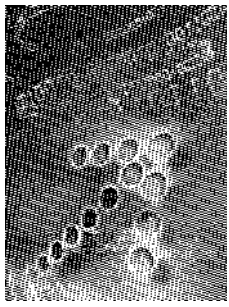
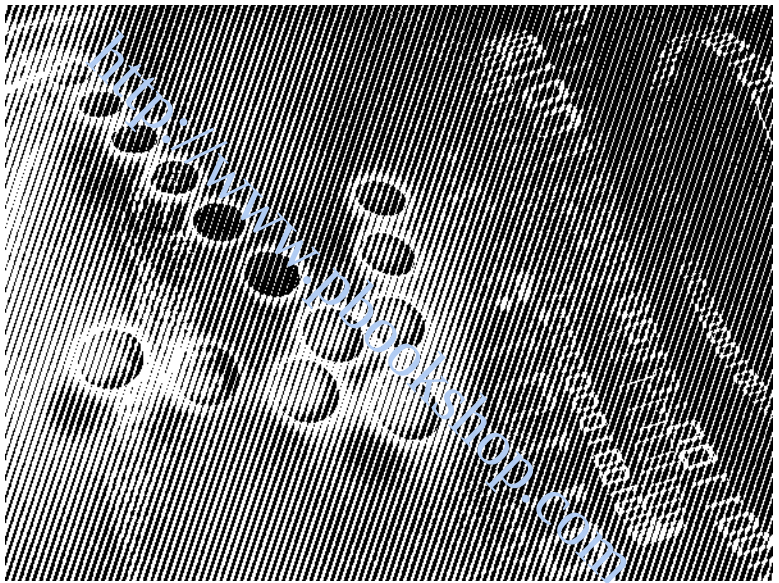
Chapter 1



Success Depends on Innovation and Innovation Depends on Information Technology



"Past results are not a guarantee of future performance. As we enter the process-centered age, this caveat should be writ large—for individuals, for companies, and for countries. It should be emblazoned on the desk of every employee, and of every citizen."

—Michael Hammer

Executive Summary

In competitive markets, innovation is the prerequisite for success. IT enables innovation. Meeting demand for innovative products and services in fast-moving global markets requires C-level executives to work more closely with IT than ever before. IT needs formal development standards and improved management practices to guarantee that new products and services will delight and amaze customers.

Another Day at the Racetrack

Rapid, unrelenting change is the price we pay for enjoying the fruits of modernity. Fresh danger and new opportunity appear suddenly, demanding immediate responses.

As recent history demonstrates, nothing can fully prepare us for the pace, complexity, and unpredictability of modern global markets. So we rely on increasingly powerful information technology (IT) to support the strategies that we invent to achieve our goals.

Without IT, we could not hope to capitalize on openings and dodge threats as they materialize. Responding quickly and effectively to changes in markets, however, requires deep portfolios of highly adaptable IT resources.

Quite frankly, IT has to be capable of changing as swiftly as the markets. A static set of IT resources will not help an organization competing in a global business environment.

But an adaptable IT system requires adaptable software. This presents both a dilemma and an opportunity.

Organizations that learn how to convert knowledge and intelligence into programming code quickly and efficiently will enjoy competitive advantages over rivals with slower, less efficient software development processes.

In the next round of global competition, having the smartest people won't be enough. What will be needed? The fastest and most creative programmers to convert your ideas into code?

Actually, no. This is an old (and dangerous) way of thinking. It assumes that IT is merely a cost incurred to support the enterprise. In fact, IT can be an incredible asset to the organization—an invaluable driver of growth and expansion. That's what this book is all about.

Winning the Race with Two Leaps in Productivity

A huge untapped potential for productivity improvement is locked up by the interoperability of IT and the other parts of the business.

But don't expect IT to unlock this potential all by itself. In fact, don't even expect IT to realize that this low-hanging fruit is there. It will continue to lie untapped until the chief executive officer (CEO) leads the charge. That is why Chapter 2 explains how to drive IT strategy from the top.

Achieving this Enterprise Leap in productivity is the responsibility of you, the CEO, the top manager of the entire enterprise. Too many people (and too many books) focus on improving the effectiveness of IT without situating that goal

in the context of improving the effectiveness of the organization as a whole. Focusing on IT instead of the organization as a whole leads to opportunities squandered.

As CEO, you must change how the enterprise views and uses IT. Chapter 3 explains why. It argues that IT is one of the enterprise's most valuable assets *because it is a tool of change of that drives productivity across the enterprise.*

To reap IT's potential as an agent of change, IT itself has to become more nimble, responsive, and productive. That's why IT must improve its own productivity. Chapters 4 and 5 explain how.

Once the CEO can verify that IT productivity has truly improved, then a new partnering of CEO and chief information officer (CIO) can begin. Now, the CIO becomes a source of ideas for improving productivity across the enterprise.

This leads to the second quantum leap in productivity, which occurs at the enterprise level. As Chapter 6 explains, this second leap is achieved by gluing together current human processes with computer programs.

Meeting in the Middle

Much has been written about the need for IT to develop a deeper understanding of business. For the past decade, CIOs have been urged, coaxed, counseled, and exhorted to act more like CEOs, chief financial officers (CFOs), chief operating officers (COOs), and other C-level executives.

I'm not going to argue about the wisdom of that advice. But I'm going to suggest that it's only half the story. The other half, the piece that is usually missing from conversations about innovation, competitiveness, and the opening of new markets is this: It is time for business to learn more about IT.

Specifically, it is time for CEOs, CFOs, COOs, and other C-level executives to start acting more like CIOs.

Why now? Why should top management expend the energy required to learn more about IT, a large amorphous aggregation of multiple technologies that is constantly moving, changing shape, and evolving into who knows what?

BusinessWeek Research Services recently polled 353 C-level executives spread across multiple sectors of the economy such as financial services, health care, pharmaceuticals, transportation, retail, and manufacturing. One goal of the survey was determining which factors drive C-level decisions about IT. Nearly half of the executives surveyed were CEOs and precisely one-third were CIOs.

Here is how executives ranked their top five crucial business goals for 2008:

1. Sales and revenue growth
2. Reaching new customers
3. Improving customer service and retention
4. Increasing profit margins
5. Increasing market share

In today's economy, it is virtually impossible to achieve any of those five goals without first having the capability to develop and market brilliant new products and services that your customers will absolutely adore.

Rapid innovation and first-rate product creation require tight working relationships between the C-suite and IT. You don't have to be joined at the hip, but as one CEO recently told me: "If you're not meeting daily with your CIO, you've got a problem."

Let's face reality: IT is to business in the twenty-first century what wind was to the Royal Navy in the eighteenth century, what coal was to the Industrial Revolution, what philosophy was to the Enlightenment, and what art was to the Renaissance.

Okay, you get the point. IT is essential and indispensable. Not only is IT central to every organization, it radiates through every part of the organization. IT touches everything. IT is everywhere.

Now is the time for top managers to meet their CIOs halfway and learn to walk a mile in their shoes.

A True Story—Mostly

The following story is mostly true. I can't tell you the names of the companies involved, but you can guess. There will be no prizes for guessing right, but you will have the satisfaction of knowing that even humongous companies with legions of planners, strategists, and business analysts make mistakes that your great-grandmother would never have made.

A few years back, an extremely large global technology supplier saw an opportunity to tear off and swallow a huge chunk of the market for Internet routers. The plan was simple: They would simply overwhelm the competition by throwing vast resources into new product development, marketing, and sales.

The leading maker of routers at that time was not a small company. But it wasn't nearly as large as the global technology supplier that had fixed its hungry eye on the router market.

The company that made routers had more collective experience and deeper knowledge of the market than any potential competitor. But the company had a weakness: It had no formal company-wide standards for testing the new software that it developed for its routers and other innovative products.

For years the developers at this company had enjoyed a sense of cowboy-like freedom, largely unhampered by rules or anything else that they imagined might have restricted their creativity.

The company paid a steep price for all of this unbridled creativity. It took forever to get their pioneering products and services out the door and into the hands of their eager customers. And the overall quality of its products—especially its high-end routers—ranked among the lowest in the industry.

The company's senior vice present (SVP)—the executive responsible for software development—knew that if they

were going to compete head to head with a global giant, they would need to get new products and services to market much faster than in the past. That meant they would need a world-class process for developing new software quickly and cost-effectively.

So the SVP issued an edict: Henceforth, standard tests will be applied to all software. Every night, all developers will check their newly written code into an automated system, where it will be scanned for errors. Every morning, the developers will receive reports showing the previous day's errors. Before doing any new work, the developers will fix the mistakes they made yesterday.

The beauty of this process is that it assures that each day's work begins and ends with clean code. Clean code virtually guarantees that the software will work—or at least that it will not fail or begin acting strangely because of hidden defects.

The SVP's edict saved the company. It began moving new products and new services to market so quickly and so efficiently that the global technology supplier decided to rethink its plan. Ironically, the larger company now uses the same standard testing process as the smaller company.

There is one more delicious detail. When the manager who had initially championed the testing process talks to his developers, they tell him they could not imagine writing software today without a system for testing it automatically. "Can you imagine," they say, "trying to test all this code manually?"

Talk the Talk *and* Walk the Walk

When Joseph Moses Juran, a founding father of modern quality management, died at the age of 103 in early 2008, his grandson David Juran was quoted in the *New York Times* obituary: "Everyone who's in business now adopts the philosophy of quality management."

He was right, of course. What he did not say, at least not to the *Times* reporter, is that it has become all too easy to adopt the philosophy of quality management without understanding or embracing the underlying processes that Joseph Juran spent most of his life exploring and writing about.

Good quality management usually results in higher productivity. The best historical example of this is Toyota's approach to the automotive manufacturing process. With Juran's help, Toyota devised a production system in which quality management was tightly integrated with other assembly-line processes.

As a consequence, defects were found and fixed long before the finished product left the plant. More important, the defects themselves were treated as valuable clues that might reveal ways for continuously improving the efficiency of Toyota's manufacturing processes.

Well, we all know how the story ends. Toyota is now the world's number-one automaker.

Beyond the Numbers

Juran viewed quality management as more than a system of esoteric numerical techniques for reducing errors in manufacturing processes. He saw quality management as a way of making the world a better place. He was a brilliant and decent man.

Although I was trained as a theoretical physicist, I consider myself a student of the great philosophers of modern business—visionary thinkers such as Juran, W. Edwards Deming, and Peter Drucker.

What their teachings share, I believe, is a sense that business is about more than business results. Business is about life. Good businesses—that is, businesses that are well organized and well managed—improve the lives of their employees, customers, and stakeholders.

Bad businesses, however, affect life negatively. When a business is poorly organized and mismanaged, especially if it is a large business, many people suffer.

If parts of this book seem like angry rants against mismanagement, it is because I believe passionately in the power and potential of a well-run business to accomplish good things.

So What Exactly Do I Do?

I make a kind of highly specialized software that makes it easier for other people to make good software. My software

is used almost exclusively by software developers, so it won't hurt my feelings if you've never heard of it.

But even if you don't know the first thing about it, my software is probably helping your home computer network run faster, your online banking transactions remain secure, and your Internet phone work without sounding like you're calling from outer space.

My software increases the productivity of software development groups—by a lot. It accomplishes this by automating every part of the software development process that can possibly be automated. If you are trying to create new software quickly, that's important.

Fred Brooks, the author of a classic book on software productivity called *The Mythical Man-Month* and highly regarded professor of computer science at the University of North Carolina, wrote an article in 1986 entitled "No Silver Bullet," in which he argued that there would never again be a giant increase in programmer productivity. Only incremental improvements were possible, he said.

As depressing as that sounds, for many years Brooks was right. But his point is no longer correct; today, it is possible to achieve astonishing improvements in productivity. And not only developer productivity, but also the productivity of the whole organization, and this is the key. I know this is true because I see it every day: radical, discontinuous improvements in quality and productivity.

I don't want to bore you with the details, so here again are the two quantum leaps condensed into two bullet points:

- **Developer Leap.** You create software from reusable "building blocks" instead of reinventing the wheel every time you need a new feature or capability. This, with automation of your software development process, leads to what I am calling "disposable software." From my perspective, disposable software is any software that is easy to produce and easy to change.
- **Enterprise Leap.** Disposable software enables you to automate all the routine business processes—processes that do not require creativity—throughout your organization. This automation creates enormous savings, and equally important, the IT infrastructure that makes it possible creates an environment that fosters constant innovation and creativity.

The computer and systems science that underlies these revolutionary changes may be complex, but the fundamental concepts are not difficult to grasp. As CEO you need to understand and embrace this new discipline, and then invite the rest of your organization to follow your lead.

Productivity Trumps Quality—and for Good Reason!

In old-fashioned software shops—of which there are hundreds of thousands in existence today—the function of the "Quality" group is testing software to find bugs.

If automobile plants worked this way, the Quality group would drive each car for 10,000 miles after it came off the assembly line to make sure nothing was broken. Sounds pretty silly, doesn't it?

The right thing to do is to make sure the right processes are in place to guarantee that each car coming off the line is perfect. The attention is to the process, not to the product.

That is the approach I take in developing software tools. While I do make software that finds bugs, the general purpose of all my tools is *preventing* bugs from ever being created in the first place!

For example, in lots of places, developers like to work with prototypes that they know are buggy, just to get a feel for the problem. That is dangerous, and my tools do not allow it. Software developers use my software to check their software for bugs before it can be put anywhere where it can do damage, even in early prototype versions.

Here is another way of saying it: People use my software to make sure that the software they create actually works—before it is loaded into a router, a network server, or a mobile phone.

My software is a critical piece of their quality management process. In many instances, my software has enabled companies to leap ahead of their competitors.

Exactly What Kind of *Ware* Are We Talking About?

Software is the soul of IT, the spirit inside the machine. But the word *software* is misleading for a variety of reasons. First, it is certainly not a "ware" in any sense that we commonly understand.

Second, the word *software* was invented as a rhetorical device for drawing a clear distinction between the program running on a computer and the computer itself, which is the hardware.

In truth, software is thought captured in lines of programming code. It is the human mind—or little pieces of it, at any rate—translated into a language that a computer can understand.

Like the human mind itself, software can be very difficult to change.

People who have never written a line of code tend to believe that software is mushy and malleable. But just the opposite is true. After software has been written, released, and loaded onto an enterprise platform or onto the hard drives of 10 million personal computers (PCs), it might as well be cast in bronze.

At the risk of treading the same path as Scott Rosenberg in his excellent book, *Dreaming in Code*, it is important for people to understand that writing good software is not an

easy process—it is inherently difficult. Rosenberg wisely quotes Donald Knuth's famous observation, "Software is hard."

The sheer complexity of writing software makes it like sailing or playing the violin. You can strive for perfection, but perfection will always elude you. You will make mistakes. You will find yourself wedged into dark corners with no map to guide you back into the light. It is a really messy process.

Now multiply all that messiness by 100 or 1,000—the number of developers you might need to create a large software project or to extensively modify an existing software implementation—and you begin to understand why software development is such an expensive and inefficient endeavor.

So when the IT leaders cringe at every change request, it is not because they want to stand in the way of optimizing information technology. They are just (understandably) terrified by the prospect of change. Imagine that you are the CIO of a global bank and your CFO has already told you that next year's IT budget will be slashed to the bone. Or that you are the chief technology officer (CTO) of a national retail chain that is still working the bugs out of a recently purchased customer intelligence platform. Or that you are the COO of a transcontinental railroad that is still reeling from last year's upgrade of its business performance management solution. How eager would *you* be to take on an "IT optimization" initiative considering the associated risks?

Disposable Software

Even if you hire the world's best programmers, you're likely to run headlong into deeply rooted institutional challenges. As anyone who has ever tried can tell you, modifying any aspect of a modern software program can escalate rapidly into a task of Homeric proportions.

As suggested earlier, most of today's commercial software might as well be carved in stone.

Now imagine if you had the capability to modify code so smoothly and so easily that software would be considered disposable.

Unlike disposable diapers or disposable razors, disposable software would be environmentally friendly. It would leave no trace when you discarded it. It would be the ultimate recyclable product—and it would be endlessly renewable.

Best of all, disposable software would enable organizations to respond effectively within days or even hours to changing conditions by translating the intellectual power of their human capital investments directly into executable programs.

So what is stopping us from making the logical switch from traditional software to disposable software?

The answer lies in the hopelessly outmoded way in which we manage the creation of new software.

People outside of the software business would be amazed—and quite possibly shocked—by the premodern management techniques commonly used to develop new software. The industrial manufacturing processes of the late nineteenth century are models of efficiency compared to software development methods of the twenty-first century.

It is important to remember that the original Industrial Revolution did not proceed rapidly or in a straight line. The first factories were more like collections of independent artisans working under a single roof. All the fine details of modern industry that we take for granted today evolved over the course of several centuries.

Fitfully, and with many detours along the way, the science of manufacturing gradually emerged. And it is still evolving. Today's manufacturers are struggling to catch up with the reality of a global economy that is increasingly based on the buying and selling of intangible products created from intellectual property. Who knows what challenges manufacturers will face tomorrow!

Like the manufacturing sector, the finance sector continuously adjusts its processes to accommodate new global realities. Although you would not believe it from reading the headlines, banks and other financial institutions actually do a fairly good job of keeping pace with changing times.

Software developers cannot make the same claim. Many are still caught in a preindustrial time warp.

What the world needs now is a quantum leap in software development efficiency. If we can bring software development processes into the twenty-first century, the rest will be easy.

In the next couple of pages, I hope to show you why disposable software is not a frivolous utopian luxury, but an absolute necessity.

Only a Fool . . .

With the exception of some highly publicized foul-ups by Microsoft, most consumer software releases are nonevents. By the time a new piece of software or a new device containing a new piece of software reaches you, it is usually safe to assume that it will work pretty much as advertised.

Unless you are really unlucky, getting a new cell phone up and running does not require a lot of effort. You figure out which friends, relatives, and coworkers to put in your contact list, you choose your ring tones, you change the wallpaper, and you're good to go.

The situation is only slightly different in the commercial world, where most software applications either function properly right out of the box or require relatively simple modifications to function properly.

The picture changes at the enterprise level, where hundreds of software applications must work in constant

harmony to sustain the enormous needs of a large organization or business.

Again, with the exception of a few spectacular hiccups, most enterprise software applications run smoothly and uneventfully for many years.

And that is the problem right there.

Enterprise computing applications are so complex that when they are running smoothly, only a fool would try to modify them. Even if you manage to convince the CIO that a major modification is absolutely necessary to achieve a significant business objective, the CIO's IT team will struggle furiously to resist anything but modest and mostly superficial changes in the existing system.

They will not resist change because they are hopeless Luddites—they will resist change because they know from bitter experience that even the smallest alteration in a complex IT environment will likely trigger a cascade of unexpected and hugely expensive consequences.

They also know in their bones that *any change in the system can easily make things a whole lot worse!*

No matter what the IT vendor or software consultant says, the truth is this: Every system modification requires a significant investment of money, time, and effort.

As a result, proposed software modifications that could streamline operations and enable more effective allocation of resources are often vetoed because the steep cost of change is perceived to outweigh the potential benefits.

Even if you work for a privately owned company and you do not have to worry about Wall Street analysts breathing down your neck, it is still hard to make the business case for a project with high initial costs and a long payback period.

So the leaky roof does not get fixed until there's a flood of water pouring through the ceiling.

True IT Productivity

Let's back up for a moment and examine the parameters of the challenge I have just described. Your IT team will try desperately not to let you break what is already working—even if it means sacrificing real opportunities for streamlining operations, opening new markets, growing revenue, and increasing profitability.

Faced with this degree of resistance, what are your options? You could fire your IT team and hire new employees—who would quickly become indistinguishable from the old employees.

You could outsource your IT operations—and spend the remaining years of your career worrying about business continuity, security, and a host of other pesky issues that do not

vanish merely because your IT department is now spread out over five continents.

Or you could deal with the problem directly at its source: You could improve the efficiency and reliability of your software development process.

If your software development processes were brought into the twenty-first century, there is a good chance that your IT team would lose its fear of change—because they would trust the processes that create or modify software.

When complex software systems can be changed, improved, or modified easily, you have achieved what I call true IT productivity. Anything short of true IT productivity is a sham—an illusion of stability that represents the temporary calm between storms.

True IT productivity enables continuous improvement. It requires a formal, standardized, and largely automated framework for fine-tuning software development life cycles. It is an ongoing, iterative process that has no beginning and no end. It becomes standard operating procedure by force of habit.

To see the value of true IT productivity, let's look briefly at an example of what happens when IT productivity collapses.

Plunging into Crisis

As many residents of California already know, the Los Angeles Unified School District experienced a meltdown of

its new $95 million payroll system. *Los Angeles Times* reporter Joel Rubin wrote this vivid description of the mess:

> *… [C]onsultants hired to implement the system urged the district to proceed as scheduled in early January 2007…they urged the district in a report to "Go! Proceed…and go-live on January 1!"*
>
> *Go live they did, plunging the district into a crisis from which it is only now emerging. Over the course of last year, taxpayers overpaid an estimated $53 million to some 36,000 teachers and others, while thousands more were underpaid or not paid at all for months.*

Marla Eby, director of communications for the teachers' union, told blogger Michael Krigsman that while the old payroll system was complex, it did work. The new system, she told Krigsman, "was rolled out too quickly, and without sufficient testing. The union requested that the system be run in parallel prior to full rollout, to ensure these problems would not occur. The school district chose not to follow this advice for budget reasons, which is ironic given all the cost overruns now."

In a better world, the school district's new payroll system would have been thoroughly tested before going live. After going live, the system would have been subjected to continuous testing. When problems surfaced, modifications would have been created, tested, and dropped into place seamlessly.

Make no mistake—problems always arise. That is the nature of complex IT systems. And, by the way, the Los

Angeles Unified School District's new payroll system was not an ad-hoc affair thrown together by amateurs. The software was supplied by SAP, the world's leading enterprise resource planning (ERP) vendor. Additionally, the district awarded Deloitte Consulting a $55 million contract to customize the software, the *Los Angeles Times* reported.

The huge costs of the new system no doubt contributed to the slow pace of fixing the problems. School officials were simply terrified by the real possibility that any modifications could easily make things even worse and wind up costing the district more money.

The result was a state of paralysis and confusion. In some instances teachers went without pay for months. Imagine being a student during that year!

We know the details of this sad story because the school district is a public entity and is subject to intense public scrutiny. As a long-time developer of enterprise software, I can testify that such situations are not limited to the public sector. The CIOs of companies in disparate industries such as travel, leisure, gaming, retail, heavy manufacturing, pharmaceutical, and health care have all shared similar stories with me.

The problem has caused considerable embarrassment, in addition to huge losses of capital, in the financial services sector. Despite spending billions of dollars to implement forward-looking financial management software systems, no single institution accurately predicted the scope and severity

of dysfunction in the world's credit markets following the collapse of the real estate bubble.

Complex enterprise systems proved incapable of coping with even the most straightforward issues. For example, all major trading institutions rely on monitoring software to issue alerts when danger looms. Clearly, those systems failed—most spectacularly in the case of Société Générale, where one rogue trader lost $7.2 billion before the bank caught on.

These software failures did not occur all at once. There were signs along the way. But modifying the software would have required significant investments of operating capital. Instead, the CIOs at many leading firms were instructed to reduce their requests for capital spending and trim their operating budgets.

In hindsight, it is easy to see how such investments would have appeared trivial compared to the trillions of dollars in financial losses suffered throughout the global economy.

Managing Creativity in the Real World

Economic slowdowns offer us an opportunity to reflect and to plan for the future. Any serious planning must include a sincere drive to increase true IT productivity.

I believe that the next significant jump in worker productivity will be preceded by a leap in true IT productivity. But we will need to have a new generation of software

development processes in place before the next economic "boom" gives everyone another good excuse to ignore the problem. That means we have to get busy now.

One of the thorniest issues facing software developers is human nature, or more precisely, our prejudices about the way we deal with creativity.

As a species, we tend to believe that creative processes are inherently individualistic. We are not accustomed to organizing creative processes and we often assume that creativity itself is largely unmanageable, as if it were a force of nature beyond our control.

When we think of creativity in the sciences, we imagine the lonely inventor in a cluttered workshop or the mad scientist in an isolated castle.

We do not imagine teams of individuals with different strengths and varying degrees of talent. Instead, we see a solitary genius with a global view, a Renaissance man with every possible skill required to get the job done . . . all by himself. What a guy!

I am embarrassed to admit that this delusion is especially prevalent among software developers.

What is missing from the lone-inventor fantasy is the reality of collaboration, which depends on many people with different viewpoints coming together to solve a complex problem.

No single individual on the team knows everything. Each individual understands a small part of the problem and each interacts with other members of the team. They share their knowledge, explain their insights, report their progress, and throw around ideas.

Eventually, the problem is solved as a result of continuous interaction among individual members of the team.

Superconductivity as Metaphor

Before I became a software designer, I was a theoretical physicist. So it is natural for me to draw comparisons between certain aspects of software development and physics. It occurred to me recently that the phenomenon of superconductivity provides a useful metaphor for helping us understand the challenges of managing software developers.

At normal temperatures, individual electrons within a potentially superconductive material collide with the material's atomic lattice, losing energy with each collision. But if you lower the temperature of the material sufficiently, its electrons pair up and begin mimicking each other's behavior. The paired electrons act as if they are one particle.

Pretty soon, all the electrons are moving in the same direction. Their individual behavior becomes a global effect. The collisions stop and the material's electrical resistance disappears. The result is a quantum leap in conductivity.

In our metaphor, software developers are the electrons and the software development project is the lattice structure surrounding them.

Under normal conditions, the developers bounce back and forth, ricocheting haphazardly from one problem to the next. And like the electrons, they lose energy with each collision. As a group, they are inefficient and relatively unproductive.

In an enterprise dedicated to true IT productivity, however, management organizes software developers into collaborative units and promotes a style of coherent behavior that is not terribly different from the behavior of the paired electrons in our superconductive material.

As with superconductivity, however, the devil is in the details. Promoting and sustaining the kind of coherent behavior that will result in true IT productivity requires a new type of IT management infrastructure.

Last but Certainly Not Least: Sarbanes-Oxley

The Sarbanes-Oxley Act of 2002 (SOX) requires all publicly traded companies in the United States to provide extensive financial information to regulatory agencies. Many executives think that complying with this law is an expensive nuisance that provides no real benefit to anyone. There is no doubt that it puts a great demand on the IT function.

Therefore, if for no other reason than to comply with this law and the vast changes it is enforcing in financial reporting,

you should want your IT staff to be a highly productive, "peak performing" group.

Goodness knows that your job as CEO is complicated enough without having to worry about breaking the law because your IT group could not provide the right data in the right format in a timely manner.

But, as you will see by the end of this book, SOX—when you look at it as I do, from the point of view of IT and business process—is not an inconvenience. Rather, it is a blessing in disguise because it forces you to do what you need to do anyway to align your business and IT processes.

In the last chapter, I will talk about how to use SOX as a forcing function to drive the vitally important improvements in IT productivity and quality that are this book's focal points. But before we can come back to this somewhat contrarian assertion, we need to look at the whole spectrum of ways in which IT needs to change. So let's get started. . . .