helpline volunteers come to understand their role on the helpline and carry out that role to help callers while maintaining self-care practices. We build upon this past work, elevating the lived experiences of callers as they interact with the helpline system that helpline volunteers support. In particular, we articulate and contextualize difficulties callers have experienced in connecting with helplines [67, 79].

2.3 Realization-Focused Justice in Providing Care

Questions around how sociotechnical systems can support or exacerbate individual needs have become a growing field of study in Human-Computer Interaction (HCI). In particular, recent work [23, 29] in HCI has emphasized the importance of a greater sensitivity to how sociotechnical systems might amplify the oppression of those who are most marginalized [22, 25], with this sensitivity embodied through orienting design practices towards the pursuit of social justice. In their work on social justice oriented design, Dombrowski et al. [29] acknowledge the existence of diverse and vibrant orientations of thinking about the practice of social justice. This conceptualization of social justice is rooted in a multidimensional [58] approach to Rawls’s framework of distributive justice [94], an approach emphasizing “justice as fairness,” embodied through “the assessment of a system based on the perspectives of those subject to its control.” However, as Dombrowski et al. note [29], the term “social justice” simply functions as a mechanism for thinking about how power and privilege shape our interactions with each other, with institutions, and with technology. Other researchers in HCI have used different conceptualizations of justice as a mechanism to analyze privilege and power, including Young’s emphasis on relational justice in the context of oppression [55, 118], indigenous frameworks around restorative justice [97, 120], and disability justice [9, 89]. Others have proposed a closer integration of conceptualizations of justice into design frameworks, including Asad’s work describing methodological practices prefiguring a more equitable relationship between researchers and non-researchers [4] and the work of the Design Justice Network [23].

To analyze the different ways that societal factors interact with the design of the helpline systems to marginalize callers needs, we engage a design justice approach [23]. In their approach to evaluate technological systems, Costanza-Chock [23] notes the importance of “centering the voices of those who are directly impacted by the outcomes of the design process” in working to understand how the design and interface of the system might marginalize certain users’ needs by design, or reproduce societal inequities. Rather than looking broadly at the existence of resources and systems, Costanza-Chock urges us to look directly at whether these resources and systems do justice to people’s individual needs. Similarly, in the fields of clinical psychology and psychiatry, researchers have recently noted the critical need to include the voices of those with lived experience with mental health in the design of interventions [43], with a focus on those with the most marginalized identities [117]. However, Costanza-Chock does not tie the design justice paradigm to any specific historical conception of justice, noting more broadly the importance of centering stakeholder and community perspectives when determining whether a system might be doing justice to those it serves.

As noted in prior HCI research [29, 55] and beyond [99, 102], though Rawls advocates for justice to be done from the perspective of those most marginalized [94], Rawls’s notions of distributive justice root themselves in an ideal form of justice as achieved via *institutions*, and overlook community [88] and relational [118] forms of oppression and justice. In his critique of Rawls’s distributive justice framework, Sen [102] draws on study of pre-colonial Indian

jurisprudence to describe the concepts of *niti* and *nyaya*: the difference between institutional justice and realization-focused justice. In Sen's conceptualization, while *niti* refers to the procedural and legal ways that justice may be enacted, *nyaya* refers to "the world that actually emerges, not just the institutions or rules we happen to have" [102] and the realization of the justice promoted by the institutions and laws created via *niti*. Sen makes it clear that this form of realization-focused justice requires a two-fold effort—both the creation of institutions that support the practice of an ideal justice, and specific attention and action around the removal of injustices from the perspective of those whose needs are not met. In prior HCI scholarship, Mudliar [72] has engaged Sen's conceptualization of *niti* and *nyaya* to illustrate challenges around biometric scanning for food entitlements in India. Padmanabhan and Abraham [83] have also acknowledged the role of *niti* and *nyaya* in shaping fairer machine learning algorithms.

Our findings align well with Sen's framework around justice. The Indian Mental Health Care Act of 2017 provides that every person in India has a right to access mental healthcare funded by the government [31]. Additionally, many mental health helpline numbers exist for individuals in distress in India [79, 87]. However, as we discuss below, our detailed interviews with those who have engaged with the helpline system find that these resources are often not accessible to callers. This occurs both as a result of the specific user experience associated with the helpline system as well as the structural factors that limit open discussion and access around mental health resources. While justice may be done with regards to *niti*-oriented institutional justice, many in distress do not find their needs met at different points on their pathway to care.

In this work, we leverage Sen's definitions of justice when engaging with a design justice approach to understand how the design of the helplines interacts with societal factors to keep callers from experiencing *nyaya*-oriented realized justice towards their immediate care needs. As we suggest design implications for a more inclusive and usable helpline system, we pay special attention to the kind of world that callers emphasize would accommodate their mental health needs outside of simply their immediate interactions with the affordances of the helpline. Inspired by a *nyaya*-oriented framework, we orient our recommendations towards that world.

3 MENTAL HEALTHCARE IN INDIA

India has a long history of community-based care for mental health concerns [113], and helplines fit into the broader historical pattern of care being sought from diverse points of contact and mediums [54]. Prior to European colonization of the Indian subcontinent, care for mental health concerns in India was framed by community healers as being a part of overall health, with little separation between body and mind [54]. It is likely that the only institutional care that individuals would receive was tied to religious institutions or hospices devoted broadly to health [20, 53, 85]. However, colonization resulted in the creation of the first asylums in India [54] as well as the globalization of the asylum system in the 19th century [20]. Institutions formed under this colonial paradigm were primarily created for the use of British soldiers and settlers [53, 103],

and framed those experiencing mental illness as being fundamentally dangerous to society as a justification for segregation of those experiencing mental illness [104].

By the 1940s, spurred by Indian commissions surveying poor treatment at these institutions [51, 54], new policies, outpatient institutions, and journals devoted to the treatment and care of those with mental illness were created and iterated on over the course of the second half of the 20th century [20, 51, 80]. These new policies centered both medical *and* community forms of care, with one notable policy example being the National Mental Health Programme, instituted in 1982 [84]. Community-centered forms of care have been noted as being particularly important in India's mental health policies given the necessity of a shared understanding of illness between care provider and those in distress for effective care [86, 91]. Identity-based factors, such as stigma [105] and caste [93], can have an influence on the kinds of care people find effective and can access. The latest of these policies, the Mental Health Care Act of 2017, affirms the Indian government's commitment to providing mental healthcare to every person in India [31].

Past research [87, 108] has described how mental health helplines in India are also primarily community-led initiatives, funded by NGOs and other philanthropic organizations, and are often started by individuals who have experienced loss from suicide or have had lived experience with suicidal ideation [87]. Helplines broadly frame themselves as facilitating non-judgmental emotional support [108] and creating a safe space for people to express their feelings [87]. However, as a result of a resource-constrained mental healthcare system, helplines often function as conduits to the broader mental healthcare system, with volunteers providing recommendations and referrals to other points of care [87].

4 METHOD

4.1 Study Design and Recruitment

The goal of this study was to understand the perceptions and experiences that individuals in India have around mental health helplines. We conducted semi-structured interviews with 18 participants from diverse backgrounds and areas of India. Participants were recruited via a combination of purposive [60] and snowball sampling [101], online depression support groups on WhatsApp and Telegram, and Twitter. Due to commonly held beliefs that helplines in India "don't get through" [67, 79], we broadened our selection criteria to include those we conceptualize as helpline stakeholders—individuals who expressed perceptions or opinions about the state of mental health helplines in India, explicitly chose to not call helplines in a state of distress, and wanted to speak to us given our research topic. Interviews were conducted until we had reached a point of saturation [12]. Interviews were done over WhatsApp and Telegram during The Coronavirus Disease 2019 (COVID-19) pandemic [81], all done over the course of June 2020.

Through these interviews, we aimed to elevate the lived experiences participants had interacting with the helpline system and the broader Indian mental health system while in a state of intense distress. Towards this goal of understanding specific participant needs and experiences, questions posed to participants were centered around their perceptions and experiences using helplines in India, with a particular focus around their understanding of gaps

Table 1: This table includes the demographic information of all participants. All names used are pseudonyms.

Name	Gender Identity (self-described)	Age	Location	Formal Mental Health Diagnoses (self-described)	Connected to a helpline?
Ashok	Male	31	Mumbai	Depression	No
Bhumika	Female	29	Delhi	Clinical Depression	No
Subhasini	Cisgender Female	23	Madurai	No specific diagnosis disclosed	Yes
Farah	Female	26	Not disclosed	No specific diagnosis disclosed	Yes
Vikram	Male	26	Bangalore	No specific diagnosis disclosed	Did not try
Diya	Female	26	Madurai	Anxiety Disorder	Yes
Suraj	Male	Undisclosed	Undisclosed	No specific diagnosis disclosed	No
Sandhya	Woman	27	Pune	Borderline Bipolar Disorder	Yes
Juhí	Female	25	Bangalore	Depression	No
Damini	Cisgender Woman	23	Mumbai	No specific diagnosis disclosed	No
Donna	Transwoman	29	Navi Mumbai, from West Bengal	No specific diagnosis disclosed	Did not try
Kiran	Female	21	Mumbai	Borderline Personality Disorder	Yes
Anu	Nonbinary	23	Mumbai	Depression	Did not try
Aashna	Cis Woman	21	Undisclosed	No specific diagnosis disclosed	Yes
Kashika	Female	22	Undisclosed	No specific diagnosis disclosed	Yes
Mitali	Female	22	Delhi	No specific diagnosis disclosed	No
Jayashri	Female	33	Bangalore	Psychosomatic Disorder	Yes
Priya	Female	24	Undisclosed	Depression and Anxiety	Did not try

and unmet needs in their experience and their strategies to fill those gaps. This often led to recommendations from participants for how to improve the service of helplines. Questions included “When did you last call a helpline?”, “How did you find information about helplines?”, and “In an ideal world, what would an experience with helplines look like?” When quoting participants, we use the exact language that they used to describe their relation to their suicidal ideation¹.

Details about the diversity of our sample and the demographics of individual participants are in Table 1. Participants had a wide range with regards to gender identity and locality, and tended to be young, located in large urban areas, at a high level of formal education, and female. Though diverse, this sample was likely not representative of all helpline callers, and we discuss some of the limitations of this sample in our Discussion. As noted, 14 participants attempted to call mental health helplines at some point, and of those, 8 were able to get successfully connected with one. To accommodate anxiety that several participants had associated with phone calls, interviews were done via the medium participants felt most comfortable with. For 14 participants, this was over a phone call, and for 4 participants, this was over synchronous chat.

Participant Safety and Risk Mitigation Measures: This study was approved by the Georgia Institute of Technology and Microsoft Research ethics and institutional review boards (IRBs). To ensure participant safety, as a part of the consent process, participants were clearly told that they are free to end the interview at any time. Post-interview, participants were also debriefed on the content of their interview and sent the accepted draft version of the paper, and were told to let the interviewer know if there are parts of their disclosure that felt too sensitive or deanonymizing for publication. Several participants noted that they greatly appreciated this transparency in our research methods. Additionally, following Draucker et al. [30], after particularly overwhelming questions (such as those

on past suicidal ideation or self-harm), participants were briefly asked after answering if they felt okay and still wanted to continue the interview. To protect privacy, all names used for participants are culturally appropriate pseudonyms.

4.2 Analysis

To analyze the data, we used an inductive and iterative approach guided by our considerations of how helplines may not do justice to the care needs of participants and who may be excluded from making use of their services. We developed codes around caller perceptions and experiences, and then manually coded answers to interview questions to find themes. Through an “open coding” process, we organized specific language and concepts described by participants into broader themes using an interpretative qualitative analysis of interview transcripts [63]. Codes included “discomfort from friends after disclosing suicidal ideation,” “jokes about helpline not working,” “recommendation: helpline referral to therapists”, and “illness-centered framing of distress.” The broader themes that arose from this coding process focused around societal factors that influenced the participant’s mental health (such as their family or their income) or factors related to the helpline interface (such as connections between the helpline and other means of care). In our findings, we describe the role of these factors in how participants understood and used helplines.

Positionality: Our commitment to studying mental health helplines is part of a longer-term engagement of understanding how mental health support is made available to people from various backgrounds via technology-mediated means in India. All authors on this paper are of Indian origin, currently living in both India and the United States, and include authors with lived experience of mental illness.

5 FINDINGS

We now walk through the stages of helpline usage, and examine how both the experience of the helpline system and societal factors influence how and whether callers’ needs are met at each stage. We find that at every step of the pathway from feeling distress to finding

¹In many cases, participants described their desire to “commit suicide.” This wording is widely not used to describe suicide in print as a result of the accompanying stigma associated with the word “commit” [7]. We still use it when quoting participants as part of our aim to represent participants’ lived experiences and understandings of their suicidal ideation as accurately as possible.

helpline-based care, the interface underlying the helpline system and complex societal factors together work to marginalize caller needs, with little potential for participants to hold helplines accountable or advocate for their specific needs. Though the helpline system is framed as an institution facilitating justice towards the needs of those in distress, as a result of its design and societal factors, the system was unable to offer experienced, *realized* justice or *nyaya* to the needs of callers at each step of care.

5.1 Finding Helplines

Participants reported being struck with intense distress or suicidal ideation, and looking for resources to relieve that distress. Finding other resources unavailable, such as friends or therapists, the mental health helpline was seen as the only available resource that might extinguish intense and short-lived distress.

5.1.1 “I need something. I need someone.” Of the 14 participants who chose to call a helpline when in intense distress, only 8 were able to get connected with one. Participants described diverse avenues to helpline discovery, with varying levels of awareness of the existence and function of helplines before discovery. They reported learning about the helpline through information about mental health resources provided by their college, from friends who had more information on which helplines worked from lived experience, from seeing lists of suicide helplines in India in the past, and from searching means of suicide or for resources on Google.

Most participants saw helplines as their last resort after other parts of their support network were found to be unavailable. Mitali described how she felt like she was violating a boundary if she called her therapist outside of their scheduled sessions. Bhumika described her experience of being forced to resort to helplines late at night, noting that she had already tried calling 5-10 friends prior to dialing helpline numbers and found that no one was available to answer her call. Similarly, Sandhya noted she started to believe that she was “disturbing people” when she called people while feeling suicidal or feeling an urge to self-harm, including her therapist.

Initiatives around suicide prevention and mental health in India have emphasized the importance of speaking openly to friends when suicidal [112]. However, in practice, speaking openly about feeling suicidal ended up being discouraging and exhausting for participants. Juhi described how her desire to talk to friends when suicidal often conflicted with her friends’ discomfort with open discussions around mental health, dissuading her from speaking openly to them. In her experience, friends would become “*defensive*” or “*alert*” when she mentioned that she was feeling suicidal to them, and she often found herself suddenly centering *their* needs, telling them “*don’t worry, I won’t die*” to alleviate their sudden panic. Similarly, Mitali described how her friends had actually asked her to stop talking to them about her mental health out of a sense of helplessness (CW):

“If I tell a friend that I feel like self-harming, he gets very uncomfortable with it. And so I respect his space and do not bring it up. Which means I cannot really talk about my mental health because both of those things happen very often. I am very often suicidal and very often, I feel like self-harming. [...] So I do

not really talk about my mental health with anyone.”—*Mitali*

Friends could also be hard to find for those experiencing continued severe mental distress. Mitali described how difficult it was to “*make new friends when you don’t have the energy to do even basic chores*.” Without friends sensitive to her mental health and willing to support her, helplines became one way that Mitali could potentially find a support network.

A majority of participants described calling during night, with several participants even noting that their feelings of distress were heightened at night. Additionally, night was noted by participants as a good time to call the helplines. Kiran explained how most live with an extended family that might not be open to conversations about mental health, and as a result, “*night is the only time when people can call up. When everybody else is asleep*.”

Participants who went to colleges that promoted the use of helpline numbers had a greater awareness of the existence of mental health helplines, and felt that helplines provided a different and less stigmatizing form of support than friends could provide. Farah described the process of asking friends for a helpline number (as opposed to direct support) when in a state of distress:

“I need an independent solution then and there, I don’t need sympathy. My friends give me sympathy and love and affection. (laughs) I need a tool. I need something. I need someone. I don’t want them to think of me like that. I’m okay being vulnerable with them, and my friends, especially my close friends, would know about my problems and my situation and all. But I want somebody who is far away. I don’t want to be identified by these experiences that I have every once in a while.”—*Farah*

Not all participants were aware of the existence of mental health helplines before calling, and were surprised to see Google recommend a mental health helpline number to them. Jayashri described her surprise (CW):

“So one night, it was really bad and I was crying for a very long time and I just wanted to kill myself and I actually—well, what I first Googled was “how to commit suicide.” [...] And then in the results, a lot of helpline numbers came. It was probably the first suggestion that came, open helplines *kar ke* [as displayed based on my query], you get these small prioritized results on Google. [...] Up until the moment I Googled for “how to commit suicide,” I had no idea that helplines even existed.”—*Jayashri*

In several different countries, including India, Google will recommend a helpline notice when an individual searches for a means of suicide or even simply, as Jayashri did, “how to commit suicide” [17]. Kiran drew a connection between Google’s recommendations and the helplines that people share with each other and on social media in her college, noting that Aasra, the helpline recommended at the time² was also most discussed by people in her circles.

²Over the course of fieldwork in June 2020, the helpline recommended by Google within its notice was actually changed from Aasra [2] to iCALL [46], potentially as a result of significant mention of a failure to get connected with Aasra by different news publications [67, 79]

Before calling, participants had little frame of reference for what their experience calling a helpline might be like, and called out of a sense of “*What do I have to lose?*” (Ashok) and wanting some form of relief in a state of imminent crisis. Subhasini similarly described her decision process as “[not wanting] to take this pain anymore” and thinking “whatever advice they give, if I listen to it, then probably I will feel better.” Mental health helplines in India do not frame themselves as being resources for solely those who are suicidal [87], but this is how they were still understood by participants, as highly sought emergency resources. Juhi described being conscientious of this fact when deciding whether to call or not, taking special care to make sure that her call did not displace other callers who may have more severe needs. Juhi viewed calling if she was not suicidal as “*wasting the time*” of volunteers.

As participants noted, dozens of helplines exist for the benefit of those in distress, but few central ways to find these numbers. While different organizations are doing their part by creating aggregated lists of numbers or by recommending numbers in search results, callers still have little understanding of what these helplines are for. Like Juhi, most assume they are only for those who are feeling suicidal, and proceed to not call even when they are feeling a lower level of distress that helplines institutionally aim to support.

5.1.2 “The trust factor”. For those who had some awareness of the existence of helplines before calling, trust factored strongly into which list of helplines participants decided to leverage when searching for help. Participants looked for websites and resources that showed that helplines were willing to accommodate and support their needs that were often identity-based. Diya described how counselors in her college encouraged students to call a recommended helpline if the counselors were unavailable, and as a result, that recommended helpline had what Diya dubbed “the trust factor.” Diya elaborated on this concept, noting that she would not have felt comfortable calling a random number without some sense of how her information would be stored or used by volunteers. Similarly, Damini described intentionally looking for resources that suited her identity-based needs:

“I looked at the Alt Story³ website and there was sort of, like, it said that they care about your social location, they care about your caste, they care about—it sort of gave off the vibe. Plus, the person who started Alt Story before, we were friends.”—Damini

Damini also noted that helplines provided in Google results could often be from other countries (most commonly the United States), and thus irrelevant and non-useable for Indian callers. As a result, helpline aggregation websites from Indian organizations were most commonly used and trusted, particularly those endorsed by celebrities [35] or associated with initiatives around gender identity [110, 111] or community care [5].

Other participants did not feel too strongly about where the number was coming from, noting that in a state of distress, they

were not paying too much attention to information about the number and were looking for a source of relief. As Juhi noted about her decision process in finding a helpline:

“Because if you...if you are in that space, you don’t even want to know the name of that suicide helpline, it’s like just give me a fucking number and I’ll just call it. Give me something that will help.”—Juhi

5.1.3 “And then they’ll call my parents”. In these interactions, participants had little idea of where their data was stored or how it was used when they called the helplines. For some, trust that a helpline would not disclose identifying data from the calls is why they picked the helpline that they ended up calling. Others believed that helplines were a generally benevolent institution, and did not fear misuse of their data. Only Farah and Diya had considered where their information might be stored as well as what kind of data might be stored before they called a helpline. Similarly, as a result of yet unimplemented laws around confidentiality [31], some participants were nervous about the disclosure of information from calls or to mental health professionals, and were not aware of whether information about the confidentiality policy of helplines existed. Though helplines claim to be confidential, participants described breaches of privacy in other mental health resources in their lives. In one example, Kiran described a situation in which her school counselor disclosed information about her sessions to her college professors. Similarly, Kashika described calling a helpline out of a fear that her therapist would tell her mother she was feeling suicidal given past experiences.

Participants were particularly unsure of what happens when a person who calls expresses a tangible plan or high motivation to end their life. In particular, they expressed a lot of fear that the police or an ambulance would be called, which dissuaded them from calling. Juhi described feeling concerned that helpline volunteers might manipulate her given her vulnerable mental state to call her parents or some other person without her best interests in mind: “[They may call] ambulance, and police, and all of this. And then, and then they’ll call my parents and then fuck my life.”

When asked how the helplines would find out Juhi’s parents’ contact information, she described that they may take advantage of her distressed mental state and encourage her to call a friend or family member even if she would not have wanted to. The fear of family involvement was also mentioned by other participants. Sandhya described a situation in which she was planning on attempting suicide, and a friend contacted her family, which resulted in a clinician coming to her home. Sandhya felt uncomfortable with the power differential between her and the clinician sent to her home, and described her desire for an experience in which a peer “[sat] down and [talked] to [her] like an adult”. Institutional forms of care, such as escalation via an ambulance or a clinician, were seen as demeaning, potentially dangerous, or fearful by participants.

Media portrayals of people calling crisis lines when distressed had a strong influence on participant ideas around what happens when someone calls a mental health helpline, including perceptions about potential levels of police involvement. Juhi described seeing Western movies in which people called “911”⁴ when feeling suicidal,

³The Alt Story is a website that does education and awareness work around mental health, and provides counseling services that are “affordable, intersectional feminist, trauma-informed, kink-aware, queer-affirmative and caste aware” in Bengaluru, Mumbai, and remotely [110].

⁴India’s emergency numbers are 100 and 112 [68].

and seeing a police and ambulance intervene: “*All these assumptions about what happens with police and ambulance came from that '911' call only. Like whatever I see in movies, you know?*”

Participants expressed beliefs that even if the police were called, they would likely not be very helpful. Jayashri described how the experiences of a friend reporting their abuser to the police went nowhere, with police officers saying “*Ah, we also do that at home, it's not a big deal*” and being unwilling to submit a First Information Report (FIR), and drew parallels to her friend’s experience to justify her doubt in the helpfulness of potential police involvement. Similarly, Damini expressed a strong belief that an involvement of the police in mental health crises would affect a caller’s mental health negatively, particularly given that suicide was only decriminalized in India in 2017 [31]. Rather than doing justice to caller needs, the involvement of the police was seen as a fearful possibility that could stigmatize callers.

Helpline policies tend to be very careful about the privacy of callers, and do not call any external party or authorities unless there is consent from the caller [87], even choosing to not intervene when a caller has begun the process of ending their life. However, participants made their decision to call as if these policies did not exist, as they were unaware of their existence. The individual agency that could be protected by these institutional policies was not realized by participants as a result of little available information on these policies or their implications for callers.

5.2 Using Helplines

Though participants expected to quickly receive support after calling a helpline, their real experiences were drastically different. Callers described a trial-and-error process of iterating through helpline numbers, a process that often forced callers to set their specific care needs aside and make do with whatever helpline they were able to connect to.

5.2.1 “The phone just keeps ringing”. Participants who found a list of numbers online dialed the first number they saw without looking at much information about the helpline. They shared their expectation that helplines would be available 24/7 and that they would be connected to a volunteer immediately based on representations of crisis helplines in media. Unfortunately, it was also often the case those helplines that claimed to be 24/7 actually did not end up being 24/7, including the helpline recommended by Google when participants searched means of suicide. As Kiran noted, though Aasra claimed to be a 24/7 helpline number, she was unable to connect with them, and at times, it would even say “*the number is not in use anymore*”. Ashok and Mitali described similar experiences with Aasra, noting that they had called as a result of it being recommended by Google’s helpline notice as a 24/7 helpline, but then being met with a busy tone (Mitali) or a process in which the line “*just kept on ringing*” (Ashok).

The constant and unending ringing associated with the helplines was cited by several participants as a particular source of additional anxiety, lending the expectation that someone might eventually pick up. As Farah described:

“It just keeps ringing and it is so horrible because (laughs) it just makes you so ANXIOUS (laughs). Yeah, it just keeps ringing until it stops. You call and you

just keep praying, please pick up, PLEASE pick up. (laughs) Because every second is so long, right? It just feels like eternity. And you can’t just put your phone away and wait for it like a normal call. Or get back to them in 5 minutes. No, you need help right THEN and you start to literally count the number of times it rings. And then at the end it will say that all of the people are busy, get in touch with us in some time or something like that.”—Farah

Three participants even cited the unending ringing as a source of comic relief, laughing at the fact that even the helplines would not pick up their call after they had exhausted their other resources. Bhumika joked, “*Well, if I need to feel suicidal, I'll make sure to fit it during that time*” and Farah joked that the helpline not answering her call was “*the teatime joke of the day*.” While participants made light of the situation to cope, they also noted that this experience of their call for help not being answered made them feel “*worthless*” or “*hopeless*,” and frustrated with how helplines frame themselves.

For those who did make the decision to try to call, this process could be frustrating and tiring enough to dissuade someone from trying again. Suraj described how the exhaustion might dissuade a potential caller (**CW**): “*A person who isn't in a good state isn't going to try 10-20 phone numbers, it isn't like he is looking for fixing an error in their code to keep digging. If they are in a such a bad state of mind, they won't try much.*” For participants who chose to not use a helpline when in distress, the perception that no one would pick up was a main reason they were dissuaded. Anu noted that when someone is “*at the edge*” and is met with no response or the perpetual ring, it “*pushes you more towards the edge*.” This urgency combined with a lack of responsiveness also formed the main reason for why Anu had consciously chosen to not try a helpline when experiencing intense distress. Similarly, Vikram decided to not call helplines when in distress due to his preference for a more consistent method of getting support, such as through text messages with an office provided counseling service. However, some participants felt that calling a helpline might end up being valuable if they were connected, and still proceeded to attempt dialing.

There are dozens of different helplines in India [67, 79, 87], but callers are still unable to reach one when they most need help. Although institutional justice may be done through the existence of these helplines, justice is not realized by those who most need it.

5.2.2 “Going through the motions”. Given the possibility of an endless ring from any one individual number, to get connected, participants understood that their best chance at success when interfacing with the helpline system was to cast a wide net. Participants described a process of progressively trying each number they could find until the line was connected. Only two participants were connected with a helpline on the first try.

Farah described this process of urgently needing to call a helpline for her mother who was staying with her during the pandemic-related lockdown and was in distress:

“So my boyfriend was also with me, and he was calling up helplines as well, and I was also calling up helplines as well, because we both had meetings to get back to at work and thought ‘well, this is a thing that needs to be taken care of.’ But [my mother] was unwell, and I

could see it—so unwell, not eating, not drinking, and I felt so horrible. But yeah, so we called these helplines and my boyfriend finally got across to somebody and I said ‘Give me the phone.’”—Farah

After being connected, participants expected their interactions with the helplines to be a one-time interaction that happened with a volunteer. As a result, participants described being surprised when helplines called them back to check in, a common practice among Indian helplines that is done with caller consent [87]. Subhasini noted her strong appreciation for these follow-ups, noting that it was “refreshing” to have someone check in on her with some familiarity with “what [she was] going through and how [she was] doing” to ensure that Subhasini was doing well. Other participants noted the importance of making a deep connection with the human on the other side of the line, and wishing that they could consistently talk to the same volunteer to avoid the need to repeat their story with each call. This paralleled similar desires expressed by Indian helpline volunteers in Pendse et al.’s [87] study of volunteer motivations and experiences on Indian mental health helplines.

Several participants described storing the contacts of helplines that they knew that they were able to get connected to in their phones for future use. Kiran imagined she might need a helpline in the future, and thus proactively called each helpline she could find when she was feeling okay to figure out which helplines worked consistently and which ones did not, and saved the ones that did work. Even for those who were not connected, the existence of the helpline itself functioned as a safety net that almost all participants felt they could rely on if they did feel suicidal again, even if the helpline did not work. Participants felt like they had agency over their mental health through the existence of helplines as a resource they could potentially rely on when in distress. Subhasini noted that she felt particularly safe knowing that even after she ended the call, there was someone she could try to call and connect with in the future if she needed it. Damini noted that traditional metrics for measuring success (such as number of calls connected or helpline availability) might be inaccurate at measuring success.

“It’s not always like, the correct sort of metric to think that, if someone who has called a helpline and has not gone through suicide, it doesn’t necessarily mean that the helpline worked. Because when you are in a mental state where you’ve decided to call the helpline, you’ve already crossed a hurdle, because you’ve already gained some amount of control over your emotions at that point. So I think just going through the motions of trying to call a helpline calmed me down to an extent.”—Damini

Damini described that if there was a centralized helpline number, some kind of AI-powered Interactive Voice Response (IVR) system that guided callers through grounded exercises while they waited to be connected “*might just calm you down in that moment when you are feeling utterly helpless*” by interrupting thought spirals. Juhi also noted that those who store the numbers or install mental health related apps may never call and simply look at the resources as a moment of pause as they process their thoughts and ponder their next steps. Participants described a broad gap between the intended

justice as provided for by helpline organizations, and how justice was realized by participants.

5.2.3 “I just wanted some temporary relief”. Participants had differing expectations and experiences with regards to the care needs the helplines framed themselves as being able to accommodate, and the care needs helplines were actually able to accommodate. Participants expressed that what made the helplines unique was having a space to be able to vent out their feelings and cry freely, to be able to participate in “*moderated grounding*” (Farah) that helped extinguish panic attacks, exercises to help a caller realize that they may be in the midst of a “*thought spiral*” (Damini, Kiran). For participants, success was framed as any guided exercise or provided space that extinguished urges to end one’s life or self-harm. When asked what a successful interaction looked like, Sandhya responded (CW):

“I did not hurt myself. I mean, that was the whole goal of the call. And that was accomplished for that particular day. I won’t say that that one call completely healed me and I never hurt myself again. That was not the case, but that was also not even the intention of the call.”—Sandhya

Similarly, Diya noted a clear division in support from therapy versus support from helplines:

“I knew that this probably wouldn’t be—this wouldn’t be therapy. She’s not going to help me through all the issues that led to this moment of my life. But I just wanted some temporary relief, and I got that.”—Diya

Participants cited the goal of a helpline interaction not necessarily being to return to a “happy state,” but for the helpline volunteer to guide them to “clarity and a sense of normal” (Farah) in which they had some level of agency over their state of crisis. However, several participants noted that they had little recourse for when helpline experiences especially did not help to meet these goals, and indicated that they wanted a mechanism to provide feedback on the experience that they had. Ashok described an experience where after a poor experience, he tried to call the helpline back to provide feedback, and being unable to be connected, he messaged the helpline’s Facebook page:

“So they have a Facebook page, and from what I remember, they were quite regular about posting about how you can reach out to us and all of that stuff. And I wrote them a message on Facebook—initially a politer one. [...] But yeah, I did send a second message after trying again on the phone. And I said ‘All right guys, thankfully, I managed to make a call to a friend so I am ok. But now I have tried you a number of times and I even messaged you on Facebook, you never replied. So this is not a way to run a hotline. So just take down your goddamn number.”—Ashok

The helpline’s page responded with an adversarial response, noting that he had “*no business telling us what to do*” and apologizing that he was unable to be connected. Ashok never called a helpline again, but noted that a simpler way to provide feedback to helpline volunteers would be useful. In this situation, the caller was forced to use his personal Facebook profile to provide feedback.

This method of providing feedback was particularly dangerous, as it often required the participants to deanonymize themselves when attempting to hold institutions accountable. Jayashri described her experience trying to provide feedback after an experience in which she was feeling especially suicidal and was met with volunteers harassing her through making “*lewd jokes*” and asking her about her “*family life*.”:

“I just wanted someone to answer the call, and actually tell them that I had this experience with your helpline and I really want to give bad feedback for this person. [...] After that, my friends told me, why do you want to take it there, especially if they escalate it to some level. Your anonymity will be compromised if you try to do this. Especially, that’s the thing—they say all this stuff about anonymity, but they have the number that I am calling from. It can be quite damaging, let it go, you’ve already had quite a bit to deal with.”—Jayashri

Given the non-linear nature of mental health recovery, participants also recognized that immediate post-call feedback may not reflect their overall perspectives on the support that the helpline offered. Participants recommended that feedback happen in some automated way after some time had passed since the interaction with helpline volunteers. Participants cited times of 12 hours to several days, noting that time to reflect on the experience of having called and processing their experience on the helpline gave them a better idea of how the experience could have been improved. Text messages were cited as one relatively welcome way for participants to provide feedback.

Given that participant needs in a moment of crisis were different based on the level and type of distress they were feeling, participants were reluctant to give feedback on the volunteer’s strategies themselves, preferring instead to provide feedback on the interface itself or on how the interaction made them feel. As Diya noted:

“The one thing that I would be okay sharing is whether I’d want to call again or not, but I don’t think I will be able to tell them ‘here, you didn’t do well’ or ‘here, you could do this better.’ Because, I don’t know, maybe another person in the same situation if they’d been answered that way [by the volunteer], maybe they would have gotten most of what they wanted.”—Diya

A good metric for feedback suggested by several participants was summarized well by Ashok—a simple question asking the caller whether the helpline made them feel “safe.”

Helpline policies are influenced both by the parent organization [100], as well as the individual centers that administer care [87]. These policies (*niti*, or institutional justice) are designed towards helping callers, but in the implementation of these policies (*nyaya*, or realized justice), there are many potential points of failure or risks of harm on the callers’ end. Callers also have no means of recourse to address these failures. This happens due to a combination of the stigma associated with speaking openly about calling a helpline, and an inability to reach the helplines, let alone to offer feedback. This can be particularly harmful given that helplines serve some of the most marginalized.

5.3 Pathways to Future Care

When helplines were unavailable, participants found it difficult to find other resources that could help them in their time of need. These resource constraints were often societal, influenced by perspectives on the identities the participant held (such as their sexual orientation), the kind of community-based and caste-aware care they needed, or stigma against mental illness in general. These systemic injustices kept participants from both being able to find support for their needs from helplines, and from being able to find sufficient alternative care elsewhere.

5.3.1 “Depression as a rich person’s problem”. Helplines were seen by participants as a stopgap measure in moments of severe crisis, and several participants noted that they had used other mental health care resources before, during, and after calling helplines. Of the 18 participants that we interviewed, 14 had seen a therapist at some point, 3 were not able to, and 1 did not disclose. However, most had only seen a therapist in the past for a small number of sessions, and were not consistently seeing a therapist at the time of interview. Additionally, 3 had exclusively seen the counselor at their college, and not sought outside resources for their mental health.

For those whose first experience with mental health resources were helplines, their experience with helplines had a strong impact on their later willingness to engage with mental health resources, both including calling the helpline again as well as exploring more formal care. Damini described how after a bad experience with a helpline, her friend was unwilling to try any other mental health resource until Damini herself booked an appointment for her friend.

“So there’s one friend who refused to go for therapy after having a bad experience [with helplines]—this is someone with whom I’ve had to engage for about a year to get her to just, let her know that all mental health help is not the same. [...] So yeah eventually I had to tell her, ‘you know, let me book a session for you.’ So it’s like, that’s the level of pursuing you have to do when someone is put off by one bad experience with a helpline. [...] So if I call a helpline one time, and I’m feeling suicidal, and it doesn’t work? In the future, calling the helplines? I might skip that step.”—Damini

Similarly, after a poor experience calling a helpline, Jayashri was never motivated to seek professional help for mental health concerns ever again, even when recommended to do so by a doctor. She later found support and relief from a rekindled relationship with her mother.

Poor experiences influenced whether people felt comfortable calling the helpline again. For example, Diya noted that she had thought about calling helplines again, but that “*the feeling of not having gotten enough out of it remained with [her]*” and she preferred to talk to a friend or a therapist when in intense distress. For many others, when helplines did not work, therapy was found desirable but not affordable. Ashok noted that, as a result of the ₹1500–2000⁵ per session cost of therapy, most in India see “*depression as a rich person’s problem*,” with therapy being something that the average

⁵This is approximately \$20–30 (USD). As of 2019, average annual income per capita is estimated by the Indian government to be ₹96,563 [90], or approximately \$1,274 (USD).

person “simply cannot afford.” Juhi illustrated her search for affordable care and its inaccessibility given her financial situation and locality, noting that she would enthusiastically make use of “scholarship for mental health.”

“If somebody could like, give me a scholarship for mental health, I would totally take it. Like, please I need it, it’s so bad. Problem was that my parents didn’t know about it, so that’s why it was a financial issue for me. Mostly, I knew that I can afford one or two sessions. I was not sure who I can go to in [hometown] because I tried, but I was not comfortable about privacy.”—Juhi

As Juhi noted, due to the inaccessibility of other forms of care, online resources were a temporary and inconsistent measure that she attempted to make use of. However, online resources were rarely cited as being enough when a participant was in need. Anu noted that they had tried YourDost and 7Cups and found them lacking, determining that they were not “*at the stage such that non-professional help would have helped.*”

Even after participants were able to find a therapist, it could be the case that the therapist was not sensitive to their needs. Priya described how she attempted to see several different therapists, but when she started to describe her experience of being queer, the (male) therapists doubted her, asking her uncomfortable questions such as whether she was sure she was queer or if she had ever had sex with a man. She did not try therapy after those poor experiences, instead relying on friends when in times of distress. Similarly, Kiran described how after a therapist had breached her privacy, Kiran’s mother traveled to her town and made appointments with 5-6 psychologists per day until Kiran had found a psychologist who was a good fit. Kiran acknowledged that the privilege of having parents who validated her mental health and were wealthy enough to spend money on it was what made it possible for her to keep trying therapists until she was able to find one who worked.

Participants who had one-off experiences with therapists described not necessarily feeling significantly better after 1-2 sessions, and recommended that helplines be able to make recommendations for effective therapists in the area of the caller. Suraj suggested that “*suggesting therapists near [the caller’s] area*” and “*taking feedback from [the caller]*” to make better recommendations for therapists might be a way for helplines to more effectively help people. Anu even noted that they had found their excellent and well-matched therapist through an online list of therapists recommended and aggregated by a mental health helpline. For several participants, helplines functioned as a complement to therapy, which they could call when their therapists were not available. Participants described later processing their experiences calling the helplines with therapists, teasing out the deeper causes of why they may have been in such intense distress when they called, and work towards healing.

5.3.2 “So there has to be a fundamental structural change”. Identity played a substantial role with regards to the kind of care participants desired from helplines. In several cases, these identity-based needs around mental health formed the reason why participants were not likely to use the helpline when in distress. All participants who were not cisgender men or male-identified expressed a gender preference with regards to the person giving care. These

preferences were rooted in the forms of care that they wanted to get, shared experiences, and avoiding potential harassment (Kiran, Jayashri) or “*slut-shaming*” from volunteers (Damini): “*Yeah, I would prefer talking to a woman definitely. Because talking to a man, we don’t always have good experiences with guys. That could be really triggering again.*”

Participants who were members of the LGBTQ community were also outspoken about the need for queer affirmative volunteers on the helpline calls, as well as information online indicating that the helpline was a safe place for callers who were queer. Damini noted the constant fear that queer people have of feeling judged when accessing resources, and noted that information online might encourage someone “*who is really worried about being judged.*” to call.

Participants were mixed on whether they thought there should be separate helplines for LGBTQ people in India as in other countries [88]. Several strongly believed that a separate helpline would help them feel safer with regards to the specific LGBTQ-related issues that they might want to discuss. Aashna noted that she would have felt “*a hundred times more scared about calling a helpline if [she] wanted to talk to [helplines] about my queerness.*” Other participants stressed that a separate helpline for members of the LGBTQ community might cause more of a reason for members of Indian society to see the LGBTQ population as a separate, stigmatized group. Anu noted that a different helpline felt like a “[separation] on the basis of identity” and that it would not “*normalize us from having these identities, or we exist, or these things happen.*” They noted that so long as the volunteer was able to actively listen without being judgmental or giving unsolicited advice, it could give those who are in distress “*a big space between yourself and the edge.*”

Participants advocated for more queer affirmative volunteers to staff the existing helplines, as well as devoting energy to making the helplines accessible past a ring of no return, before devoting energy to a queer-affirmative helpline. Anu (who identified as nonbinary and queer) called it a “*breather*” when a mental health professional was queer, and noted:

“And a lot of people from the community should be given opportunities to volunteer and work on these helplines. [...] Because a lot of times, the fear of judgment harms us and keeps us away. It is shrouded in shame and guilt. So it stops us from reaching out. [...] So when a psychiatrist, or therapist, or helpline is queer affirmative or from the community, that helps a lot. Like okay, people start to have hope. That there ARE people like me out there, people who have life figured out. That it’s not completely hopeless.”—Anu

Anu went on to note that societal changes were necessary to make the helplines more queer affirmative, past simply making a separate line or having more volunteers on the helpline who were queer: “*We should not be thrown in the shadows again. If people are sensitized about it and the stigma around it is reduced, that would help so much.*” Both Anu and Priya described how they relied on a community of often queer or LGBT-identified friends around them when they were feeling especially distressed. Priya characterized formal mental health care as administered by a stigmatizing and “*conservative*” society.

Donna, a member of an organization supporting Bahujan⁶ mental health also described how the helpline system does not work for people from adivasi⁷ backgrounds by enforcing a “medicalization” of distress rooted in “upper caste savarna”⁸ notions of normality and ways of understanding mental health, as opposed to indigenous ones.

“So there has to be a fundamental structural change, that’s what I am talking about. We need to recognize the wisdom that is existing in the [indigenous] communities, how they look at mental health. Because even if we have a helpline specifically for tribals, but the mental health professionals are all savarna, again the diagnosis will still be the same, the level of understanding will still be the same.”—Donna

Donna urged for a greater consideration of the different ways caste and identity might impact the way a caller might want to be cared for, particularly those who might be multiply marginalized [19, 22, 25], such as someone who is Bahujan and queer. To accommodate these specific needs, Donna stressed the importance of having separate but “interwoven” resources for mental health. Donna also emphasized the importance of including modes of providing and receiving care not formally recognized as mental health care, citing the example of how their mother would talk about her life and family with other “sisters” in the neighborhood as a form of “therapeutic” relief from domestic violence.

5.3.3 “We can’t take anybody with a disorder”. Several participants had a sense of shame about calling helplines in a time of need. When Farah disclosed to the interviewer that she had called several helplines and the interviewer responded positively, she exclaimed “Oh, huh, I didn’t [tell you I had called several helplines] because I thought you’d freak out. Like ‘I didn’t want a freak for this, I wanted somebody with a one-time experience.’” Similarly, Subhasini felt embarrassed that she felt the need to call the helplines, given that she was a mental health professional.

Participants described this stigma having an influence on when and how they accessed helplines, such as making sure to call during night so their family members did not know they had called. Sandhya noted that her parents were “scared” of mental illness, seeing it as “something abnormal,” as she believed her parents “[didn’t belong] to a generation where these things don’t happen a lot.”

In some cases, this stigma also had a direct impact on the kinds of resources participants were able to access in a time of need. Kiran described how she had been denied help from an online chat resource on a helpline’s website as a result of her borderline personality disorder (BPD) diagnosis. When she asked for resources and therapists specific to those with BPD, she was met with the answer of “we can’t take anybody with a disorder” and “you’ll need to see a specialist.” Even with this stigma, participants still found

⁶A label used by the participant and their organization to describe those who are part of historically lower caste and indigenous communities in India, including Dalits, Adivasis, OBCs, Pasmandas, Nomadic Tribes [26].

⁷Indigenous Indian

⁸Savarna literally means “part of the caste system” in Sanskrit, and is used to describe those who were historically placed within a caste as a part of the caste system, as opposed to being seen as invisible and existing below the institution of caste itself [37]. This term is often used to describe those with the privilege of being of a higher caste [92].

diagnoses validating and useful in explaining and normalizing their feelings. Kiran noted that she “loved” her BPD diagnosis.

Of the participants interviewed, 9 mentioned that they had been formally diagnosed with some form of mental health issue. Given a lack of access to institutions that could formally validate their lived experiences, participants also doubted the validity of what they were feeling, noting that even though they felt like what they were experiencing were “panic attacks,” they had never been formally diagnosed. In describing their distress during interviews, participants would describe a symptom, but make sure to add a caveat afterwards expressing their lack of confidence in using clinical terminology for what they were experiencing, having never seen a psychiatrist. Self-doubt in whether expressions of clinical distress could be considered valid were heard from both participants who had not been formally diagnosed with a mental health issue and those who had been formally diagnosed with a mental health issue.

6 DISCUSSION

Through investigating the lived experiences of callers, we found a key gap between how care was intended by those who support helplines, and how that same care was experienced in practice by those who needed it most. In particular, the institutional promise of a quickly accessible active listener was rarely successfully realized for participants, an injustice highlighting the gap between *niti* and *nyaya* in how helplines were understood and experienced by participants. Participants who needed help immediately were met with endless ringing, a lack of quickly accessible information about the helplines’ operating protocols or volunteers, and uncertainty over the ability of the helplines to provide non-judgmental care to those with marginalized identities. At each turn of a participant’s attempted pathway to care, the system intended to facilitate their care interacted with societal factors in making the specific needs of participants invisible. Individual factors (such as gender identity, sexual orientation, or level of distress) influenced the kinds of care that participants felt were accessible to them, and thus influenced where and how they looked for resources when in need. When attempting to access what they understood to be accessible care via the helpline system, barriers rooted in the design of the helpline system limited their ability to engage with the resource, and further influenced their future interactions with other forms of care. These intersections between individual needs, societal factors, and the design of the system make clear the need for researchers to deeply consider how structural factors create hurdles along an individual’s pathway to care.

We found that participant needs were not met even though the institutions to meet those needs existed, such as legal promises of accessible healthcare [31] and a diversity of resources. This suggests that the Rawlsian notion of solely approaching justice based on the existence of resources or institutions [94] is not enough. We thus follow Sen’s [102] two-fold approach of both centering the realized, experienced injustices by callers, and being conscientious of what a greater justice for those with mental health concerns might look like. We use Costanza-Chock’s design justice framework [23] to articulate the harms perpetuated by the intersections of helpline design and societal marginalization. We view each harm as an injustice, and make recommendations for what a more just helpline

might look like from the callers' perspectives, drawing inspiration from Sen's recommendation of *removing injustices* [102].

- (1) **Signaling Wait Time:** While multiple helplines numbers are available and able to dialed by anyone with a phone, this availability does not translate to accessibility. Participants are often met with a perpetual ringing, a busy tone, or a notice that a line has been disconnected, and expressed a desire to better understand how long it might take before their needs are met. Low-cost systems that estimate and report the wait-time associated with matching a caller to an agent poised to answer their call have been used in other domains [38], and would be a welcome addition to the mental health helpline system. Additionally, if wait-times are particularly high, call-backs could be leveraged [10, 106] with caller consent. Automated follow-ups could text the caller asking how they are doing and recommend resources based on their response until the line is free for the caller.
- (2) **Intelligent Call Routing:** While several helplines framed themselves as accessible at any time, participants found that these lines were not accessible when they needed them most (such as at night). Participants operated assuming that some of these helplines would fail to work, and would begin a process of iteratively trying helplines till one worked. This process could easily be quickly automated and parallelized to save callers time and energy, and interactive IVR-based systems could potentially be used to walk participants through grounding exercises with their consent. Predictive modeling of caller behavior used to efficiently route callers to the most free center in other domains [6, 27] could also be leveraged here.
- (3) **Supporting Pathways to Care:** Though there are legal provisions for accessible mental health care in India [31], participants described the process of trying to find care outside of helplines quite difficult and expensive. Several participants recommended that there be linkages with other forms of care for those who want more consistent care than what helplines could provide. These linkages might look like a separate menu for callers that provides location-specific recommendations for affordable therapists endorsed by the helpline, or specific referrals to other helplines or resources, such as a helpline specific to gender identity or sexual orientation (with caller consent).
- (4) **Mechanisms for Feedback:** In cases when helplines did not help participants, participants had little means of being able to hold the institutions accountable. Potential feedback mechanisms based on participant recommendations might include a separate number in which callers can anonymously leave open-ended feedback, automated text messages that ask the participant to report how they feel after some period of time, or a simple post-call question of whether the call made the caller feel "safe."

As Costanza-Chock notes, simple improvements to make the design of a system more usable do not necessarily mean that the system is more accessible [23]. Sen notes that realization-focused justice means that justice must be experienced from the perspective of individuals in need [102]. For helplines to do justice to the needs

of those they serve, they must both be accessible and also *understood* as accessible, so callers can pursue the care that suits their needs rather than what care happens to be available. Drawing from disability justice [89], as part of a design justice approach, Costanza-Chock stresses the importance of analyzing how interlocking and intersecting societal systems influence who can and cannot access care that meets their needs. We thus make recommendations for helpline organizers and journalists to act towards destigmatizing the practice of calling helplines and making them more accessible to those most marginalized.

- (1) **Information about Calling Experience:** The policy of helplines is to actively listen to anyone who calls in distress. As a result of stigma around open expression of mental distress and care, there were few open narratives about the experience of calling a helpline, and participants had little idea of what to expect before they called. Participants believed that they could only call when exceptionally distressed, and decided not to call even when it might have benefited them. News articles about suicide often end with a mental health helpline number at the end in case information is triggering [16], and helpline numbers are often copied and shared across social media after crisis [95]. However, there is little information attached about what happens *after* one calls the helpline. Adding a short description of what kinds of distress are supported by helplines and what happens when someone calls would enable callers to make better decisions about what resource to reach out to when distressed.
- (2) **Communicating Safety and Agency:** Helplines frame their services as being accessible to all, regardless of background [87]. Participants still felt wary of calling the helplines through fears of the helplines calling the police without their consent or fears of being judged by helpline volunteers due to their sexual orientation. More information about the forms of care practiced on a helpline, including queer affirmative policies and the fact that Indian helplines do not involve the police in care [87], would help callers feel safer when calling and feel more open to discussing their unfiltered thoughts and emotions.
- (3) **Volunteer Backgrounds and Diversity:** Participants expressed how mental health professionals in their lives had discounted their sexual orientation or their modes of experiencing care, and noted that this was an institutional resulting from a lack of awareness or diversity by those who decide what care looks like, and for who it is practiced. Participants urged a greater diversity of volunteers on the helpline, both from queer and Bahujan backgrounds, and the creation of separate but *interwoven* resources for those who are marginalized. Helpline administrators could intentionally target recruitment to members of these communities, and incorporate their experiences when designing policies on how helpline counseling is conducted.

It is important to note that though discussions of identity-based marginalization did arise in our study, our sample of participants was predominantly urban, young, and at a high level of formal education. Sen characterizes the practice of justice as being one that combines work done towards the pursuit of an ideal world

(*niti*), and immediate work to correct injustice based on the lived experiences of those most impacted (*nyaya*). The individual and structural recommendations we make here might make care slightly more accessible, but are incremental steps towards social justice and care for those experiencing mental health concerns. Further work is needed to understand how individuals from other communities not sampled might be marginalized by the design of the helpline, societal factors, or at other points on their pathway to care.

7 CONCLUSION

In this work, through interviews with 18 callers and stakeholders, we explore how the interface associated with mental health helplines in India interacted with complex societal factors to marginalize individual caller needs. We investigate the different ways that helplines are perceived and used, and find significant technical and societal barriers to successful care. Through engaging with a design justice [23] approach to understand how caller needs are not met by the design of the helpline, rooting our analysis in Amartya Sen's ideas around realization-focused justice [102], we explore what a more just pathway to care that utilizes the helpline system might look like for callers.

ACKNOWLEDGMENTS

This work was possible due to the willingness of our participants to share their most intimate (and often painful) experiences searching for care and relief. Participants in this study made time to share their stories with us, even in the context of widespread stigma and a global pandemic (The Coronavirus Disease 2019 or COVID-19), and we are immensely grateful. In addition, special thanks go out to Devyani Nighoskar for laying down one early foundation of this study [79], and being a sounding board throughout the process of writing this work. For this work, Pendse was supported partly through a Microsoft gift to De Choudhury and Kumar, and Pendse and De Choudhury were partly supported through NIH grant R01MH117172 to De Choudhury.

REFERENCES

- [1] [n.d.]. FCC 12-149: Facilitating the Deployment of Text-to-911 and Other Next Generation 911 Applications.
- [2] Aasra. 2020. (2020). <http://www.aasra.info/helpline.html>
- [3] Gregory Armstrong, Michelle Kermode, Shoba Raja, Sujatha Suja, Prabha Chandra, and Anthony F Jorm. 2011. A mental health training program for community health workers in India: impact on knowledge and attitudes. *International journal of mental health systems* 5, 1 (2011), 17.
- [4] Mariam Asad. 2019. Prefigurative design as a method for research justice. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–18.
- [5] The Banyan. 2020. (2020). <https://thebanyan.org/>
- [6] Ho Bao, Scott A Henning, Christopher E Pearce, and James David Williams. 2011. System and method for routing calls. US Patent 7,974,277.
- [7] Susan Beaton, Peter Forster, and Myfanwy Maple. 2013. Suicide and language: Why we shouldn't use the C'word. *InPsych* (2013).
- [8] Robinder P Bedi, Pavithra A Thomas, Damanjit Sandhu, and Sachin Jain. 2020. Survey of counselling psychologists in India. *Counselling Psychology Quarterly* 33, 1 (2020), 100–120.
- [9] Cynthia L Bennett, Daniela K Rosner, and Alex S Taylor. 2020. The care work of access. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.
- [10] RagHurma Bhat, Mukul Jain, and Joseph F Khouri. 2011. Dynamic callbacks from a contact center. US Patent App. 12/580,716.
- [11] Kamaldeep Bhui and Dinesh Bhugra. 2002. Mental illness in Black and Asian ethnic minorities: Pathways to care and outcomes. *Advances in Psychiatric Treatment* 8, 1 (2002), 26–33.
- [12] Patrick Biernacki and Dan Waldorf. 1981. Snowball sampling: Problems and techniques of chain referral sampling. *Sociological methods & research* 10, 2 (1981), 141–163.
- [13] Michael L Birnbaum, Asra F Rizvi, Keren Faber, Jean Addington, Christoph U Correll, Carla Gerber, Adrienne C Lahti, Rachel L Loewy, Daniel H Mathalon, Leigh Anne Nelson, et al. 2018. Digital trajectories to care in first-episode psychosis. *Psychiatric Services* 69, 12 (2018), 1259–1263.
- [14] Jonathan K Burns and Andrew Tomita. 2015. Traditional and religious healers in the pathway to care for people with mental disorders in Africa: a systematic review and meta-analysis. *Social psychiatry and psychiatric epidemiology* 50, 6 (2015), 867–877.
- [15] Chee-Hon Chan, Ho-Kit Wong, and Paul Siu-Fai Yip. 2018. Exploring the use of telephone helpline pertaining to older adult suicide prevention: A Hong Kong experience. *Journal of affective disorders* 236 (2018), 75–79.
- [16] Prabha S Chandra, Padmavathy Doraiswamy, Anuroopa Padmanabh, and Mariamma Philip. 2014. Do newspaper reports of suicides comply with standard suicide reporting guidelines? A study from Bangalore, India. *International journal of social psychiatry* 60, 7 (2014), 687–694.
- [17] Qijin Cheng and Elad Yom-Tov. 2019. Do search engine helpline notices aid in preventing suicide? Analysis of archival data. *Journal of medical internet research* 21, 3 (2019), e12235.
- [18] James CS Chiang, Alex SY Chow, Raymond CK Chan, CW Law, and Eric YH Chen. 2005. Pathway to care for patients with first-episode psychosis in Hong Kong. *Hong Kong Journal of Psychiatry* 15, 1 (2005), 18–23.
- [19] Kirstyn Yuk Sim Chun and Anneliese A Singh. 2010. The bisexual youth of color intersecting identities development model: A contextual approach to understanding multiple marginalization experiences. *Journal of Bisexuality* 10, 4 (2010), 429–451.
- [20] Alex Cohen, Vikram Patel, and Harry Minas. 2014. A brief history of global mental health. *Global mental health: Principles and practice* (2014), 3–26.
- [21] Eleanor Cole, Gerard Leavey, Michael King, Eric Johnson-Sabine, and Amanda Hoar. 1995. Pathways to care for patients with a first episode of psychosis: a comparison of ethnic groups. *The British Journal of Psychiatry* 167, 6 (1995), 770–776.
- [22] Patricia Hill Collins. 1990. Black feminist thought in the matrix of domination. *Black feminist thought: Knowledge, consciousness, and the politics of empowerment* 138 (1990), 221–238.
- [23] Sasha Costanza-Chock. 2020. *Design justice: Community-led practices to build the worlds we need*. MIT Press.
- [24] Catherine M Coveney, Kristian Pollock, Sarah Armstrong, and John Moore. 2012. Callers' experiences of contacting a national suicide prevention helpline. *Crisis* (2012).
- [25] Kimberle Crenshaw. 1990. Mapping the margins: Intersectionality, identity politics, and violence against women of color. *Stan. L. Rev.* 43 (1990), 1241.
- [26] The Blue Dawn. 2020. (2020). <https://twitter.com/thebluedawn56>
- [27] Vladimir N Deryugin, Dimitriy A Torba, and Igor Neyman. 1999. Method for routing calls to call centers based on statistical modeling of call behavior. US Patent 5,926,538.
- [28] Amit Dias, Fredric Azariah, Stewart J Anderson, Miriam Sequeira, Alex Cohen, Jennifer Q Morse, Pim Cuijpers, Vikram Patel, and Charles F Reynolds. 2019. Effect of a lay counselor intervention on prevention of major depression in older adults living in low-and middle-income countries: a randomized clinical trial. *JAMA psychiatry* 76, 1 (2019), 13–20.
- [29] Lynn Dombrowski, Ellie Harmon, and Sarah Fox. 2016. Social justice-oriented interaction design: Outlining key design strategies and commitments. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 656–671.
- [30] Claire Burke Draucker, Donna S Martolf, and Candice Poole. 2009. Developing distress protocols for research on sensitive topics. *Archives of psychiatric nursing* 23, 5 (2009), 343–350.
- [31] Richard M Duffy and Brendan D Kelly. 2020. India's Rights of Persons with Disabilities Act, 2016. In *India's Mental Healthcare Act*, 2017. Springer, 61–80.
- [32] S Faizan, BN Ravesh, LS Ravindra, and K Sharath. 2012. Pathways to psychiatric care in South India and their socio-demographic and attitudinal correlates. In *BMC proceedings*, Vol. 6. BioMed Central, 1–1.
- [33] Stephanie S Felder, Jamie Seligman, Cicely K Burrows-McElwain, Maryann E Robinson, and Erik Hierholzer. 2014. Disaster trauma: federal resources that help communities on their road to recovery. *Disaster medicine and public health preparedness* 8, 2 (2014), 174–178.
- [34] Raymond Fink, Sam Shapiro, and Sidney S Goldensohn. 1970. Family physician referrals for psychiatric consultation and patient initiative in seeking care. *Social Science & Medicine* (1967) 4, 3 (1970), 273–291.
- [35] The Live Love Laugh Foundation. 2020. (2020). <https://www.thelivelovelauthfoundation.org/helpline.html>
- [36] Myfanwy Franks and Ros Medforth. 2005. Young helpline callers and difference: exploring gender, ethnicity and sexuality in helpline access and provision. *Child & Family Social Work* 10, 1 (2005), 77–85.
- [37] Robert Eric Frykenberg. 2008. Avarna and Adivasi Christians and Missions: a paradigm for understanding Christian movements in India. *International bulletin of missionary research* 32, 1 (2008), 14–20.

- [38] Padma R Gargya, Charles M White, and Stuart Dufour. 2004. System and method for implementing wait time estimation in automatic call distribution queues. US Patent 6,714,643.
- [39] R Gater, Almeida E Sousa, B De, G Barrientos, J Caraveo, CR Chandrashekhar, M Dhadphale, D Goldberg, AH Al Kathiri, M Mubbashar, et al. 1991. The pathways to psychiatric care: a cross-cultural study. *Psychological medicine* 21, 3 (1991), 761–774.
- [40] David Goldberg and Peter Huxley. 1980. Mental health in the community: The pathways to psychiatric care. *London (UK): Tavistock Publications* (1980).
- [41] Philip Henson, Hannah Wisniewski, Chris Hollis, Matcheri Keshavan, and John Torous. 2019. Digital mental health apps and the therapeutic alliance: initial review. *BJPsych Open* 5, 1 (2019).
- [42] Peter E Hodgkinson and Michael Stewart. 1991. *Coping with catastrophe: A handbook of disaster management*. Taylor & Francis/Routledge.
- [43] Melanie A Hom, Brian W Bauer, Ian H Stanley, Joseph W Boffa, Dese'Rae L Stage, Daniel W Capron, Norman B Schmidt, and Thomas E Joiner. 2020. Suicide attempt survivors' recommendations for improving mental health treatment for attempt survivors. *Psychological services* (2020).
- [44] Anne Huguet, Sanjay Rao, Patrick J McGrath, Lori Wozney, Mike Wheaton, Jill Conrod, and Sharlene Rozario. 2016. A systematic review of cognitive behavioral therapy and behavioral activation apps for depression. *PloS one* 11, 5 (2016), e0154248.
- [45] Peter Huxley. 1996. Mental illness in the community: the Goldberg-Huxley model of the pathway to psychiatric care. *Nordic Journal of Psychiatry* 50, sup37 (1996), 47–53.
- [46] iCALL. 2020. (2020). <http://icallhelpline.org/>
- [47] Yesmin Iqbal, Rubina Jahan, and Muhtasabbib Rumman Matin. 2019. Descriptive characteristics of callers to an emotional support and suicide prevention helpline in Bangladesh (first five years). *Asian journal of psychiatry* 45 (2019), 63–65.
- [48] Jemimah A Johnson, Janhavi Devdutt, Seema Mehrotra, Poornima Bhola, Paulomi Sudhir, and Amit Sharma. 2020. Barriers to Professional Help-seeking for Distress and Potential Utility of a Mental Health App Components: Stakeholder Perspectives. *Cureus* 12, 2 (2020).
- [49] Ravindra Lai Kapur. 1946. The story of community mental health in India. *Mental health: An Indian perspective* 2003 (1946), 92–100.
- [50] Ravi L Kapur. 1979. The role of traditional healers in mental health care in rural India. *Social Science & Medicine. Part B: Medical Anthropology* 13, 1 (1979), 27–31.
- [51] Sarbjit Khurana and Shweta Sharma. 2016. National mental health program of India: a review of the history and the current scenario. *Int J Community Med Public Health* 3 (2016), 2696–2704.
- [52] Arthur Kleinman. 1978. Concepts and a model for the comparison of medical systems as cultural systems. *Social Science & Medicine. Part B: Medical Anthropology* 12 (1978), 85–93.
- [53] K Krishnamurthy, D Venugopal, and AK Alimchandani. 2000. Mental hospitals in India. *Indian journal of psychiatry* 42, 2 (2000), 125.
- [54] Anant Kumar. 2004. History of mental health services in India. *Journal of Personality and Clinical Studies* 20, 1–2 (2004).
- [55] Neha Kumar, Nassim Jafarinaini, and Mehrab Bin Morshed. 2018. Uber in Bangladesh: The Tangled Web of mobility and justice. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–21.
- [56] Chandrakanti Lahariya, Shyam Singhal, Sumeet Gupta, and Ashok Mishra. 2010. Pathway of care among psychiatric patients attending a mental health institution in central India. *Indian journal of psychiatry* 52, 4 (2010), 333.
- [57] James D Livingston and Jennifer E Boyd. 2010. Correlates and consequences of internalized stigma for people living with mental illness: A systematic review and meta-analysis. *Social science & medicine* 71, 12 (2010), 2150–2161.
- [58] Hennie Löter. 2011. *Poverty, ethics and justice*. University of Wales Press.
- [59] Kathleen MacDonald, Nina Faiman-Adelman, Kelly K Anderson, and Srividya N Iyer. 2018. Pathways to mental health services for young people: a systematic review. *Social psychiatry and psychiatric epidemiology* 53, 10 (2018), 1005–1038.
- [60] Martin N Marshall. 1996. Sampling for qualitative research. *Family practice* 13, 6 (1996), 522–526.
- [61] Suresh Bada Math, Guru S Gowda, Vinay Basavaraju, Narayana Manjunatha, Channaveerachari Naveen Kumar, Arun Enara, Mahesh Gowda, and Jagadisha Thirthalli. 2019. Cost estimation for the implementation of the Mental Healthcare Act 2017. *Indian Journal of Psychiatry* 61, Suppl 4 (2019), S650.
- [62] Hendrika Meischke, Brooke Ike, Ian Painter, Devora Chavez, Mei Po Yip, Steven M Bradley, and Shin-Ping Tu. 2015. Delivering 9-1-1 CPR instructions to limited English proficient callers: a simulation experiment. *Journal of immigrant and minority health* 17, 4 (2015), 1049–1054.
- [63] Sharan B Merriam and Robin S Grenier. 2019. *Qualitative research in practice: Examples for discussion and analysis*. John Wiley & Sons.
- [64] Brian L Mishara, François Chagnon, Marc Daigle, Bogdan Balan, Sylvaine Raymond, Isabelle Marcoux, Cécile Bardon, Julie K Campbell, and Alan Berman. 2007. Which helper behaviors and intervention styles are related to better short-term outcomes in telephone crisis intervention? Results from a silent monitoring study of calls to the US 1-800-SUICIDE network. *Suicide and Life-Threatening Behavior* 37, 3 (2007), 308–321.
- [65] Brian L Mishara and Marc Daigle. 2001. Helplines and crisis intervention services: Challenges for the future. *Suicide prevention: Resources for the millennium* (2001), 153–171.
- [66] Brian L Mishara and David N Weisstub. 2016. The legal status of suicide: A global review. *International journal of law and psychiatry* 44 (2016), 54–74.
- [67] Shubhangi Misra. 2020. Most suicide helplines are of little help as they don't work when people need them. *The Print* (2020).
- [68] Pranav D Modi, Rajavi Solanki, Tripti S Nagdev, Pallavi D Yadav, Nyayosh K Bharucha, Ajay Desai, Paresh Navalkar, Sunil B Kelgane, and Deepak Langade. 2018. Public awareness of the emergency medical services in Maharashtra, India: A questionnaire-based survey. *Cureus* 10, 9 (2018).
- [69] Parimala Moodley and Rachel E Perkins. 1991. Routes to psychiatric inpatient care in an Inner London Borough. *Social Psychiatry and Psychiatric Epidemiology* 26, 1 (1991), 47–51.
- [70] Craig Morgan, Rosemarie Mallett, Gerard Hutchinson, Hemant Bagalkote, Kevin Morgan, Paul Fearon, Paola Dazzan, Jane Boydell, Kwame McKenzie, Glynn Harrison, et al. 2005. Pathways to care and ethnicity. 1: Sample characteristics and compulsory admissions: Report from the AeSOP study. *The British journal of psychiatry* 186, 4 (2005), 281–289.
- [71] Craig Morgan, Rosemarie Mallett, Gerard Hutchinson, and Julian Leff. 2004. Negative pathways to psychiatric care and ethnicity: the bridge between social science and psychiatry. *Social science & medicine* 58, 4 (2004), 739–752.
- [72] Preeti Mudiar. 2020. Whither Humane-Computer Interaction? Adult and Child Value Conflicts in the Biometric Fingerprinting for Food. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.
- [73] Robin J Munro. 2002. Political psychiatry in post-Mao China and its origins in the cultural revolution. *Journal of the American Academy of Psychiatry and the Law Online* 30, 1 (2002), 97–106.
- [74] Vanessa Murphree, Bryan H Reber, and Frederick Blevens. 2009. Superhero, instructor, optimist: FEMA and the frames of disaster in Hurricanes Katrina and Rita. *Journal of Public Relations Research* 21, 3 (2009), 273–294.
- [75] James Y Nazroo, Kamaldeep S Bhui, and James Rhodes. 2020. Where next for understanding race/ethnic inequalities in severe mental illness? Structural, interpersonal and institutional racism. *Sociology of Health & Illness* 42, 2 (2020), 262–276.
- [76] Martha Neary and Stephen M Schueller. 2018. State of the field of mental health apps. *Cognitive and Behavioral Practice* 25, 4 (2018), 531–537.
- [77] H Blaise Nguendo-Yongsi. 2020. Access to health care in African cities: therapeutic pathways of city dwellers with mental health problems in Yaoundé-Cameroun. *Cities & Health* (2020), 1–10.
- [78] Mark Nichter. 1981. Idioms of distress: Alternatives in the expression of psychosocial distress: A case study from South India. *Culture, medicine and psychiatry* 5, 4 (1981), 379–408.
- [79] Devyani Nighoskar. 2018. 40 Unanswered Calls Show Us The Grim Reality Of Suicide Helplines In India. *Homegrown* (2018).
- [80] S Haque Nizamie and Nishant Goyal. 2010. History of psychiatry in India. *Indian journal of psychiatry* 52, Suppl1 (2010), S7.
- [81] World Health Organization et al. 2020. Coronavirus disease 2019 (COVID-19): situation report, 72. (2020).
- [82] Oladapo Oyebode, Felwah Alqahtani, and Rita Orji. 2020. Using machine learning and thematic analysis methods to evaluate mental health apps based on user reviews. *IEEE Access* 8 (2020), 111141–111158.
- [83] Deepak Padmanabhan and Savitha Sam Abraham. 2020. Representativity Fairness in Clustering. In *12th ACM Conference on Web Science* (Southampton, United Kingdom) (WebSci '20). Association for Computing Machinery, New York, NY, USA, 202–211. <https://doi.org/10.1145/3394231.3397910>
- [84] R Padmavati. 2005. Community mental health care in India. *International Review of Psychiatry* 17, 2 (2005), 103–107.
- [85] SR Parkar, VS Dawani, JS Apte, et al. 2001. History of psychiatry in India. *Journal of postgraduate Medicine* 47, 1 (2001), 73.
- [86] Sachin R Pendse, Naveena Karusala, Divya Siddarth, Pattie Gonsalves, Seema Mehrotra, John A Naslund, Mamta Sood, Neha Kumar, and Amit Sharma. 2019. Mental health in the global south: challenges and opportunities in HCI for development. In *Proceedings of the 2nd ACM SIGCAS Conference on Computing and Sustainable Societies*. 22–36.
- [87] Sachin R Pendse, Faisal M Lalani, Munmun De Choudhury, Amit Sharma, and Neha Kumar. 2020. "Like Shock Absorbers": Understanding the Human Infrastructures of Technology-Mediated Mental Health Support. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.
- [88] L.L. Piepzna-Samarasinha and E. Dixon. 2020. *Beyond Survival: Strategies and Stories from the Transformative Justice Movement*. AK Press. <https://books.google.com/books?id=t2iRDwAAQBAJ>
- [89] Leah Lakshmi Piepzna-Samarasinha. 2018. *Care work: Dreaming disability justice*. arsenal pulp press.
- [90] Ministry of Statistics Press Information Bureau, Government of India and Programme Implementation. 2020. First Advance Estimates of National Income, 2019–20. (2020).

- [91] R Raguram, Mitchell G Weiss, Harshad Keval, and SM Channabasavanna. 2001. Cultural dimensions of clinical depression in Bangalore, India. *Anthropology & Medicine* 8, 1 (2001), 31–46.
- [92] Pallavi Rao. 2018. Caste and the LoSHA discourse. *Communication Culture & Critique* 11, 3 (2018), 494–497.
- [93] Sharadamba Rao. 1966. Caste and mental disorders in Bihar. *American Journal of Psychiatry* 122, 9 (1966), 1045–1055.
- [94] John Rawls. 1971. *A theory of justice*. Harvard university press.
- [95] Patricia R Recupero. 2020. Social Media and the Internet. *The American Psychiatric Association Publishing Textbook of Suicide Risk Assessment and Management* (2020), 321.
- [96] Bernadette Richardson. 2012. Wireless emergency routing on empirical data: shaving time saves lives. *International Journal of Emergency Services* (2012).
- [97] Lionel P Robert, Casey Pierce, Liz Marquis, Sangmi Kim, and Rasha Alahmad. 2020. Designing fair AI for managing employees in organizations: a review, critique, and design agenda. *Human–Computer Interaction* (2020), 1–31.
- [98] Rebecca Rodrigues, Arlene G MacDougall, Guangyong Zou, Michael Leibermanbaum, Paul Kurdyak, Lihua Li, Salimah Z Shariff, and Kelly K Anderson. 2020. Risk of involuntary admission among first-generation ethnic minority groups with early psychosis: a retrospective cohort study using health administrative data. *Epidemiology and Psychiatric Sciences* 29 (2020).
- [99] Valerian Rodrigues. 2011. Justice as the Lens: Interrogating Rawls through Sen and Ambedkar. *Indian Journal of Human Development* 5, 1 (2011), 153–174.
- [100] Vanda Scott. 2001. Volunteer Action in Preventing Suicide. *Suicide prevention: Resources for the millennium* (2001), 265.
- [101] Irving Seidman. 2006. A guide for researchers in education and the social sciences.
- [102] Amartya Kumar Sen. 2009. *The idea of justice*. Harvard University Press.
- [103] S Sharma and RK Chadda. 1996. Recommendations of WHO workshop on “Future role on mental hospitals in mental health care.” In: Mental Hospitals in India: Current Status and Role in Mental Health Care. *Institute of Human Behaviour and Allied Sciences, Delhi* (1996).
- [104] Shridhar Sharma and LP Varma. 1984. History of mental hospitals in Indian sub-continent. *Indian journal of psychiatry* 26, 4 (1984), 295.
- [105] Rahul Shidhaye and Michelle Kermode. 2013. Stigma and discrimination as a barrier to mental health service utilization in India. *International health* 5, 1 (2013), 6–8.
- [106] Kevin Shinseki. 2018. System and method for providing chat-based customer callbacks. US Patent App. 16/058,044.
- [107] Amresh K Shrivastava, Megan E Johnston, Larry Stitt, Meghana Thakar, Gopa Sakel, Sunita Iyer, Nilesh Shah, and Yves Bureau. 2012. Reducing treatment delay for early intervention: evaluation of a community based crisis helpline. *Annals of general psychiatry* 11, 1 (2012), 20.
- [108] Sujata Sriram, Aparna Joshi, and Paras Sharma. 2016. Telephone Counselling in India: Lessons from iCALL. In *Counselling in India*. Springer, 201–216.
- [109] Amresh Srivastava, Megan Johnston, Larry Stitt, Meghana Thanksr, Sunita Iyer, et al. 2013. Suicidal ideation in callers to a crisis hotline in Mumbai. *Journal of Public Health* 5, 7 (2013), 305–308.
- [110] Alternative Story. 2020. (2020). <http://alternativestory.in/>
- [111] Your Story. 2019. These suicide prevention helplines ensure support is just a call away. (2019). <https://yourstory.com/herstory/2019/09/suicide-prevention-helplines-support-mental-health-women>
- [112] It's Ok To Talk. 2020. (2020). <http://itsoktotalk.in/>
- [113] Rangawsamy Thara, Ramachandran Padmavati, Jothy R Aynkran, and Sujit John. 2008. Community mental health in India: A rethink. *International Journal of Mental Health Systems* 2, 1 (2008), 1–7.
- [114] John Torous, Jennifer Nicholas, Mark E Larsen, Joseph Firth, and Helen Christensen. 2018. Clinical review of user engagement with mental health smartphone apps: evidence, theory and improvements. *Evidence-based mental health* 21, 3 (2018), 116–119.
- [115] Sarah J Tracy. 2002. When questioning turns to face threat: an interactional sensitivity in 911 call-taking. *Western Journal of Communication (includes Communication Reports)* 66, 2 (2002), 129–157.
- [116] Kai Wang, Deepthi S Varma, and Mattia Prosperi. 2018. A systematic review of the effectiveness of mobile apps for monitoring and management of mental health symptoms or disorders. *Journal of psychiatric research* 107 (2018), 73–78.
- [117] Sara M Williams, Laura M Frey, Dese'Rae L Stage, and Julie Cerel. 2018. Exploring lived experience in gender and sexual minority suicide attempt survivors. *American journal of orthopsychiatry* 88, 6 (2018), 691.
- [118] Iris Marion Young. 2011. *Justice and the Politics of Difference*. Princeton University Press.
- [119] Zahoor Zafrulla, John Etherton, and Thad Starner. 2008. TTY phone: direct, equal emergency access for the deaf. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, 277–278.
- [120] Howard Zehr. 2005. Changing Lenses: A New Focus for Crime and Justice, 3rd edn. Scottsdale.

Chapter 16

What do Prototypes Prototype?

Stephanie Houde and Charles Hill
Apple Computer, Inc.
Cupertino, California, USA

16.1 Introduction	367
16.2 The Problem with Prototypes.....	367
16.2.1 What is a Prototype?	368
16.2.2 Current Terminology.....	368
16.3 A Model of What Prototypes Prototype	369
16.3.1 Definitions.....	369
16.3.2 The Model	369
16.3.3 Three Prototypes of One System	369
16.4 Further Examples	371
16.4.1 Role Prototypes	372
16.4.2 Look and Feel Prototypes	374
16.4.3 Implementation Prototypes	376
16.4.4 Integration Prototypes	377
16.5 Summary	379
16.6 Acknowledgments	380
16.7 Prototype Credits	380
16.8 References	380

16.1 Introduction

Prototypes are widely recognized to be a core means of exploring and expressing designs for interactive computer artifacts. It is common practice to build prototypes in order to represent different states of an evolving design and to explore options. However, since interactive systems are complex, it may be difficult or impossible to create prototypes of a whole design in the formative stages of a project. Choosing the right kind of more focused prototype to build is an art in itself, and communicating its limited purposes to its various audiences is a critical aspect of its use.

The ways that we talk, and even think, about proto-

types can get in the way of their effective use. Current terminology for describing prototypes centers on attributes of prototypes themselves, such as what tool was used to create them, and how refined-looking or -behaving they are. Such terms can be distracting. Tools can be used in many different ways, and detail is not a sure indicator of completion.

We propose a change in the language used to talk about prototypes, to focus more attention on fundamental questions about the interactive system being designed: What role will the artifact play in a user's life? How should it look and feel? How should it be implemented? The goal of this chapter is to establish a model that describes any prototype in terms of the artifact being designed, rather than the prototype's incidental attributes. By focusing on the purpose of the prototype—that is, on *what it prototypes*—we can make better decisions about the kinds of prototypes to build. With a clear purpose for each prototype, we can better use prototypes to think and communicate about design.

In the first section we describe some current difficulties in communicating about prototypes: the complexity of interactive systems; issues of multi-disciplinary teamwork; and the audiences of prototypes. Next, we introduce the model and illustrate it with some initial examples of prototypes from real projects. In the following section we present several more examples to illustrate some further issues. We conclude the chapter with a summary of the main implications of the model for prototyping practice.

16.2 The Problem with Prototypes

Interactive computer systems are complex. Any artifact can have a rich variety of software, hardware, auditory, visual, and interactive features. For example, a personal digital assistant such as the Apple Newton has an operating system, a hard case with various ports, a graphical user interface and audio feedback. Users experience the combined effect of such interrelated features; and the task of designing—and prototyping—the user experience is therefore complex. Every aspect of the system must be designed (or inherited from a previous system), and many features need to be evaluated

in combination with others.

Prototypes provide the means for examining design problems and evaluating solutions. Selecting the focus of a prototype is the art of identifying the most important open design questions. If the artifact is to provide new functionality for users—and thus play a new *role* in their lives—the most important questions may concern exactly what that role should be and what features are needed to support it. If the role is well understood, but the goal of the artifact is to present its functionality in a novel way, then prototyping must focus on how the artifact will *look and feel*. If the artifact's functionality is to be based on a new technique, questions of how to *implement* the design may be the focus of prototyping efforts.

Once a prototype has been created, there are several distinct audiences that designers discuss prototypes with. These are: the intended *users* of the artifact being designed; their *design teams*; and the supporting *organizations* that they work within (Erickson, 1995). Designers evaluate their options with their own team by critiquing prototypes of alternate design directions. They show prototypes to users to get feedback on evolving designs. They show prototypes to their supporting organizations (such as project managers, business clients, or professors) to indicate progress and direction.

It is difficult for designers to communicate clearly about prototypes to such a broad audience. It is challenging to build prototypes which produce feedback from users on the most important design questions. Even communication among designers requires effort due to differing perspectives in a multi-disciplinary design team. Limited understanding of design practice on the part of supporting organizations makes it hard for designers to explain their prototypes to them. Finally, prototypes are not self-explanatory: looks can be deceiving. Clarifying what aspects of a prototype correspond to the eventual artifact—and what don't—is a key part of successful prototyping.

16.2.1 What is a Prototype?

Designing interactive systems demands collaboration between designers of many different disciplines (Kim, 1990). For example, a project might require the skills of a programmer, an interaction designer, an industrial designer, and a project manager. Even the term “prototype” is likely to be ambiguous on such a team. Everyone has a different expectation of what a prototype is. Industrial designers call a molded foam model a prototype. Interaction designers refer to a simulation of on-

screen appearance and behavior as a prototype. Programmers call a test program a prototype. A user studies expert may call a storyboard which shows a scenario of something being used, a prototype.

The organization supporting a design project may have an overly narrow expectation of what a prototype is. Shrake (1996) has shown that organizations develop their own “prototyping cultures” which may cause them to consider only certain kinds of prototypes to be valid. In some organizations, only prototypes which act as proof that an artifact can be produced are respected. In others, only highly detailed representations of look and feel are well understood.

Is a brick a prototype? The answer depends on how it is used. If it is used to represent the weight and scale of some future artifact, then it certainly is: it prototypes the weight and scale of the artifact. This example shows that prototypes are not necessarily self-explanatory. What is significant is not what media or tools were used to create them, but *how they are used by a designer* to explore or demonstrate some aspect of the future artifact..

16.2.2 Current Terminology

Current ways of talking about prototypes tend to focus on attributes of the prototype itself, such as which tool was used to create it (as in “C”, “Director™”, and “paper” prototypes); and on how finished-looking or -behaving a prototype is (as in “high-fidelity” and “low-fidelity” prototypes). Such characterizations can be misleading because the capabilities and possible uses of tools are often misunderstood and the significance of the level of finish is often unclear, particularly to non-designers.

Tools can be used in many different ways. Sometimes tools which have high-level scripting languages (like HyperCard™), rather than full programming languages (like C), are thought to be unsuitable for producing user-testable prototypes. However, Ehn and Kyng (1991) have shown that even prototypes made of cardboard are very useful for user testing. In the authors' experience, no one tool supports iterative design work in all of the important areas of investigation. To design well, designers must be willing to use different tools for different prototyping tasks; and to team up with other people with complementary skills.

Finished-looking (or -behaving) prototypes are often thought to indicate that the design they represent is near completion. Although this may sometimes be the case, a finished-looking prototype might be made early in the design process (e.g., a 3D concept model for use

in market research), and a rough one might be made later on (e.g., to emphasize overall structure rather than visual details in a user test). Two related terms are used in this context: "resolution" and "fidelity". We interpret resolution to mean "amount of detail", and fidelity to mean "closeness to the eventual design". It is important to recognize that the degree of visual and behavioral refinement of a prototype does not necessarily correspond to the solidity of the design, or to a particular stage in the process.

16.3 A Model of What Prototypes

Prototype

16.3.1 Definitions

Before proceeding, we define some important terms. We define *artifact* as the interactive system being designed. An artifact may be a commercially released product or any end-result of a design activity such as a concept system developed for research purposes. We define *prototype* as any representation of a design idea, regardless of medium. This includes a pre-existing object when used to answer a design question. We define *designer* as anyone who creates a prototype in order to design, regardless of job title.

16.3.2 The Model

The model shown in Figure 1 represents a three-dimensional space which corresponds to important aspects of the design of an interactive artifact. We define the dimensions of the model as *role*; *look and feel*; and *implementation*. Each dimension corresponds to a class of questions which are salient to the design of any interactive system. "Role" refers to questions about the function that an artifact serves in a user's life—the way in which it is useful to them. "Look and feel" denotes questions about the concrete sensory experience of using an artifact—what the user looks at, feels, and hears while using it. "Implementation" refers to questions about the techniques and components through which an artifact performs its function—the "nuts and bolts" of how it actually works. The triangle is drawn askew to emphasize that no one dimension is inherently more important than any other.

Goal of the Model: Given a design problem (of any scope or size), designers can use the model to separate design issues into three classes of questions which frequently demand different approaches to prototyping. Implementation usually requires a working system to be built; look and feel requires the concrete user expe-

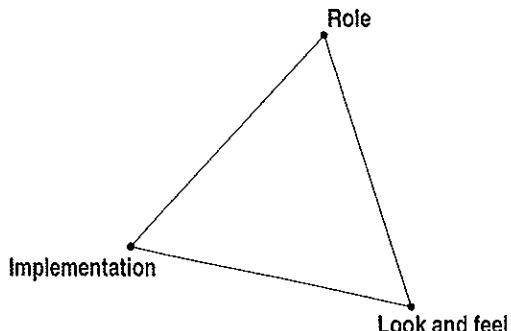


Figure 1. A model of what prototypes prototype.

rience to be simulated or actually created; role requires the context of the artifact's use to be established. Being explicit about what design questions must be answered is therefore an essential aid to deciding what kind of prototype to build. The model helps visualize the focus of exploration.

Markers: A prototype may explore questions or design options in one, two or all three dimensions of the model. In this chapter, several prototypes from real design projects are presented as examples. Their relationship to the model is represented by a marker on the triangle. This is a simple way to put the purpose of any prototype in context for the designer and their audiences. It gives a global sense of what the prototype is intended to explore; and equally important, what it does not explore.

It may be noted that the triangle is a relative and subjective representation. A location toward one corner of the triangle implies simply that *in the designer's own judgment*, more attention is given to the class of questions represented by that corner than to the other two.

16.3.3 Three Prototypes of One System

The model is best explained further through an example from a real project. The three prototypes shown in Examples 1-3 were created during the early stages of development of a 3D space-planning application (Houde, 1992).

The goal of the project was to design an example of a 3D application which would be accessible to a broad range of non-technical users. As such it was designed to work on a personal computer with an ordinary mouse. Many prototypes were created by different members of the multi-disciplinary design team during the project.

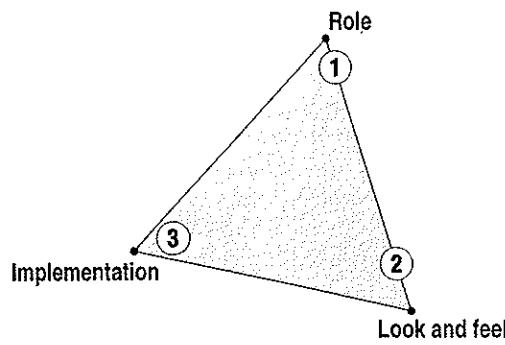
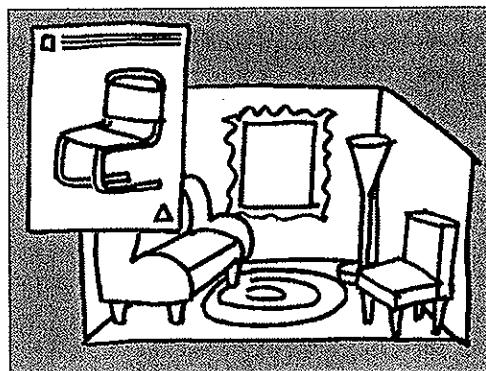


Figure 2. Relationship of three prototypes (Examples 1 - 3) to the model.

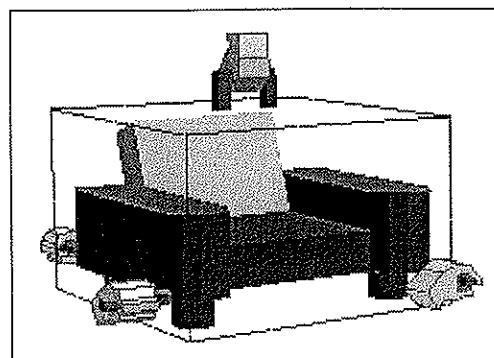


Example 1. Role prototype for 3D space-planning application [E1: Houde 1990].

The prototype shown in Example 1 was built to show how a user might select furniture from an on-line catalog and try it out in an approximation of their own room. It is an interactive slide show which the designer operated by clicking on key areas of the rough user interface. The idea that virtual space-planning would be a helpful task for non-technical users came from user studies. The purpose of the prototype was to quickly convey the proposed role of the artifact to the design team and members of the supporting organization.

Since the purpose of the prototype was primarily to explore and visualize an example of the role of the future artifact, its marker appears very near the role corner of the model in Figure 2. It is placed a little toward the look and feel corner because it also explored user interface elements in a very initial form.

One of the challenges of the project was to define an easy-to-use direct manipulation user interface for moving 3D objects with an ordinary 2D mouse cursor. User testing with a foam-core model showed that the

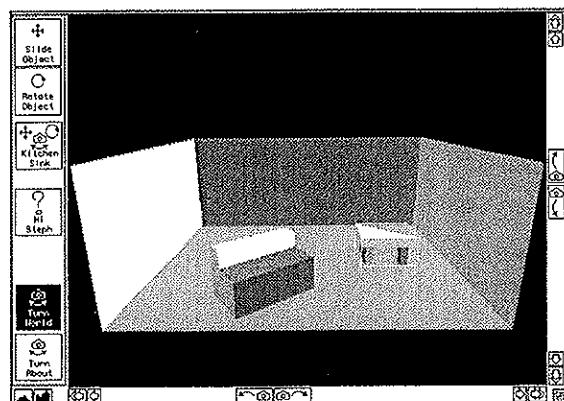


Example 2. Look and feel prototype for 3D space-planning application [E2: Houde 1990].

most important manipulations of a space-planning task were sliding, lifting, and turning furniture objects. Example 2 shows a picture of a prototype which was made to test a user interface featuring this constrained set of manipulations. Clicking once on the chair caused its bounding box to appear. This "handle box" offered hand-shaped controls for lifting and turning the box and object chair (as if the chair was frozen inside the box). Clicking and dragging anywhere on the box allowed the unit to slide on a 3D floor. The prototype was built using Macromedia Director (a high level animation and scripting tool). It was made to work only with the chair data shown: a set of images pre-drawn for many angles of rotation.

The purpose of the Example 2 prototype was to get feedback from users as quickly as possible as to whether the look and feel of the handle-box user interface was promising. Users of the prototype were given tasks which encouraged them to move the chair around a virtual room. Some exploration of role was supported by the fact that the object manipulated was a chair, and space-planning tasks were given during the test. Although the prototype was interactive, the programming that made it so did not seriously explore how a final artifact with this interface might be implemented. It was only done in service of the look and feel test. Since the designer primarily explored the look and feel of the user interface, this prototype's marker is placed very near the look and feel corner of the model in Figure 2.

A technical challenge of the project was figuring out how to render 3D graphics quickly enough on equipment that end-users might have. At the time, it was not clear how much real-time 3D interaction could be achieved on the Apple Macintosh™ II fx computer — the fastest Macintosh then available. Example 3



Example 3. Implementation prototypes for 3D space-planning application [E3: Chen 1990].

shows a prototype which was built primarily to explore rendering capability and performance. This was a working prototype in which multiple 3D objects could be manipulated as in Example 2, and the view of the room could be changed to any perspective. Example 3 was made in a programming environment that best supported the display of true 3D perspectives during manipulation. It was used by the design team to determine what complexity of 3D scenes was reasonable to design for. The user interface elements shown on the left side of the screen were made by the programmer to give himself controls for demonstrating the system: they were not made to explore the look and feel of the future artifact. Thus the primary purpose of the prototype was to explore how the artifact might be implemented. The marker for this example is placed near the implementation corner (Figure 2).

One might assume that the role prototype (Example 1) was developed first, then the look and feel prototype (Example 2), and finally the implementation prototype (Example 3): that is, in order of increasing detail and production difficulty. In fact, these three prototypes were developed almost in parallel. They were built by different design team members during the early stages of the project. No single prototype could have represented the design of the future artifact at that time. The evolving design was too fuzzy—existing mainly as a shared concept in the minds of the designers. There were also too many open and interdependent questions in every design dimension: role, look and feel, implementation.

Making separate prototypes enabled specific design questions to be addressed with as much clarity as possible. The solutions found became inputs to an integrated design. Answers to the rendering capability questions addressed by Example 3 informed the design

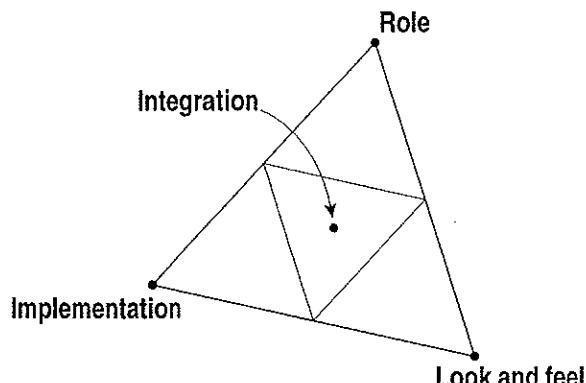


Figure 3. Four principal categories of prototypes on the model.

of the role that the artifact could play (guiding how many furniture objects of what complexity could be shown). It also provided guiding constraints for the direct manipulation user interface determining how much detail the handle forms could have). Similarly, issues of role addressed by Example 1 informed the implementation problem by constraining it: only a constrained set of manipulations was needed for a space-planning application. It also simplified the direct manipulation user interface by limiting the necessary actions, and therefore controls, which needed to be provided.

It was more efficient to wait on the results of independent investigations in the key areas of role, look and feel and implementation than to try to build a monolithic prototype that integrated all features from the start. After sufficient investigation in separate prototypes, the prototype in Example 3 began to evolve into an integrated prototype which could be described by a position at the center of our model. A version of the user interface developed in Example 2 was implemented in the prototype in Example 3. Results of other prototypes were also integrated. This enabled a more complete user test of features and user interface to take place.

This set of three prototypes from the same project shows how a design problem can be simultaneously approached from multiple points of view. Design questions of role, look and feel, and implementation were explored concurrently by the team with the three separate prototypes. The purpose of the model is to make it easier to develop and subsequently communicate about this kind of prototyping strategy.

16.4 Further Examples

In this section we present twelve more examples of

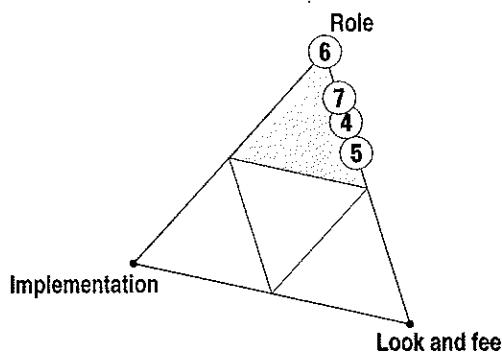


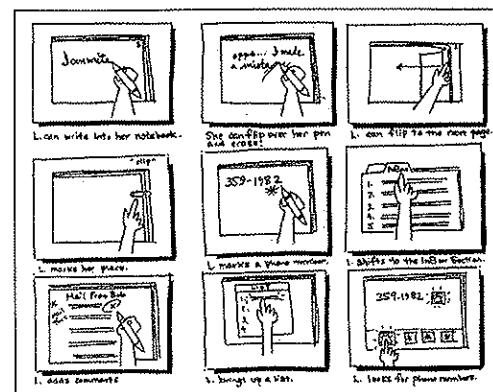
Figure 4. Relationship of role prototypes (Examples 4 - 7) to the model.

prototypes taken from real projects, and discuss them in terms of the model. Examples are divided into four categories which correspond to the four main regions of the model, as indicated in Figure 3. The first three categories correspond to prototypes with a strong bias toward one of the three corners: *role*, *look and feel*, and *implementation* prototypes, respectively. *Integration* prototypes occupy the middle of the model: they explore a balance of questions in all three dimensions.

16.4.1 Role Prototypes

Role prototypes are those which are built primarily to investigate questions of what an artifact could do for a user. They describe the functionality that a user might benefit from, with little attention to how the artifact would look and feel, or how it could be made to actually work. Designers find such prototypes useful to show their design teams what the target role of the artifact might be; to communicate that role to their supporting organization; and to evaluate the role in user studies.

A Portable Notebook Computer: The paper storyboard shown in Example 4 was an early prototype of a portable notebook computer for students which would accept both pen and finger input. The scenario shows a student making notes, annotating a paper, and marking pages for later review in a computer notebook. The designer presented the storyboard to her design team to focus discussion on the issues of what functionality the notebook should provide and how it might be controlled through pen and finger interaction. In terms of the model, this prototype primarily explored the role of the notebook by presenting a rough task scenario for it. A secondary consideration was a rough approximation of

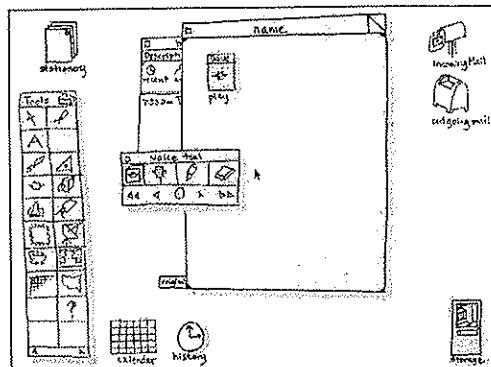


Example 4. Storyboard for a portable notebook computer [E4: Vertelney 1990].

the user interface. Its marker, shown in Figure 4, is therefore positioned near the role corner of the model and a little toward look and feel.

Storyboards like this one are considered to be effective design tools by many designers because they help focus design discussion on the role of an artifact very early on. However, giving them status as prototypes is not common because the medium is paper and thus seems very far from the medium of an interactive computer system. We consider this storyboard to be a prototype because it makes a concrete representation of a design idea and serves the purpose of asking and answering design questions. Of course, if the designer needed to evaluate a user's reaction to seeing the notebook or to using the pen-and-finger interaction, it would be necessary to build a prototype which supported direct interaction. However, it might be wasteful to do so before considering design options in the faster, lighter-weight medium of pencil and paper.

An Operating System User Interface: Example 5 shows a screen view of a prototype that was used to explore the design of a new operating system. The prototype was an interactive story: it could only be executed through a single, ordered, sequence of interactions. Clicking with a cursor on the mailbox picture opened a mail window; then clicking on the voice tool brought up a picture of some sound tools; and so on. To demonstrate the prototype, the designer sat in front of a computer and play-acted the role of a user opening her mail, replying to it, and so forth. The prototype was used in design team discussions and also demonstrated to project managers to explain the current design direction. According to the model, this prototype primarily explored the role that certain features of the operating



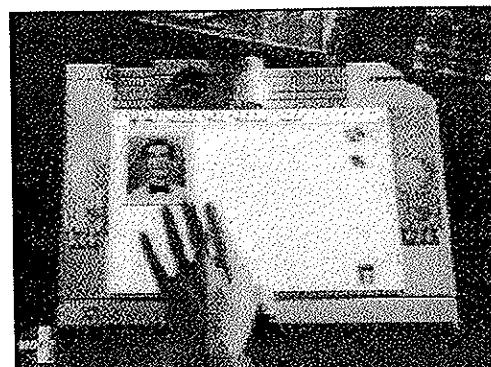
Example 5. Interactive story for an operating system interface [E5: Vertelney and Wong 1990].

system could play in a user's daily tasks. It was also used to outline very roughly how its features would be portrayed and how a user would interact with it. As in the previous example, the system's implementation was not explored. Its marker is shown in Figure 4.

To make the prototype, user interface elements were hand-drawn and scanned in. Transitions between steps in the scenario were made interactive in Macromedia Director. This kind of portrayal of on-screen interface elements as rough and hand-drawn was used in order to focus design discussion on the overall features of a design rather than on specific details of look and feel or implementation (Wong, 1992). Ironically, while the design team understood the meaning of the hand-drawn graphics, other members of the organization became enamored with the sketchy style to the extent that they considered using it in the final artifact. This result was entirely at odds with the original reasons for making a rough-looking prototype. This example shows how the effectiveness of some kinds of prototypes may be limited to a specific kind of audience.

The Knowledge Navigator: Example 6 shows a scene from Apple Computer's Knowledge Navigator™ video. The video tape tells a day-in-the-life story of a professor using a futuristic notebook computer (Dubberly and Mitch, 1987). An intelligent agent named "Phil" acts as his virtual personal assistant, finding information related to a lecture, reminding him of his mother's birthday, and connecting him with other professors via video-link. The professor interacts with Phil by talking, and Phil apparently recognizes everything said as well as a human assistant would.

Based on the model, the Knowledge Navigator is identified primarily as a prototype which describes the role that the notebook would play in such a user's life.

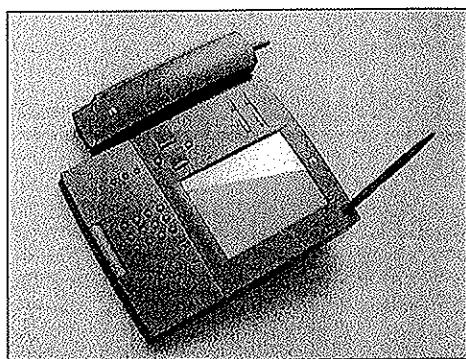


Example 6. Knowledge Navigator™ vision video for a future notebook computer [E6: Dubberly and Mitch 1987].

The story is told in great detail, and it is clear that many decisions were made about what to emphasize in the role. The video also shows specific details of appearance, interaction, and performance. However, they were not intended by the designers to be prototypes of look and feel. They were merely place-holders for the actual design work which would be necessary to make the product really work. Thus its marker goes directly on the role corner (Figure 4).

Thanks to the video's special effects, the scenario of the professor interacting with the notebook and his assistant looks like a demonstration of a real product. Why did Apple make a highly produced prototype when the previous examples show that a rapid paper storyboard or a sketchy interactive prototype were sufficient for designing a role and telling a usage story? The answer lies in the kind of audience. The tape was shown publicly and to Apple employees as a vision of the future of computing. Thus the audience of the Knowledge Navigator was very broad—including almost anyone in the world. Each of the two previous role design prototypes was shown to an audience which was well informed about the design project. A rough hand-drawn prototype would not have made the idea seem real to the broad audience the video addressed: high resolution was necessary to help people concretely visualize the design. Again, while team members learn to interpret abstract kinds of prototypes accurately, less expert audiences cannot normally be expected to understand such approximate representations.

The Integrated Communicator: Example 7 shows an appearance model of an Integrated Communicator created for customer research into alternate presentations of new technology (ID Magazine 1995). It was one of three presentations of possible mechanical configurations and interaction designs, each built to the same



Example 7. Appearance model for the integrated communicator [E7: Udagawa 1995].

high finish and accompanied by a video describing on-screen interactions. In the study, the value of each presentation was evaluated relative to the others, as perceived by study subjects during one-on-one interviews. The prototype was used to help subjects imagine such a product in the store and in their homes or offices, and thus to evaluate whether they would purchase such a product, how much they would expect it to cost, what features they would expect, etc.

The prototype primarily addresses the role of the product, by presenting carefully designed cues which imply a telephone-like role and look-and-feel. Figure 4 shows its marker near the role corner of the model. As with the Knowledge Navigator, the very high-resolution look and feel was a means of making the design as concrete as possible to a broad audience. In this case however it also enabled a basic interaction design strategy to be worked out and demonstrated. The prototype did not address implementation.

The key feature of this kind of prototype is that it is a concrete and direct representation, as visually finished as actual consumer products. These attributes encourage an uncoached person to directly relate the design to their own environment, and to the products they own or see in stores. High quality appearance models are costly to build. There are two common reasons for investing in one: to get a visceral response by making the design seem "real" to any audience (design team, organization, and potential users); and to verify the intended look and feel of the artifact before committing to production tooling. An interesting side-effect of this prototype was that its directness made it a powerful prop for promoting the project within the organization.

16.4.2 Look and Feel Prototypes

Look and feel prototypes are built primarily to explore

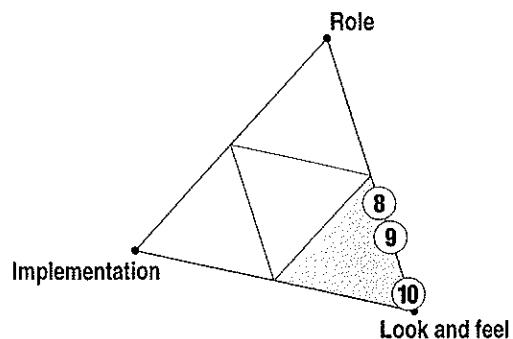


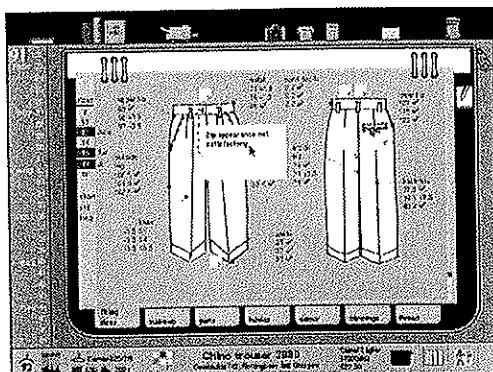
Figure 5. Relationships of the look and feel prototypes (Examples 8-10) to the model.

and demonstrate options for the concrete experience of an artifact. They simulate what it would be like to look at and interact with, without necessarily investigating the role it would play in the user's life or how it would be made to work. Designers make such prototypes to visualize different look and feel possibilities for themselves and their design teams. They ask users to interact with them to see how the look and feel could be improved. They also use them to give members of their supporting organization a concrete sense of what the future artifact will be like.

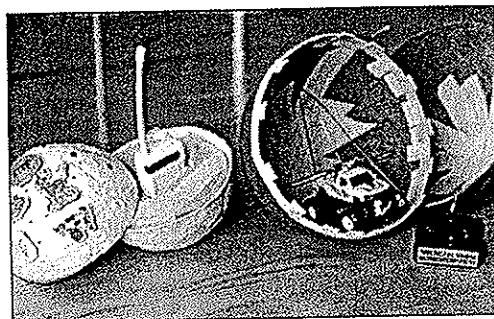
A Fashion Design Workspace: The prototype shown in Example 8 was developed to support research into collaboration tools for fashion designers (Hill et al, 1993; Scaife et al, 1994). A twenty-minute animation, it presented the concept design for a system for monitoring garment design work. It illustrated in considerable detail the translation of a proven paper-based procedure into a computer-based system with a visually rich, direct manipulation, user interface. The prototype's main purposes were to confirm to the design team that an engaging and effective look and feel could be designed for this application, and to convince managers of the possibilities of the project. It was presented to users purely for informal discussion.

This is an example of a look and feel prototype. The virtue of the prototype was that it enabled a novel user interface design to be developed without having first to implement complex underlying technologies. While the role was inherited from existing fashion design practice, the prototype also demonstrated new options offered by the new computer-based approach. Thus, Figure 5 shows its marker in the look and feel area of the model.

One issue with prototypes like this one is that inexperienced audiences tend to believe them to be more



Example 8. Animation of the look and feel of a fashion design workspace [E8: Hill 1992].

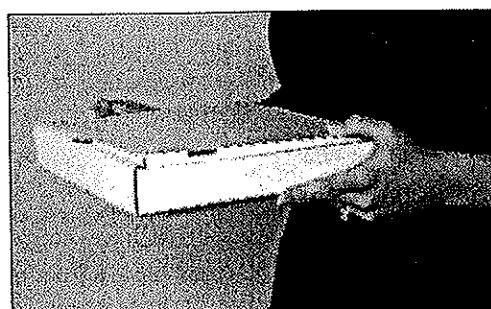


Example 9. Look and feel simulation prototypes for a child's toy [E9: Bellman et al, 1993].

functional than they are just by virtue of being shown on a computer screen. When this prototype was shown, the designers found they needed to take great care to explain that the design was not implemented.

A Learning Toy: The "GloBall" project was a concept for a children's toy: a ball that would interact with children who played with it. Two prototypes from the project are shown, disassembled, in Example 9. The design team wanted the ball to speak back to kids when they spoke to it, and to roll towards or away from them in reaction to their movements. The two prototypes were built to simulate these functions separately. The ball on the left had a walkie-talkie which was concealed in use. A hidden operator spoke into a linked walkie-talkie to simulate the ball's speech while a young child played with it. Similarly, the ball on the right had a radio-controlled car which was concealed in use. A hidden operator remotely controlled the car, thus causing the ball to roll around in response to the child's actions.

As indicated by the marker in Figure 5, both prototypes were used to explore the toy's look and feel from a child's viewpoint, and to a lesser extent to evaluate the



Example 10. Pizza-box prototype of an architect's computer [E10: Apple Design Project, 1992].

role that the toy would play. Neither seriously addressed implementation. The designers of these very efficient prototypes wanted to know how a child would respond to a toy that appeared to speak and move of its own free will. They managed to convincingly simulate novel and difficult-to-implement technologies such as speech and automation, for minimal cost and using readily available components. By using a "man behind the curtain" (or "Wizard of Oz") technique, the designers were able to present the prototypes directly to children and to directly evaluate their effect.

An Architect's Computer: This example concerned the design of a portable computer for architects who need to gather a lot of information during visits to building sites. One of the first questions the designers explored was what form would be appropriate for their users. Without much ado they weighted the pizza box shown in Example 10 to the expected weight of the computer, and gave it to an architect to carry on a site visit. They watched how he carried the box, what else he carried with him, and what tasks he needed to do during the visit. They saw that the rectilinear form and weight were too awkward, given the other materials he carried with him, and this simple insight led them to consider a softer form. As shown by its marker, this is an example of a rough look and feel prototype (Figure 5). Role was also explored in a minor way by seeing the context that the artifact would be used in.

The pizza box was a very efficient prototype. Spending virtually no time building it or considering options, the students got useful feedback on a basic design question—what physical form would be best for the user. From what they learned in their simple field test, they knew immediately that they should try to think beyond standard rectilinear notebook computer forms. They began to consider many different options

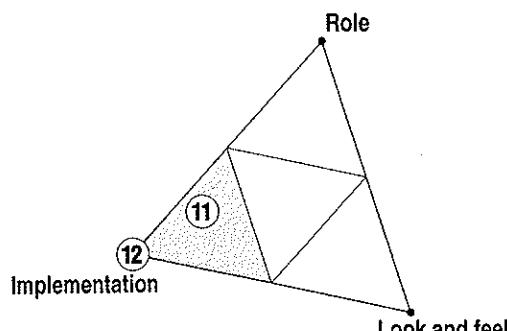


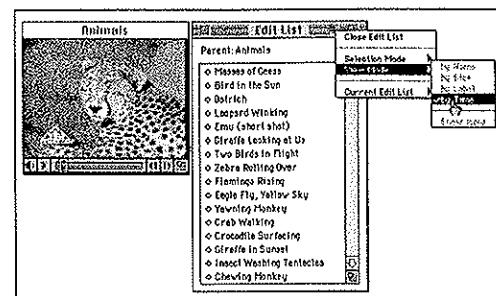
Figure 6. Relationships of implementation prototypes (Examples 11 and 12) to the model.

including designing the computer to feel more like a soft shoulder bag.

16.4.3 Implementation Prototypes

Some prototypes are built primarily to answer technical questions about how a future artifact might actually be made to work. They are used to discover methods by which adequate specifications for the final artifact can be achieved—without having to define its look and feel or the role it will play for a user. (Some specifications may be unstated, and may include externally imposed constraints, such as the need to reuse existing components or production machinery.) Designers make implementation prototypes as experiments for themselves and the design team, to demonstrate to their organization the technical feasibility of the artifact, and to get feedback from users on performance issues.

A Digital Movie Editor: Some years ago it was not clear how much interactivity could be added to digital movies playing on personal computers. Example 11 shows a picture of a prototype that was built to investigate solutions to this technical challenge. It was an application, written in the C programming language to run on an Apple Macintosh computer. It offered a variety of movie data-processing functionality such as controlling various attributes of movie play. The main goal of the prototype was to allow marking of points in a movie to which scripts (which added interactivity) would be attached. As indicated by the marker in Figure 6, this was primarily a carefully planned implementation prototype. Many options were evaluated about the best way to implement its functions. The role that the functions would play was less well defined. The visible look and feel of the prototype was largely incidental: it was created by the designer almost purely to demonstrate the available functionality, and was not intended to be used by others.



Example 11. Working prototype of a digital movie editor [E11: Degen, 1994].

This prototype received varying responses when demonstrated to a group of designers who were not members of the movie editor design team. When the audience understood that an implementation design was being demonstrated, discussion was focused productively. At other times it became focused on problems with the user interface, such as the multiple cascading menus, which were hard to control and visually confusing. In these cases, discussion was less productive: the incidental user interface got in the way of the intentional implementation.

The project leader shared some reflections after this somewhat frustrating experience. He said that part of his goal in pursuing a working prototype alone was to move the project through an organization that respected this kind of prototype more than “smoke and mirrors” prototypes—ones which only simulate functionality. He added that one problem might have been that the user interface was neither good enough nor bad enough to avoid misunderstandings. The edit list, which allowed points to be marked in movies, was a viable look and feel design; while the cascading menus were not. For the audience that the prototype was shown to, it might have been more effective to stress the fact that look and feel were not the focus of the prototype; and perhaps, time permitting, to have complemented this prototype with a separate look and feel prototype that explained their intentions in that dimension.

A Fluid Dynamics Simulation System: Example 12 shows a small part of the C++ program listing for a system for simulating gas flows and combustion in car engines, part of an engineering research project (Hill, 1993). One goal of this prototype was to demonstrate the feasibility of object-oriented programming using the C++ language in place of procedural programs written in the older FORTRAN language. Object-oriented programming can in theory lead to increased

```

IntList& IntList::operator=(const IntList& oldList)
{
register long n = oldList.size();
if (n != size) setSize(n);
register int* newPtr = &values[n];
register int* oldPtr = &oldList.values[n];
while (n--) --newPtr = --oldPtr;
return *this;
}

```

Example 12. C++ program sample from a fluid dynamics simulation [E12: Hill, 1993].

software reuse, better reliability and easier maintenance. Since an engine simulation may take a week to run on the fastest available computers and is extremely memory-intensive, it was important to show that the new approach did not incur excessive performance or memory overheads. The program listing shown was the implementation of the operation to copy one list of numbers to another. When tested, it was shown to be faster than the existing FORTRAN implementation. The prototype was built primarily for the design team's own use, and eventually used to create a deployable system. The marker in Figure 6 indicates that this prototype primarily explored implementation.

Other kinds of implementation prototypes include demonstrations of new algorithms (e.g., a graphical rendering technique or a new search technology), and trial conversions of existing programs to run in new environments (e.g., converting a program written in the C language to the Java language).

Implementation prototypes can be hard to build, and since they actually work, it is common for them to find their way directly into the final system. Two problems arise from this dynamic: firstly, programs developed mainly to demonstrate feasibility may turn out in the long term to be difficult to maintain and develop; and secondly, their temporary user interfaces may never be properly redesigned before the final system is released. For these reasons it is often desirable to treat even implementation prototypes as disposable, and to migrate successful implementation designs to a new integrated prototype as the project progresses.

16.4.4 Integration Prototypes

Integration prototypes are built to represent the complete user experience of an artifact. Such prototypes bring together the artifact's intended design in terms of role, look and feel, and implementation. Integrated

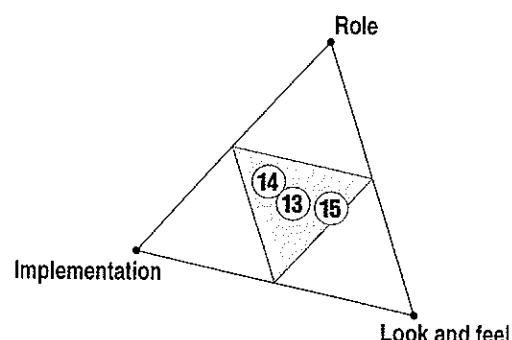
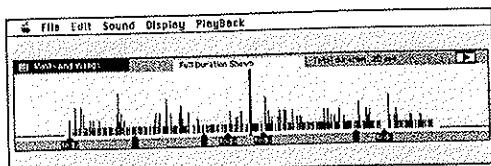


Figure 7. Relationships of integration prototypes (Examples 13 - 15) to the model.

prototypes help designers to balance and resolve constraints arising in different design dimensions; to verify that the design is complete and coherent; and to find synergy in the design of the integration itself. In some cases the integration design may become the unique innovation or feature of the final artifact. Since the user's experience of an artifact ultimately combines all three dimensions of the model, integration prototypes are most able to accurately simulate the final artifact. Since they may need to be as complex as the final artifact, they are the most difficult and time consuming kinds of prototypes to build. Designers make integration prototypes to understand the design as a whole, to show their organizations a close approximation to the final artifact, and to get feedback from users about the overall design.

The Sound Browser: The "SoundBrowser" prototype shown in Example 13 was built as part of a larger project which investigated uses of audio for personal computer users (Degen et al, 1992). The prototype was built in C to run on a Macintosh. It allowed a user to browse digital audio data recorded on a special personal tape recorder equipped with buttons for marking points in the audio. The picture shows the SoundBrowser's visual representation of the audio data, showing the markers below the sound display. A variety of functions were provided for reviewing sound, such as high-speed playback and playback of marked segments of audio.

This prototype earns a position right in the center of the model, as shown in Figure 7. All three dimensions of the model were explored and represented in the prototype. The role of the artifact was well thought-out, being driven initially by observations of what users currently do to mark and play back audio, and then by iteratively designed scenarios of how it might be done more efficiently if electronic marking and viewing functions were offered. The look and feel of the prototype went through many visual design iterations.



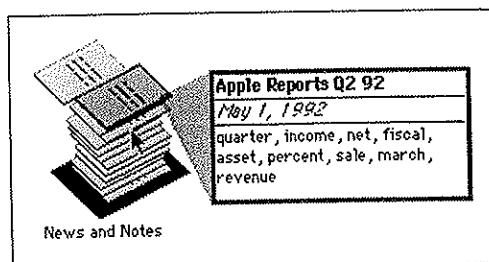
Example 13. Integrated prototype of a sound browser [E13: Degen, 1993].

The implementation was redesigned several times to meet the performance needs of the desired high-speed playback function.

When the SoundBrowser was near completion it was prepared for a user test. One of the features which the design team intended to evaluate was the visual representation of the sound in the main window. They wanted to show users several alternatives to understand their preferences. The programmer who built the SoundBrowser had developed most of the alternatives. In order to refine these and explore others, two other team members copied screen-shots from the tool into a pixel-painting application, where they experimented with modifications. This was a quick way to try out different visual options, in temporary isolation from other aspects of the artifact. It was far easier to do this in a visual design tool than by programming in C. When finished, the new options were programmed into the integrated prototype. This example shows the value of using different tools for different kinds of design exploration, and how even at the end of a project simple, low-fidelity prototypes might be built to solve specific problems.

The Pile Metaphor: The prototype shown in Example 14 was made as part of the development of the “pile” metaphor—a user interface element for casual organization of information (Mander et al, 1992, Rose et al, 1993). It represented the integration of designs developed in several other prototypes which independently explored the look and feel of piles, “content-aware” information retrieval, and the role that piles could play as a part of an operating system. In the pile metaphor, each electronic document was represented by a small icon or “proxy”, several of which were stacked to form a pile. The contents of the pile could be quickly reviewed by moving the arrow cursor over it. While the cursor was over a particular document, the “viewing cone” to the right displayed a short text summary of the document.

This prototype was shown to designers, project managers, and software developers as a proof of concept of the novel technology. The implementation de-

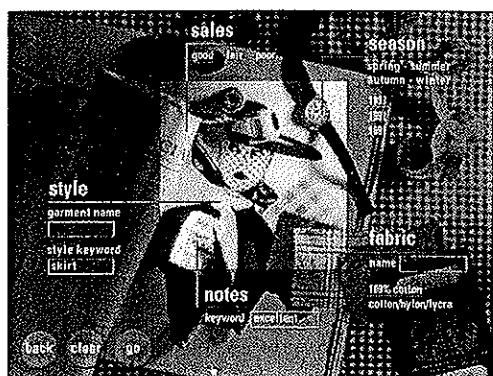


Example 14. Integration prototype of the “Pile” metaphor for information retrieval [E14: Rose, 1993].

sign in this prototype might have been achieved with virtually no user interface: just text input and output. However, since the prototype was to be shown to a broad audience, an integrated style of prototype was chosen, both to communicate the implementation point and to verify that the piles representation was practically feasible. It helped greatly that the artifact’s role and look and feel could be directly inherited from previous prototypes. Figure 7 shows its marker on the model.

A Garment History Browser: The prototype in Example 15 was a working system which enabled users to enter and retrieve snippets of information about garment designs via a visually rich user interface (Hill et al, 1993; Scaife et al, 1994). The picture shows the query tool which was designed to engage fashion designers and provide memorable visual cues. The prototype was designed for testing in three corporations with a limited set of users’ actual data, and presented to users in interviews. It was briefly demonstrated, then users were asked to try queries and enter remarks about design issues they were currently aware of.

This prototype was the end-result of a progression from an initial focus on role (represented by verbal usage scenarios), followed by rough look and feel prototypes and an initial implementation. Along the way various ideas were explored, refined or rejected. The working tool, built in Allegiant SuperCard™, required two months’ intensive work by two designers. In retrospect the designers had mixed feelings about it. It was highly motivating to users to be able to manipulate real user data through a novel user interface, and much was learned about the design. However, the designers also felt that they had had to invest a large amount of time in making the prototype, yet had only been able to support a very narrow role compared to the breadth shown in the animation shown in Example 8. Many broader design questions remained unanswered.



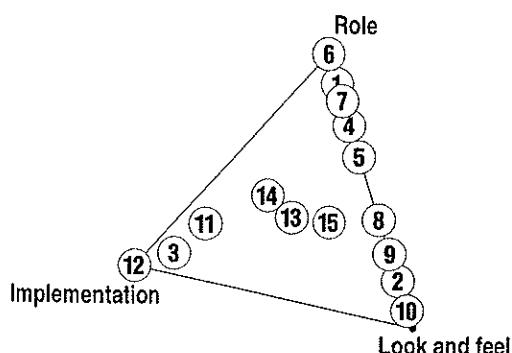
Example 15. Integrated prototype of a garment history browser [E15: Hill and Kamish, 1992].

16.5 Summary

In this chapter, we have proposed a change in the language used by designers to think and talk about prototypes of interactive artifacts. Much current terminology centers on attributes of prototypes themselves: the tools used to create them, or how refined-looking or -behaving they are. Yet tools can be used in many different ways, and resolution can be misleading. We have proposed a shift in attention to focus on questions about the design of the artifact itself: What role will it play in a user's life? How should it look and feel? How should it be implemented? The model that we have introduced can be used by designers to divide any design problem into these three classes of questions, each of which may benefit from a different approach to prototyping.

We have described a variety of prototypes from real projects, and have shown how the model can be used to communicate about their purposes. Several practical suggestions for designers have been raised by the examples:

- Define “prototype” broadly. Efficient prototypes produce answers to their designers’ most important questions in the least amount of time. Sometimes very simple representations make highly effective prototypes: e.g., the pizza-box prototype of an architect’s computer [Example 10] and the storyboard notebook [Example 1]. We define a prototype as any representation of a design idea—regardless of medium; and designers as the people who create them—regardless of their job titles.
- Build multiple prototypes. Since interactive artifacts can be very complex, it may be impossible to create an integrated prototype in the formative stages of a



1. 3D space-planning (role)
2. 3D space-planning (look and feel)
3. 3D space-planning (implementation)
4. Storyboard for portable notebook computer
5. Interactive story, operating system user interface
6. Vision video, notebook computer
7. Appearance model, integrated communicator
8. Animation, fashion design workspace
9. Look and feel simulation, child's toy
10. Pizza-box, architect's computer
11. Working prototype, digital movie editor
12. C++ program listing, fluid dynamics simulation
13. Integrated prototype, sound browser
14. Integrated prototype, pile metaphor
15. Integrated prototype, garment history browser

Figure 8. Relationships of all examples to the model.

project, as in the 3D space-planning example [Examples 1, 2, and 3]. Choosing the right focused prototypes to build is an art in itself. Be prepared to throw some prototypes away, and to use different tools for different kinds of prototypes.

- Know your audience. The necessary resolution and fidelity of a prototype may depend most on the nature of its audience. A rough role prototype such as the interactive storyboard [Example 4] may work well for a design team but not for members of the supporting organization. Broader audiences may require higher-resolution representations. Some organizations expect to see certain kinds of prototypes: implementation designs are often expected in engineering departments, while look-and-feel and role prototypes may rule in a visual design environment.
- Know your prototype; prepare your audience. Be clear about what design questions are being explored with a given prototype—and what are not. Communicating the specific purposes of a prototype

to its audience is a critical aspect of its use. It is up to the designer to prepare an audience for viewing a prototype. Prototypes themselves do not necessarily communicate their purpose. It is especially important to clarify what is and what is not addressed by a prototype when presenting it to any audience beyond the immediate design team.

By focusing on the purpose of the prototype—that is, on what it prototypes—we can make better decisions about the kinds of prototypes to build. With a clear purpose for each prototype, we can better use prototypes to think and communicate about design.

16.6 Acknowledgments

Special thanks are due to Thomas Erickson for guidance with this chapter, and to our many colleagues whose prototypes we have cited, for their comments on early drafts. We would also like to acknowledge S. Joy Mountford whose leadership of the Human Interface Group at Apple created an atmosphere in which creative prototyping could flourish. Finally, thanks to James Spohrer, Lori Leahy, Dan Russell, and Donald Norman at Apple Research Labs for supporting us in writing this chapter.

16.7 Prototype Credits

We credit here the principal designer and design team of each example prototype shown.

[E1] Stephanie Houde [E2] Stephanie Houde [E3] Michael Chen (1990) © Apple Computer, Inc. Project team: Penny Bauersfeld, Michael Chen, Lewis Knapp (project leader), Laurie Vertelney and Stephanie Houde.

[E4] Laurie Vertelney. (1990), © Apple Computer Inc. Project team: Michael Chen, Thomas Erickson, Frank Leahy, Laurie Vertelney (project leader).

[E5] Laurie Vertelney and Yin Yin Wong. (1990), © Apple Computer Inc. Project team: Richard Mander, Gitta Salomon (project leader), Ian Small, Laurie Vertelney, Yin Yin Wong.

[E6] Dubberly, H. and Mitch, D. (1987) © Apple Computer, Inc. The Knowledge Navigator (videotape.)

[E7] Masamichi Udagawa. (1995) © Apple Computer Inc. Project team: Charles Hill, Heiko Sacher, Nancy Silver, Masamichi Udagawa.

[E8] Charles Hill. (1992) © Royal College of Art, London. Design team: Gillian Crampton Smith, Eleanor Curtis, Charles Hill, Stephen Kamlish, (all of the RCA), Mike Scaife (Sussex University, UK), and Philip Joe (IDEO, London).

[E9] Tom Bellman, Byron Long, Abba Lustgarten. (1993) University of Toronto, 1993 Apple Design Project, © Apple Computer Inc.

[E10] 1992 Apple Design Project , © Apple Computer, Inc.

[E11] Leo Degen (1994) © Apple Computer Inc., Project team: Leo Degen, Stephanie Houde, Michael Mills (team leader), David Vronay.

[E12] Charles Hill (1993). Doctoral thesis project, Imperial College of Science, Technology and Medicine, London, UK. Project team: Charles Hill, Henry Weller.

[E13] Leo Degen (1993) © Apple Computer Inc., Project team: Leo Degen, Richard Mander, Gitta Salomon (team leader), Yin Yin Wong.

[E14] Daniel Rose. (1993). © Apple Computer, Inc. Project team: Penny Bauersfeld, Leo Degen, Stephanie Houde, Richard Mander, Ian Small, Gitta Salomon (team leader), Yin Yin Wong

[E15] Charles Hill and Stephen Kamlish. (1992) © Royal College of Art, London. Design team: Gillian Crampton Smith, Eleanor Curtis, Charles Hill, Stephen Kamlish, (all of the RCA), and Mike Scaife (Sussex University, UK).

16.8 References

Degen, L., Mander, R., Salomon, G. (1992). Working with Audio: Integrating Personal Tape Recorders and Desktop Computers. *Human Factors in Computing Systems: CHI'92 Conference Proceedings*. New York: ACM, pp. 413-418.

Dubberly, H. and Mitch, D. (1987). *The Knowledge Navigator*. Apple Computer, Inc. videotape.

Ehn, P., Kyng, M., (1991) Cardboard Computers: Mocking-it-up or Hands-on the Future., Design at Work: *Cooperative Design of Computer Systems* (ed. Greenbaum, J., and Kyng, M.). Hillsdale, NJ: Lawrence Erlbaum. pp. 169-195.

Erickson, T., (1995) Notes on Design Practice: Stories and Prototypes as Catalysts for Communication.

- "Envisioning Technology: The Scenario as a Framework for the System Development Life Cycle" (ed. Carroll, J.). Addison-Wesley.
- Hill, C. (1993) *Software Design for Interactive Engineering Simulation*. Doctoral Thesis. Imperial College of Science, Technology and Medicine, University of London.
- Hill, C., Crampton Smith, G., Curtis, E., Kamlish, S., (1993) Designing a Visual Database for Fashion Designers. *Human Factors in Computing Systems: INTERCHI'93 Adjunct Proceedings*. New York, ACM, pp. 49-50.
- Houde, S., (1992). Iterative Design of and Interface for Easy 3-D Direct Manipulation. *Human Factors in Computing Systems: CHI'92 Conference Proceedings*. New York: ACM, pp. 135-142.
- I.D.Magazine, (1995) Apple's Shared Conceptual Model, *The International Design Magazine: 41's Annual Design Review, July-August 1995*. USA. pp. 206-207
- Kim, S. (1990). Interdisciplinary Collaboration. *The Art of Human Computer Interface Design* (ed. B. Laurel). Reading, MA: Addison-Wesley. pp.31-44.
- Mander, R., Salomon, G., Wong, Y.Y. (1992). A 'Pile' Metaphor for Supporting Casual Organization of Information. *Human Factors in Computing Systems: CHI'92 Conference Proceedings*. New York: ACM, pp. 627-634.
- Rose, D.E., Mander, R., Oren, T., Ponceleón, D.B., Salomon, G., Wong, Y. (1993). Content Awareness in a File System Interface: Implementing the 'Pile' Metaphor for Organizing Information. *Research and Development in Information Retrieval: SIGIR Conference Proceedings*. Pittsburgh, PA: ACM, pp. 260-269.
- Scaife, M., Curtis, E., Hill, C. (1994) Interdisciplinary Collaboration: a Case Study of Software Development for Fashion Designers. *Interacting with Computers*, Vol 6. no.4, pp. 395-410
- Schrage, M. (1996). Cultures of Prototyping. *Bringing Design to Software* (ed. T. Winograd). USA: ACM Press. pp.191-205.
- Wong, Y.Y. (1992). Rough and ready prototypes: Lessons from graphic design. *Human Factors in Computing Systems: CHI'92 Conference, Posters and Short-Talks*, New York: ACM, pp.83-84.

What Do Hackathons Do? Understanding Participation in Hackathons Through Program Theory Analysis

Jeanette Falk
jfo@edu.au.dk
Aarhus University
Aarhus, Denmark

Gopinaath Kannabiran
goka@itu.dk
Department of Computer Science, IT
University of Copenhagen
Copenhagen, Denmark

Nicolai Brodersen Hansen
nbha@cs.aau.dk
Department of Computer Science,
Aalborg University
Aalborg, Denmark

ABSTRACT

Hackathons are increasingly embraced across diverse sectors as a way of democratizing the design of technology. Several attempts have been made to redefine the format and desired end goal of hackathons in recent years thereby warranting closer methodological scrutiny. In this paper, we apply program theory to analyze the processes and effects of 16 hackathon case studies through published research literature. Building upon existing research on hackathons, our work offers a critical perspective examining the methodological validity of hackathons and exemplifies how specific processes for organizing hackathons are modified for different purposes. Our main contribution is a program theory analysis of hackathon formats that provides an exploration and juxtaposition of 16 case studies in terms of causal relations between the input, process and the effects of hackathons. Our cataloguing of examples can serve as an inspirational planning resource for future organizers of hackathons.

CCS CONCEPTS

- Human-centered computing → Interaction design process and methods.

KEYWORDS

hackathons, participatory design, research methods, program theory

ACM Reference Format:

Jeanette Falk, Gopinaath Kannabiran, and Nicolai Brodersen Hansen. 2021. What Do Hackathons Do? Understanding Participation in Hackathons Through Program Theory Analysis. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan*. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3411764.3445198>

1 INTRODUCTION

Hackathons are increasingly used for different purposes in various contexts involving technology. Even though hackathons were originally intended for and attended by software developers, in recent years hackathon events have been conducted across domains by

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI '21, May 8–13, 2021, Yokohama, Japan

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-8096-6/21/05...\$15.00
<https://doi.org/10.1145/3411764.3445198>

engaging with people who do not necessarily have a software development background. As these events are increasingly embraced across diverse sectors, some hackathon organizers have sought to tailor the format in relation to their needs and context. For instance, corporate driven technology innovation has embraced hackathons for “[...] speeding-up the early phases of their innovation process up to the development and evaluation of prototypes”[16]. In contrast, hackathons in scientific communities foreground mentoring and “[...] allow community newcomers to develop technical artifacts that are perceived as useful by the community that organizes the event” [35]. In another instance, hackathon based events have been used “[...] for creating the circumstances under which grassroots innovation might flourish” [47]. These three examples show us that hackathons are organized for various contexts through modified formats with different end goals. In such instances, hackathons undergo a simultaneous change in format and desired end goal i.e. how to conduct a hackathon and towards what end are repeatedly redefined. Studying such attempts frames our research questions: What do hackathons do? How are hackathons adapted to specific contexts and challenges? How might we conceptualize hackathons as a specific form of collaboration? Despite hackathons gaining more traction in HCI research, relatively little is understood about their methodological validity and thereby warranting closer scrutiny. In this work, we will study how hackathon processes are formatted and organized in relation to the desired end goal of a project in order to aid future research through the use of hackathons.

In general, hackathons have been commended as a way to empower a broad and diverse audience through participation in various phases of designing technology. Hackathons are often framed as a way of democratizing the development of technology through participation. Though hackathons are increasingly embraced across sectors, their formats and use are not completely problem free. Criticism against hackathons include the lack of sustainable outputs in the form of prototypes as well as a lack of diversity in participation among others. Further, scholars have also argued that “participation’ is not a sufficient condition for changing power relations: forms of participation exist and presently thrive that do not question, but further, dominant power patterns around the development of IT” [2]. Exploring these tensions, we position our work as contributing to the ongoing critical dialog about hackathons in HCI [1, 12, 25, 46] by studying the processes and effects of hackathons. Falk Olesen and Halskov performed a literature review of 381 research publications and point out that “[...] the relation between how hackathons are organized and the outcomes is a valuable objective for future research” [14]. Researchers interested in utilizing hackathons as a form of participation through designing for and

with specific groups can benefit from a systematic analysis of the means and intended outcomes of hackathons in their situated contexts. Building upon existing research, we offer a complimentary critical perspective by studying the processes and effects of 16 hackathon case studies selected from published research literature. While Falk Olesen and Halskov's work [14] provides an exhaustive literature review of hackathons, our work will engage with 16 case studies by applying program theory (PT) to exemplify how specific components for organizing hackathons are modified for different purposes. Our methodological approach of analysing 16 cases contrasts with and builds upon existing research on hackathons in three ways.

- First, a case study analysis “[...] focuses on understanding the dynamics present within single settings” [13] whereas a literature review is concerned with generalizing a wide range of relevant works. As Yin has argued, a case study approach is suitable for examining contextual conditions relevant to the phenomenon under study [53]. Put another way, a literature review aims to describe *what* has been said whereas a case study analysis aims to understand *how* specific cases operate in their contexts.
- Second, a case study analysis deliberately introduces a theoretical lens [31] – in our work, PT – whereas a literature review attempts to build a theoretical perspective that emerges from existing works.
- Finally, our work aims to produce a critical juxtaposition of 16 hackathon cases selected for closer analysis whereas a literature review aims to produce a systematic and exhaustive summary of existing works. We offer PT diagrams that juxtapose the modified processes and effects of 16 hackathon cases to facilitate cross case analysis. By doing so, our work critically evaluates the methodological validity of hackathons in selected cases as well as across cases through PT. Unlike a literature review, a case study analysis does not aim for an exhaustive survey.

PT comes out of the field of evaluation and offers “[...] a way to make explicit the assumed causalities of projects and programs” [7] by focusing on the “[...] underlying assumptions about how programs are expected to work” [42]. Since our work seeks to understand the processes and effects of hackathons, we apply PT to analyze 16 case studies based on existing research literature. PT has recently been used by HCI researchers to “[...] make evaluations more precise and increase learning, since making processes explicit enables investigation into why a project or program did or did not work” [22]. Hackathons are increasingly embraced in HCI both as part of conferences such as CHI4Good but also as ways to generate research data and explore new research endeavours [32, 40, 45]. We offer our work as a methodological inquiry about hackathons by pursuing our research question – what do hackathons do in their contexts? Through a PT analysis of 16 case studies, we provide a cross case analysis where “[...] mobilization of case knowledge occurs when researchers accumulate case knowledge, compare and contrast cases, and in doing so, produce new knowledge” [27]. Based on our case study analysis, we discuss the formats of hackathons and offer suggestions for researchers interested in organizing hackathons. In line with Falk Olesen and

Halskov's work [14], we are interested in developing a more systematic approach for organizing and using hackathons as part of research.

Our contributions include: a catalog of examples on hackathon formats modified for specific purposes, see Table 2 for an overview. Each example is analyzed according to PT and provides structuring about how the hackathon was organized as well as explicating assumptions of a range of adaptions to the formats. We furthermore identify three main motivations for modifying the hackathon formats, see Tables 3, 4, and 5 for overviews of PT diagrams for each motivation. Based on our cross case analysis of 16 case studies, we provide suggestions for researchers who are interested in using modified hackathons for further knowledge production.

The rest of the paper is structured as follows: First, we introduce related work on hackathon formats, and the theoretical framework of PT. Secondly, we describe the method for selecting papers for the PT analysis, and the analysis procedure. Thirdly, we analyze the selected papers based on PT. The analysis section is structured into three categories: Participation, Sustainable Outcomes, and Learning, which represents the main motivations for modifying hackathon formats. Finally, we discuss retrospective insights on analyzing modified hackathon formats using PT. We also discuss some prospective considerations for organizers and HCI researchers.

1.1 Limitations

We selected our cases for modified hackathon formats using the ACM Digital Library for our search query. By doing this, we have restricted our scope of analysis to research literature that is published through ACM. This precludes any research about hackathon formats that may have been published outside of ACM in other related fields such as healthcare, where there is an increased use of hackathons. Therefore, our analysis and claims must be understood within the limited scope of our research and not as universal claims about hackathons. As mentioned earlier, our aim with this paper is not to conduct an exhaustive analysis of all the ways hackathon formats have been adapted to specific formats. Rather, we want to highlight cases which may be particularly relevant for the HCI community and provide closer analysis of the selected cases through PT. Further, we have analyzed the 16 cases solely based on published research literature which may not include all the relevant details of a project.

2 RELATED WORK

In this section, we introduce related HCI research to our research interest in hackathon formats and how they may be organized for specific purposes. Then, we present the theoretical framework of PT.

2.1 Conceptualizing Hackathon Formats

There is no strict definition of what exactly constitutes a hackathon as such. Although hackathon formats are varied, there is a commonality of elements which often occur in hackathons and the formats often share a very similar structure [11]. Research literature describes hackathons in very similar ways, such as: “Hackathons are events where people who are not normally collocated converge for a few days to write code together.” [50]; “Hackathons bring

together participants from different backgrounds to address a problem through the creation of a computational intervention over the course of a day or two.” [40]; “In general, hackathons are time-bounded events, typically of two to five days, during which people gather together and form teams, each of which attempts to complete a project of interest to them.” [38]; On a general level, we subscribe to these ways of describing hackathon formats, which we frame as typically intensive and accelerated design processes where participants explore and design ideas for design cases or challenges during a short time-frame.

In recent years, there have been some contributions which systematically conceptualize the elements which generally constitute hackathon formats in order to support the design and organization of hackathons in different contexts. Based on empirical studies of 10 hackathons and a review of published literature, Pe-Than et. al. contributes with a discussion on how hackathons can be organised by listing design choices which organisers may consider in order to meet particular purposes [38]. Their contribution focus on corporate hackathons, and while there may be relevance for non-corporate hackathons as well, the discussed design decisions and the purposes which the design decisions support are specifically oriented towards creating commercial advantage in a corporate context. In this paper, we explore how hackathon formats have been modified and organized for several different contexts.

In extension of Pe-Than et. al.’s study [38], Nolte et. al. have recently developed a kit for supporting the organisation of hackathons. They outline 12 general decisions which a hackathon organizer should consider [36]. We share a research interest with Nolte et. al. who focus on hackathons which aim: “[...] to foster a specific goal for a specific audience in a specific domain.” [36]. We diverge from Nolte et. al.’s focus on how to organize hackathons on a couple of perspectives: We conduct a retrospective case study on 16 particular hackathons, and scaffold our analysis using PT. We thereby focus on the experiences and insights from organizing hackathon formats for specific purposes, and seek to explicate as well as juxtapose the assumptions underlying how the specific hackathons were organized. We further argue that PT can be used prospectively for organizers as a tool to structure how to evaluate specific attempts at organizing hackathon formats.

Porras et. al. seek to: “[...] explore the various approaches to implementing hackathons and their outcomes for different stakeholders [...]” [39]. Based on the authors’ own experiences with hackathons, a review of literature as well as empirical data from students and industry participants, the authors sum up their insights from organising hackathons over the years in a taxonomy of seven different hackathon formats, mostly organized in educational contexts. Though we are also interested in the various approaches of implementing hackathons, we do not delimit our study to educational contexts.

Taylor and Clarke frame hackathons as participatory design activities which designers and HCI researchers can learn from [46]. Taylor and Clarke engage in better understanding: “[...] how hackathons are being appropriated for different audiences and what we might learn from these events to inform the configuration of our own participatory activities.” [46]. In order to accomplish this, they participated in and studied six hackathons which specifically invited non-technical participants. The hackathons which they

study can be seen as being organized towards accommodating this non-technical audience. Therefore, there are overlaps between our study and theirs: Similar to Taylor and Clarke, we also study for instance the hackathon Self-Harmony by Birbeck et.al. [4]. However, we do not only look at hackathon formats which have been modified for a non-technical audience, though these are part of our study, but we are interested in cases which in general pursue tailoring hackathon formats for a specific purpose.

Motivated by an aim to systematically conceptualize the phenomenon of hackathons, Kollwitz and Dinter developed a taxonomy based on a systematic literature review of 189 publications [28]. The taxonomy cover ten dimensions which each contain a spectrum of characteristics of the different ways in which a hackathon format is typically organised. While Kollwitz and Dinter seek to conceptualize the characteristics of hackathon formats on different dimensions, we seek to explore the hackathon formats which are explicitly organized for a particular purpose. Such a modified hackathon format may share characteristics as described in Kollwitz and Dinter’s taxonomy, however, they introduce and explore often novel ways of organizing hackathons tailored for specific purposes.

Another literature review which we draw upon, is the aforementioned work of Falk Olesen and Halskov [14]. Based on a comprehensive literature review of how researchers conduct research with and on hackathons, they identify three overarching motivations for organizing hackathons in this context: Learning, structuring processes, and enabling participation. While these overlap with our findings, they elaborate the motivations for structuring processes and enabling participation with several sub-categories based on multiple examples from the review. As mentioned before, our study contrasts with theirs by focusing on a few select cases in order to provide detailed analysis and comparison.

Based on five hackathons which de Götzen et. al. themselves participated in or organized, the authors explore how the hackathons: “[...] were prepared and run and how their different formats affected both the selection of participants and the outcomes of the hackathons.” [11]. We share the research interest in how hackathon formats can take on different forms, but we differ from their focus on hackathons in service design.

We see the contributions in this section as important steps towards more systematic research on hackathons, where insights on different kinds of formats and how to organize these formats are synthesized and structured, which furthermore enable comparison of the formats. The systematization of hackathon organizers’ accumulated experience can inform how other organizers may choose to organize as well as evaluate their particular hackathon format. We believe the latter in particular can contribute with intermediate level knowledge contributions [23], which can further support HCI researchers using modified hackathon formats as part of their research methods. In our paper we contribute to this line of hackathon research.

2.2 Program Theory

PT is originally intended for the field of evaluation, and is not a theory as such, but an approach which can be used for understanding processes and making clear how to evaluate a program [3, 22]. The aim is to make evaluations of programs more precise,

Table 1: A recreation of the Program Theory representations used by Hansen et al. [22].

Input	Process		Output	Effect	
	Mechanism	Activity		Outcome	Impact
Tangible and intangible resources needed	The fundamental principle generating an effect	The medium or way in which the mechanism is brought into action	Immediate tangible and intangible product emerging from the process	Short- and medium term consequences derived from the output	Long term effects of the program or project

and thereby contribute to knowledge on how different programs work. Recently, PT have been applied in the field of participatory design [22], playful and participatory city-making [44] and has also been discussed as a way to approach and contribute to research on hackathons [14]. Hansen et al. [22] build on PT to conceptualize how participatory design can be viewed as a specific set of programs that uses certain mechanisms (such as for instance mutual learning or shared reflections) and aims at specific effects such as quality of life, workplace democracy or emancipation [22]. They visualize the relationships between inputs, processes and effects in a tabular format (see Table 1) that we also draws on in the analysis. PT operates with the notions of inputs, process and effects [22]:

- *Inputs* to a program are the provided resources which are needed to initiate and complete a program. This could for instance be provided design materials or tools in a hackathon, or a design challenge.
- The *process* of a program consists of the *activities* which are conducted during the program by the participants. These activities support *mechanisms*, which are the general principles that generate effects. An example from a hackathon setting could be workshops during the hackathon to improve participants' proficiency in prototyping, which may lead to better quality prototypes.
- The program can generate *effects*, which are distinguished into *outputs*, *outcomes*, and *impacts*: Outputs are the immediate tangible and intangible products of the process, for example prototypes developed during a hackathon. Outcomes are short and midterm effects, such as consequences, benefits or drawbacks of the program. In the context of hackathons, this could be a startup company which started as an idea in a hackathon. Impacts are long term effects which are often “[...] achieved in conjunction with other programs.” [22]. Impacts are, however, difficult to determine whether it was caused by a certain program. A long term impact of hackathons could for example be the production of “[...] new imaginaries of place, belonging, and hope in the performance of citizenship.” [12].

In terms of inputs, process, and effects, PT can be used to make explicit “[...] the underlying assumptions about how programs are expected to work.” [42].

In the participatory design context, Bossen, Dindler and Iversen argue that more explicit, systematic evaluations of participatory design, such as in applying PT, can: “[...] enhance accountability, learning and knowledge building.” [6]. Similarly, we argue that this can be obtained by applying PT for the study of hackathon formats. PT can be used to structure and articulating evaluations of these modified hackathon formats, and explicate assumptions of why a format was designed in the way it was. This, in turn, make way for

comparing hackathon formats, and can contribute towards more formative research on hackathon formats [17].

3 METHODOLOGY

In this section we describe our method for selecting and analyzing the papers using PT.

3.1 Selecting the Papers

We searched for “hackathon” in the abstract, title and author keywords of publications, divided by the boolean operator OR, in the ACM Digital Library (ACM DL), published in the year range of 01/01/2010 to 23/07/2020, the last date representing when we conducted the latest search. There were no hits before the year 2010 with this search query. All items which were not either a journal paper, conference paper, book or book chapter were filtered from the sample. This resulted in 78 publications from the ACM DL.

Before reading the 78 abstracts, the three authors discussed and agreed on a set of guidelines for filtering the sample of abstracts. We based the development of the guidelines on our interest in finding papers which intentionally and clearly adapt a hackathon format for a specific purpose. The following guidelines supported us in filtering out publications which:

- Only made cursory mentions of hackathons, such as when a hackathon was used to produce an outcome, with the outcome being the focus of the publication.
- Only used a hackathon as a method to evaluate a prototype. In these cases the hackathon format was not modified for a specific purpose as such, but would often follow a “typical” hackathon format.
- Reported on only the participant perspective of hackathons, such as motivations and experiences of participating in a hackathon.
- Described the organization of a hackathon, but did not organize the format for a specific purpose.
- Did not revolve around the organization and facilitation of a hackathon.

Following the above guidelines, two authors read the 78 abstracts of the papers. During this reading, the two authors compared and discussed insights in order to align the filtering process. This resulted in 41 papers chosen for further reading.

We iterated our reading process, and read the full paper closely. During the reading of the 41 papers, we filtered out papers which for instance included too little detail about the organization of the hackathon, or turned out to not live up to our focus on papers that described how a hackathon format was organized for a specific purpose.

In some cases, the paper authors themselves clearly indicated that the conducted hackathon was modified, by writing for example: “However, we also recognised inclusivity challenges with the hackathon format and made significant changes to cater to a wider audience.” [48]; or “[...] a novel initiative that adapts the conventional hackathon [...]” [52]; or “[...] we present a reimagining of the hackathon model [...]” [24]. In these cases, we included the papers.

In one case, two papers referred to the same hackathon [47, 48], so we included the more recently published paper [48] which also described the hackathon in enough details for us to analyze the format according to PT. This resulted in 16 papers chosen for further analysis, see Table 2.

As described in the introduction, we approach the 16 papers as case studies, in the sense that we focus on understanding the dynamics within the single settings. In the next subsection we describe our analysis procedure.

3.2 Analysis Procedure

Our analysis of the 16 papers was driven by PT. We coordinated the analysis of the 16 papers by identifying the following focus points for each paper:

- The domain which the hackathon was conducted in, for example education, research, or civic engagement.
- The main approach used for organizing a hackathon format for a specific situation, for example through choice of material.
- The reason for why the format was modified.
- The input elements for the hackathon format.
- The process elements for the hackathon format, including activities and mechanisms.
- The output elements for the hackathon format, both intended and described.

For each paper, we filled in these focus points in a collaborative online spreadsheet. These focus points helped us in terms of identifying overarching domains and research contexts in which the hackathons were organised. Two authors both analysed all 16 papers using the focus points. During the analysis of the papers, the authors kept an ongoing discussion of insights.

In order to identify potential patterns across the 16 papers, we used an online collaborative whiteboard tool to visually group the papers into categories. We identified three purposes for which the hackathons were modified: Participation, Sustainable Outcomes, and Learning.

These three categories are not mutually exclusive, but reflect the main purpose for which the hackathon was modified. A paper could still reflect intentions for for example broadening participation, while the main purpose for the hackathon adaptions were based on supporting learning.

4 ANALYSIS

In a PT perspective, the intended effect of a program is pursued through planned inputs, and mechanisms facilitated by activities during the process. In the following subsections, we describe the motivations and considerations for how the hackathon formats were tailored for a specific purpose, and how the tailored hackathon format was evaluated, if it was evaluated. This analysis forms the

point of departure for the discussion where we discuss the patterns found in the analysis. To make it easier to follow how PT informed our analysis we use the schema from Table 1 to visualise the relationships between inputs, processes and effects of the many different hackathons.

4.1 Participation

We identified nine papers which aimed at broadening participation through tailoring hackathon formats [4, 24, 29, 30, 33, 35, 41, 48, 49]. Two of the papers used a co-design approach with end-users [4, 24], where hackathon participants were either end-users themselves or worked closely together with end-users to co-design prototypes, while Richard et.al. [41] are more focused on empowering the hackathon participants by inviting them to partake in the development of technology. Taylor et.al. [48] also focus on empowering the participants of the hackathons, but do this by specifically engaging the participants with their own neighbourhood community. The case of Konopacki et. al.’s hackathon did not involve technology development, but used the hackathon format to structure the participants’ engagement with a platform for engaging citizens in lawmaking [29]. In Kopeć et. al.’s hackathon end-users, older adults, were invited to partake in the hackathon with developers [30]. Similarly in Narain et. al.’s hackathon, end-users were invited to participate, and were additionally carefully matched with developers [33]. The participation of newcomers in a scientific hackathon was the focus in Nolte et. al.’s case [35], where mentors supported the novice participants during the hackathon. A focus in Thomer et. al.’s hackathon for co-designing with end-users was the choice of design material as a way to support non-designers [49].

In the following sections, we analyse these cases in detail according to PT. Table 3 provide an overview of the salient elements of the cases in terms of input, process and effects.

4.1.1 Co-Designing with end-users. The hackathons in two of the papers concern designing and developing technology both for and with those affected by the theme of the hackathon: self-harm [4], and breastfeeding [24]. The authors of the two papers [4, 24] share a motivation for raising awareness about their hackathon themes by conducting the hackathons. The way in which the authors modify the two different hackathons in order to achieve this motivation differ from each other.

Input: The organisers of both hackathons undertook *sense-checking activities*: In Birbeck et. al.’s hackathon [4], it was important to consult frequently with local mental health organizations to ensure the creation of a sensitive and tactful space. For Hope et. al.’s hackathon [24], it was important for the organisers to work with their own biases, since the majority of the organizers were white, college-educated, cis-gendered, and heterosexual, and the target audience of the hackathon were mothers and parents who face the most challenges regarding breastfeeding: “[...] mothers of color, low-wage workers, and/or LGBTQ+ parents.” [24].

The two hackathons diverge in how *participants were recruited*. In Birbeck et. al.’s hackathon [4], participants were recruited through channels such as mailing lists. Their recruitment operated after a first-come-first-served principle. The hackathon attracted a wide range of participants, including a target audience of end-users with lived experience of self-harm [4]. In an earlier version of

Table 2: An overview of the selected 16 papers which organize hackathon formats for specific purposes. They are categorized after three main motivations for modifying the formats: Participation, Sustainable Outcomes, and Learning.

Category	Paper
Participation	Hackathons as Participatory Design: Iterating Feminist Utopias [24] Self Harmony: Rethinking Hackathons to Design and Critique Digital Technologies for Those Affected by Self-Harm [4] StitchFest: Diversifying a College Hackathon to Broaden Participation and Perceptions in Computing[41] Strategies for Engaging Communities in Creating Physical Civic Technologies[48] MUDAMOS: a civil society initiative on collaborative lawmaking in Brazil[29] Older adults and hackathons: a qualitative study[30] ATHack: Co-Design and Education in Assistive Technology Development [33] How to Support Newcomers in Scientific Hackathons - An Action Research Study on Expert Mentoring [35] Co-designing Scientific Software: Hackathons for Participatory Interface Design [49]
Sustainable outcome	Mapathons and Hackathons to Crowdsource the Generation and Usage of Geographic Data [20] Lab Hackathons to Overcome Laboratory Equipment Shortages in Africa: Opportunities and Challenges [52] Post-Hackathon Learning Circles: Supporting Lean Startup Development [9]
Learning	Short datathon for the interdisciplinary development of data analysis and visualization skills [43] Experience Report: Thinkathon – Countering an “I Got It Working” Mentality with Pencil-and-Paper Exercises [10] Developing Course Projects in a Hack Day: An Experience Report [19] The community garden hack: participatory experiments in facilitating primary school teacher’s appropriation of technology [26]

Table 3: A comparative overview of selected elements which contributed to modifying hackathon formats for specific purposes in terms of broadening participation.

	Input	Mechanism	Process	Output	Effect Outcome	Impact
Co-Designing with end-users [4, 24]	Sense-checking activities Prioritised participant recruitment Welcoming venue space Inspiration sources	Centering marginalized voices of participants Microphone	Volunteers supporting participants with tools and materials Mentors	Designs presented at science fair Seven designs	Raising awareness of theme Press and media Collaborations Community Policy Personal	Reimagining breastfeeding in the US Making hackathons more inclusive and accessible
Broadening Participation in Computation [41]	Targeted recruitment Theme: Wear & Care Venue Design materials: LilyPad Arduino Hack boxes Recommendations [34]	Cross-team social interactions and collaboration Helping novice participants with technology development	Sharing content from hack boxes Mentors	Wearable designs	Changed perceptions on coding and computation	Broadening female participation in computation
Empowering Communities [48]	Participant recruitment through local events Makers recruited through makerspace	Allowing families and parents to participate	Three events of six hours Slow introduction of technology	Three prototypes	Relationships between residents and makers Creating excitement around technology possibilities Confidence with technology	Empowerment of citizens
Engaging Citizens in Lawmaking [29]	Mobile app Mudamos	Collaboration Diversity of participants strengthen bill proposal	Multistakeholder panel Fishbowl conversion Draft bill creation in three steps	Test of bill Publication on Mudamos	Engaging citizens in lawmaking	
Inviting End-User Participants [30]	Targeted invitation of older adult participants Meeting prior to hackathon between organizers and participants			19 designs	Reflections on challenges regarding participation of end-users	
Matching End-Users and Participants [33]	Recruitment of co-designers (end-users) Proposal of projects by co-designers Presentation of projects at social event Project preferences submitted Documentation prize			Assistive technology prototypes	Knowledge about accessible product design	
Mentoring Newcomers [35]	Mentors Presentation on problem space by mentor	Mentors as experts assisting newcomers	Mentors helping with problem scoping and technical support	3 projects		
Design Materials for Co-Designing with End-Users [49]	Materials for paper prototyping Workshop on prototyping Workbooks	Capturing work practices	Filling out workbooks	Completed workbooks Interface designs	Sharing with broader community	

their hackathon format, Hope et. al. used a similar participant recruitment as Birbeck et. al, however for the hackathon described in [24], they made a prioritised recruitment of participants in order to reach their target audience, by using channels such as: “[...] personal outreach through our partners, specific recruitment at Historically Black Colleges & Universities (HBCUs), and outreach to community organizations.” [24].

Both hackathons reflected on the *venue* and how the location could support specific needs during the hackathons. In the case of Birbeck et. al.’s hackathon [4], they needed a space for participants who would need to withdraw if they for example became upset because of the particular hackathon theme (self-harm). In the case of Hope et. al.’s hackathon, the venue space was designed in order to: “[...] make the space more welcoming to a wider spectrum of people.” [24]. This included spaces which imitated living rooms, an art exhibition, a zine library, and a “Baby Village” [24].

Inspiration sources in different forms were also offered to the participants of both hackathons. Birbeck et. al. offered inspiration packs with three challenges (in the form of questions) to guide the initial idea generation [4], while Hope et. al. created and distributed a book with narratives and innovations from parents to the hackathon participants[24]. Additionally, quotes from these books were hung up on the walls of the venue.

Process: During the process, Birbeck et. al strived for a continuous sense-checking of the participants’ ideas, and did this by having mentors at the hackathon who could provide critique and ensure that participants engaged sensitively with the topic. These mentors were both people with lived experience of self-harm, as well as professionals working with mental health. Additionally, participants were asked to document their process, so that this process documentation could be used to later engage stakeholders in critical dialogue about the design outputs of the hackathon and whether the outputs engage with the topic of self-harm in a sensitive way [4]. During the process, Hope et. al. focused on centering the marginalized voices of the participants, and had volunteers to help the participants with the provided tools and materials of the hackathon. The organisers also provided a microphone, which the participants at all times could use during the hackathons, in order to make requests and: “[...] leverage expertise in the room and provide a way for people to take ownership over the process [...]” [24].

Effect: The output of Birbeck et. al.’s hackathon [4] was seven different designs developed by the participants, as well as the post-hackathon stakeholder discussion about these designs. The authors reflect that, while it is important to engage end-users in the design process, it is equally important to ensure objective and expert perspectives on the outcomes of the design processes, as these stakeholder discussions: “[...] can be seen as integral to how these design outcomes could be successfully integrated into self-harm care pathways.”[4]. In order to build relationships and meaningful dialogue around the output, in form of the designs, Hope et. al. held a science fair. They furthermore report five different kinds of impact of the hackathons: Press and media, collaborations, community, policy, and personal. A research related output are five design principles for organizing an event for centering marginalized voices.

Though the two papers both strive for similar outcomes of their programs, facilitating co-design with end-users, the planned inputs and activities facilitating these outcomes are quite different. This emphasises how the *theory* of the program, (hackathons can be settings for co-designing with end-users, and furthermore be a meaningful setting where respectively sensitive topics can be explored and marginalized voices are centered), can potentially be pursued through a range of different activities.

4.1.2 Broadening Participation in Computation. Like the two above-mentioned cases, the format of Richard et. al.’s hackathon, called StitchFest, was carefully and comprehensively organized to “[...] broaden not only participation in computing by reaching larger numbers of female participants but also perceptions of computing through the event composition and theme.” [41].

Input: In order to accomplish this, Richard et.al. designed StitchFest based on a set of recommendations released from the National Center for Women and Information Technology [34]. Additionally, the authors drew on research that indicate that the use of design materials, themes and spaces can influence participation and perception [21]. As the main design material or platform for the StitchFest, the authors chose the LilyPad Arduino, which combines computation with crafting and sewing. The authors continued the theme of crafting, by developing a design challenge called “Wear and Care”. Another input to the hackathon, was the distribution of “hack boxes” for each team. The hack boxes each contained the same primary components, but had different sensors and actuators in them. The reason behind the different content of the boxes, was based on the authors’ assumption that this would prompt the participants to interact with each other, exchanging both materials and experiences with using the materials.

Process: During the hackathon, the hack boxes contributed to an activity of sharing materials between participants, and in that way support cross-team and collaborative interactions, rather than isolated teamwork. Mentors were also present during the hackathon, assisting the participants with the technology development.

Effect: The immediate outputs of the hackathon were wearable designs and longer term outcomes were how the participants, who were interviewed a few weeks after the hackathon, reflected on how the hackathon broadened their perceptions regarding coding and computation [41].

However, despite the hackathon being carefully designed and followed recommendations and research meant to support a broader participation in computation, the authors reflect: “[...] a list of recommendations alone is no guarantee for success.” [41]. In conducting a second iteration of the hackathon the authors reflected: “While the second iteration of StitchFest benefitted from our experiences in setting up the Spring 2014 [...] we found that this was not sufficient to bring in larger number of female participants as we did in the Fall.” [41]. Despite some seemingly positive effects from the hackathon, this reflection points towards the need for further exploring and evaluating the assumed relations between planned inputs and intended effects of hackathons. However, the StitchFest suggest some interesting future endeavours for research on hackathon organization modified for broadening participation, in terms of: targeted

recruitment, thematic framing, space arrangements, kinds of materials and material distribution [41].

4.1.3 Empowering Communities. The hackathon of Taylor et. al. was organized to accommodate the needs of families and parents, who did not have much experience with technology development [48]. This was done through rethinking the program of the hackathon, introducing technology slowly, and facilitate the creation of inventor kits after the hackathon, meant to enable the residents to build new civic technologies. They motivate the choice of conducting a hackathon for this purpose because of hackathon formats: “[...] intensive nature and their focus on active participation and creative thinking, as opposed to more discursive consultations.” [48]. However, at the same time, the authors are aware of inclusivity challenges with hackathon formats, and therefore modified their hackathon to: “[...] cater to a wider audience.” [48].

Input: Neighbourhood residents were recruited through local events in the city where the hackathon would take place. Makers were also recruited through the city’s makerspace and personal connections.

Process: One significant activity of Taylor et.al.’s hackathon was dividing the hackathon program into a series of three events of six hours, as they assumed this would allow parents and families to participate, contrary to a single weekend-long event. The authors introduced technology slowly: At the first event, participants mainly worked with craft materials which were easy to work with for all the participants. The second event: “[...] followed a more conventional hackathon model,” [48], where teams built prototypes using both craft materials and simple electronics. The last event made use of the digital fabrication equipment at the city’s makerspace in order to finalise prototypes.

Effect: The immediate output of the hackathon was three prototypes. The intended outcome of the three events was building relationships between residents and makers, which could potentially facilitate future learning and collaboration between the two groups. An outcome of the hackathon was the residents becoming more comfortable with technology, and being able to see technology as something they could employ for local needs in the neighbourhood. An assumption was that the residents would become comfortable with technology, because participating in developing technology would provide them insights into this process and demystify it. Following the hackathon, the authors held several informal and formal gatherings and events in order to among other things maintain a sense of community for the residents.

The authors reflect that it is important that the participants’ feeling of ownership of being able to develop civic technology for local needs extends beyond the prototypes to the process itself. For that purpose, the authors suggest making activities more open, and supporting lightweight, drop-in engagement [48].

4.1.4 Engaging Citizens in Lawmaking. Konopacki et.al.’s hackathon is an example on a hackathon format which was specifically tailored towards engaging citizens in drafting bills.

Input: Their case took point of departure in an app, Mudamos, developed to enable citizens to: “[...] participate in lawmaking and express their support for draft bill proposals introduced in the legislature using electronic signatures.” [29]. To start supporting citizens in drafting bills through the app (the intended outcome of

the hackathon) the authors organized a hackathon to: “[...] draft bills collectively addressing a single issue and within a timeframe.” [29].

Process: During the hackathon the authors organized a range of activities to support this outcome, including a multistakeholder panel, and a “fishbowl conversation” with the audience. Next, the hackathon was structured into three steps: “[...] address the draft bill main objective, write down general definitions and, finally, draft bill devices.” [29]. Participants worked in groups during this process, and were supported by mentors. These activities were driven by an assumption that the more diverse sectors who participate during the hackathon, the stronger the bill proposal.

Effect: An immediate output of this modified hackathon is a test of the bill and its publication on the Mudamos app. The test is presentations of the proposals to consultants in order to improve the bill. Finally, the bill is published on Mudamos, and signatures for the bill can be collected through the app.

The Mudamos hackathon is the only example in our sample where the output does not as such involve technology development. Instead, a specific technology, the app, is the input and platform around which the process revolves. The hackathon format is then organized around activities which drives discussion and content creation for the technology.

4.1.5 Inviting End-User Participants. Kopéc et. al. organized a hackathon where older adults were invited to participate in teams of young programmers [30]. They argue that: “[...] it is important to optimize the process of developing solutions that would suit the needs of this demographic [the older adults].” [30].

Input: The main way in which the hackathon format was organized for this purpose, focus on the targeted invitation of older adult participants. Prior to the event the hackathon organizers met with the older adult participants in order to explain the hackathon to them, as well as discuss any concerns from the older adult participants regarding participation. The main concern from the older adult participants was about their own technical aptitude, whereto the organisers assured them that the their technical skills were less important than: “[...] their willingness to share their personal insights and experience with the programmers.” [30].

Effect: In evaluating the case of Kopéc et. al.’s hackathon, the authors observed a number of issues regarding the participation dynamic between the young programmers and older adult participants. In one scenario the programmers did not make contact with the older adult participants, whose insights were oftentimes disregarded as non-representative for the older adult population. In a second scenario, the older adults were only consulted sporadically by the young programmers, and here the older adults’ experiences were oftentimes also deemed as non-representative. In a third scenario, the older adult participants played an active role in the development teams, like the organisers had envisioned.

Kopéc et. al. reflect that it is not enough to only invite a target group to participate in a hackathon, as it cannot be assumed that all participants, whether they are part of the target group or the developers, can participate on equal terms [30]. End-users needs to not only be invited as an input to the hackathon, but to be actively supported and perceived as participants with valid insights during

the process of a hackathon, if full collaboration and participatory design is envisioned.

4.1.6 Matching End-Users and Participants. An intended outcome of the hackathon organised by Narain et. al. [33] is educating student participants about designing accessible products. The outcome is facilitated by conducting a hackathon which include a rather elaborate matching process between the student participants and the co-designers, who live with a disability.

Input: We see the elaborate matching process as an important input for this case. The co-designers were recruited from interest lists at rehabilitation facilities, schools, and disability service centers at universities. Then, the co-designers proposed potential projects, and were encouraged to consider among other things the feasibility and scope of the projects. The projects were then presented to all the hackathon attendees at a social dinner event two weeks prior to the “build day” of the hackathon. During the event, the co-designers and participants mingled and discussed ideas. After the social event, the participants submitted project preferences, and matched co-designers and participants were encouraged to get to know each other before the build day. Another interesting input to Narain et. al.’s hackathons is the inclusion of a documentation prize, which can potentially facilitate outputs which are more easily shared. The authors report that, as a result of including the prize: “[...] many teams created high-quality instructions and drawings to help others understand their work.” [33]. However, the authors note that their hackathon format still lack: “[...] support for the continuation of projects after the event.” [33].

Effect: An immediate output of the hackathon was assistive technology prototypes.

Despite the careful matching process, some end-user participants did not feel that their perspectives were perceived as valid, quite similar to Kopéc et. al.’s older adult participants [30]. One participant reflected: “Two of my friends participated in the event and had very negative experiences because the hackers did not take any of their input. It wasn’t collaborative at all. They were not open to the co-designer’s suggestions and went off and did things without including the co-designer at all. In fact, they did the exact opposite of what the co-designer wanted and something he knew would not work.” [33]. This further suggest that it is not enough to only invite participants, such as end-users, there needs to be activities during the process which can support mechanisms of collaboration between participants and end-users.

4.1.7 Mentoring Newcomers. Nolte et al. [35] presents a study on how adding *mentors* might be needed in scientific hackathons, that aim to support scientific communities by developing technical artefacts useful to the research being conducted. In short, while a person might be competent to participate in a hackathon in general, the scientific hackathons can often be excluding to newcomers because of the large amount of lingo, complex challenges and especially existing technical infrastructure and requirements.

Input: Nolte et al. [35] ran a study where they introduced mentors, that is domain experts already working on research problems and using tools related to them, to help the hackathon newcomer groups out before, during and after the hackathon. The mentors for example would provide a short webinar on the problem space before the hackathon and be available during the hackathon to help

with problem scoping and support on technical issues related to the domain.

Process: Nolte et al.[35] highlights how the mentors were not meant to be stakeholders owning the problem, but rather experts assisting, and in the case where these two roles were confused this seemed detrimental to the progress of the hackathon. This is ascribed to one of the strengths of hackathons, in allowing hackathon teams to take ownership over their work and so, care should be taken to ensure that this is allowed while mentors act in a more outside and neutral role.

Effect: As an output of the hackathon, three projects were developed.

In a PT perspective the mentors contributed both with inputs (in the sense of domain knowledge, webinars, logins and tools), and participated in activities (by helping scope, discuss and assist during the hackathon) and also interestingly with sustaining the effort beyond the scope of the actual hackathon. As the members were community members, they were well-positioned to help take a next step with the projects.

4.1.8 Design Materials for Co-Designing with End-Users. The choice of design material can in the hackathon of Thomer et. al. be seen as a way to modify the hackathon format to be more welcoming for non-designers. The hackathon of Thomer et.al. is conducted in the research field of taxonomy, which often operate with niche UX/UI needs and underdeveloped GUI’s [49]. These GUI’s can act as barriers for entry of researchers with less computational backgrounds [49]. In order to develop better GUI’s which accommodate for those niche needs, Thomer et. al. organised a three day hackathon around co-designing interfaces by inviting people with expertise in either taxonomy, information science or software development.

Input: The participants were provided with materials for creating paper prototypes and digital wireframe tools, and were introduced to prototyping methods. Thomer et. al. reported that despite the participants not having prior design experience, the participants easily created interface designs with the provided materials for paper prototyping. The participants, however, struggled with the tools provided for creating digital wireframes. Another input were the workbooks provided to the participants [49].

Process: These workbooks were intended to support the capture of the participants’ work practices as well as storing the interface designs during the hackathon process. The completed workbooks would then serve as an output, which could: “[...] be shared with the broader development community, and act as a blueprint for future software design.” [49].

Effect: The motivation behind including the workbooks, creating more shareable outputs, is similar to the motivation for the input of a documentation prize in Narain et. al.’s hackathon [33].

We see this design material choice as a way of tailoring the hackathon format to invite non-designers to partake in design, as materials for especially paper prototyping does not require a certain expertise, such as a design or computation background, in order to be used. This is similar to the case of the above-mentioned Stitchfest hackathon [41].

4.2 Sustainable Outcomes

The three papers in this category, revolved around the theme of creating more sustainable outputs and outcomes of hackathons. Two papers modified the hackathon format as a vital part in a circular process, and sought to make the formats, and particularly the outcomes, more sustainable in each their way: [20, 52]. One paper implemented a post-hackathon activity intended to support advancing the immediate hackathon outputs and consider business models for the outputs: [9]. Table 4 provide an overview of the salient elements of the three cases in terms of PT.

4.2.1 Rethinking the Hackathon Format as Part of Circular Processes.

The two hackathon formats by Gama et. al. [20] and Webb et. al. [52] can be seen as formats that address the critique point of hackathons as producing only short lived outputs, often in the form of the immediate output of prototypes.

Input: Gama et. al. conducted a circular process of hackathons and mapathons exploring the Blue Economy, which is: “[...] the sustainable use of marine and ocean resources for economic growth and improved livelihoods in coastal areas.” [20]. The input for the mapathons, which are hackathons for generating map data through remote mapping, was unmapped geographic data, while the input for the hackathons was the outcome of these mapathons: mapped geographic data.

Process: During the mapathons, geographic data of unmapped areas was generated and improved. During the subsequent hackathons, prototypes based on this generated geographic data were developed, and the geographic data was evaluated[20].

Effect: The output of the hackathons were the prototypes as well as feedback for improving the geographic data, which was then collected and used as input for the next mapathon [20].

Webb et. al.’s hackathon format, the LabHack, is a case on modifying a hackathon by addressing an issue particularly relevant for the participants themselves. Emphasising that hackathons have become: “[...] a valuable educational and practical tool across the CHI community [...]” [52], Webb et. al. motivate their modified hackathon format by focusing on laboratory equipment shortages in STEM education in Africa. The goal, and thereby the intended outcome, with the modified hackathon format was to: “[...] motivat[e] and challeng[e] students to design and build their own frugal, reproducible pieces of laboratory equipment.” [52].

Input: In order to approach this issue, Webb et. al. tailored a hackathon format based on the Open Hardware movement, where laboratory equipment plans are disseminated through online resources [37]. University students and technical club hobbyists were invited to the hackathon, and were encouraged to draw on these already available resources and make their own design plans available.

Process: One process element for Webb et. al.’s modified hackathon, was the preparation phase, where participants did most of the design and development of the equipment prototypes before the actual hackathon event, which were mainly dedicated to learning sessions. According to the authors, the longer preparation phase for example enable design iterations and thereby improved quality of the designs [52]. In addition, participants have more time for developing skills necessary for building their prototype [52].

Effect: Eleven prototypes were developed, and additionally contributions to the Open Hardware movement were made. Like Gama et. al. [20], Webb et. al. [52] rethought the hackathon format into a circular process, in the sense that the LabHack is meant to be applied whenever there is a need for laboratory equipment and thereby the hackathon can potentially contribute to the shared repository of equipment plans.

Summarizing, the two hackathon formats analyzed in this section each explore how to potentially improve the longevity and quality of hackathon outputs and outcomes. In Gama et. al.’s case, they organized mapathon and hackathon formats into a circular process where each conducted format is used to build on and improve outcomes from a previous format. This presents an interesting approach to potentially improve the quality, as well as prolong the lifetime of hackathon outcomes. Webb et. al.’s case deliberately aims at contributing with *reproducible* outputs, in the form of design plans that can be shared beyond the immediate hackathon event in a central repository, hence potentially facilitating an “[...] ongoing and self-sustaining process to be applied whenever there is some equipment need [...]” [52].

4.2.2 Post-Hackathon Activity. We identified one paper which explored an activity that mainly concerned the output of a hackathon, and how to advance the outputs towards realization and practice, for instance by transitioning a hackathon team into a startup [9]. Chan et. al. introduce the concept of post-hackathon *learning circles*, to: “[...] connect hackathon teams with key stakeholders, reflect on prototypes and consider business models.” [9].

In terms of PT, the learning circle activity can be described as having potential for supporting a mechanism for improving the connection between the designed outputs and stakeholders, hence improving the chances that the immediate outputs could have an impact on end-users.

4.3 Learning

This section shows how hackathons activities and mechanisms have also been found useful in educational settings, primarily to solve issues faced by more traditional instructional approaches. In a PT perspective this could be construed as experimenting with alternative programs in the form of inputs, mechanisms and activities to either strengthen existing approaches and tools for instruction or enabling students to engage with new types of technology they have not encountered before. Table 5 provide an overview of the identified PT elements.

While all the papers in this category are concerned with learning, they also generally take different approaches both in terms of inputs, mechanisms and activities. Salinas et al. [43] discuss how to use the specialized “Datathon” format in an adapted shorter and more intensive version, while Cutts et al. [10] transform a traditionally digital programming introductory class into a shorter paper-based hackathon format that enables students to grasp fundamental concepts. Similarly, Gama [19] adapted the hackathon format into a full day of solving challenges in groups, thus aiding student motivation and hands-on experience with topics in a course on web technologies. Last, Karimi et. al. [26] used a modified hackathon format to allow teachers to better understand the possibilities that technology might offer.

Table 4: A comparative overview of highlighted elements which contributed to modifying hackathon formats for specific purposes in terms of facilitating more sustainable outcomes.

	<i>Input</i>	<i>Mechanism</i>	<i>Process</i>	<i>Activity</i>	<i>Output</i>	<i>Effect Outcome</i>	<i>Impact</i>
Rethinking the Hackathon Format as Part of Circular Processes [20, 52]	Unmapped geographic data	Improving geographic data	Mapping unmapped geographic data	Mapped geographic data	Improving and prolonging hackathon outcomes	Supporting growth of sustainable Blue Economy	
	Improved geographic data from previous mapathon		Developing applications using mapped data	Applications			
			Improving mapped data using feedback from hackathon	Evaluations of mapped geographic data			
	Need for laboratory equipment	Educational opportunity	Learning sessions and workshops	11 frugal and reproducible designs	Contributing to Open Hardware movement	Addressing equipment shortage which undermine STEM education in Africa	
	Arduino kits for teams	Skill development	Elaborate preparation and prototype creation phase before physical event				
Post-Hackathon Activity [9]	Open Hardware resources						
		Improving connection between output and stakeholders	Learning circles	Four solution suggestions			

Table 5: A comparative overview of highlighted elements which contributed to modified hackathon formats for specific purposes in terms of facilitating learning.

	<i>Input</i>	<i>Mechanism</i>	<i>Process</i>	<i>Activity</i>	<i>Output</i>	<i>Effect Outcome</i>	<i>Impact</i>
Shortening timeframes to build skill [43]	Setup with computers	Enable participants to gain basic data analysis and visualization skills	Six hour time frame			Basic literacy skills	
	Userfriendly software						
Alternative educational program [10, 19]	Pen and paper	Mastery-based learning	Collaborative group activities solving code exercises	Improved code comprehension			
		Peer-learning	Tutor assistance				
Demystifying design materials [26]	Challenge Data files	Challenge-based learning	10 hour timeframe	Prototypes	Higher motivation		
	non-entry level design materials	Demystifying technology	Setup and troubleshooting of prototyping technology				

4.3.1 Shortening Timeframes to Build Skill. While hackathons draw on time constraints for some of their power, there is also potentially excluding aspects or new potentials in even further constraining or expanding the timeframe in which the hackathon is executed in.

One example of this is the 'Short Datathon' format presented by Salinas et al. [43]. Salinas et. al. observed that with the rise of Big Data, so-called 'Datathons', that is hackathons that focus on a particular set of data, are becoming increasingly popular, yet it is difficult for newcomers to participate as they do not possess the basic data literacy skills.

Input: Salinas et al. [43] adapted the Datathon format for an educational setting running as part of an "Internet Week" focusing on new technologies. The outset consisted of creating an initial infrastructure with existing computers, an easy to use piece of software and a series of vetted data sources to ensure participants do not have to spend time on setting up for getting started.

Process: Rather than trying to tweak a larger Datathon, the authors created a six hour hackathon that enabled participants to gain basic data analysis and visualization skills, thus allowing them to participate in larger Datathons. The chief means of this is a shortening of the timeframe: it can be daunting to participate in a full weekend event if one does not feel confident about being able to participate. This in turn hurts the diversity of existing datathons. With the infrastructure ready and a set schedule for the six hours

long Datathon, collaboration between participants on the actual data exploration and visualisation could begin.

Effect: In terms of PT, the intended outcome, basic literacy skills for newcomers, shaped both the inputs and process. The authors shortened the format to six hours as well took on a more structured form, to achieve their goal of supporting newcomers. We consider this paper an example of how it might not be enough to do recruitment to achieve the necessary inputs for a given program: rather, we might need to run other programs first.

4.3.2 Alternative Educational Program. One example of transforming an educational program is the 'Thinkathon' discussed and developed by Cutts et al. [10]. While students were able to solve programming problems at home using assistance from the internet or their development software, they would struggle with 'code comprehension' in the final exams and further in their studies.

Input: To enable students to strengthen their understanding of the core concepts of programming, the authors developed the 'Think-a-Thon': three evenings where students in groups would, using only pen and paper, work their way through 200 exercises.

Process: The designed activities were, in PT terms, the collaborative group activities with tutor assistance as students worked their way through the exercises with pen and paper. The employed PT mechanisms are referred to by Cutts et al. [10] as Mastery-Based learning by way of exposure and practice with one problem-set

that challenges students and builds upon the previous one. The ability to be co-located and work on similar problems with tutor assistance established a mechanism of peer-learning in the sense that students would discuss informally between groups how to solve certain issues. This mechanism, a novel coupling of reading and lab exercises in a hackathon-like format using pen and paper, allowed the students to ignore scaffolding such as code editors and frameworks, focusing purely on the code and the fundamental concepts such as statements, expressions or variables.

Effect: Cutts et al. [10] report how the outcomes and impact of this approach was meant to both prepare them for the coming exams but also develop confidence and understanding in their own ability to solve programming problems without external assistance.

A second example is provided by Gama [19] who reports on how they transformed the final part of an introductory distributed systems course for undergraduates as an elective in an education on Information Systems. Gama [19] focuses on how adapting a hackathon model might establish 'Challenge-based Learning'.

Input: Students received data of the geolocation of taxis and busses, and would use this to simulate the real-time streaming of the data to create a system using distributed technologies.

Process: A 10 hour 'hack day' was organized where the students would turn up and in groups start developing a system using the data and series of distributed web technologies. The idea was to put into practice what the students had learned during the semester, and this idea was motivated by the reflection among staff at the education that assigning such projects as homework seemed less efficient due to creating a disconnect between concepts and technologies, and the actual project work. This was accomplished by means of an activity where students were tasked with developing a prototype of an urban mobility system. Throughout the activity the students had the chance to receive feedback on their design choices. The core mechanism of Challenge-Based learning can be considered driven partly by this challenge and intense teamwork by means of being co-located, receiving support and discussing with other students.

Gama [19] reflects that the setup might have benefited from an even more careful structuring of the data the students received and also that there tended to be a quick division of labour among students that meant that not every student got to work on all parts of the projects. The latter is suggested to be solved by a forced task rotation thus making the format proposed by Gama [19] even more structured.

Analysing these two examples in a PT perspective is closely linked to the two core mechanisms, Mastery Based Learning and Challenge-Based Learning, discussed by Cutts et al. [10] and Gama [19] respectively. What hackathons do in these two education-based examples is to establish intense time-constrained activities that are a) *co-located* b) *framed in terms of specific tech and challenges* c) *and with ample in-room support for students*.

4.3.3 Demystifying Design Materials. Another example on a modified hackathon for supporting learning, was the hackathon organized by Karimi et. al. [26].

Input: They specifically selected design materials which were not entry-level 'plug and play' materials as input for a short

hackathon lasting five hours [26]. In addition to the choice of material, Karimi et. al. designed the overall challenge and theme for the hackathon to be about augmenting a community garden with technology, a setting meant to be familiar to the teachers.

Process: The choice of design material was motivated by Karimi et. al's objective of "demystifying technology", so that teachers could develop experience with the technology in order to be able to apply the technology into their own classrooms as part of their teaching.

Effect: Karimi et. al. believe that this approach is more sustainable and has longer-term impact compared to if the teachers were only presented for the 'plug and play' types of technology. However, despite this motivation, the authors conclude that the: "[...] oneday hackathon workshop did not provide sufficient background to allow teachers to gain the confidence needed to implement their own digital technology project in a classroom." [26]. They still reflect on the hackathon as a potential format for providing teachers with experiences with technology that move beyond only appropriating 'plug and play' technologies into their classrooms, and note that, in general, appropriating technology into their teaching will unlikely ever be straightforward for teachers, given their constraints [26].

Though the hackathon was only partially successful [26], we see the attempt of providing experience with the "mysterious" and unfamiliar (the technology) in a familiar setting (a community garden) as an interesting approach to balancing the input for a hackathon format, where the goal is to expose non-designer participants to the messy reality of designing with technology that is not 'plug and play'.

5 DISCUSSION

What do hackathons do in their specific contexts? To engage with this question, we have performed a PT analysis on 16 case studies studying the processes and effects of hackathons. We have sought to explicate the underlying assumptions of 16 case studies that modify hackathon formats for specific purposes through our analysis in the previous section. We have offered our case study analysis as a critical effort to prepare the grounds for researching and evaluating hackathons formatively [17] by investigating which adaptions worked well under which conditions, what kind of problems were faced, and how the overall process for organizing hackathon can be improved. Carrying forth insights gained from our analysis, we now turn to discussing the role of hackathons and how they may be modified for future HCI research.

First, we discuss the retrospective insights on modified hackathon formats, in terms of the three categories: Participation, Sustainable Outcomes, and Learning. Following this, we discuss prospective considerations for future research and organization regarding hackathon formats.

5.1 Retrospectives on the Modified Hackathons

Researchers have noted that it is "difficult in particular for novice organizers to decide how to run an event that fits their needs" [36]. We have provided a catalogue of 16 case studies and presented a PT analysis of these cases studying their modified processes and effects. By juxtaposing and comparing these cases, we hope that future research with and on hackathons can systematically build on

the knowledge and experiences in relation to organizing hackathon formats for specific purposes. Oftentimes, modifying a hackathon format was motivated by general criticism of hackathon formats, such as the lack of sustainable long-term outcomes and lack of diversity in participation. The analyzed cases illustrate steps towards exploring formats which may accommodate these critique points.

5.1.1 Participation. As mentioned in the introduction, the HCI community has an ongoing critical dialog about hackathons. Lodato, Gregg and DiSalvo noted that though hackathons can produce new imaginaries in the performance of citizenship, hackathons risk reinforcing specifically a technological citizenship, while neglecting to ask who do not participate, who are not able to participate or is not represented in hackathons [12]. Or in the words of Irani: “There was no time to care by drawing in those who have been silenced [...] There was only time for the entrepreneurial spirit.” [25]. Through analyzing the nine cases which broadly revolved around the topic of participation, we noticed several different ways of modifying hackathons to invite a broader audience to partake in technology development, see [4, 24, 29, 30, 33, 35, 41, 48, 49].

An input used to invite a broader audience to take part in developing technology was the choice of which design material to provide for participants. This reflects the findings from the related work of Taylor and Clarke [46], who studied how non-technical participants can participate in hackathons appropriated for a non-technical audience. While the choice of design material in hackathon formats can be used to lower the barrier for developing technology for non-technical participants, design material can also be used because it is a barrier. This was illustrated in the hackathon of Karimi et.al. [26].

An often-mentioned suggestion for broadening participation in hackathons, is inviting or recruiting specific people who may not normally participate in hackathons. However, the authors from [30, 33] illustrate that it may not suffice with targeted recruitment. From the perspective of PT, in order to move towards an intended impact of a broader and more diverse participation in hackathons, for example by minorities or end-users, there needs to be carefully designed activities and mechanisms during the process which can support their participation.

In their guidelines for organizing corporate hackathons, Pe-Than et.al. reflect that teams with mixed skilled participants most likely will consist of novices and experts [38]. In order to bridge the gap between novices and experts, Pe-Than et.al. suggest for example brainstorming as a useful technique: “[...] that allows everyone to feel their ideas are heard and seems particularly effective in helping those who identify as minorities.” [38]. Brainstorming is, however, a technique which is used for generating ideas, which is usually needed in the *beginning* of a design process. To keep bridging the gap between different groups of people during the process of hackathons, a range of different and synergistic activities throughout the process may be necessary. As illustrated in the analysis of the hackathons of Birbeck et.al. and Hope et. al. [4, 24], a range of different activities and mechanisms may accomplish similar intended effects, such as facilitating communication and collaboration between different groups of people. Therefore it is important to evaluate experiences with modifying hackathons

for specific purposes, to explicate whether assumptions behind initiatives hold water, and share these insights.

5.1.2 Sustainable Outcomes. The output of hackathon formats have been criticized on several dimensions, for example of being of poor quality, lacking in utility and usability [18].

Through PT, we can juxtapose and compare the different approaches to creating more sustainable outcomes. These approaches were particularly pronounced in the cases of [9, 20, 52]. However, we noticed other initiatives which were motivated with contributing to more shareable outcomes, for example the workbooks in [49], and the documentation prize in [33]. Making outcomes of hackathons shareable could potentially help improve some of the issues surrounding hackathon outcomes. The inputs of the workbooks and the documentation prize aim at making the design processes more transparent in two different ways: The workbooks were intended to capture *work practices* and make them shareable, while the documentation prize was intended to motivate participants to make high-quality instructions so others could understand their work. In that way, the captured work practices become a form of shareable output. Even though an immediate output in itself is not good quality, shareable outcomes can be more valuable for longer term effects, such as contributing to a shared knowledge repository, as was the case by Webb et.al. [52].

In line with the approach in Chan et. al.’s hackathon [9], the hackathon by Birbeck et. al. offer inspiration for how to support the output of hackathons. In the case of Birbeck et. al. this was done through post-hackathon stakeholder discussion about the hackathon prototypes. They argue that involving stakeholders in post-hackathon discussion like these, can make way for potentially integrating the design outcomes into “self-harm care pathways.” [4].

While these concrete examples may potentially contribute towards more sustainable hackathon outcomes, we need more studies to explore these initiatives and their effects. Furthermore, while these initiatives aim at making work practices transparent and outcomes shareable, sustainable outcomes should also be considered in terms of participation. In a participatory design context, involving end-users as participants carefully and meaningfully in design processes is essential for ensuring outcomes which authentically meet end-users needs [5]. Therefore, the question of how sustainable outcomes are may also depend on who participates and how they participate. As we observed from the cases, the participation of people has to be thoroughly considered in order to go beyond participants as only inputs towards employing activities and mechanisms which support their participation throughout the hackathon.

5.1.3 Learning. Learning and developing skills has been one of the major promises of hackathons [51] and given the focus of education on novel ways of engaging with difficult topics it was unsurprising that we found a set of papers on education such as [10, 19]. We consider such efforts attempts at using alternative *programs*, where classical homework, classroom instruction and exercises would be the normal way of doing it. However, the studies also show that hackathons are no silver bullet in this area. Rather, the examined corpus show how the positive learning outcomes reported is the result of a careful tweaking of both input and activities to ensure

that the vital mechanisms of Mastery Based Learning and Challenge-Based Learning become effective and leads to outcomes appropriate for the teaching goals outlined in the papers. What PT offers in this case is a way of being precise about the interplay between inputs, activities, mechanisms and outcomes.

5.2 Considerations for Organizing Hackathons

Even though there are several problems with using hackathons as a research method, based on our analysis we believe that hackathons offer unique strengths and advantages that can be carefully leveraged for future HCI research. While we are in agreement with several of the existing critiques on hackathons, we argue that modified hackathons can and should be utilized as a part of future HCI research since they can offer a form of methodological democracy in the face of epistemological hegemony. Epistemological hegemony “represents a concern for the domination of one view of knowledge and the subordination of all other forms” [8]. By modifying the processes and desired end goal of hackathons, researchers have the opportunity to include those who have been historically marginalized when considering the design of technology mediated futures. Through our PT analysis and comparison of 16 cases studies, we have foregrounded the researchers’ point of view who have attempted to explore and add various dimensions to the democratic aims of participation through modified hackathons.

Pushing our argument further, we argue that modified hackathon formats for research should be understood and evaluated as exercising methodological democracy by both researchers and participants, albeit not without problems. Based on our cross-case analysis, we proposition that modified hackathon formats should aspire to make two types of contribution for HCI research. One type of contribution is concerned with the desired end goal of the research project while the other type of contribution is about exercising methodological democracy by redefining hackathon formats for further knowledge production. The latter contribution extends beyond the scope of the individual project and can only be made sense of in relation to how other researchers are utilizing hackathons in their specific contexts over a period of time. Since researchers are often, though not always, positioned as the driving and defining force of hackathon activities and formats, exercising methodological democracy is attributed to the researchers’ agency by default. However, participants’ contributions towards methodological democracy in the face of epistemological hegemony is currently underexplored and requires further engagement in future HCI research utilizing modified hackathons.

Understanding modified hackathons as exercising methodological democracy involves acknowledgement of participant contributions beyond individual project outcomes towards the development of broader HCI methodology and theory related processes and mechanisms. Because participants’ contributions towards methodological democracy may not always be immediately evident and most likely evolve beyond a specific project, we speculate that this aspect is often overlooked and/or overshadowed by insights that are circumscribed within the scope of individual projects through research writing. However, there are a few research works that have critically foregrounded participants point of view thereby

questioning, re-ordering, and expanding the processes and outcomes of hackathons. For instance, DiSalvo et al have argued that “civic hackathons produce new imaginaries of place, belonging, and hope in the performance of citizenship [wherein] these imaginaries provide crucial affective mechanisms for mobilizing energy and ultimately resources to secure political outcomes in local communities” [12]. Similarly, exploring feminist hackerspaces, Fox et al have noted that “designing how the space should look, feel, and run, members reframe activities seldom associated with technical work as forms of hacking [thereby] shift concerns for women in technology from questions of access (who is included) to questions of recognition (who is visible) while grappling with productive ambiguities in between” [15]. In both these cases, participants’ points of view are intentionally foregrounded as driving methodological decisions with respect to the research project but also towards the development of broader HCI methodology and theory related to hackathons.

For researchers who are interested in using modified hackathon formats, based on our cross case analysis, we recommend framing their work and claiming contributions in relation to exercising methodological democracy. Research utilizing modified hackathon formats for further knowledge production should be evaluated beyond individual project goals and aim to push back on epistemological hegemony by exercising methodological democracy. This does not mean that all modified hackathon formats are necessarily problem free but should rather be treated as critical attempts at questioning and rearranging existing power relationships with respect to research knowledge production.

ACKNOWLEDGMENTS

We thank our anonymous reviewers and Midas Nouwens for helpful feedback. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 740548).

REFERENCES

- [1] Seyram Avle, Silvia Lindtner, and Kaiton Williams. 2017. How methods make designers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 472–483.
- [2] Evi E. Beck. 2002. P for Political: Participation is Not Enough. *Scandinavian Journal of Information Systems* 14 (2002), 5–13.
- [3] Leonard Bickman. 1987. The functions of program theory. *New directions for program evaluation* 1987, 33 (1987), 5–18.
- [4] Nataly Birbeck, Shaun Lawson, Kellie Morrissey, Tim Rapley, and Patrick Olivier. 2017. Self Harmony: rethinking hackathons to design and critique digital technologies for those affected by self-harm. In *proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 146–157.
- [5] Susanne Bødker and Morten Kyng. 2018. Participatory design that matters—Facing the big issues. *ACM Transactions on Computer-Human Interaction (TOCHI)* 25, 1 (2018), 1–31.
- [6] Claus Bossen, Christian Dindler, and Ole Sejer Iversen. 2016. Evaluation in participatory design: a literature survey. In *Proceedings of the 14th Participatory Design Conference: Full papers Volume 1*. Association for Computing Machinery, New York, NY, USA, 151–160.
- [7] Claus Bossen, Christian Dindler, and Ole Sejer Iversen. 2018. Program theory for participatory design. In *Proceedings of the 15th Participatory Design Conference: Short Papers, Situated Actions, Workshops and Tutorial-Volume 2*. Association for Computing Machinery, New York, NY, USA, 1–4.
- [8] Mark Brough. 2013. Chapter 4 - Toward Cultural Epidemiology: Beyond Epistemological Hegemony. In *When Culture Impacts Health*, Cathy Banwell, Stanley Ulijaszek, and Jane Dixon (Eds.). Academic Press, San Diego, 33 – 42. <https://doi.org/10.1016/B978-0-12-415921-1.00004-X>

- [9] Tina Chan, Josephine McMurray, AnneMarie Levy, Heidi Sveistrup, and James Wallace. 2020. Post-Hackathon Learning Circles: Supporting Lean Startup Development. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–8.
- [10] Quintin Cutts, Matthew Barr, Mirella Bikanga Ada, Peter Donaldson, Steve Draper, Jack Parkinson, Jeremy Singer, and Lovisa Sundin. 2019. Experience report: thinkathon-countering an’ got it working’ mentality with pencil-and-paper exercises. *ACM Inroads* 10, 4 (2019), 66–73.
- [11] Amalia De Götzen, Luca Simeone, Joanna Saad-Sulonen, Begüm Becermen, Nicola Morelli, et al. 2020. The hackathon format: an analysis of its possible interpretations under a service design perspective. *DS 101: Proceedings of NordDesign 2020, Lyngby, Denmark, 12th–14th August 2020 DS 101* (2020), 1–12.
- [12] Carl DiSalvo, Melissa Gregg, and Thomas Lodato. 2014. Building belonging. *interactions* 21, 4 (2014), 58–61.
- [13] Kathleen M. Eisenhardt. 1989. Building Theories from Case Study Research. *The Academy of Management Review* 14, 4 (1989), 532–550. <http://www.jstor.org/stable/258557>
- [14] Jeanette Falk Olesen and Kim Halskov. 2020. 10 Years of Research With and On Hackathons. In *Proceedings of the 2020 ACM on Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 1073–1088.
- [15] Sarah Fox, Rachel Rose Ulgado, and Daniela Rosner. 2015. Hacking Culture, Not Devices: Access and Recognition in Feminist Hackerspaces. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing* (Vancouver, BC, Canada) (CSCW ’15). Association for Computing Machinery, New York, NY, USA, 56–68. <https://doi.org/10.1145/2675133.2675223>
- [16] Frank J Frey and Michael Luks. 2016. The innovation-driven hackathon: one means for accelerating innovation. In *Proceedings of the 21st European Conference on Pattern Languages of Programs*. Association for Computing Machinery, New York, NY, USA, 1–11.
- [17] TW Frick and CM Reigeluth. 1999. Formative research: A methodology for creating and improving design theories. *Instructional-design theories and models: A new paradigm of instructional theory* 2 (1999), 633–652.
- [18] Kiev Gama. 2017. Preliminary findings on software engineering practices in civic hackathons. In *2017 IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*. IEEE, IEEE Press, Piscataway, NJ, USA, 14–20.
- [19] Kiev Gama. 2019. Developing course projects in a hack day: an experience report. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 388–394.
- [20] Kiev Gama, Victoria Rautenbach, Cameron Green, Breno Alencar Gonçalves, Serena Coetze, Nicolene Fourie, and Nishanth Sastry. 2019. Mapathons and Hackathons to Crowdsource the Generation and Usage of Geographic Data. In *Proceedings of the International Conference on Game Jams, Hackathons and Game Creation Events 2019*. Association for Computing Machinery, New York, NY, USA, 1–5.
- [21] Amanda L Griffith. 2010. Persistence of women and minorities in STEM field majors: Is it the school that matters? *Economics of Education Review* 29, 6 (2010), 911–922.
- [22] Nicolai Brodersen Hansen, Christian Dindler, Kim Halskov, Ole Sejer Iversen, Claus Bossen, Ditte Amund Basballe, and Ben Schouten. 2019. How participatory design works: mechanisms and effects. In *Proceedings of the 31st Australian Conference on Human-Computer-Interaction*. Association for Computing Machinery, New York, NY, USA, 30–41.
- [23] Kristina Höök and Jonas Löwgren. 2012. Strong concepts: Intermediate-level knowledge in interaction design research. *ACM Transactions on Computer-Human Interaction (TOCHI)* 19, 3 (2012), 1–18.
- [24] Alexis Hope, Catherine D’Ignazio, Josephine Hoy, Rebecca Michelson, Jennifer Roberts, Kate Krontiris, and Ethan Zuckerman. 2019. Hackathons as participatory design: iterating feminist utopias. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–14.
- [25] Lilly Irani. 2015. Hackathons and the making of entrepreneurial citizenship. *Science, Technology, & Human Values* 40, 5 (2015), 799–824.
- [26] Arafeh Karimi, Peter Worthy, Paul McInnes, Marie Bodén, Ben Matthews, and Stephen Viller. 2017. The community garden hack: Participatory experiments in facilitating primary school teacher’s appropriation of technology. In *Proceedings of the 29th Australian Conference on Computer-Human Interaction*. Association for Computing Machinery, New York, NY, USA, 143–151.
- [27] Samia Khan and Robert VanWynsberghe. 2008. Cultivating the Under-Mined: Cross-Case Analysis as Knowledge Mobilization. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* 9, 1 (2008). <https://doi.org/10.17169/fqs-9.1.334>
- [28] Christoph Kollwitz and Barbara Dinter. 2019. What the Hack?—Towards a Taxonomy of Hackathons. In *International Conference on Business Process Management*. Springer, Springer International Publishing, Cham, 354–369.
- [29] Marco Konopacki, Debora Albu, and Fabro Steibel. 2019. MUDAMOS: a civil society initiative on collaborative lawmaking in Brazil. In *Proceedings of the 12th International Conference on Theory and Practice of Electronic Governance*. Association for Computing Machinery, New York, NY, USA, 175–180.
- [30] Wiesław Kopeć, Bartłomiej Balcerzak, Radosław Nielek, Grzegorz Kowalik, Adam Wierzbicki, and Fabio Casati. 2018. Older adults and hackathons: a qualitative study. *Empirical Software Engineering* 23, 4 (2018), 1895–1930.
- [31] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research methods in human-computer interaction*. Morgan Kaufmann, Cambridge, MA, USA.
- [32] Ian Li, Jon Froehlich, Jakob E Larsen, Catherine Grevet, and Ernesto Ramirez. 2013. Personal informatics in the wild: hacking habits for health & happiness. In *CHI’13 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 3179–3182.
- [33] Jaya Narain, Ishwarya Ananthabhotla, Samuel Mendez, Cameron Taylor, Hosea Siu, Lora Brugnaro, and Adriana Mallozzi. 2020. ATHack: Co-Design and Education in Assistive Technology Development. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–7.
- [34] National Center for Women and Information Technology 2013. *National Center for Women and Information Technology. Top 10 Ways to Increase Girls’ Participation in Computing Competitions*. National Center for Women and Information Technology. Retrieved September 16, 2020 from <http://www.ncwit.org/resources/top-10-ways-increase-girls-participation-computing-competitions/top-10-ways-increase-girls>
- [35] Alexander Nolte, Linda Bailey Hayden, and James D. Herbsleb. 2020. How to Support Newcomers in Scientific Hackathons - An Action Research Study on Expert Mentoring. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 025 (May 2020), 23 pages. <https://doi.org/10.1145/3392830>
- [36] Alexander Nolte, Ei Pa Pa Pe-Than, Abasi amefon Obot Affia, Chalalai Chaihirunkarn, Anna Filippova, Arun Kalyanasundaram, Maria Angelica Medina Angarita, Erik Trainer, and James D. Herbsleb. 2020. How to organize a hackathon – A planning kit. [arXiv:2008.08025 \[cs.CY\]](arXiv:2008.08025 [cs.CY])
- [37] Open Source Hardware Association 2020. *Open Source Hardware Association*. Open Source Hardware Association. Retrieved September 14, 2020 from <https://www.oshwa.org/definition/>
- [38] Ei Pa Pa Pe-Than, Alexander Nolte, Anna Filippova, Christian Bird, Steve Scallen, and James D Herbsleb. 2019. Designing corporate hackathons with a purpose: the future of software development. *IEEE Software* 36, 1 (2019), 15–22.
- [39] Jari Porras, Jayden Khakurel, Jouni Ikonen, Ari Happonen, Antti Knutas, Antti Herala, and Olaf Drögehorn. 2018. Hackathons in software engineering education: lessons learned from a decade of events. In *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*. Association for Computing Machinery, New York, NY, USA, 40–47.
- [40] Emily Porter, Chris Bopp, Elizabeth Gerber, and Amy Voids. 2017. Reappropriating hackathons: the production work of the CHI4Good day of service. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 810–814.
- [41] Gabriela T Richard, Yasmin B Kafai, Barrie Adleberg, and Orkan Telhan. 2015. StitchFest: Diversifying a College Hackathon to broaden participation and perceptions in computing. In *Proceedings of the 46th ACM technical symposium on computer science education*. Association for Computing Machinery, New York, NY, USA, 114–119.
- [42] Patricia J. Rogers, Anthony Petrosino, Tracy A. Huebner, and Timothy A. Hacs. 2000. Program theory evaluation: Practice, promise, and problems. *New Directions for Evaluation* 2000, 87 (2000), 5–13. <https://doi.org/10.1002/ev.1177>
- [43] Myriam Raquel Noguera Salinas, Maria Claudia Figueiredo Pereira Emer, and Adolfo Gustavo Serra Neto. 2019. Short datathon for the interdisciplinary development of data analysis and visualization skills. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE Press, Piscataway, NJ, USA, 95–98.
- [44] Geertje Slingerland, Stephan Lukosch, Mariëlle den Hengst, Caroline Nevejan, and Frances Brazier. 2020. Together We Can Make It Work! Toward a Design Framework for Inclusive and Participatory City-Making of Playable Cities. *Frontiers in Computer Science* 2 (2020), 51. <https://doi.org/10.3389/fcomp.2020.600654>
- [45] Karen Tanenbaum, Theresa Jean Tanenbaum, Amanda M. Williams, Matt Ratto, Gabriel Resch, and Antonia Gamba Bari. 2014. Critical Making Hackathon: Situated Hacking, Surveillance and Big Data Proposal. In *CHI ’14 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 17–20. <https://doi.org/10.1145/2559206.2560476>
- [46] Nick Taylor and Loraine Clarke. 2018. Everybody’s hacking: participation and the mainstreaming of hackathons. In *CHI 2018*. Association for Computing Machinery, Association for Computing Machinery, New York, NY, USA, 1–2.
- [47] Nick Taylor, Loraine Clarke, and Katerina Gorkovenko. 2017. Community Inventor Days: Scaffolding Grassroots Innovation with Maker Events. In *Proceedings of the 2017 Conference on Designing Interactive Systems (Edinburgh, United Kingdom) (DIS ’17)*. Association for Computing Machinery, New York, NY, USA, 1201–1212. <https://doi.org/10.1145/3064663.3064723>

- [48] Nick Taylor, Loraine Clarke, Martin Skelly, and Sara Nevay. 2018. Strategies for engaging communities in creating physical civic technologies. In *CHI 2018*. Association for Computing Machinery, Association for Computing Machinery, New York, NY, USA, 1–12.
- [49] Andrea K Thomer, Michael B Twidale, Jinlong Guo, and Matthew J Yoder. 2016. Co-designing scientific software: Hackathons for participatory interface design. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 3219–3226.
- [50] Erik H Trainer, Arun Kalyanasundaram, Chalalai Chahirunkarn, and James D Herbsleb. 2016. How to hackathon: Socio-technical tradeoffs in brief, intensive collocation. In *proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. Association for Computing Machinery, New York, NY, USA, 1118–1130.
- [51] Jeremy Warner and Philip J Guo. 2017. Hack. edu: Examining how college hackathons are perceived by student attendees and non-attendees. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. Association for Computing Machinery, New York, NY, USA, 254–262.
- [52] Helena Webb, Jason RC Nurse, Louise Bezuidenhout, and Marina Jirotnka. 2019. Lab Hackathons to Overcome Laboratory Equipment Shortages in Africa: Opportunities and Challenges. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–8.
- [53] Robert K. Yin. 2014. *Case study research: design and methods*. Sage Publication, Thousand Oaks, CA, USA.