tools, gardening tools. Tools for painting and drawing, tools for baking and cooking. Gloves and bats for baseball, fishing rods, boats, cars. A pleasurable tool is one that is made with quality, that fits the mind and body, that makes it fun to do a task. Tools that help. Tools that do not get in the way.

Appliances should add pleasure to our lives. They can remove the drudgery of tasks and make them enjoyable, fun even. Hence "pleasurability" as a design axiom for information appliances.

4
What's Wrong with the PC?

If there is any device that characterizes modern technology, it is the personal computer (PC). It gives all of us computational power that earlier scientists and business people could only dream of in their wildest fantasies. The world has been transformed once again, this time through the power of technologies available to the individual: communications, computation, access to information of all sorts in multifarious shapes and forms, all available through our personal computers, the instrument of power. But despite this, the PC is hardly a technological blessing; it is as much a curse as a wonder, and it is attacked as much as it is praised.

What's wrong with the PC? Everything. Start with the name. The personal computer is not personal nor is it used to do much computing. Mostly, it is used for writing, reading, and sending things to one another. Sometimes it is used for games, entertainment, or music. But most of the time it is using us. When I prowl the halls of my workplace, I often see people on their hands and knees beside their computer. No, not praying, but installing new things, rebooting, checking the cable connections, or just muttering under their breath. The *personal* computer isn't very personal. It's big and clumsy, sitting there on the desk, occupying space, requiring more and more time to maintain, requiring lots of help from one's family, friends, and neighbors. Rather than being personal, friendly, and supportive, it is massive, impersonal, abrupt, and rude.

Rebooting. Even the word is strange technical jargon. The word *booting* is derived from *bootstrap,* a small loop at the side or rear of a boot to help the owner pull it on. To bootstrap is to pull yourself up by those straps, not that many people have them on their boots anymore. A computer, when first turned on, starts up with no knowledge. No matter that you have long been using it, when first turned on it is completely ignorant of the past. In order to get started, it must find its operating system, that massive program that is the infrastructure for all else. But the computer doesn't even know how to find the operating system; It has to start off with a tiny program that has been permanently implanted in its memory and use that to load a larger program that in turn lets it load even larger programs that let it load the operating system. And, of course, the whole purpose of the operating system is to give you, the user, some way of calling up the stuff you want to work on. This process of bringing in a program whose sole purpose is to load yet a larger program is called *bootstrapping*. But why am I telling you this? Why do you need to know? You shouldn't have to know or care.

Boot, RAM, DRAM, ROM, floppy disk, hard disk, megahertz, gigabyte: Why should we want to know any of these terms? Answer: We don't. We are told we need to know because we are driven by technology and technologists.

The personal computer tries to be all things to all people. It casts all the activities of a person onto the same bland, homogeneous structure of the computer: a display screen, a keyboard, and some sort of pointing device. This is a certain guarantee of trouble. Any single set of tools is a compromise when faced with a wide range of tasks. It's like trying to do all your cooking with a knife, a fork, a spoon, and one saucepan over an open fire. It can be done. Campers do it. In olden times, people managed with less. But the myriad cooking utensils, stoves, cooktops, ovens, and so forth are in the kitchen for a reason; they make life easier, they do a better job.

A second problem is that of complexity. Try to make one device do many things and complexity increases. Try to make one device suffice for everyone in the whole world and complexity increases even more. The single, general purpose computer is a great compromise, sacrificing simplicity, ease of use, and stability for the technical goals of having one device do all.

We humans are social beings. We work best with other people. The real promise of the new technologies comes with the merger of the communication and computing industries, keeping people in touch with one another, communicating, socializing, working together, playing together. Sure, there are times when we do things alone. Creative work requires solo, silent, concentrated periods. Reading is a solitary activity. But much of our time is spent in talking to others, whether by mail, on the phone, or in person. The exciting new services are social and interactive, yet here we are, trying to build them on a device whose first name is "personal."

How many hours a week do you spend keeping your computer working, updating hardware or software, reading instruction manuals, help files, or the monthly PC magazine? Too many. How many hours a day do you spend keeping your TV set or telephone or refrigerator working? Updating it? Reading instruction manuals and help files? Not very many. There is a lesson to be learned from that contrast.

Today's PC has gotten too big, too expensive, too complex, demanding more and more attention. It is a general purpose machine, which means it can do anything. This is not a virtue.

Take another look at the Swiss Army knife, one of those knives with umpteen blades. Sure, it is fun to look at, sure it is handy if you are off in the wilderness and it is the only tool you have, but of all the umpteen things it does, none of them are done particularly well. Yes, my Swiss Army knife has a screwdriver and scissors and corkscrew—it even has a knife blade—but when I am home, I much prefer to use a real screwdriver, a real scissors, a real corkscrew, and even a real knife. Not only are the simpler devices superior, but they are easier to use; with the Swiss Army knife, I invariably pry up the wrong blade until I find the one I am seeking.

Now take another look at the PC: It does everything, serves all masters, works all around the world. The end result is that it comes with large instruction manuals, multiple layers of menus and screen icons and toolbars. Each item presents me with options that I neither understand nor care about. Today, the PC has become more complex than the old main-frame computer it was intended to replace. Why is this? In part because of the business model of the personal computer business, in part because we have let ourselves be trapped.

But finally, and much more important, I don't normally need to compute anything, so why do I want a computer? Sure, I need to write, yes, I am a habitual user of email, and yes, I use my personal computer for all sorts of activities; but the machine itself is an imposing technology, and I don't want to be controlled by a technology. I just want to get on with my life, enjoy my activities and friends. I don't want a computer, certainly not one like today's PC, whether or not it is personal. I want the benefits, yes, but without the PC's dominating presence. So down with PCs; down with computers. All they do is complicate our lives.

Don't get me wrong: There have indeed been many important virtues of our new technologies. After all, it is things that make us smart, things that allow us to represent our ideas in a permanent manner, things that allow ideas to be transmitted from one generation to another, that allow people to collaborate over time and space. We shouldn't give up the virtues. Moreover, I'm technically savvy, well versed in the technology. In my home, where I write these words, I have four desktop computers, two laptops, two laser printers, and an ethernet network. The problem is that the clumsiness and flaws of the technology tend to overwhelm the virtues, at least for most people. So, what we need is to keep the virtues of the machines without the overhead, without the clumsiness. We need to move to the third generation of the PC, the generation where the machines disappear from sight, where we can once again concentrate upon our activities and goals in life. We are ready for the generation of information appliances.

The First Two Generations of the PC

We are now in the second generation of the personal computer. The first generation was the era of the Apple II and the IBM PC. Before that, computers were large and expensive. They were sold to companies, governments, and universities. Then, suddenly, for the first time, computers were available as small, relatively inexpensive machines that could

be bought and used by average individuals. These were awkward machines, quite puny by today's standards, but they were able to perform some useful functions, primarily word processing, creating spreadsheets, and playing games. These machines were limited in capability, difficult to learn, and difficult to use. But these early machines made a difference, primarily because they empowered their users. For the first time people could do their own accounts and budget projections without waiting for information technologists in their company to get around to them. The first word processors enhanced the ease of writing and revising. Computer games started to evolve. And thousands of individuals developed educational tools for schools. For the first time people were in control of their computing tools.

The second generation of the PC was that of the Graphical User Interface—the GUI (pronounced "gooey"). This is where we are today. The first successful machine of this generation was the Apple Macintosh, which followed upon the unsuccessful Xerox Star and Apple Lisa. Soon, other companies followed, IBM with OS/2 and Microsoft with Windows. In the GUI generation, the primary philosophy is "ease of use," making the complex machinery of the personal computer relatively simple to operate. And therein lies the rub: The machine is indeed complex, and the GUI goal is to sugarcoat this complexity so that it won't be noticed. Alas, complex things are truly complex, and an attractive image on the screen doesn't overcome the fundamental problems. Rather than trying to make a complex machine easy, the better way would be to make a simple machine in the first place.

What's Wrong with the Graphical User Interface

The Graphical User Interface was right for its time, but wrong for today. Why? First, it has outgrown its usefulness. The basic interface design was developed back in the days when personal computers were small by today's standards. The essential design principle was to make everything visible, so that instead of memorization of archaic commands, one could see the entire array of possible commands, file names, and directory names. Second, the basic operation was by selection, dragging, and direct manipulation. These principles worked well as long as the machines themselves were small.

The graphical user interface really worked. Buy a new computer program, take it home, stick the floppy in the computer, and use it right away. Don't bother to open the manual; why would you need to do that? Just pull down each menu and look at it, and you have seen all the commands. Don't understand some? Just try them out—you couldn't do any harm, and everything is reversible anyway.

But those were the "good old days." The computer had a really small memory (128K, or 128 thousand bytes), and a small floppy diskette (that could only hold 400K, 400,000 bytes), and no network or hard drive. The programs were small and simple. And you couldn't store anything permanently in the computer. So making everything visible worked, for there wasn't that much to be visible. And learning by trying out everything worked, because there wasn't that much to try out.

Today, machines have expanded in power thousands of times. Today, all machines have internal disk storage, and many are connected to networks that enable them to receive millions of documents from locations all over the world. My home computer has almost ten thousand files in it, most of which are meaningless to me. My company network has hundreds of locations all over the world, each with thousands or even millions of documents. The design philosophy of making everything visible fails miserably in this context.

What's the matter with the graphical interface today? The solution doesn't scale. Making everything visible is great when you have only twenty things. When you have twenty thousand, it only adds to the confusion. Show everything at once, and the result is chaos. Don't show everything, and then stuff gets lost.

But although the computer has changed dramatically since the 1980s, the basic way we use it hasn't. The internet and the World Wide Web give much more power, much more information, along with more things to lose track of, more places to get lost in. More ways to confuse and confound. It is time to start over.

Reality check: The early Apple Macintosh computer was small and convenient, but the screen was small, too—small and inconvenient. Tiny is more like it. No color, just black and white. And it was slow. S-L-O-W. And no hard disk, just floppies, so you had to store your stuff on piles of floppies. If you didn't label those floppies and then keep the labels up to date (and who could ever do that?) it was awfully hard to find the stuff again. And if you think they don't hold much today, well, they held a lot less then. No large capacity storage devices, no CD-ROMs, no DVD, no hard disk, no networking. But easy to use? Yup. Nice and easy. You truly did not need to read the manuals. Those were the good old days, which, like most good old days, are happier in memory than in reality.

Why You Really Don't Want to Use a Computer (even though you think you do)

Do you really want to use a computer? Do you want to use a word processor? Of course not. The fact that you think you do is the triumph of marketing and advertising over common sense. Now, maybe if you are a confirmed technology addict, or a computer programmer, sure you love using computers, but not the rest of us. We want to get on with our lives.

I don't want to use a computer. I don't want to do word processing. I want to write a letter, or find out what the weather will be, or pay a bill, or play a game. I don't want to use a computer, I want to accomplish something. I want to do something meaningful to me. Not "applications," not some bizarre complex computer program that does more than I ever want to know about and yet doesn't really do exactly what I need. I want computing that fits my activities. I want the technology hidden away, out of sight. Like electric motors. Like the computers that control my car.

Once upon a time, cars were difficult to use. They had all those controls and meters and gauges. Spark adjustment, fuel priming, choke, and throttle. And there was little standardization, so every car worked differently. Some were steered with wheels, some with tillers, some with levers. The speed of the engine was adjusted by foot, by hand, by pedals, levers, or knobs. In the real early days you had to take your mechanic along with you when you went for a drive. To start it you had to prime the fuel line, adjust the spark setting, set the choke, open the throttle, and then stand outside the car beside the engine and crank it over by hand, being careful that it didn't start at the wrong time and break your arm. There were gauges for all sorts of things.

Today, now that the cars are extremely reliable, all you really need is a speedometer to tell you how fast you are moving, a fuel gauge to tell you when you are running low, and that's about it. Anything else can be done with warning lights or messages that only come on when they are needed, ideally to warn you a bit before serious problems arise, when there is still enough time to take corrective action or get help.

Even the fuel gauge isn't what you want. You don't really want to know how much fuel you have left (no, honest, you don't); what you really want to know is how far you can drive. Some cars provide this information. The normal fuel gauge can't do this because it is a simple float that rides up and down on the surface of the fuel, allowing its level to be translated into how much fuel is left in the gas tank. To translate fuel level into how many more miles of driving is possible requires some computation. The fuel level has to be converted to the amount of fuel, either liters or gallons. Then, an estimate of the efficiency has to be made: How much gasoline have you been using per mile or kilometer recently? Multiply the efficiency by the amount of fuel and you have the predicted range. Do the same computation in kilometers and liters as well as in miles and gallons so you can accommodate inhabitants of both the more advanced countries that use metric measurements and lesser advanced countries that don't. These computations require a computer; hence the moral of this story.

Computers ought to be like the embedded ones that tell you how far you can drive with the amount of fuel remaining in the fuel tank: invisible, automatic, and useful. It's invisible, so you don't have to do anything to it. It provides valuable information. Drive more efficiently, and the remaining distance goes up; drive less efficiently and the distance goes down. It wouldn't be difficult to add a time estimate: "Twenty minutes to empty."

This is the way the fuel tank meter ought to be: Get rid of the current gauge that tells what fraction of the fuel tank still has fuel in it and replace it with one that says how far or how long we can go. Notice, too, that this computer-controlled gauge is very limited in its functionality; It tells the range of driving with the remaining fuel. Nothing more, nothing less.

This is the way computers ought to be, not just in the car, but in the home, at schools, and in the office. Useful for doing things, for getting answers, for having fun, presenting us with the information we need to know, information we can use directly without further thought. According to this model they will be far easier to use. They will be designed specifically to fit the task, to fit the needs of their users. This also means that they will be specialized, so we are apt to need many of them. No problem, because they will be like all our other appliances: We buy just the ones we want, just the versions that fit our lives. Their simplicity and utility make up for their specialization.

Why the Personal Computer Is So Complex

The major problem with today's PC is its complexity. The complexity of the PC is pretty fundamental; it is built into its foundation. There are three major reasons for the complexity: the attempt to make a single device do too many things; the need to have a single machine suffice for every person in the world; and the business model of the computer industry.

Make a single device do everything, and each task will be done in a manner that is adequate, but not superior. As I have explained before, a multiple purpose device cannot be optimized for any single task; it has to be a compromise. Its physical shape, the nature of its controls and displays, are all compromises. Imagine a musical instrument that combines the violin, guitar, flute, and piano keyboard. Can it be done? Oh yes, your handy-dandy music synthesizer program will produce the sounds of all these instruments from a typewriter keyboard. But will it be inspired music? Will a real musician use it? Of course not.

The second cause of the computer's inherent complexity is that computer companies make products intended to be used by hundreds of millions of people all around the world. "Know your user" is the mantra of good design, but how can you possibly know your user when it could be millions of people, of every age, every educational level, and every social and cultural group, while hoping to satisfy every conceivable need and style of work? Because each country, each culture, and for that matter, each individual, has different interests and needs, this means that the product has to have a large set of features and operations in order to satisfy everyone who might use it. No matter that any individual is apt to use only a very small number of features or commands; to satisfy the world market, the product must have everything. Making one device try to fit everyone in the world is a sure path toward an unsatisfactory product; it will inevitably provide unnecessary complexity for everyone.

Finally, there is the business model, the strategy that the computer industry follows in order to ensure that it can make a profit year after year. There is nothing wrong with this: A company that fails to make money soon goes out of business, and then it cannot be of use to anyone, even if its products were loved and respected. But the strategy adopted by the computer industry is also one that dooms it to an ever-increasing level of complexity in its products. Let us look at this issue in greater detail.

How the Business Model of the PC Guarantees Complexity

All companies need to make a profit in order to survive. It doesn't do anyone any good to make great products if the company fails for a lack of funds. Now, how do you make a profit in the computer business? By selling computers and software, right? Yes, but the problem is that the vast majority of people, if they need a computer at all, only need one.

And each user of a machine only needs one word processor, one spreadsheet, one email program.

Think about it. Suppose you could buy a computer and software that would make you completely happy. From your point of view, that's good. But, oops, from the point of view of the computer manufacturers, that's a problem. How would they make money if their customers were so happy that they wouldn't need to buy anything else? Horrors. There is only one solution to their dilemma: They need to make you unhappy with what you have and make you want something else. Isn't this wonderful: An industry whose business model is based upon the need to make their customers unhappy.

Every year, the computer companies have to convince you that this year's version of software has features in it that you simply can't live without, even though you have lived without them all your life up to now. Moreover, because you think you are already happy with your software, the new software has to do everything that the old software did, while adding those exciting new features that you can't live without. Each year the hardware manufacturers make their systems more powerful, with more memory capacity, and faster, better graphical display capability, all absolutely necessary, of course, if you are to be able to use that new software with all those new features you suddenly can't live without. Year after year after year there will be new releases of that original, perfectly adequate software, with new feature after new feature added. Year after year you will need faster, bigger, more capable hardware. The result: guaranteed complexity.

All industries have a problem of ensuring a continuing revenue stream from their customers. They need a business model: a plan for ensuring a continual stream of revenue. The business model for some industries is trivial. If you sell consumable goods such as food, people eat it, and therefore need to buy more. Similarly, in the newspaper business, fresh news continually arrives, so customers naturally seek your services to find out about the latest events. In other businesses the life of the product is made artificially short, by making it into a fashion. Once things become fashion, a whole new industry of trendsetters emerges to law": "Software is a gas, it expands to fill its container." Moreover, he added, "It's a good thing for the computer industry that computer power expands so rapidly. This way we can build bigger and fancier software that require you to get a bigger and faster computer, so we can use up all that space too."[2]

Think about it. When you go into a store to buy a stove or refrigerator, television set or telephone, you are bombarded with rows of almost identical-looking items, each barely distinguishable by price, perhaps by appearance, and by a comparison list of features. Whether or not you can use the device, whether it really does the job for you is usually not a major decision point, even if it should be. You have been taken over by the channel.

Whatever happened to the consumer? Whatever happened to the notion that one should solve the consumer's needs, which are really expressed in such terms as having fun, doing homework, writing letters, and the like, and not in megahertz and megabytes? What happened?

The computer industry works under a peculiar view of the world. The goal is to manufacture a machine that can do everything, that fits all people with the same basic hardware and software, that provides applications that have little to do with real work, and that grow ever more complex over time. It's a great business to be in, but a horrible way to affect people's lives. There are better ways to serve the customer, better ways to make money.

Activity-Based Computing

With today's PC, we buy the hardware, the computer, in order to support computer programs, also known as *applications*. Applications: what a terrible term. What a terrible concept. Applications have little to do with the tasks that people are attempting to accomplish. Look. We don't do word processing; we write letters, or memos, or reports, or notes to ourselves. Some of us write books. I do not want to go to my computer to do word processing. I don't want to go to my computer at all. What I do

want is to be able to write, with a tool that fits my needs. When I write, I need some way of getting my ideas onto paper or screen, some way of reviewing them, of outlining and restructuring. I need to be able to incorporate notes I have made and sometimes drawings or photographs.

When I write business memos, I need a different writing tool. I may want to insert some budget tables and calendar schedules. When I write a letter I may want the letters I am responding to, and perhaps a calendar, and maybe my address book. Each task has its own special requirements, each of which encompasses several different applications. Today's applications have far too much power for the use I make of them, yet lack all the necessary components for any given task.

On top of all of this, life is filled with interruptions. It is a rare event when I can finish a task at one sitting. At almost any task I am interrupted by other people, by the telephone, by the next scheduled event, or because I need other material and can't proceed without it. Sometimes more urgent matters intrude, and sometimes I simply need to stop and do other things, such as eat, or sleep, or socialize.

People do activities, and the software ought to support this. At Apple Computer, we called this approach "Activity Based Computing" (ABC), and together with a hardy band of souls[3] we tried to interest the company in the notion that software based upon ABC would fit the lives of our customers better than the traditional application model that we and our competitors were selling. (What happened? It's a long story, so see the endnote.[4])

The basic idea is simple: Make it possible to have all the material needed for an activity ready at hand, available with little or no mental overhead. Tools, documents, and information are gathered together into packages maximally designed for the particular activities in which they participate, without interfering with other activities. Of course, it must be possible to make changes in the choices and to switch rapidly and easily among the activities. Finally, items not needed for the current activity are hidden so they do not distract and do not take up valuable work space.

An *activity* is a goal-directed set of tasks. Activities, tasks, and actions provide a hierarchy; an activity is composed of tasks, which in turn are composed of actions. Examples include "doing one's mail," "doing the weekly home banking activity of paying bills and balancing the checkbook," "writing a technical report." Tasks are lower-level activities, aimed at fulfilling particular subgoals of an activity. The activity of doing the mail will have in it the tasks of "read the new mail," "write a new message," and "forward the mail from Sonia to Fred." Similarly, the activity of "banking" might have subactivities, such as "balance the checkbook," or "send payment to a merchant." Actions are a lower level of interest and refer to such things as selecting a particular menu command, or typing a particular name or phrase, or naming a file. In the scientific research field of activity theory,[5] of which this set of specifications is a special case, there are further distinctions. Thus, an *operation* would be the physical movement of the hand in order to move the mouse so as to highlight a menu, with yet another operation being to move the appropriate finger so as to depress the mouse button and select the item of interest. Operations make up actions. Actions make up tasks, and tasks make up activities.

Work within any single activity can take a long time. It may involve numerous people. As a result, it is necessary to allow different people to share the activity spaces, and to figure out how to coordinate the work so that one person's actions do not interfere with another's.

Because activities take place over extended periods, it is necessary to make it possible to return to the tasks without disruption. If you are interrupted while doing an activity, you should be able to resume at a later time, whether it be an hour or a month, and find the activity space exactly as it was left, with all the items and tools in the same place and same state. If the cursor was in the middle of a highlighted word, it is still in the same place, with the same highlighting still present. All this would aid memory, would aid the resumption of the task, and would be built to reflect the way people really work. In everyday life there are multiple interruptions and it is important to be able to resume work at some later time exactly where one left off. This requires restoration of the exact context of the activity.

Activity spaces could be shared with other people or copied from one machine to another. A company might wish to provide standardized activity spaces for its procedures, such as for filling out expense reports or purchase orders. Small software companies might provide specialized activity spaces, much as stationery stores provide a wide variety of forms and notebooks. Users could then further customize these spaces to meet their particular requirements.

This would be a very different way of doing software than the homogeneous, super-duper general purpose software packages we now must use. Instead of long menu commands, one would have a chest of tools from which to select, much like working on a project in the home, where you select only those tools needed for the task and have only those at the worksite; activity spaces would allow just the needed selection. But, just as in the home where it is possible to go back and get another tool, it is possible in an activity space to add or subtract tools as needed.

Activity spaces are not a magical cure to all that ails the PC. The "C" in ABC still stands for "computing." Many of the negative characteristics I have described for the PC would be unchanged. All activities would still be mapped onto the very same set of interface tools: a screen, a keyboard, and a pointing device. And the same machine would still be doing everything, with the extra requirement that it manage the variety of activity spaces and tools that each user would need. Activity spaces are probably difficult to implement in today's world of the personal computer, although there have been numerous attempts.[6]

A far better approach is to implement ABC without the "C"—without the computer. The goal is to make it so that the tools match the activities. There is an alternative way of getting to this state: Build special purpose devices, information appliances, where each device is tuned especially for an activity. With separate devices, some of the properties come automatically. If you are interrupted, just put the device away. When you wish to resume, pick it up and get to work. There would be no interference from other activities, no problem of keeping the original state. If we made special devices for activities, we could tailor them appropriately. The banking activity could have a special check printer and dedicated connection to the bank and to your stock broker. The letter writer could have a built-in address book and print letters and envelopes.

The main barrier to the introduction of technology that is aligned with people's real needs and desires, with people's real activities, is the mindset of the computer industry. This industry has grown up being dominated by technology. The result has been the development of powerful tools that have become essential to modern life. The computer industry feels vindicated: It has been highly successful. It has prevailed in the face of skepticism. And it did it all through the power of modern information processing technology. Why should it change?

The problem is that the resulting device is technology-centric. To make it usable by the vast majority of people who lack the detailed technical skills, the industry has been forced to add all sort of add-ons: wizards, help systems, telephone support lines, books, training courses, internet sites that feature the answers to "Frequently Asked Questions," whose numbers have grown so large that we now need help systems to navigate through all those answers. All these add-ons contribute to the complexity; now, in addition to the ever-increasing complexity of the computer applications, we must cope with the ever-increasing complexity of the help systems and support services. The computer industry is stuck in a rut from which it can't escape. Its very success has driven it further and further down a path of no return. Its business strategy is caught in the endless loop of added features, continual upgrades, and, as a result, ever-increasing complexity and every increasing help systems to let us cope. The only way out is to start all over.

There are many hurdles in the way of information appliances, but the goal is worth it: devices that fit the person, that fit the task. Devices that are easy to use, not only because they will be inherently simpler, but because they fit the task so well that to learn the task is to learn the appliance.

Now let us take a look at the fundamental issues, the better to understand how to do better. In the next chapter I look at problems with the PC and the attempts to overcome them. I conclude that these are all

doomed to fail. The problems are too fundamental; there is no simple magical cure. In chapter 6 I point out that the infrastructure is wrong. In chapter 7 I examine the mismatch between the needs and abilities of people and the requirements of machines. People are analog and biological; information technology is digital and mechanical. Being digital may be good for machines, but it is bad for people. This sets the framework for chapter 8's examination of why things have become so difficult to use.

In the final chapters of this book I propose an alternative approach: a human-centered development process coupled with a set of disruptive technologies, the better to yield a family of information appliances designed to fit human tasks, tailored for human needs and abilities.