ONE

IT'S JUST A PRODUCT!

Computer software is so immensely powerful, so fearsomely complex, and so deeply embedded in our daily routine that it's usually considered to be quite unlike all other consumer goods. To most of us, its workings are invisible and incomprehensible. Yet there's nothing magic about software. It's just a product—a man-made tool that is developed, manufactured, marketed, licensed, and sold.

Software has much in common with every other product, with one very important difference:

> *We have come to accept that the software we use won't work in the way we expect.*

We maintain high expectations for every major product we buy, especially when we purchase or lease any form of business equipment—except one. Our experience with information technology has taught us that whenever we plan to install new software,

whether it's a basic word-processing program or a multimillion-dollar enterprise system, the most realistic viewpoint is to hope for the best and expect the worst.

Unfortunately, our low expectations concerning software products are justified. After a 10-year global survey of more than 50,000 information technology projects purchased and developed for firms both large and small, The Standish Group, a Massachusetts market research firm specializing in software, reported in 2004 that less than one-third of these projects were delivered on time, on budget, with the required features and functions. More than half of them came in late, were over budget, or lacked required features, and the remainder were canceled or never used.

We possess every resource we need to produce business technology that fulfills all its promise. By making small, low-tech adjustments to existing software systems and by changing the way we specify, evaluate, and develop new systems, we can overcome their shortcomings and exponentially improve their performance—but only if we begin to change the way we think about them.

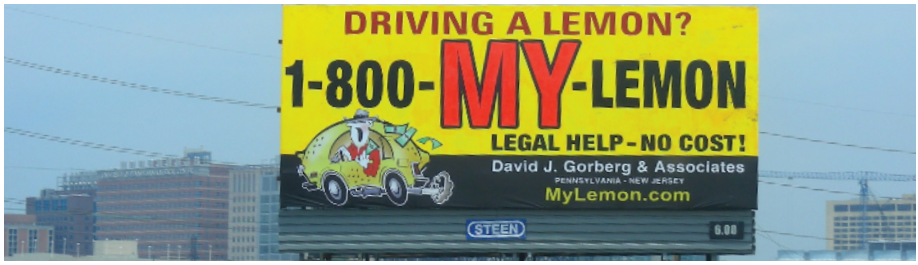
THE WORLD'S BIGGEST LEMONS

Whenever we buy a product, we have certain expectations.

If we decide to order a powerful new luxury car, we expect that all of its features will perform in a predictable way, from the steering wheel to the sound system. We have every reason to expect that this car will give us a smooth ride and an absolutely great experience. But imagine being told by the dealer that once you take delivery you should expect a period of adjustment because at first the gauges on the dashboard may seem confusing, and the steering wheel may be a little tricky to maneuver. What would you think if you researched consumer reviews of this model and found out that, overall, your chances of being satisfied with your purchase were only one in three? Would you be reassured if the dealer explained that the purchase price included the cost of sending a manufacturer's representative to ride alongside you for several hours each week to provide training? There's no question that you'd take your business elsewhere. But what if every dealer of every automotive brand in the United States and Germany and Japan told you the same thing?

Why is so much of our software so unsatisfactory? It's not for a lack of resources or planning. For most businesses, information technology is a priority that may represent their biggest investment. Highly detailed specifications for these systems are written by expert technologists in collaboration with some of the best minds in the business community, and these products are developed by brilliant technicians. Breathtaking advances in technology have become commonplace, and the business of testing software has become a $13 billion world market. Yet we still struggle to operate the software that serves our everyday needs. We greet each successive upgrade cautiously, anticipating problems with new features while hoping that old features will be easier to use; but time after time, when we face a computer screen and touch a keyboard, we find that something is amiss.

Disappointed buyers of expensive software "solutions" often find that they have little recourse. If your car turns out to be a lemon, you have strong protection under both federal and state laws. In fact, if you pay more than $25 for almost any product that doesn't work properly despite its written warranty, the Magnuson-Moss Act will back you in court, and every state has enacted the Uniform Commercial Code, which enables buyers to obtain satisfaction for goods that fail to perform according to the terms of their contracts. But standard purchase and licensing agreements for software



Is it a lemon? If it runs, it's not a lemon. (If the handling is rough—tough!)

contain limitations on liability, so buyers of unsatisfactory software often have little recourse unless they have the foresight, the negotiating muscle, and the expertise to modify these contracts. As a result, the owner of a $15 million software product that turns out to be too difficult to use has much less chance of obtaining a refund or a replacement than the buyer of a defective DustBuster.

Not even a lemon law would protect the buyer of an enterprise system that is engineered to be technically perfect but designed so clumsily that almost no one can understand how to use it. When it comes to buying software solutions, the most effective way to obtain full value from your investment is to ask the right questions.

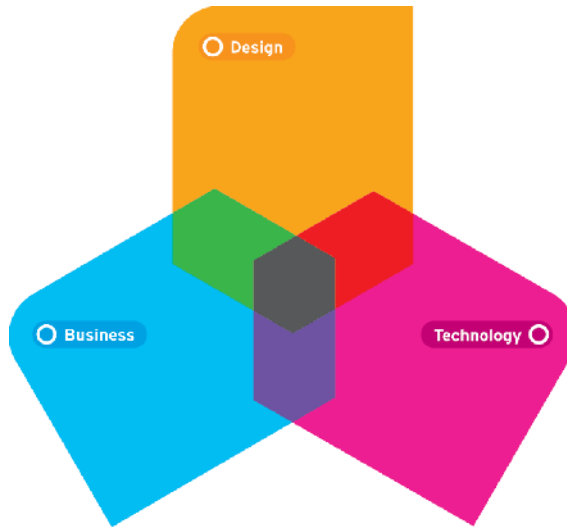
THE CHECKLIST

Buyers of business software generally use two kinds of checklists to evaluate these new products: one for the business requirements and another for the technology.

To identify the business requirements, business owners usually make a list of all the tasks that the software needs to support, and they use that inventory as a yardstick to compare the features and functions of various software systems.

To answer questions about the technology of competing software products, the chief information officer asks:

What's the vendor's reputation?
What's the industry buzz on the vendor's platform?
What are the hardware considerations?
How does rollout happen?
How do we receive updates?
Are the updates reliable?
What kind of support does this vendor offer?



Sweet spot—the interaction of business, technology, and design

But all too often, no one asks questions that will reveal whether the software will be a good fit—whether it will do the job the way it needs to be done, whether it can be easily understood, or even whether anyone in the company will use it. Evaluating the features and functions of a well-established product from SAP, Oracle, Microsoft, or another major vendor is one thing; successfully using that product in your own company is an altogether separate activity.

A buyer's checklist needs to do far more than confirm that a software product meets every business and technical requirement; it also needs to ask how well it fulfills certain human requirements:

How clearly does it communicate to the people who will be using it?
Is the information presented in a logical sequence?
How consistent is this system?
How flexible is it?
How forgiving is it?
What's it like to use this product?

Technology can't answer these questions, and business analysts often overlook them. Yet this information means the difference between a system that supports efficiency and innovation and one that incurs enormous hidden costs in errors, delays, and lost opportunities.

Despite the best efforts of experts in business and technology to specify their requirements, nearly everyone recognizes that something is deeply wrong with most computer software: It works, but it seems capricious because it doesn't always work the way we think it will, and much of the time it just doesn't make sense to us. As the prison captain portrayed by actor Strother Martin famously observed in the film *Cool Hand Luke,* what we have here is failure to communicate.

Not getting the message—Paul Newman in *Cool Hand Luke*

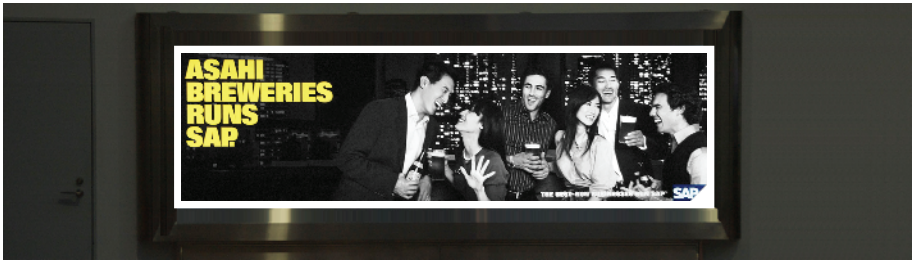
FAILURE TO COMMUNICATE

The way we do our work has profoundly changed, and yet we've only begun to shape the tools we use every day.

We've become accustomed to having astonishingly powerful information technology at our disposal: Electronic systems enable us to gather, store, and share complex information with a few keystrokes. Energy traders use sophisticated software to monitor market activities and to balance supply and demand on a minute-to-minute basis.

Manufacturers use software to take orders, manage inventory, link assembly lines with loading docks, calculate delivery routes and schedules, and connect customers with call centers around the world, displaying each transaction in fine detail on any number of computer screens. In many hospitals, physicians use handheld computers to record information about their patients and to order diagnostic tests and medication.

The effectiveness of these tools is determined not only by the accuracy of the data and the precision of the code that drives that data, but also by how well the information is organized and how clearly it's presented on the screen. For all the millions of professionals actively engaged in utilizing information technology—whether they manage the flow of electricity, staff a customer-service hotline, or head an intensive care unit—the quality of the software they use will determine whether the process of exchanging information becomes



Billboard ads in airports across the globe boast of the relationships that bind business and technology.

a cost-effective, satisfying experience or a continuing source of errors and delay. In transportation, health care, weather forecasting, and many other sectors, this can be a matter of life or death.

By 2008, estimated annual revenues from the sale of U.S. packaged software had risen to $149 billion, a 46 percent increase from 2000, according to the U.S. Department of Commerce. Hundred-million-dollar enterprise systems and $49 personal accounting applications have become everyday wonders of technology. For individuals, these systems offer to save time and money and to provide better communications and computing capabilities; for industry, their promises of increased efficiency and a healthier bottom line make their ownership and use a necessity. But the reality of ownership often falls well short of its promise, and dramatically short of what might have been.

Often these deficiencies are only minor inconveniences, and sometimes they're even mildly amusing. One morning not long ago, my desktop printer inexplicably began to print a report in Chinese, even though the monitor in front of me displayed the text in English. Technical glitches in software are so common that we're not really surprised when something bizarre happens. But when the quality of software compromises our business, our personal privacy, or our safety, no one is amused. A $10 billion error briefly disrupted the New York Stock Exchange in 2007 when a trader for Morgan Stanley placed an order for $10.8 million of stocks. The electronic order-entry tool he was using had a built-in multiplier of 1,000, and when the system automatically raised the order to $10.8 billion, more than 81 million shares were traded before the order could be cancelled. The Exchange fined Morgan Stanley $300,000.

The volatility of supply and demand makes businesses especially vulnerable to flaws in software products that manage order processing and inventory. A famous fiasco occurred when a new computerized fulfillment system at Hershey Foods broke down in the fall of 1999, leaving retailers' shelves empty of candy just before Halloween and causing $150 million in projected sales to vanish. In 2005, a $20 million enterprise resource planning system for the medical equipment corporation Invacare caused so much disruption in order processing that the company lowered its fourth-quarter sales estimates by $30 million.

Some software bugs are potentially deadly. In Louisville, Kentucky, the city's computerized emergency dispatch system suddenly froze one evening in November 2005. Earlier that day, software for the dispatch system had been upgraded, but by about nine P.M. dispatchers were unable to disconnect 911 calls as they were completed. The lines quickly clogged, and for two hours no new callers could get through. A year earlier, at an air-traffic control center in Palmdale, California, a high-tech, touch-screen communication system shut itself down, snarling air traffic across the country until the system could be restarted three hours later. During that period at least five planes came dangerously close to one another, and 800 flights in Southern California were diverted, delayed, or canceled because controllers at Palmdale couldn't communicate with planes in the air or with other air-traffic control centers. The problem was

traced to a computer program that caused communication systems at air-traffic control centers across the country to shut down unless someone manually reset them every 30 days.

The Associated Press reported in January 2009 that software glitches at the Veterans Administration (V.A.) had caused errors and delays in treatment at its medical centers around the country. Because the V.A.'s electronic system did not clearly show physicians' stop orders for medications, some patients received prolonged doses of drugs such as the blood thinner heparin, which can be fatal in excessive doses.

The full cost of the glitches, bugs, and errors in software products is unknown, but a study by the U.S. National Institute of Standards found in 2002 that software errors cost the American economy $59.5 billion annually. The effect of technical failure is immediate and clear, but technical flaws represent only part of the problem. More subtle but equally serious is the damage caused by software that is precisely engineered to perform every function, but whose users can't make it work.

Very often, even software whose performance is technically perfect seems counterintuitive. As a result, countless working systems are defeating the purposes for which they were created because they cause so much difficulty for the individuals who use them.

What if you bought a fast, powerful new car, only to discover that before you could lower the windows, view the gas gauge, or turn on the news, you had to stop the car and put it into reverse? If your dealer told you, "That's just the way it is; they're all that way," would you regard that as acceptable?

The cost of software that's hard to use is impossible to estimate, but it's a pervasive problem, and one that's at least as significant as the cost of technical flaws. In most cases, the fundamental flaw can be traced to a missing feature—lack of clear communication with human users.

The more expensive a product, the more grandiose the marketing hyperbole, and software is no exception. These sales pitches for software systems are as implausible as they are for any other product:

"Software is unlike anything else."

Although they can be very big and strong, software systems have a lot in common with every other product that is manufactured, marketed, and sold to the business community—and there's no reason they can't be made to better serve you.

"New software technology will give you a competitive edge."

Bundles of new features won't solve old problems unless they provide easier access to information within your organization. A system that is difficult to use will slow your operations and can even bring them to a standstill.

"If you fully specify your business needs, technologists can engineer the solution."

Technology can't give us all the information we need unless we specify the human requirements as well as the business requirements.

Clear communication, valuable business intelligence, and strong decision support are based on analysis and interpretation of data. To be meaningful, these processes must include information about the requirements of the people who use the technology.

"Software solutions that have been developed specifically for your industry will meet all your business needs."

How similar is your business to those of your competitors across the street, across the country, or across the ocean? Are most companies in your industry about the same size, and do they use the same work processes and share the same corporate culture? Do they even speak the same language?

Businesses are as varied as the human beings who run and staff them, and most systems will need some adjustment to fulfill the business requirements and meet the needs of the men and women who use them to do their work.

"As soon as everyone gets used to it, your system will be easy to use."

The need for intensive training is a symptom of a serious defect, and sophisticated systems often languish on desktops despite extended training programs.

Training should be an opportunity to learn how to use a business system to improve productivity, not a series of lessons in how to manipulate the counterintuitive features of a balky tool.

"If your software system isn't delivering the results you want, you need an upgrade, and you may want a customized system."

Even an old legacy system may be a technological wonder that just needs to be taught to speak English.

Low-tech, low-cost changes such as clarifying language, simplifying access codes, creating shortcuts, reordering sequences, submerging marginal functions, removing irrelevant features, and eliminating visual clutter can have a big payoff. The bigger the system, the more flexibility you may have to tweak it, but nearly every business system can be reconfigured in small ways that can dramatically improve performance and productivity.

Before you write off your sunk costs, consider whether the system you have can be adjusted to deliver what you need.

Early adopter—my grandfather, Harold Flavell Westcott, M.D., in 1918, extracting the benefits of owning a car

THE WRENCH ON THE FRONT SEAT

The information technology available to us is as fast as the speed of light, more powerful than most of us can imagine, and as reliable as the clock ticking in the hall. But this technology doesn't serve us nearly as well as it could, simply because we do not develop software through processes that genuinely consider those who will use these products.

Today, most business software is like so much of our public transportation: uncomfortable and unpredictable, avoided by those who have the means to do so, and tolerated by those without.

My grandfather was a surgeon who was stationed in the American Southwest with the U.S. Army during World War I, and during this time he owned a Model T. When I asked my grandmother what she remembered about that first car—was it the speed, or the freedom of the open road, or was it just the feeling of having the wind in her hair?—she told me that what she remembered most was sitting on the wrench. The car was supplied with a few basic tools, and a wrench was stored on the front passenger seat. She

A balanced perspective—my grandparents, Harold and Grace, at left, with friends

said that the experience that she remembered in that car was sitting on the tools.

My grandmother was something of a snob, and she also said that the car made an auto mechanic of her husband because every so many miles he had to reach underneath the hood and make adjustments with the tools that the manufacturer had supplied to him. This was regarded as normal, part of the responsibility of ownership. It was as if the manufacturer had said, "If you are going to own this technology, you are going to accept these things."

My grandmother wasn't hoping for a car that had power windows or an automatic transmission, and she wasn't expecting the car to talk to her. The only innovation she wanted was not to have the imprint of a wrench on her bottom. She was asking for just a little more comfort. Furthermore, she didn't like seeing her husband transformed from respected surgeon to amateur mechanic. This last concern cannot be underestimated, because in making decisions about the things we buy, we're strongly affected by what these things say about us and how they make us feel. And that's the level of conversation that you have today whenever you ask people about their experiences with software: The usability of the system is the proverbial wrench under everyone's bottom.

Clumsy, ineffective software makes everyone who uses it feel uncomfortable. It causes employees to perform poorly, so it gives them a negative impression of their value as workers, and it also makes a company seem unconcerned for its employees and disconnected from the needs, the expectations, and the goals of its customers.

By today's standards, the Model T was a lemon: hard to start, hard to steer, and often in need of adjustment. But long before the Model T morphed into Mustangs and Explorers, Ford began to design and engineer its products to make them more practical and more appealing to customers. Rather than adding extra padding to the seats or moving the wrench, the manufacturer eliminated the need for a readily accessible tool kit by designing a product that doesn't require constant maintenance.

Today we take it for granted that we can start a new car with the flick of a wrist and that it will give us a smooth, quiet ride, keep us warm

Dashboard data from a Mercedes at a glance and with the press of a finger

in winter and cool in summer, bring us the latest news and our favorite music, show us how to reach our destination, and even help protect us in a collision, all without requiring us to do little more than refuel or recharge it from time to time and change the oil every few thousand miles.

To design business software that we can use with the same assurance, we need to consider the history of product design and manufacturing. We need to think about how those older processes—those ways of modeling objects to suit the needs of the people who use them—can be applied to the process of building new software and modifying existing systems. Imagining these collections of features and functions to be different from all other consumer goods in the history of commerce has brought us to where we are today: We possess tremendously powerful technology, but our experience with that technology routinely makes us miserable.

INVENTING AN EXPERIENCE

Successful manufacturers of consumer goods create something more than good products. They create great experiences.

One of the multitudes of innovative and influential consumer products launched during the twentieth century was the MP1, a stylish, portable communications device created by a pioneering technology company. The company had been founded by a visionary engineer who understood the necessity for his company to adopt new technology and to respond to changes in the marketplace. He was very concerned with the physical form of his products, because he knew that an attractive form would make them more popular, and he recruited architects and designers to join his development team. The company's first products were unremarkable—manual typewriters manufactured at a small factory near Turin, Italy, that

could turn out 20 machines a week. But Camillo Olivetti, who founded the company in 1908, adopted new forms of labor organization and new techniques of mass production, and the company prospered. By 1930, the company was producing 13,000 typewriters a year and had opened its first European subsidiary.

In 1932 Olivetti introduced the first portable typewriter. This handsome machine, with an innovative design by Aldo Magnielli that made the chassis independent of its frame, was the MP1. As beautiful as it was practical, the MP1 won quick acceptance. New models followed, including the Studio 42, which was designed for both home and office use, and the Lexikon 80, a manual typewriter designed in 1948 by architect Marcello Nizzoli and sheathed in enameled aluminum, a machine so sleek and elegant that it was added to the design collection of the Museum of Modern Art.

The MP1, the Studio 42, and the Lexikon 80 were soon succeeded by Olivetti's own electric typewriters and personal computers as the company established itself as a global leader in communications technology now known as Telecom Italia S.p.A. Yet even today these manual typewriters, long obsolete, continue to attract our admiration as iconic examples of innovative, practical products designed to please. This isn't the case with any software ever invented. At least, I have yet to hear anyone say, "I love my word processing software!"

Portable word processor: Olivetti's MP1

Designed to be versatile: Olivetti's Studio 42

When I asked my grandmother about her memories of her first car, I expected an expression of nostalgia to cross her face, but instead I saw something quite different. Try to imagine yourself many years from now, recalling the computer that now sits on your desk, remembering your experience with it, and smiling with satisfaction. More likely you'll react just as my grandmother did to an old photo of her Model T: "Ugh—that old thing!"

Olivetti's innovative typewriters were the products of a well-established process that capable businesspeople have long used to design and manufacture consumer products. That process began with a consideration of the physical form that the product would take. First a designer would make sketches and models to describe the shape and size of the product, and that proposal would be evaluated according to how well it met the needs or desires of its intended market. The next step was to test a prototype with members of its target audience, and often those tests led to revisions in the design. Once approved, the design would be refined and expressed in detailed manufacturing specifications; then, at each stage of the manufacturing process, the product would be measured for accuracy and the design would be validated, keeping in mind as a constant point of reference the relationship between the product and the consumer.

Nearly 70 years later, Apple Computer used a similar process to create its first digital music player. The iPod was developed over an eight-month period in 2001 by a team of nearly 300 designers, programmers, and engineers in the United States and India. The development team built it feature by feature, with special attention to the user interface, and when the first prototypes were built, Apple CEO Steve Jobs was among those who tested each one to see exactly how well it worked.

The iPod was visionary not only because it was smaller, more powerful, and more stylish than any other device of its kind, but also because its unique wheel made it simple to operate. The iPod was designed and engineered to be compact, capacious, elegant, and clear—so clear that a novice could easily load, locate, and play thousands of songs. This was a new object that didn't look quite like anything else, and it had impressive capabilities. But the iPod's most compelling aspect was its combination of extraordinary technology and brilliant design.

The design process moves bright ideas like the MINI Cooper from concept to commercialization.

The iPod succeeded not because it packaged great features in a new form, but because anyone could easily use those features. The iPod was so much easier to operate than its competitors, and so much more convenient to carry, that using it became a new experience. The innovation of iPod was to create something we didn't know we wanted—not only a new object, but also a new form and a new experience!—all so well executed that it became instantly desirable to people everywhere. Not only did millions of buyers find the iPod useful and attractive, but something else happened: People made it part of their daily routines. Apple reinforced this concept by marketing the iPod on its web site as "an ideal companion," and its print ads featured silhouettes of young people wearing iPods and dancing with joy. Using the iPod became an experience that nearly everyone wanted.

Easy to love—The iPod is designed to offer an experience that feels like second nature.

iPod photo courtesy of Apple

The iPod and the MP1 are the successful products of a traditional development process that is driven by concern for its users. However, when we consider the products of information technology, it's clear that the skills, the rigor, and the goals that drive other product companies have not effectively influenced the design and development of most software.

Developers of new business software pack their products with features, but little of this advanced technology represents any improvement in the experience of using it. More often, new software products are merely new objects—new forms, new features, or even new compilations of old features, which rely on these innovations alone for their success. Business executives consider these products highly desirable because of the increased efficiencies they promise, but those who use them typically find them less than efficient. In fact, the professionals for whom these products are intended may be unwilling or even unable to use them despite the rewards they might derive from them.

An innovative form or a new feature can go only so far to satisfy its users. Olivetti's MP1 may have boosted morale among those who used it, but its sleek design didn't make the task of typing easier or faster. In itself, the iPod is both a beautiful device and a wonderful experience: The gesture of my finger to elicit an almost magical response from the technology puts a smile on my face. The result of that interaction is a delightful experience with my music, my

media—things I've chosen according to my personal taste. But this delight contrasts starkly with the scowl that I wear when I interact with the broader continuum of Apple iPod ownership—which extends to my on-screen experience with iTunes. Here, even technology's coolest music player demonstrates profound weakness in understanding how software should satisfy its end user. Managing lists, understanding the limits of this device, and making a purchase is so difficult that my 11- and 12-year-old daughters have to struggle to buy movies even though they have permission to charge them to my account. When my computer-savvy preteen daughters have so much trouble spending my money, something is wrong.

Not even Apple has fully solved the problem of how to give the people who use its products a great experience—far from it. There's no doubt that Apple makes a sexy personal computing cabinet. The company pays an exceptional amount of attention to the design of the cabinetry that encases the everyday on-screen challenges of its user population, giving us innovative shapes and CPUs incorporated into pristine boxes and monitors with shiny, rounded corners that sit elegantly on our desks—all very beautiful. But on-screen, can anyone argue that the Apple O/S experience is truly far superior to the user environment of the icon-driven Windows desktop and its maze of file directories? I have wrestled with both Apple and PC

Designed to please—Some of the most useful, successful products owe their popularity, in part, to design.

iPhone 3G photo courtesy of Apple

Some of the best design is invisible.

DESIGNED

In the late 1980s I moved to New York City to work as a freelancer for one of the hot design firms at that time. This firm's work was largely print-based, but I was there to consult on the company's account with Citibank, designing what would become the customers' experience with the bank's new automated teller machines.

During my time in New York I was able to run with a lot of graphic designers and see the so-called best work of that time. What I saw the leaders of the design community calling the best was often what I called *useless:* Broadway production posters with typographic treatments that may have captured the feeling of a performance but were almost impenetrable when it came to deciphering performance times, locations, and ticket prices; product packaging with beautiful color palettes and cool imagery, but also with type that was so tiny, even my 21-year-old eyes couldn't read it.

When I discussed the problems I had recognizing these confusing creations as examples of exceptional design, many fellow designers scoffed, saying that the very characteristics that troubled me were the things that made these objects special—an expression of their designers' individual visions. *Yikes!* To me, these were the traits that ruined my experience with each object. It seemed clear that utility and quality of experience were not standard design priorities. What I believed to be an opportunity to succeed by communicating clearly, directly, and gracefully to an audience was seen by other designers as an opportunity to express their own voices.

TO DISAPPEAR

It's said that great actors disappear into their roles. Whenever Laurence Olivier took the stage, he seemed to become the character he was portraying, masterfully presenting the playwright's vision to an audience through the use of carefully honed skills. The results for the audience were the emotions and experiences as intended by the writer, the director, the producer, and, of course, the actor.

As technology advances beyond the screen, there are enormous opportunities to design satisfying experiences with our electronic tools. Unlike icons and data fields on a computer display, much of our new technology is represented by little or no physical form. Consider the voice-activated phone system in your car: Simply press a button, and an audible prompt triggers a list of choices in your mind. This experience reflects a true understanding of the business goals that inspired this object's existence as well as the needs, expectations, and capabilities of its target audience, all orchestrated and designed into a satisfying experience with a product that is nearly invisible. In the ether of this human-machine interface, there's no place for a designer's thumbprint.

desktops and have grown equally frustrated, although for different reasons. Despite the zealous loyalty of Apple and PC partisans, each group of products exists within an environment that is miles from an end user's ideal experience—although perhaps in different directions.

In any consumer product, technology and usability are so intertwined that neither can succeed without the other. Rarely are the two elements in perfect balance, but when the developers of a product possess the will and the know-how to create an innovative form that positively influences the experience of using it, the appeal and the value of that product will be almost unlimited.

JUST WHAT WE NEED

All great products have one thing in common: They seem to be designed just for you.

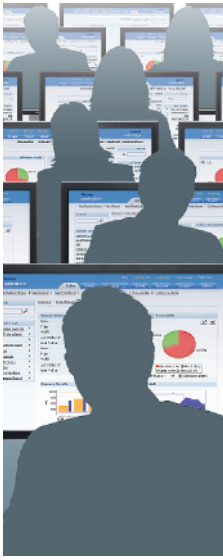
Successful products appeal to the hearts and minds of the men and women who use them, and most consumer products are developed through a process of testing and evaluation that keeps the focus on humans. The software development process is very different.

The process of developing business software has been driven by priorities unlike those of traditional product design. Typically, software developers bring just two concerns to the process—the business requirements and the technology needed to execute them. Far too little thought is given to the practical problems of the people who will use these systems as many as eight hours a day, and almost no thought at all to how the knowledge and abilities of these individuals might influence the design of the software.

Many executives and software developers are convinced that if technicians can write the code to satisfy the business requirements, then the human beings who use the systems will fall in line and adapt their behavior to suit the software. (That swearing you did last time you used Microsoft Word: That was an example of what happens when you're forced to adapt your behavior to a technologist's idea of how you should be working.) Not enough consideration is given to whether these basic business tools could be more effective and more pleasing, or how these tools are affecting the relationships between employers, their customers, and their employees.

It's true that managers can accomplish great things on a large scale by persuading (or ordering) large numbers of workers to behave in a certain way. In theory, if you post identical information on a thousand computer screens in your organization, you can teach a thousand workers to consistently respond to that information in the same way. But in reality this is not going to happen, because it's just too easy for human beings to do otherwise. When the act of performing a task is as intimate as an individual sitting in front of a computer screen, with a private view of that screen, that individual has choices. He or she can choose to work quickly or slowly, to try something or not, and to be effective or ineffective. And this is where the absence of design—the inattention to human needs—becomes most noticeable and creates the greatest risk to a business investment in technology.

Software manufacturers are generally confident that their products will succeed on the strength of their technology. But products that don't appeal to their users can be self-defeating. Whenever software systems create obstacles—technical jargon, ambiguous messages, illogical sequences, or visual clutter—the people who use these systems will respond in a variety of ways.



Building software that doesn't make it possible for its users to follow a clear, straight path is like forcing drivers to navigate a succession of cloverleaf intersections and unmarked detours in order to continue along an expressway leading due north. The results will be similar: Just as some drivers will lose their way and others will cut across the median strip to seek an alternate route, those who use unsympathetic software will disobey commands whenever possible and try to find a more direct approach by navigating around the barriers. All of them will feel badly every mile of the journey, and they'll be weary and frazzled by the time they reach their destination, if they arrive at all.

Part of the process of developing software should be to identify the most appropriate route from the perspective of the user, within the constraints imposed by business and technology. It's not always possible to build highways exactly where we want them to go, and software developers often discover a conflict between what's best for a business, what best serves its customers, and what's best for the

employees who will use the system. However, it's possible to find a good compromise, one that utilizes the best available technology to make a finished product that people consider to be not only valuable and but also practical and easy to use.

If a product as mundane as a wristwatch or as technologically complex as a digital music player can bring us satisfaction and pleasure because of the way they work, the way they look, and the way they make us feel, surely we should expect the same results from the software we use every day.

Technology alone cannot satisfy our need for more effective, more satisfying software systems. The secret of developing software that communicates clearly is the traditional process of product design that balances business and technical requirements with the needs of the men and women who use these products.

We want technology that fulfills every business and technical requirement, but our best software products also seem intuitive. This doesn't happen by accident; it represents a commitment to specify, test, and evaluate the quality of the users' experience with that product every step of the way. This is an astute business decision, because the quality of the experience a product offers will determine how well it succeeds.

It's not unreasonable to insist on business software that is easier to use, less mysterious, and far more effective—and in the chapters that follow, we'll show how to specify satisfaction.