1 Growing Up in the Digital Age

There is a fundamental problem for people living in the digital age. People and digital artifacts are different. People struggle to understand and use their computers, cameras, and phones. Certainly, many of these devices are quite complex, but many features of them are also poorly designed. This is not to blame designers; they are people living in the digital age too. They have grown up with the same understanding of people and the same background of computers as the rest of us. We do feel, however, that if designers were more sensitized to the problems people have using digital technologies, they might come up with better designs. If they were more literate in things digital, they could better express their intentions through their designs.

The digital is about to come of age. For sixty years, we have seen a steady development of digital technologies. Now everything is going digital. Once-separate things like cameras and phones are now converging, creating new artifacts. Indeed, it is right to say that the very fabric of the world is increasingly becoming digitally enhanced. The designers of these new environments, artifacts, and forms of interaction need to understand the things they create and the impact that they may have on people's lives.

Unfortunately, there is good reason to believe that our understanding of people, and particularly how people think, has been dominated by a serious misconception for the last sixty years. This misconception is that there is an objective world that people understand by holding and manipulating symbols in their heads that, in turn, stand for the things in the world. In place of this view is a conception that thinking is inherently embodied, rather than disembodied, literal, and based on the logical manipulations of abstract symbols. It is creative, figurative, and derived from our experiences as humans living in a physical, cultural, and social environment.

With this alternative view of cognition we can revisit the design of digital technologies. We can understand where the constructs used in the design of digital artifacts have come from and better explain how they may be used. We can help software designers and engineers understand their technology. Most important, we can develop literacy in digital technology design that will enable people to interact with and through those technologies in a satisfying and fruitful way.

Things Digital

Digital technologies, of the sort that we are familiar with today, first arrived in the form of electronic computers at the end of the Second World War. The concept of an electronic calculating machine had arisen during the 1930s, and found a real need in the work of the code-breaking group working at Bletchley Park in Buckinghamshire, England, and the war effort in the United States. Soon, general-purpose computers with names such as ENIAC and EDVAC were being developed. At much the same time, psychologists were doubting the prevalent theories of psychology. These theories focused on the observed behaviors of people, and a fierce debate raged between the behaviorists and the "cognitivists" about how best to describe and understand the psychology of people. Whereas behaviorists had looked at the observed behaviors of people, cognitivists believed that people had mental representations of things in the world that they stored, manipulated, and transformed in their minds.

Since the 1950s, there have been many developments in both computing and cognition. They have drawn from each other and have contributed to the understanding of each other. The fundamental design of a computer's memory and the way in which it processes data are much the same now as they were in the 1950s. Data—whether text, video, audio, or graphics—is represented as bits (binary digits), manipulated in the computer's main memory, and stored on some more permanent secondary storage device. Data can be transmitted over communications links to other devices. In a similar way, people could be described as having a long term memory for more permanent storage, a short-term memory for transitory data, and some processing capability, and of course they can transmit as well as receive data through hearing, seeing, smelling, touching, or tasting.

Digital technology has been very successful. The speed and capacity of computers have increased at a phenomenal rate—far beyond anything the early pioneers dreamed of. Thus we now have digital technologies (that is, computers) that are ubiquitous and pervasive, embedded in the fabric of the world. People increasingly wear computers in their clothes or jewelry, or carry computers in their phones or audio players. Ambient technology describes computers that are embedded in the fabric of buildings. These new media are characterized by their interactivity. They are interactive media—people alter the behaviors, presentation, and content of the media through their interactions with them. Interactive media continue to evolve and converge so, for example, a phone might include a camera, display live video transmissions, and connect automatically to receive e-mail when in range of a communications network. In interactive media, data is digital so it is transferable between devices, transmittable across wireless networks, and transmutable from one representation to another. The stuff of digital media that allows all this to happen is known as software.

Alongside the successes of digital media, however, there have been some notable failures. In the 1960s, the new field of artificial intelligence (AI) was established, and courses were set up at the University of Edinburgh in Scotland and Stanford University in the United States. Expectations were high. An AI computer would understand people when they talked to it. Computers would be able to wander around a house, recognize faces, and think in much the same way that people do. From the 1970s onward, huge amounts of investment have been put into "natural-language" processing to try to get computers to understand and use language as we do. In the 1990s, the Japanese "fifth-generation" project predicted highly "intelligent" machines within five years with automatic translation between languages. In the United States, President Ronald Reagan envisaged an umbrella of intelligent missiles in the "Star Wars" project. None of these has been realized.

Cognition, or cognitive psychology, has also come a long way since the 1950s. In particular there have been great advances in neurology, which is concerned with how the brain functions, where the functions are located, and how these functions can be identified and repaired, if necessary. But we are still no closer to understanding how people understand what each other say, how they recognize objects, nor how they play games like chess. If we could only understand the representations that people have (so the argument goes), we could program computers to do the same. Yet, some commentators have asserted that computers could never be intelligent in any way approximating what we understand by human intelligence because of the way that computers represent the outside world. They might do some clever things (which indeed they have done), but this has to do with size and speed, with brute force rather than understanding.

This difference between the representations of things that computers have and people's ways of understanding and acting has also made interacting with the computer difficult. The human-computer interaction (HCI) discipline grew out of the computing and cognition discussions of the 1970s, and became a central arena in which the psychologists could explore their theories. Software systems became to the cognitivists what the rat-in-a-maze had been to the behaviorists. People could be studied and questioned about the thinking that they were engaged in. The first theories of HCI were based on the idea that people are "human information processors" (see, for example, Card, Moran, and Newell 1983). Theories of human information processing had developed during the 1960s and 1970s. They examined how people could encode, transform, create, and output information. Applying this to people interacting with computers seemed an ideal laboratory in which to explore the cognitivists' ideas.

Many metaphors were used during this time to try to characterize HCI. HCI was often called the human-computer dialogue in the early days, suggesting that the person was communicating and discussing things with the computer. The notion that people interacting with computers could be considered a dialogue had its origin in the early design of computers too. In the early 1950s, "instructions" were given to computers by setting switches. Each switch could be set; either it was on or off (hence binary). Six or eight such switches (binary digits or bits) taken as a group would represent a character. With eight switches there were 256 different combinations (2^8), which was enough to represent all the letters of the alphabet, the numbers 0 to 9, special characters such as commas and periods, and spaces, with some to spare. Eight bits were known as a byte. As the process of using computers became more automated, paper tape was used to provide a sequence of instructions. Holes were punched in the tape to represent the bytes, and the sequence became known as a program, instruction set, or procedure. The process worked much the same as the original automated knitting machines, the Jacquard loom. By the late 1950s, general sets of instructions were stored in the computer to enable it to undertake frequently used procedures (such as storing and retrieving data). These were known as supervisor programs, monitor programs, or the operating system.

And so the scene was set for the first period of the digital age. Metaphors were used to describe the actions of computers and the representations that they used. Data was stored in "files." Programs were written in a programming "language." Programs were "executed" on the computer. People engaged in a "dialogue" with the computer. Since then, of course, we have a very different form of HCI and such metaphors have proliferated. We have "menus" of "commands." We "cut" and "paste." We "open" and "close" files. We "quit" programs and "import" data.

The Felt Sense of Thinking

In 1980, George Lakoff and Mark Johnson published their book *Metaphors We Live By.* This groundbreaking work argued that language and thought were based on a limited number of fundamental, conceptual metaphors. Metaphor was much more than a literary trope; it was central to how humans thought. Many metaphors were not recognized as such because they had been so ingrained into our ways of thinking and talking that we no longer saw them at all. The coauthors gave examples such as "knowing is seeing" (for instance, I see what you mean) and "up is good" (one is climbing the ladder of success). The computing examples above would not be recognized as metaphors by many; they are what we do with computers. This discovery of the systematic embedding of metaphors was accompanied by another key insight. These metaphors were based on embodied experience. These fundamental, conceptual metaphors derive from the fact that we are people living in the world: "three natural kinds of experience—experience of the body, of the physical environment, and of the culture—are what constitute the basic source domains upon which metaphors draw" (Rohrer 2005, 14).

Lakoff and Johnson's work and that of some other commentators represented a major shift in cognitive theory that may, in turn, have significant impact on how we view cognition (the debate on this continues). Given the intertwined history of computing and cognition, we suspect that the various theories that Lakoff and Johnson's ideas spawned will have a significant impact on software engineering (SE) and HCI too. In this book we explore exactly these issues.

Consider the following story based on reports of an air accident (Imaz and Benyon 1996, 106–107):

On 27th November 1983 there was a Boeing 747 accident near the Barajas airport. It was an Avianca (Colombian Airlines) regular flight from Paris to Bogota with a stop at Madrid. Even if the total scenario is quite complex, it could be said that the main cause of the accident was the misuse of the Ground Proximity Warning System (GPWS). According to the official report on the accident, the GPWS was insistently warning about the fact that the plane was below a minimum altitude.

The captain reacted saying: "OK, OK" meaning "I know it is a GPWS malfunction, so I will continue the approach procedure." The official reports says:

Captain answers "Bueno, bueno" (OK, OK).

However some people very close to this Official Commission (and some magazines of that time) say that the real captain answer—registered in the CVR (Cockpit Voice Recorder)—was:

"Calla gringo" (Shut up gringo).

"Gringo" is an expression used by some Latin-American people referring to (north) American people. In this context the captain was Colombian (south American), the aircraft north American and a warning system (GPWS) that speaks English. There is also a known fault in the artifact that in certain circumstances it gives a false warning. This determines a mistrust reaction by the captain. The captain has used a way of anthropomorphizing the GPWS by using the metaphor: THE DEVICE IS A STUPID PERSON.

This story illustrates that the relationship the captain establishes with the GPWS is not neutral. It shows some type of mistrust or a pejorative attitude. It appears that the GPWS has caused some unconscious reaction as if it were a real person. As this GPWS was a device that in certain circumstances gave erroneous information, this malfunction determined that in a critical moment, the captain did not accept the warning as a real one, but just another malfunction.

This analysis draws on a concept that we might call "sociocultural embodiment." The interaction of the captain and the device is not a simple, disembodied exchange. An HCI specialist could look at the design of the device as well as that of the human-computer interface and conclude that all HCI guidelines (derived from a traditional view of cognition) had been met. Such an analysis, we argue, would miss some key aspects of the human-computer relationship that derive from the embodied nature of cognition.

Tim Rohrer (1995) discusses "zooming windows" as an example of the power of physiologically embodied metaphors. When a person double clicks on a file icon on a computer, the image zooms out toward the person much in the same way as a page gets larger if you move a book toward your face. Similar animations have been adopted on personal data assistants and in the "genie" effect on the Macintosh operating system OS X when an item is moved on or off the temporary holding location, the Dock. Rohrer (1995, 8) says that:

zooming is more than just a nice touch however; it is one of the best examples of how user interface design can draw on common patterns of feelings. Zooming is a pattern of feelings that takes place in and through time; the realization that all feeling takes place in and through time is the most important step in thinking about users' bodies. . . . Zooming windows are an extension of the PHYSICAL WORLD metaphor, which draws on the common pattern of feeling we experience when an object approaches us. Though computer events usually happen fairly instantaneously by our standards, zooming windows are a deliberate attempt to make the PHYSICAL WORLD metaphor of the user interface to include both three-dimensional space and time.

Rohrer (2005) emphasizes the felt sense of embodied interactions. It is not just a case of knowing some interaction is good or bad; it is sense of feeling it. In this book, we explore the design of computing systems—including the process of SE and HCI—from the perspective of "embodied cognition." This term covers a number of theoretical positions that oppose the traditional "objectivist" view of cognition that has dominated the discipline since its beginning. Our interests are with computers, how people use computers, what they try to do with computers, and designing for new interactive media. In the remainder of this chapter we introduce some of the key concepts in both people and computers.

A Short History of Cognition

Throughout the 1940s and 1950s, disciplines such as psychology, linguistics, computer science, and philosophy found that they were thinking and talking about similar concepts. This gave rise to a new discipline, called cognitive science. Indeed, one history of the subject (Gardner 1985) points to September 11, 1956, as the real starting point—the middle day of a symposium on information theory at MIT when papers were presented by Noam Chomsky (on linguistics), George Miller (on human memory), and Allen Newell and H. A. Simon on computing. Each person was to become a central figure in their own field.

Cognitive Psychology was the title of Ulrich Neisser's (1967) book that applied these concepts specifically to understanding human thought from an individual perspective. The computer-processing metaphor was used just as it is, taking *meaning* as almost synonymous with information, and describing people in terms of an input, a process, and an output: "Cognitive psychologists investigate human information processing—how people encode, transform, create, and output information" (Haberlandt 1994, 25).

The computer metaphor THE HUMAN IS AN INFORMATION PROCESSOR was used to understand cognition. The two main assumptions of cognitive psychology are representation and process. Within cognitive psychology, there are many debates about what these representations might be like, such as whether the representations are analogous to the physical entities or whether they are abstract. But the question of representation is not controversial. Moreover, for the adherents of a strong cognitivism, knowledge has uniformly the same structure and format, whether it is a transient image, a memory, the meaning of a word, or a problem to be solved.

Once the question of representation has been established, the next issue concerns how to store such representations in memory. Cognitive psychologists ask, for example, whether there is a specific storage location (as in a computer), which is always the reference of a representation. Others argue that information is distributed over many locations as a neural network. Cognitivism is a term usually reserved for an individual, isolated approach to cognitive psychology. It has a whole repertoire of processes, strategies, moves, operations, procedures, algorithms, plans, goals, and so on that manipulate the representations. For example, if a person is to remember a phone number the processes would involve listening, writing the number down, trying to remember it, and recalling it when needed. In terms of cognitive psychology this would be encoding, recoding, storing, and retrieval. Encoding means that the number is recognized, next it is recoded for writing it down, and assuming that the number has been stored in memory, then it may be retrieved for later use.

This notion of cognition dominated psychology throughout the 1960s and 1970s. In the 1980s, a new computing paradigm emerged: parallel distributed processing (PDP). People such as David Rumelhart and James McClelland (1986) used this different computing metaphor to discuss cognition. The classical paradigm of centralized processes and an executive program in control was replaced by a new one: THE BRAIN IS A NEURAL NETWORK. In this model, thousands of relatively independent processing nodes are linked into a network. The strength of the different connections is altered during processing until the network settles down (or "relaxes") to provide a solution.

Another development in cognition is the notion of "distributed cognition." Starting his work in the 1980s, Ed Hutchins published *Cognition in the Wild* in 1995. Since then, he has been working with colleagues at the University of California, San Diego, to develop these ideas. One of them, Jim Hollan (et al. 2000, 175) states that unlike traditional theories, distributed cognition "extends the reach of what is considered cognitive beyond the individual to encompass interactions between people and with resources and materials in the environment."

In traditional views of cognition the boundaries of the unit of analysis are those of individuals, while in distributed cognition the unit is the cognitive process, wherever it may occur and taking into consideration the functional relationships of elements participating in the process. Distributed cognition is also concerned with the range of mechanisms that are assumed to participate in cognitive processes. This is a broader spectrum of cognitive elements than those assumed to exist in the limits of an individual brain. As an example of distributed cognition, researchers point to flying an aircraft. The successful completion of a flight is shared across the instruments in the cockpit, a pilot and copilot, the air traffic control staff on the ground surrounded by all their computers and notebooks, and so on.

Hollan and colleagues (2000) identify three different kinds of distribution of cognitive processes: across people, across representations, and across cultures. Socially distributed cognition focuses on the role that a group of people have in thinking and knowing and on the phenomena that emerge as a result of these social interactions. Second, cognitive processes make use of external as well as internal representations. These external representations are things such as notes, entries in logbooks, specialist measuring instruments, and other cognitive or information artifacts. An important ramification of this view is that designers cannot simply automate something. By changing the information artifacts that are involved in an activity, the distribution of the cognition required to undertake the activity changes.

People are social agents who live in cultural environments. Hence, there is an intertwined relationship between agents and culture. On the one hand, culture emerges as a result of the activity of human beings, and on the other hand, in its various forms of artifacts and social practices, culture shapes cognitive processes. Hollan and colleagues claim that this influence is particularly important in processes that are distributed over social agents, artefacts, and environments. This has a crucial knock-on effect for our analysis. If concepts are culturally shaped, then so is cognition. How we think about things is affected by history.

A conceptually related view of cognition is activity theory. Activity theory has its origins in the works of the Russian psychologist Lev Vygotsky beginning in the 1920s, but has only recently been recognized in the Western scientific community (see, for example, Leont'ev 1978; Bødker 1991; Bannon 1991; Nardi 1995). The concept of *activity* consists of a subject (one or more individuals), an object (held by the subject and motivating the activity), actions (goal-directed and conscious processes that must be undertaken to fulfill the object), and operations (former actions that have become routine and unconscious with practice). An activity is "a system that has structure, its own internal transitions and transformations, its own development" (Leont'ev 1978, 50).

An activity is directed toward a certain object (that is, a purpose or goal), and is mediated by one or more artifacts (which may include pieces of software, "thinking" artifacts such as language, and so on). Activities can only be understood given some knowledge of the object and the motive behind the activity. Significantly, activities need to be seen within a cultural and historical context; the term "cultural historical activity theory" (CHAT) is often used to emphasize this.

Since its beginnings in the 1950s, then, cognition has gone through many transformations as researchers have grappled with better ways of understanding and describing how people think. This quick tour through some of the main influences inevitably leaves out much detail and many other competing theories. The purpose of the tour is to illustrate that cognition is not a simple well-understood concept or theory. Yet these folk views of cognition have been profoundly influential on the design of computer systems and digital media.

Concepts of Software

There are many levels at which a computer (or any digital medium) can be described. There is the overall "architecture" of secondary storage, primary storage, processing unit, input and output mechanisms, and power source. Each of these has an impact on what can be done with the device, and each is becoming smaller and faster all the time. People are now predicting "speckled computing"—fully functional computers that are one millimeter cubed that can be sprayed from a spray can or spread around like “smart dust.” We have seen the amazing increases in secondary storage that allow people, for example, to put their whole music, photo, or video collection on a device no bigger than a pack of playing cards. We have mobile phones that contain more computing power than the standard computer of five years ago. Such features of digital media are incredible, but our interest here is in what we can make these devices do, using software. We take the basic digital architecture as given. We are interested in some conceptual description of what the computer does, the constructs of computer programming languages, and the different ways in which programming the computer has been conceptualized over the years. Even at this level, there is a huge amount that can be said, and there are specialist groups that research the psychology of programming and related areas.

A computer works by manipulating the contents of the locations in its memory. In the early days, computers were programmed in a machine *language.* This specified the location in the computer’s memory where data or instructions were stored, and where the result of any calculation should be placed. Soon after this, symbolic programming languages were developed that used more abstract terms to specify the actions of the program. Words such as move, add, store, and so on, included the specification of a storage location using names such as rate, pay, and so forth. These more abstract descriptions were then “assembled” into the machine code. This assembly language was still quite obscure, so more abstract means were used to program the computer, and more and more of the processing was moved from software into hardware. The main hardware revolution happened in the 1970s, when general-purpose microchips were developed. These have subsequently become incredibly small, and can be tailored and manufactured for specific purposes such as controlling the fuel input to a car engine or telling the time. The trade-off between hardware and software is one of flexibility. A more general-purpose hardware can be programmed to do many different things. Hardware with a more specific function is less flexible.

The more abstract programming languages began appearing in the 1950s and 1960s, and included FORTRAN (standing for “formula translation”), the common business-oriented language (COBOL), and the beginner’s all-purpose symbolic instruction code, BASIC. These languages adopted a procedural paradigm and are known as procedural languages. Sequences of instructions are written in the order in which they are to be executed. Other competing programming paradigms included the functional language LISP (for "list processing language") and the logic programming language, Prolog. These latter two were particularly popular for programming AI applications. Another important change that took place during this period was that the preferred term for describing instructing the computer changed from *computer programming* to *software engineering.* Software development was depicted using an industrial plant metaphor: SOFTWARE IS CONSTRUCTION.

The "object-oriented" (OO) paradigm of computer programming began at the Xerox Corporation's Palo Alto Research Center (PARC) with the development of the programming language Smalltalk in the early 1970s. In OO methods, the domain of interest (some "sphere of activity") is represented in terms of the objects that exist, the relationships between objects, and the messages that are passed between objects. This was an important change. The metaphor for thinking about software development changed from SOFTWARE IS A SEQUENCE OF STEPS to SOFTWARE IS A COLLECTION OF OBJECTS.

There were many competing methods during the 1990s for OO design and programming. Toward the end of the century, three of the main competing OO methods became united in the Unified Modeling Language (UML).

Objects are defined as: "an encapsulation of attributes and exclusive services [behaviors]; an abstraction of something in the problem space" (Coad and Yourdon 1992, 31). They correspond to "real-world" entities. The benefits of OO techniques include abstraction and encapsulation (otherwise known as "information hiding"). All computing is concerned with abstractions: with finding generic methods of representing things so that people can attend to the significant aspects and not get confused by the details. Objects are "viewed" from the outside, and people need not be concerned about how they are implemented. Objects can send and receive "messages"; they encapsulate the structure and processing that enables them to deal with those messages. Other features of the OO paradigm include polymorphism (that different object classes will treat the same message in different ways) and inheritance; objects can inherit characteristics from more abstract objects. The most popular OO programming languages are Java and C++.

The OO paradigm is still the dominant approach for software development. Recently, however, software objects have been programmed to be more independent. Rather than sitting and waiting to be instructed to do something, objects can be given higher-level "intentions" that they actively seek to achieve. Such systems are known as software agents. Agents operate in a variety of domains. For example, agents can move around a computer network, optimizing the flow of network traffic or checking for breaches of security. Agents in mobile phones actively seek out the strongest signal and automatically switch connections. Software in car engines optimizes fuel flow in order to maximize efficiency.

Increasingly, software agents are taking on these mundane tasks, leaving us, as people, to concentrate on the more interesting aspects of life. There are dangers and concerns of course. Computer viruses are software agents. Computers and other digital technologies connect autonomously with one another and can exchange data without us being aware of it. The advantages of the agent approach, though, is that they can be given relatively abstract, high-level instructions and be left to satisfy those as best they can. And so we see another change in how we think about software: SOFTWARE IS A SOCIETY OF AGENTS.

Human-Computer Interaction

People and software come together in the discipline of HCI. The concerns of HCI were expressed intermittently during the early part of the digital age. J. C. R. Licklider's "Man-Computer Symbiosis" in 1960 and Brian Shackel's 1959 paper are counted among the first writings to address the issues of people making use of devices along with the difficulties they might face. But the subject really started to attract interest with the publication of Ben Shneiderman's *Software Psychology* in 1980 and Don Norman's "The Trouble with UNIX: The User Interface Is Horrid" in 1981.

As with the development of both psychology and software, HCI was originally concerned with a person using a computer. Stuart Card, Thomas Moran, and Allen Newell published *The Psychology of Human-Computer Interaction* (1983) and introduced the discipline to the information-processing view of people that was so dominant in psychology. Applied to HCI, this perspective resulted in a number of detailed methods for analyzing and designing human tasks. Task analysis was to dominate HCI for the next twenty years. The basic conceptualization of HCI was that a person had a goal that they wanted to achieve. This goal could be accomplished by undertaking a number of tasks in a given order. Tasks consisted of subtasks and actions. Thus people formed a plan of action and followed it through: HCI IS FOLLOWING INSTRUCTIONS.

During this period, the emphasis was firmly on the human-computer interface: all those parts of the system that the user comes into contact with, whether physically, perceptually, or conceptually. HCI was practically synonymous with interface design.

Later in the 1980s, a new field emerged that focused on people working together through computers. This became known as computer-supported cooperative work. The stress here is on multiperson, distributed systems and issues of awareness of others, supporting collaboration and peripheral information. The centrality of tasks to HCI was also challenged by Lucy Suchman, who published *Plans and Situated Actions* in 1987. In this work, she argues that people do not simply follow plans; they react to changing situations.

When considering HCI, we recognize an initial constitutive metaphor: THE INTERACTION IS A DIALOGUE, CONVERSATION, OR COMMUNICATION. This had its origin with the introduction of operating systems in the mid- to late 1950s. These new "supervisor programs" were the intermediary software needed to represent the computer in all the interactions with the operator, programmer, or user. The consequence of using a linguistic metaphor applied to the code is that the interaction between humans and computers could be considered as a DIALOGUE. The user interface at this time (up until the late 1970s) was known as a command line interface. The interaction between person and computer consisted of the person typing in instructions, or commands, and the computer undertaking some processing and displaying the output.

A new metaphor, INTERACTING WITH THE COMPUTER IS DIRECT MANIPULATION, was formally characterized in 1983 by Ben Shneiderman. Yet it had begun to take form many years previously when Ivan Sutherland was defending his PhD thesis, *Sketchpad: A Man-Machine Graphical Communications System* (1963). It is interesting to note that Sutherland conceptualized this interaction in terms of communication—the dominant paradigm at that time—although it was a true interactive computer graphics. Shneiderman (1983, 57) described the new computing of the late 1970s that were using interactive computer graphics as follows:

In trying to understand the commonalities across these diverse interfaces, I began to notice a certain pattern. The first was a visual representation of the world of action. The objects of interest and the actions of interest were shown on the screen. . . . These new systems showed the objects of interest, and when actions were taken the results were also shown immediately and continuously on the screen. A second set of principles that also seemed important were that the actions were rapidly executed, incrementally described and reversible. Furthermore, in direct manipulation systems, pointing, selecting and dragging replace the need to type commands. For example, you could drag an icon towards a folder or you could bring it back.

While Shneiderman is credited with coining the term "direct manipulation," it is fair to say that the designers at Xerox PARC were responsible for creating the interfaces that he was observing. Shneiderman detected some patterns or principles of interaction—visibility of the objects of interest; rapid, reversible, incremental actions—to introduce a new paradigm and a new metaphor. It was the Xerox Star computer, the Apple Lisa, and finally the Apple Macintosh in 1984 that demonstrated the principles.

Finally, another approach called ubiquitous computing (Weiser 1991, 1993) appears as enhancing or generalizing direct manipulation. Weiser (1991, 94) says that:

> the idea of a "personal" computer itself is misplaced, and that the vision of laptop machines, dynabooks and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology. Such machines cannot truly make computing an integral, invisible part of the way people live their lives. . . . Such a disappearance is a fundamental consequence not of technology, but of human psychology. Whenever people learn something sufficiently well, they cease to be aware of it. . . . Only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals.

Here a new slogan is emerging: THE COMPUTER IS DISAPPEARING or THE COMPUTER IS BECOMING INVISIBLE, according to Norman (1998). In order for computers to disappear from our awareness, they would have some features that everyday objects have: a natural, intuitive form that indicates functions and content that resides in the background, or periphery, coming to the fore when needed. At this point it is interesting to listen to some skeptical voices about ubiquitous computing, who observe that "we will still be frustrated, but at a higher level of functionality, and there will be more of us willing to be frustrated" (Odlyzko 1999, 1).

Framing the Problem

And so we arrive at the present, and the digital age continues apace. The problem we have is how to design digital artifacts, how to design for new spaces that are full of computing devices, and how to design for people who are wearing these devices. Information, data, and multimedia content are easily passed from one person or place to another, transformed from one medium to another, and displayed on one device or another. What new forms of interaction will there be, what are the potential pitfalls, and what are the potential benefits?

Unfortunately, we cannot answer these questions yet. What we can do is to develop literacy in designers, and provide designers with an understanding of the underlying concepts of the digital medium and its interaction with people. We may even influence some changes in design—moving it away from the idea that we are trying to make something literally happen toward an understanding that interaction with and through digital media is fundamentally figurative.

We have seen that SE methods have changed over the years according to the metaphors that people have used to conceptualize what they are doing: THE SYSTEM IS AN INDUSTRIAL PLANT; THE SYSTEM IS A COLLECTION OF OBJECTS; and THE SYSTEM IS A SOCIETY OF AGENTS. HCI is still grounded in both metaphors: HCI IS COMMUNICATION and HCI IS DIRECT MANIPULATION. There are new proposals, such as informal interaction or sketching interfaces (Landay and Myers 2001), that try to assist people with new, informal activities, such as writing, drawing, or designing. Pattie Maes (1994) has conceptualized HCI as A SOCIETY OF INSECTS, and Alan Kay (1993) has a vision of HCI as the INDIRECT MANAGEMENT OF AGENTS. Indeed, there have been debates between protagonists of these different metaphors for HCI (Shneiderman and Maes 1997).

We are moving into yet another new era: THE COMPUTER IS DISAPPEARING. We expect interaction with digital media to become much more physical and less screen based. People will interact with digital technologies through touch, manipulation, and gesture; interaction will increasingly be embodied. People will move through environments embedded with digital artifacts, and will interact with and through technologies in new ways. These new environments promise to be highly complex in terms of their accessibility, functionality, and usability. Conceptually, it will be difficult to determine what can be done, how it can be done, and where it can be done.

We also suspect that our traditional understanding of cognition is flawed. Hence, the principles of HCI are based on an inappropriate view of how people think and act, and on the relationships between action and cognition. SE methods are predicated on an objectivist view of the world that is inappropriate.

Finally, we point to a fundamental difficulty for the design of digital media: people and digital media are different. This is the problem of HCI-SE. In order to design digital artifacts, we have to specify instructions in some way that the computer can process, but this is inevitably fundamentally different from how people want to work. There is a mismatch between the rich, complex, nuanced activities of people and the inflexible demands of digital artifacts.

In the next few chapters we explore these three intertwined issues. We introduce a view of cognition that is grounded in ideas of embodiment: we think and act the way we do because we have bodies and live in human societies. We use this to provide an alternative perspective on HCI and SE, the conceptual foundations of these subjects, and methodologies for the analysis and design of human-computer systems. The aim here is to offer insight and develop literacy in the analytic approach that we adopt. We also look at design and how embodied cognition can change the way we approach design by foregrounding the figurative (as opposed to the literal) nature of interaction, and then move toward a critical approach to the design of digital media.