SUMMER 2007 VOL.48 NO.4

MITSloan
Management Review

Eric Bonabeau

Understanding and Managing Complexity Risk

REPRINT NUMBER 48416

Understanding and Managing Complexity Risk

Increased complexity of a company's systems — products, processes, technologies, organizational structures, legal contracts and so on — can create dangerous vulnerabilities. Three complementary strategies can help mitigate the risk.

Eric Bonabeau

In March 2000, a fire struck a semiconductor plant in New Mexico, leaving Ericsson Inc. short of millions of chips that the Swedish telecom giant was counting on to launch a new mobile phone product. As a result, Ericsson was ultimately driven from the market (it would later re-enter through a joint venture with Sony Corp.) while its rival Nokia Corp. flourished. Ericsson had failed to recognize the New Mexico plant as a bottleneck in a complex, interconnected global supply chain.

Ericsson is not the only company to suffer a catastrophe due, in part, to the complexity of its own systems. In February 1995, Barings Bank, Britain's oldest merchant bank (it had financed the Napoleonic wars, the Louisiana Purchase and the Erie Canal) went from strength and prestige to bankruptcy over the course of days. The failure was caused by the actions of a single trader — Nick Leeson — who was based in a small office in Singapore. Soon after Leeson's appointment as general manager of Barings Securities Singapore, he used a secret account to hide losses he sustained engaging in the unauthorized trading of futures and options. The complexity of the Barings systems enabled Leeson to fool others into thinking that he was making money when in fact he was losing millions. But after the January 1995 Kobe, Japan, earthquake had rocked the Asian financial markets, Leeson's accumulated losses — some $1.4 billion — became too enormous to hide, eventually leading to Barings' collapse.

In the past, companies have tried to manage risks by focusing on potential threats outside the organization: competitors, shifts in the strategic landscape, natural disasters or geopolitical events. They are generally less adept at detecting internal vulnerabilities that make breakdowns not just likely but, in many cases, inevitable. Vulnerabilities enter organizations and other human-designed systems as they grow more complex. Indeed, some systems are so complex that they defy a thorough understanding. In August 2006, a defective software program aboard a Malaysia Airlines jetliner flying from Perth, Australia, to Kuala Lumpur, Malaysia, supplied incorrect data about the aircraft's speed and acceleration. This confused the flight computers, which sent the Boeing 777 on a 3,000-foot roller-coaster ride. With more than five million lines of code, aircraft software programs have become too large and complex to be tested thoroughly and are fielded without any guarantee that they will always work.

Eric Bonabeau *is the CEO and chief scientific officer of Icosystem Corp., a Cambridge, Massachusetts-based consultancy that specializes in the use of complexity science to develop innovative business strategies. Comment on this article or contact the author through smrfeedback@mit.edu.*

Legal codes and agreements have also become increasingly complicated, often resulting in loopholes that others can exploit. In the spring of 2006, Boston Scientific Corp., a medical device manufacturer, acquired its rival Guidant Corp., outbidding Johnson & Johnson in the process. Earlier, in order to gain rapid antitrust approval, Boston Scientific and J&J had both signed deals with pharmaceutical giant Abbott Laboratories in anticipation of the Guidant acquisition. J&J was first to strike a conditional licensing deal with Abbott, but the agreement failed to include a noncompete clause so Abbott was then able to help Boston Scientific — which it happily did. After Boston Scientific outbid J&J, it sold Guidant's lucrative stent business to Abbott to alleviate regulators' monopoly concerns. Thus, by exploiting a weakness in a complex legal agreement, Abbott was able to claim the best part of Guidant's business in return for its assistance of Boston Scientific.

Time is often an enemy not only because of obsolescence and increasing wear and tear but also, ironically, because of the human ability to adapt to untenable situations and resist change. Consider, for example, the information technology system of the U.S. airlines Comair Inc. In 2004, after its acquisition by Delta Air Lines Inc., Comair was forced to shut its business for several days because of an overloaded legacy crew-management system. The result: 3,900 cancelled or delayed flights, nearly 200,000 stranded passengers and a loss of $20 million in operating costs and lost revenue. Not only had the system exceeded its capacity (Comair's business had been growing healthily for years), it also suffered from the addition of layers and layers of applications. Yet Comair's IT department was reluctant to upgrade the system, as users had grown used to it. To make matters worse, Comair's acquisition by Delta created additional complications: Tight constraints on capital expenditures and friction between Comair employees and their new owners had eroded the IT department's commitment. This combination of technological and organizational complexities created the "perfect storm" for an internal meltdown.

How Complexity Breeds Fragility

As companies increase the complexity of their systems — products, processes, technologies, organizational structures, contracts and so on — they often fail to pay sufficient attention to the introduction and proliferation of loopholes and flaws. As a result, many firms are continually making fixes, which then adds to the total cost of ownership and often creates new problems. One estimate is that 20%–50% of all fixes to software bugs introduce new, unknown problems.[1] Moreover, backward compatibility in software and other products might be desirable to consumers, but it in fact creates the conditions for brittleness. Vincent Weafer, senior director at Symantec Security Response (a division of Symantec Corp., based in Cupertino, California) has warned that making Microsoft Corp.'s new operating system, Vista, compatible with earlier versions of Windows creates vulnerabilities.[2]

To exacerbate matters, the possibility of random failure rises as the number of combinations of things that can go wrong increases, and the opportunity for acts of malicious intent also goes up. Build new applications on top of legacy systems, and errors creep in between the lines of code. Merge two companies, and weaknesses sprout between the organizational boundaries. Build Byzantine corporate structures and processes, and obscure pockets are created where bad behavior can hide. Furthermore, the enormous complexity of large systems like communications networks means that even tiny glitches can cascade into catastrophic events.

The Inevitability of Disaster

In fact, catastrophic events are almost guaranteed to occur in many complex systems, much like big earthquakes are bound to happen. Indeed, the statistics of events in manmade systems is starting to closely resemble that of destructive natural phenomena. Consider the massive blackout that occurred on August 14, 2003, when a power line tripped somewhere in northeastern Ohio. That tiny accident triggered a series of cascading failures that propagated throughout the sprawling, complex, interconnected North American power grid, costing the United States between $4 billion and $6 billion, according to the U.S. Department of Energy. John Doyle, a mathematician at the California Institute of Technology in Pasadena, examined data from the incident and found that the blackout was no statistical anomaly: It was bound to happen, even "overdue." Although the DOE might disagree with that assessment, the fact is that regulators and policy-makers are currently ill-equipped to design and administer such complex systems. (Through a postmortem analysis, the DOE would later learn that northeastern Ohio is an electricity choke point, to both Michigan and the upper Midwest as well as to New York and Pennsylvania to the east.)

The August 2003 blackout was an accident. In many other cases, malicious intent is the culprit, and that is one of the most frightening thoughts of all: Any sufficiently complex system will inevitably have internal weaknesses or loopholes, some of which are bound to be uncovered and exploited. Myriad threats abound, from international terrorists to high school hackers. "It's not if, it's when are you going to get hacked," says Kevin Mitnick, the notorious hacker who served five years in federal prison for high-profile, corporate break-ins. According to him, "Computer systems are complex. There will always be ways to break in."[3]

The possibilities for mischief are endless: people who manipulate search-engine results by creating bunches of fake links, competitors that leech off a company's intellectual property by taking advantage of a badly written patent, ordinary air travelers who save money by gaming overly complicated frequent-flyer programs. The result is often an arms race between defender and attacker. As companies add increasingly sophisticated firewalls to protect their computer networks, for instance, hackers develop more ingenious ways to breach those systems.

What Really Causes Failures?

The internal weaknesses of a system tend to reveal themselves in times of external turbulence and stress. In other words, the cracks might not become apparent until something unusual occurs on the outside, that is, when it is probably too late. But an external trigger must not be confused with the cause of — or be blamed for — a catastrophic event. The Kobe earthquake and the ensuing large fluctuations in the Asian financial markets did not "cause" the fall of Barings; one too many bookings on Comair did not "cause" the meltdown of the airline's information system; and a line tripping in Ohio did not "cause" the blackout of August 2003. These external triggers merely served to highlight the existence of internal flaws, which deserve the true blame for those catastrophes. When exposed, the internal weaknesses sometimes become obvious in retrospect, but that's not always the case. For example, whereas the collapse of Barings can be traced to a single trader in a dysfunctional culture, the structural weaknesses of the United States power grid are still poorly understood.

Merge two companies, and weaknesses sprout between the organizational boundaries. Build Byzantine corporate structures, and obscure pockets are created where bad behavior can hide.

When the potential consequences are catastrophic, companies must also take into account events that are unlikely to occur. And in the case of malicious attacks, disasters might be even more likely to occur than predicted by random failure theory because such calculations do not factor in intent. When malicious intent is present, an attacker needs only one vulnerability to wreak havoc.

Forestalling Fatal Flaws

This raises the question: Without the benefit of perfect foresight, how can businesses uncover and forestall the fatal flaws lurking within their organizations? That complexity is a source of inevitable risk is not a new idea. Over 20 years ago, sociologist and organizational theorist Charles Perrow analyzed the near-cataclysmic events at the Three Mile Island nuclear power plant and developed his "normal accident theory": the idea that, as the complexity and interconnectedness of systems increase, major accidents become inevitable and thus, in a sense, "normal."[4] But what's new are the practical tool sets that have been developed to help companies better manage the risks of complex systems.

There are three complementary strategies for mitigating risk: (1) assess the risk to make better-informed decisions, such as purchasing an insurance policy to cover the risk, (2) spot vulnerabilities and fix them before catastrophic events occur, and (3) design out weaknesses through resilience. These ideas have been around for years, but researchers have had to reinvent them in the context of extremely complex, interconnected, cascade-prone systems. As such, the three approaches are relevant not only to engineers and software developers but also to anyone writing complicated contracts, designing complex products, figuring out how parts of a large corporation should work together and so on.

From the Bottom Up: Predict and Act

Catastrophic events often result not from a single cause but from interconnected risk factors and cascading failures. Each risk factor taken in isolation might not cause a disaster, but risk factors working in synergy can. The bad news is that complex, interconnected systems generate many, sometimes unexpected or counterintuitive, vulnerabilities. But the good news is that if a small, localized, single event can trigger cascading failures, then perhaps a small, localized, single intervention could act as a circuit breaker. Of course, finding that circuit breaker requires a deep understanding of how the behavior of a complex system emerges from its many constituent parts. Two different approaches make that kind of analysis possible: agent-based modeling and network theory.

Agent-based modeling is a technique that simulates complex systems from the bottom up in order to capture their emergent properties. The computer models can be of business environments, such as a trading floor of buyers and sellers, in which many virtual participants interact autonomously. Each of those software agents is encoded with a degree of artificial intelligence, enabling them to make decisions and act based on a set of rules. Additionally, the agents may change or evolve, allowing unanticipated behaviors to emerge.

Agent-based models can be run repeatedly — even millions of times — to capture rare but large events that result from unlikely synergies between risk factors. Such low-frequency, high-impact events constitute the so-called "long tail" of the risk distribution. In the past, traditional methods to estimate risk failed to capture the real statistics of long tails. Indeed, certain risks can't be insured today because insurers don't know exactly how to quantify them. Or the estimates have been inaccurate because of the law of small numbers, that is, the tendency to draw broad conclusions from a tiny number of events. Agent-based modeling, with its ability to run millions of scenarios in silico, or on computers, enables the long tail to be quantified. Actuarial calculations then become possible, and insurance products can be devised.

Consider how the Société Générale Asset Management of France has used agent-based modeling to assess its operational

risk, loosely defined as the risk of large losses resulting from organizational dysfunctions in processes, people or systems. For such calculations, the past is typically a poor predictor of the future because of limited statistics and changing conditions. Instead, the best way to assess risk and discover potentially catastrophic synergies is to simulate how the organization operates, something that SGAM was able to do by creating a computer model of virtual agents to represent employees interacting with one another as they performed certain tasks, such as converting monetary currencies or buying and selling stock. Such simulations have the added benefit of enabling companies to design mitigating strategies and to proactively manage their risk.

Agent-based modeling can also be used to assess security risks in a firm's IT infrastructure, to study the organizational network that results from the merger of two companies and to investigate a number of other complex systems. For example, a consumer goods corporation used the technology to design a better incentive structure for its country managers. The change enabled the company to lower its operating costs while minimizing the risk of catastrophic fluctuations in product stock.

Another approach is network theory, which enables companies to evaluate the risk of interconnected business systems. Much research has recently focused on "scale-free networks." In such systems — the Internet is a classic example — some nodes (called hubs) are highly linked to others, while most nodes have only a much smaller number of connections. Scale-free networks can be damaged severely by a targeted attack on just a few of their hubs, and computer viruses that spread on scale-free networks are much more difficult to eradicate. Other research has studied the conditions in which cascading failures can result from a simple accident, such as the tripping of a line in northeastern Ohio. In one such application, network science methods have investigated vulnerabilities in the European power grids, and the results suggest the use of stable islands — clusters of nodes that can be isolated quickly from the rest of the network to protect them from a cascading failure.

If a tiny, localized, single event can trigger cascading failures resulting in a catastrophe, then it just might be that a small, localized single intervention could act as a circuit breaker.

Network theory can also be used to study how people interact with each other in an organization. Research on such social networks has found that companies typically contain a handful of individuals (often unknown to management) who hold the organization together and whose resignation, dismissal or retirement would threaten the firm's basic functions. This vulnerability occurs because as organizations grow more complex and difficult to navigate, employees have to find unofficial, nonstandard ways of doing their jobs, thereby creating a shadow organization of social contacts that is very different from the company's official top-down hierarchy. Thus, the first step for management is to identify the shadow organization from the bottom up, including the people who are playing the roles of unofficial hubs. And the next step is to make the roles of those individuals more official and to rethink the organizational structure and processes so that they better reflect how work is actually being done.

Diversity-Based Testing

The best bet for uncovering weaknesses that are already embedded in a complex system is to test it with "attacks," for example, by having hackers try to break into a computer network. But the difficulty lies in designing attacks that really test the system — as opposed to merely confirming its designer's assumptions. Take, for example, the military practice of red teaming, in which troops destined for overseas combat are pitted against a fictitious enemy. The exercise is often scripted, which defeats its purpose because it doesn't prepare the troops for surprise tactics that might be used in real combat. In other situations, such as that of Comair, users of a system adapt to its flaws and are unable to see the full extent of the problems. What is needed for systems testing is an open mind — or, even better, many open minds. As Eric Raymond, the noted advocate of open-source software, once wrote, "Given enough eyeballs, all bugs are shallow." That philosophy is one reason for the success of open-source programs, like the Linux operating system, which are commonly acknowledged to be more robust than software developed in a proprietary manner.

With open-source software, developers regularly challenge each other's code, finding bugs, which leads to fixes, then more bugs, more fixes and so on. That process of diversity-based testing can involve thousands of developers who are geographically dispersed around the world. A related approach is extreme programming, in which two or more programmers work on writing the same code and then criticize and rewrite each other's code. The apparent duplication of effort is more than made up for by a significant improvement in the quality of the code, including a substantial reduction in the number of errors.

The principles of open-source software are now being taken a step further. In the field of cryptography, the National Security Agency has been actively promoting key-breaking from "ethical

cryptohackers." And diversity-based testing has other intriguing applications. Consider, for example, the U.S. Customs and Border Protection's National Targeting Center in Reston, Virginia. Workers there sift through reams of data in search of miniscule clues that might indicate the presence of terrorist activity. To find such "red flags," the agents use their own set of screening rules, on the order of 180 or so, and every few months they get together and exchange information on which rules seem to work and which don't. As a result of this peer-to-peer exchange, the agents alter their own set of rules, increasing the diversity — and effectiveness — of the screening.

Although a diversity of perspectives is often desirable, leveraging it is not always easy. One problem is that organizations often suppress the expression of diverse perspectives, either explicitly or through some implicit group dynamics. Examples include groupthink (when a team rapidly converges toward a single, although not necessarily optimal, opinion) and self-censorship (when the leader of a team expresses an opinion that then inadvertently makes other views socially unacceptable). To avoid such pitfalls, organizations need to make a concerted effort to let diversity express itself. Workers in the aviation industry, for instance, can file anonymous reports with the U.S. Aviation Safety Reporting System about any incidents that could pose a threat to safety. To encourage participation, the information cannot be used by the Federal Aviation Administration's enforcement authorities but is available to more than 150,000 aviation professionals and enthusiasts. Similarly, the U.S. Veterans Administration created an external system for workers to report patient safety information on adverse events and close calls that are not filed with the VA's internal system. The goal is for people to report incidents without the fear of future litigation so that the information can be used to improve healthcare practices and procedures.

Some companies are tapping into the diversity of their own workforces by deploying cutting-edge methods like artificial information markets. In such markets, popularized by James Surowiecki's book *The Wisdom of Crowds,* participants buy and sell stocks or futures in certain events. For example, employees might be asked to assess the risk that a new innovation might fail or the likelihood that customer demand for a popular product would soon decline. Artificial information markets can often provide more reliable assessments than those from even so-called "experts," similar to the way in which a crowd's average guess for the number of jellybeans in a large bowl will typically be much more accurate than the estimate from any one person.

Just because a company doesn't embrace open testing does not mean that its hidden vulnerabilities will remain hidden. Indeed, this key lesson should never be forgotten: If a system has a loophole, it will be discovered; it's just a matter of time. Consider Web sites like FlyerTalk.com, FareAlert.net and AirfareWatchdog.com that uncover pricing errors (a $39 roundtrip fare from Boston to

London, for example) and loopholes that consumers can exploit. Many airlines keep an eye on those Web sites to correct pricing errors and close loopholes quickly after they are reported.

If diversity can't be found in the real world, it can be created in silico. In a process called artificial evolution, computers are used to mimic the biological evolution of parasites and their hosts. Biological evolution has excelled at discovering weaknesses in virtually all living things, often by creating new organisms to exploit those vulnerabilities. Artificial evolution recreates that process on a computer, effectively breeding strategies to exploit a system's weaknesses. The approach, which can generate unexpected and surprising types of attacks (and failures), has been deployed successfully in a number of applications. Thanks to evolutionary testing, DaimlerChrysler Corp. was able to discover unexpected failure modes in its automated car-parking system, and the U.S. Navy was able to find unexpected vulnerabilities in the complex fluid control system of its future destroyer.

Robust Designs

Cellular assemblies, the immune system and animal societies are but a few examples of self-organizing biological systems. Self-organization affords systems a unique ability to respond, reorganize and counterattack in situations of stress, external challenges or internal failures. Instead of being hardwired to tackle a small set of predefined situations, self-organizing systems are designed to alter their structures and behaviors when the world around them changes or when some of their constituent units fail. Engineers have been fascinated by this concept for decades but only now are they finding it practical to build and run self-organizing, industrial-strength systems.

Consider data storage. The high cost of robust data lies in the complexity of its management, not in its actual storage. That is, fully protected enterprise-enabled storage, including installation, protection and management, costs $850 per gigabyte, whereas the actual disk space costs just $1 per gigabyte. As the price of data storage continues to soar because of management and administration costs (and not because of hardware), several companies are developing storage networks in which data manages itself according to a few simple, local-based rules such as keep recently used files close to users. Robustness is ensured by the number of replicas of the data, and the system provides automatic healing and data recovery in the vicinity of failures. The results are impressive: Not only has the total cost of ownership of data storage been dramatically reduced, the system is so robust it can survive virtually any catastrophic event.

One side benefit of self-organization is that it relies on modular components from the start. In other words, a self-organizing system is developed by first designing its constituent modules. Modularity allows for greater flexibility, which can lead to increased robustness. When a part begins to malfunction, it can be replaced quickly or swapped out altogether, with the remaining modules recombining to adjust for its absence. Although modular software development (exemplified by the use of object-oriented programming languages) has been around for years, modularity approaches are only now being investigated for other applications. The U.S. Department of Defense, for example, has been experimenting with technology that relies on both self-organization and modularity for aerial surveillance and reconnaissance missions. Instead of single, expensive unmanned aerial vehicles like those used during the second Iraq campaign, the DoD is considering using self-organizing swarms of cheap UAVs. The new approach has the theoretical benefit of increased resilience: If one module fails, the mission can still be completed because the swarm can reorganize quickly. Moreover, the ratio of human operators to UAVs is significantly reduced.

The capacity to self-organize affords systems a unique ability to respond, reorganize and counterattack when they are under stress or exposed to external challenges or internal failures.

Diversity is another key to robustness. Consider the "malware" that was recently spread to harm smartphones. Because each handset manufacturer had implemented a slightly different version of the Symbian operating system, the viruses tended to infect only a fraction of devices. Indeed, software can be made less vulnerable when programmers purposely write code variations that have no functional effects but make it less likely for a single threat to have widespread impact. Research into code diversity is still in its infancy, but it has already attracted the interests of organizations like the U.S. Defense Advanced Research Projects Agency, which is funding a program investigating the application of the principles of biological immunity to make operating systems more diverse and hence less vulnerable.

Regarding Simplicity

Complexity is often viewed as the cost of doing business, even as a means to obtaining a competitive advantage. But is complexity truly a necessary cost, and what kind of competitive advantage can it provide if competitors can do pretty much the same thing but more simply? We might take complexity for granted, but

shouldn't we instead be striving for — or at least be considering — simplicity whenever possible, for example, with respect to software, the tax code, the health care system and so on? We hate complexity and crave simplicity. And yet…

Simplicity is not happening, and it's not about to occur in any meaningful and sustainable way. Much of the reason for that is self-interest, not just the self-interest of software vendors looking to milk the newest release of a cash cow product but people's own self-interest. Take, for example, the U.S. tax code. Although everyone agrees that it's far too complex, the system represents so many interconnected vested interests that any significant simplification would create a public outcry, initiated by those affected negatively. Consequently, people can't agree how best to simplify the tax code and the only direction is toward more complexity, not less, and that progression will continue unless war or revolution resets the entire system.

The same is true of pretty much all complex systems: Touch anything and chances are the change will hurt someone or inadvertently make the system more fragile, not more robust. Ironically, what forced Ericsson out of the mobile handset market was not necessarily the plant fire in New Mexico but the decision made earlier to simplify the company's supply chain by using just a single source for some of its components. The fire then exposed the fragility of the simpler supply chain. A key lesson here is that the drive toward simplicity sometimes comes at the expense of robustness and reliability. That is, simplifying a complex system is tricky because any action can have drastic unintended consequences.

Even when simplicity should create value, it often destroys it. Consider American Airlines' decision in April 1992 to simplify its fare structure. (Ironically, it was American Airlines, a subsidiary of AMR Corp. of Fort Worth, Texas, that had pioneered complex, differentiated pricing, which had clearly been a competitive advantage when it was introduced following the deregulation of the U.S. domestic airlines industry.) The new plan reduced the number of fares on any route from dozens of prices to just four fares: first class, coach, 7- and 21-day advance purchase. Intense price competition from the other airlines ensued, destroying value for the entire industry and forcing American Airlines to drop the plan within weeks. In the end, complexity was restored. (Of course, Southwest Airlines Co. of Dallas thrives with a simple fare structure, but the company has a different business model. The point is that going from complex to simple is an uphill battle. Disruptive simplicity from new entrants, however, is a different story.)

Moreover, simplicity is not, well, that simple. Simplicity at one level can be deceiving when it relies on layers of increasing complexity at other levels. Making a technology simpler for the end user, for example, rarely means simplifying the technology itself. In fact, it often means the opposite, as more of the complexity management is simply transferred from the user to the technology. Making Microsoft's Vista simpler to use, for instance, did not make the operating system less complex inside. User simplicity can even be dangerous when it instills a false sense of comfort and security to those unaware of the underlying complexity. And modularity and diversity — two approaches that have been discussed in this article — can thwart a system's simplicity, as complexities can easily creep into the interfaces between diverse components.

Obviously, the surest way to reduce risk is to disconnect, thereby blocking potential synergies between risk factors and cascading failures. Remove the connections and, to a large extent, risk disappears. In fact, a widespread joke in IT security is that the best way to protect a computer from viruses is to disconnect it completely from any network or other computer. There have always been tradeoffs between security and convenience, but the balance between the two has moved far past the point at which being completely disconnected is a viable option. In other words, complexity is here to stay. It might not be desirable but it is inevitable because, more often than not, simplicity is simply not practical.

Charles Perrow, the noted sociologist and organizational theorist, has written that recent works in the risk area "ask how we can make risky systems with catastrophic potential more safe, a question that takes for granted that they must run hotter, be bigger, more toxic, and make super demands upon members. We should address the improvement of safety but, in addition, address the role of production pressures in increasingly privatized and deregulated systems that can evade scrutiny and accountability."[5] Perrow's views on deregulation aside, he has a point: production pressures and cost-cutting can increase risk. Barring any major wars, revolutions or terrorist-induced calamities, complexity and risk will likely continue to grow, and the best we can do is walk on the edge of the complexity cliff without falling off.

REFERENCES

1. F.P. Brooks, "The Mythical Man-Month: Essays On Software Engineering" (Reading, Massachusetts: Addison-Wesley, 1975).

2. S. Hamm, "Heading off the Hackers," BusinessWeek, Aug. 10, 2006, www.businessweek.com.

3. G.T. Huang, "The Talented Mr. Mitnick," Technology Review, March 2005, available at www.technologyreview.com/Infotech/14231/.

4. C. Perrow, "Normal Accidents: Living With High-Risk Technologies" (Princeton, New Jersey: Princeton University Press, 1999); see also S.D. Sagan, "The Limits of Safety: Organizations, Accidents, and Nuclear Weapons" (Princeton, New Jersey: Princeton University Press, 1993).

5. C. Perrow, "Normal Accidents," 379.

Reprint 48416.

MITSloan
Management Review

PDFs ■ Reprints ■ Permission to Copy ■ Back Issues

Electronic copies of MIT Sloan Management Review articles as well as traditional reprints and back issues can be purchased on our Web site: *sloanreview.mit.edu* or you may order through our Business Service Center (9 a.m.-5 p.m. ET) at the phone numbers listed below.