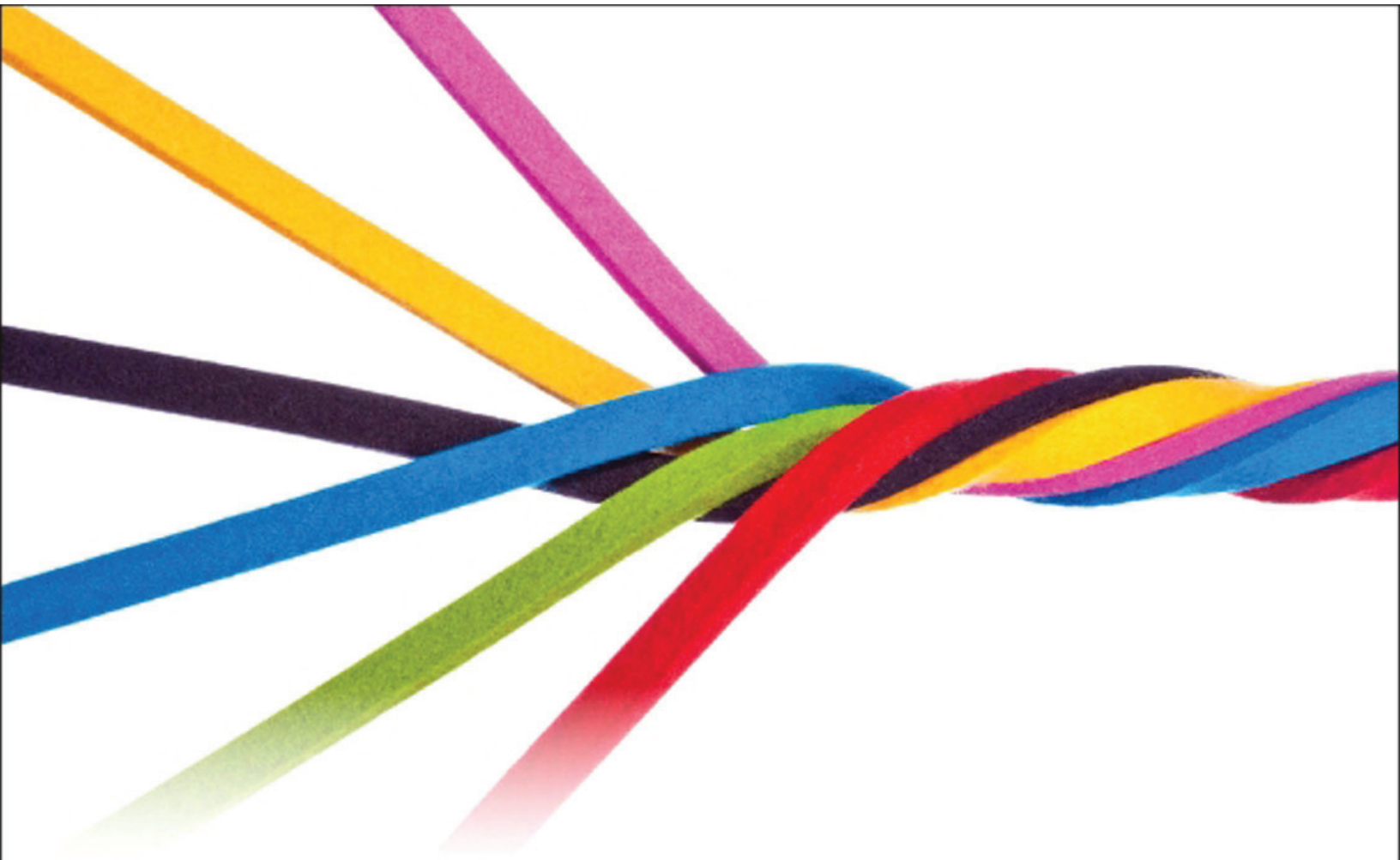

Human-Computer Interaction

An Empirical Research Perspective

MORGAN KAUFMANN

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

MK
MORGAN KAUFMANN

CHAPTER

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 *awk*ing 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.

[1] One 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 experiment 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.

[2]The 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.

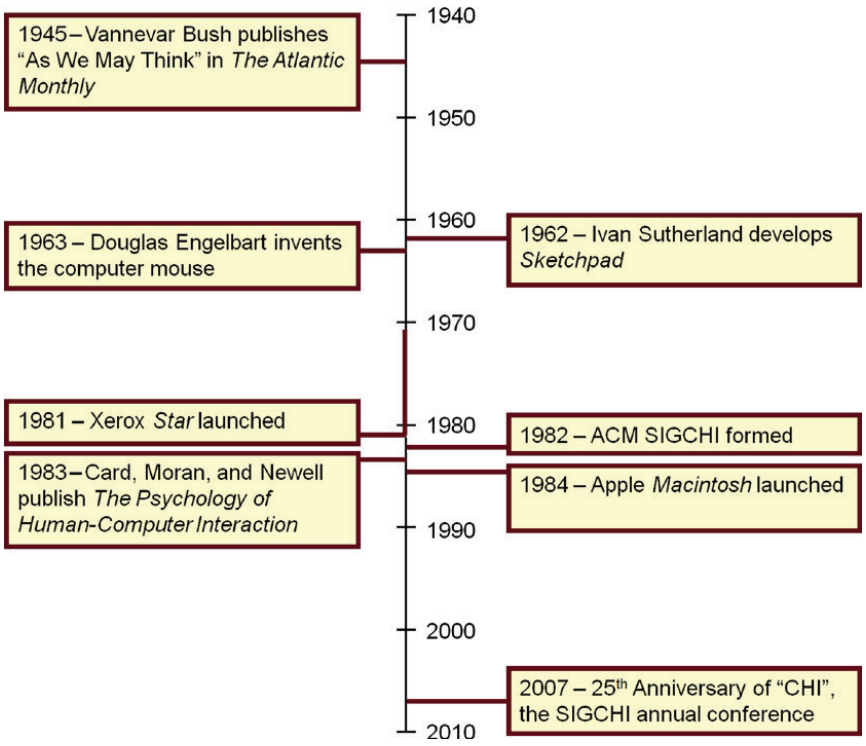

FIGURE 1.1

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

[3] Google 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

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

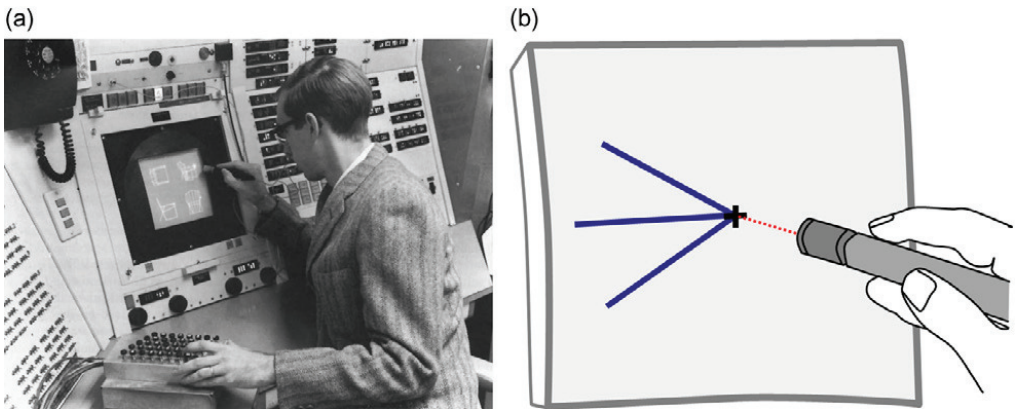

FIGURE 1.3

(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

[5]Engelbart'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.

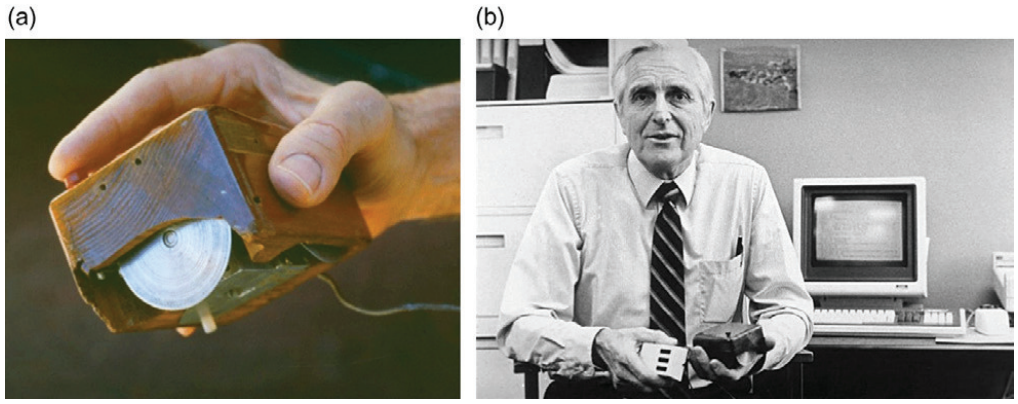

FIGURE 1.4

(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 *Grafacon*. 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.

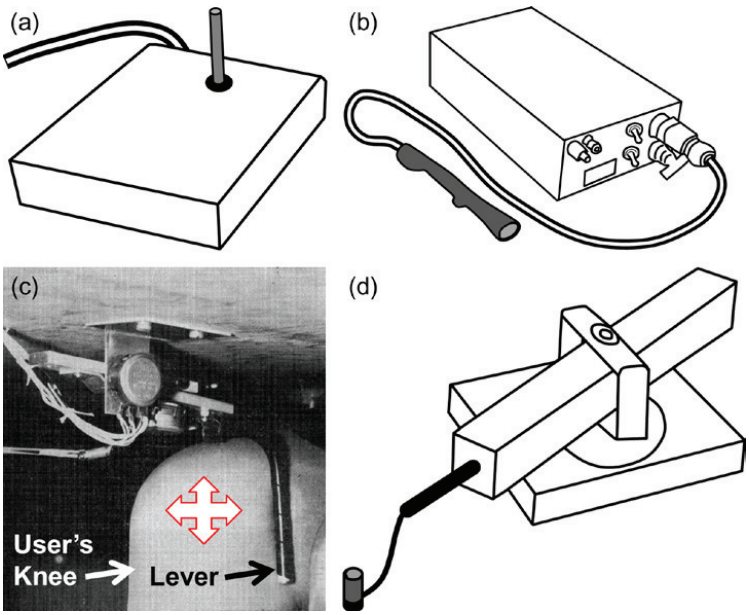

FIGURE 1.5

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

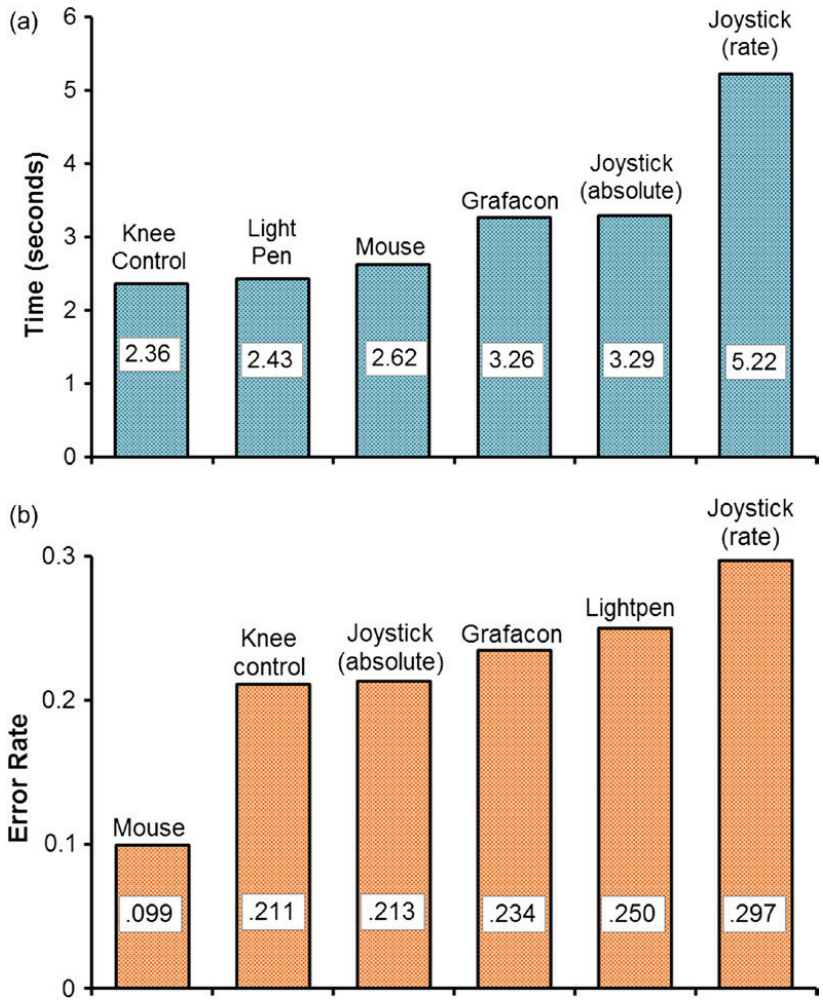

FIGURE 1.6

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.

[6] Attendance 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.

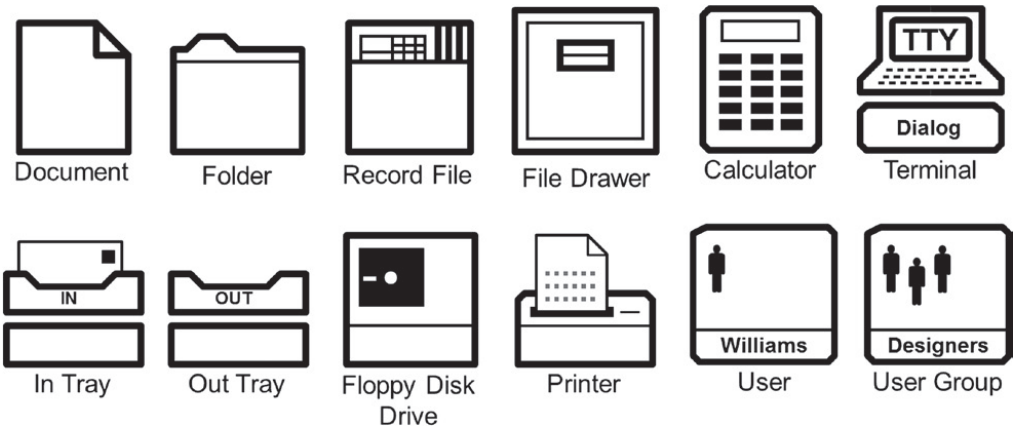

FIGURE 1.8

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.

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

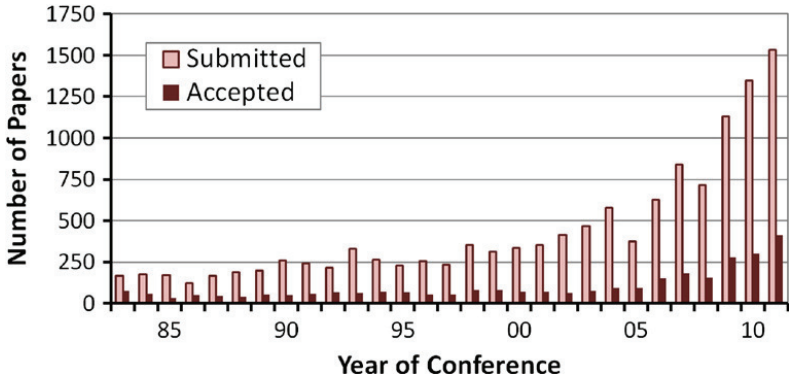

FIGURE 1.9

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.

[8]Data 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

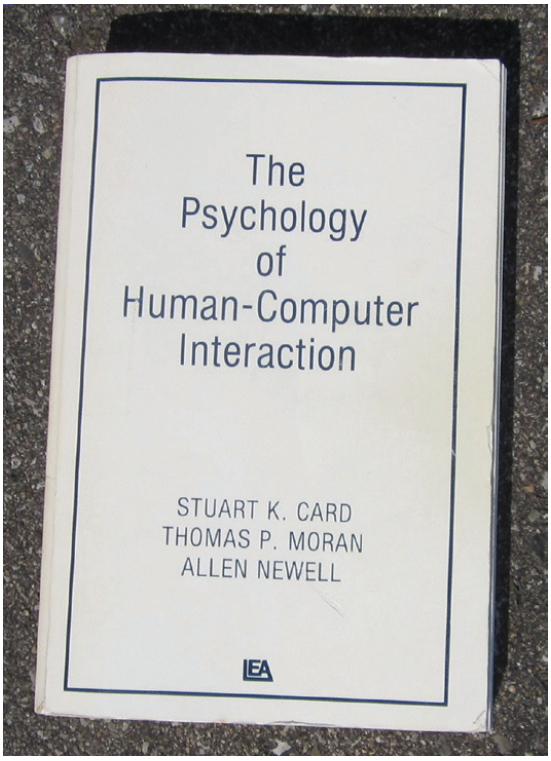

FIGURE 1.10

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

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

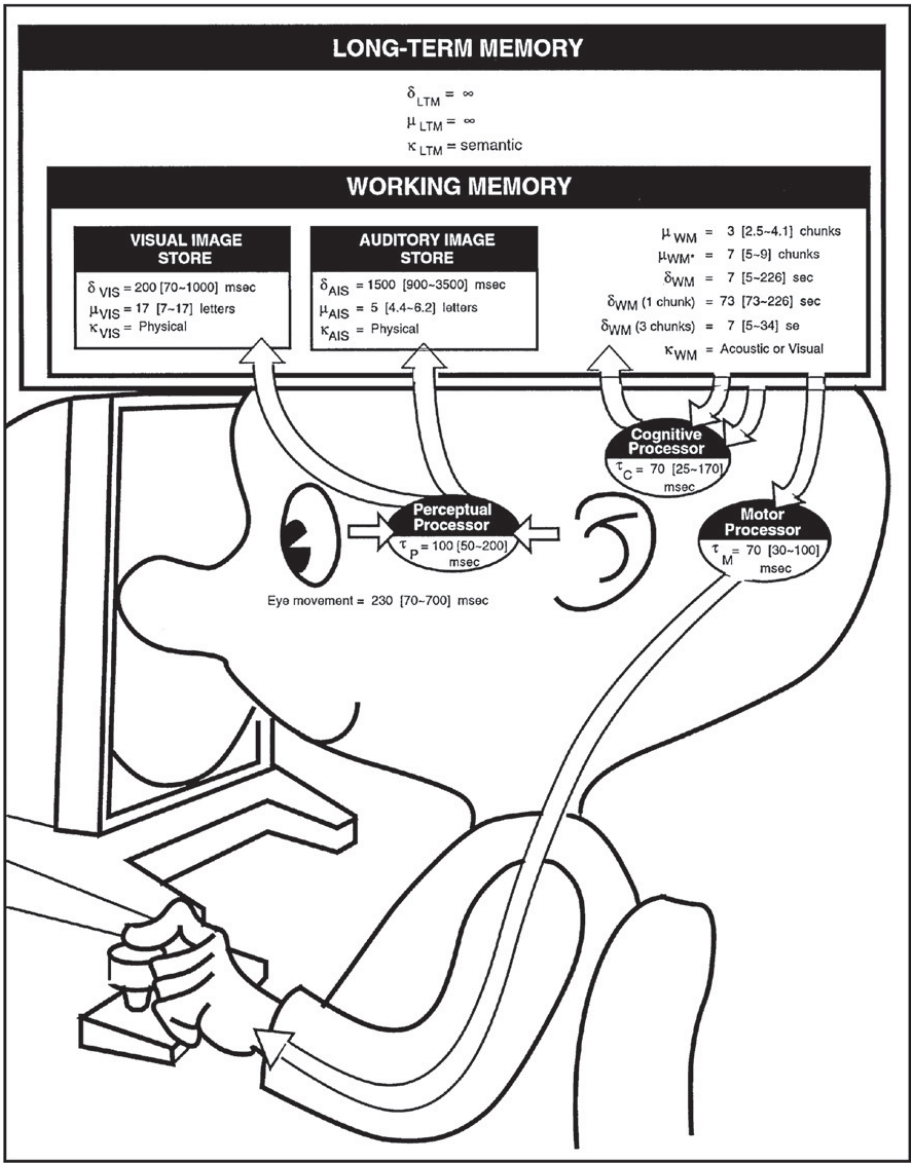

FIGURE 1.11

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

[10]Of 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:

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

(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

[11] Search using "1984 Apple Macintosh commercial."
[12] Known 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).

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

[14] www.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	*Lisa* project started . Goal of producing a powerful (and expensive!) personal computer.
1979	September – *Macintosh* 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 *Alto*. The GUI ideas in the *Alto* influence *Lisa* and *Macintosh* development.
1980	December – Apple goes public through initial public offering (IPO) of its stock.
1981	May – Xerox *Star* launched at the National Computer Conference (NCC) in Chicago. Members of the *Lisa* design team are present and see the *Star* demo. They decide to re-vamp the *Lisa* interface to be icon-based. August – IBM PC announced. Highly successful, but embodies traditional text-based command-line interface.
1982	*Lisa* and *Macintosh* development continue. Within Apple, there is an atmosphere of competition between the two projects.
1983	January – *Lisa* released. *Lisa* incorporates a GUI and mouse input. Sells for $10,000 U.S. In the end, *Lisa* is a commercial failure. December – brochures distributed in magazines (e.g., *Time*) pre-announcing the *Macintosh*.
1984	January 22 – *Macintosh* ad plays during Super Bowl XVIII. January 24 – *Macintosh* 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

[15] www.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.

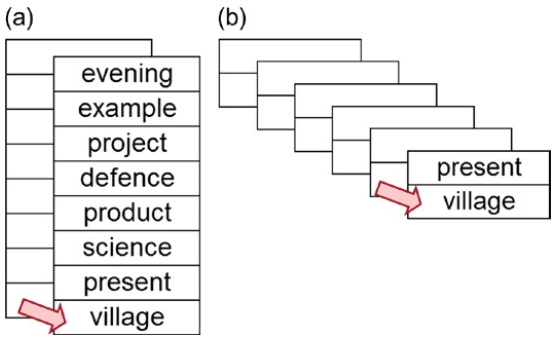

FIGURE 1.14

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