Say the word “hacker” and an indelible image undoubtedly comes to mind. Yes, it’s different for each of us but probably not by much. It’s unfortunate that over the years the negative nuance of the stereotypical hacker has taken center stage in the global spotlight, while the brilliant, skillful, and, yes, ethical expertise of its growing population is often ignored, rejected, or held suspect.

This month’s special section explores the practical lessons, knowledge, and know-how that computer science can gain from the hacker community—the white hats, as they are known, who are paid or inspired to weed out system vulnerabilities, who alert security system designers, owners, or users of their security weaknesses as well as how they might have already been exploited. Guest editor Gregory Conti may appear an unlikely advocate for this cause, but as an Academy Professor of Computer Science at the U.S. Military Academy at West Point, he finds the literature and technical conferences hosted by the hacking community professionally rewarding. He strongly believes that hacking and computer science are inherently intertwined and should foster far more cooperation and collaboration. He hopes this section narrows that divide through its technical articles and commentaries that present some of the great—good—work coming from the world of hackers.

Also in this issue are two perspectives on software piracy. A study by Bagchi et al. presents some empirical results on the causes of global piracy and how effective the war against it is working. And Cronan et al. trace the incidences of software misuse—including piracy—finding it starts early, despite the many security measures in place.

David Kang and Roger Chiang describe a systematic approach for managing IT changes from a project management perspective. Christopher Carr examines the relationship between IS departments and IS users to determine how to better appreciate and manage both sides for mutual benefit. And Barry Shore explores the challenges of enterprise integration when companies have multiple sites around the globe, each acclimated to its own way of working.

Ronald Boisvert and Mary Jane Irwin, co-chairs of ACM’s Publications Board, explain the reasons behind the development of ACM’s Plagiarism Policy, urging all authors to become familiar with its provisions. Pamela Samuelson wonders if standards should be eligible for copyright protection. And, finally, in his farewell address, ACM President David Patterson explores the major challenges that lie ahead for the association, confident we are in excellent shape to meet them.
BRITANNICA BRISTLES
In the battle for precision, the Encyclopædia Britannica recently called for Nature magazine to retract an article it published last December impugning the accuracy of Britannica’s Web site. The article compared the accuracy of two online reference resources—the renowned leader Britannica and the upstart Wikipedia, which is created, written, and edited completely by non-experts. The findings indicated the accuracy presented by the reigning authorities was not as pristine as expected. Britannica issued a statement last April (corporate.britannica.com/Britannica_nature_response.pdf), then followed with a half-page advertisement in the London Times criticizing the study and demanding a retraction. Nature responded it was indeed surprised by the findings of the comparison study, conducted by 42 experts, but the tallied results found the experts picked up errors (the great majority of them minor) at a rate of about three per online Britannica item compared to about four per Wikipedia item. Nature stands by its story; responses to the points raised by Britannica can be found at www.nature.com/nature/britannica/index.html.

SNOOZE HEADLINES
The days of the catchy newspaper headlines full of wit and style and the occasional double entendre are fading to blah as a growing number of news outlets are bowing to the ‘bots. Search-engine ‘bots, that is, designed to scour online newspapers, magazines, and TV news headlines worldwide to rank and list news items for the likes of Google and Yahoo. The New York Times reports the news media can no longer deny the global online audience their news may attract; that often means making it easy for ‘bots to find their stories. News organizations have been rewriting or distilling the traditional eye-catching headlines with more subdued, logical ones for software to catch.

One California newspaper recently renamed some of its sections to make them easier for ‘bots to find; “Real Estate” became “Homes”; “Scene” became “Lifestyles”; and the dining out section became a bland “Taste/Food.” Ironically, search engines often rework their algorithms to avoid self-serving manipulation by news outlets. As one news executive opined: “Part of the craft of journalism for more than a century has been to think up clever titles and headlines, and Google comes along and says, ‘The heck with that.’”

THE GEEK FACTOR
Contrary to popular belief, more Americans are interested in science news than ever before, though the fear of nerdom often keeps it a secret. A new study from Imaginova—a New York-based multimedia company that researches people’s interest in space, science, and technology—found a growing number of people remain silent about their scientific interests so as not to be perceived as geeks. The findings of the

"Americans like to look at the Internet as a frontier. Chinese prefer to look at it as a fenced field. Where it is productive, you may roam; where there is danger, you may not.”

—David Wolf, an American technology consultant working in Beijing.
study, based on online polling and focus groups and designed to reach beyond those with confessed scientific interests, found a huge population segment, age 18–54, to be “intellectually curious” people who fall into three distinct groups: Science with Passion (14%), with a prime interest in nature, medicine, and the environment (53% of them female); Money, Success, and Science (11%), a career-driven population with a prime interest in technology and science TV programming (64% of them male); and Style with Science (15%), with a prime interest in technology, weather, and nature (57% of them male). “We found that consumer interest in science and science-related products are both larger and more complex than previously thought,” said a researcher for the study. In fact, 85% of the intellectually curious respondents are intrigued by scientific breakthroughs and innovation and tend to visit science-oriented Web sites frequently.

**Sharper Image**

Researchers from the University of Arizona have designed glasses that change focusing power by harnessing the images seen by the eyes. BBC News reports the lenses for these glasses use electrodes to alter the optical properties of liquid crystal between layers of glass. The research team contends it may one day replace bifocal and varifocal lenses, though U.K. experts suggest the lenses would be so heavy they would require especially thick frames. Still, where area-divided lenses tend to change focus slowly and limit the peripheral vision of the wearers, the researchers point out this breakthrough will allow wearers a clear view of the entire lens area. The model uses thick lenses to support the weight of the liquid crystal; it operates at a low voltage that can switch focal power less than one second after registering the vision seen through the lens on a person’s eye. If power is suddenly lost, the lens reverts to a configuration with no added focusing power.

**Riding the Brain Train**

Japan, home to some of the world’s leading designers and vendors in the video game arena, is awash with new clientele that has taken their industry by storm. Senior citizens throughout the country, particularly women, are the fastest-growing segment of the video and computer game population—all bent on keeping their mental state sharp and active. Brain-training programs are fast becoming Japan’s hottest-selling market with 60–80+ year olds enrolling in classes and starting their days with invigorating mental gymnastics of quick math and memory-jogging quizzes on Sony PlayStations and Nintendo GameCubes, among others. Reuters reports this brain-training wave was inspired by the work of Ryuta Kawashima, a professor of brain science at Tohoku University, who created a program around a theory that a daily dose of mental calisthenics can train the brain at any age and improve memory in all patients, even those diagnosed with dementia. For video game makers eager to expand their customer base beyond teens, software featuring Kawashima’s brain-training program has been a huge and surprising electronic gold mine. “We were able to ride the brain craze,” said one Nintendo spokesperson, who admitted the firm never expected such success from this market. Indeed, the older population in Japan is so taken by this phenomenon that even the Rubik’s cube has made a comeback, with sales of the famous puzzle increasing fivefold in Japan last year. 

Send items of interest to cacm@acm.org
I agree with David A. Patterson’s “President’s Letter” (“Computer Science Education in the 21st Century,” Mar. 2006) that for CS to be interesting to students, it must also be inherently challenging and relevant. I have too often had to turn away job applicants because they wanted to sit undisturbed writing a masterpiece from scratch. Industry has no time for that. It needs requirements collected and analyzed, user documentation written, code tested, existing code adapted, and occasionally new code written but only as glue for preexisting blocks.

In the lithography community where I come from, we’d all be fired if we delayed a billion-dollar fab because somebody decided to field some “bright idea” in software. It may be somewhat boring, but good software engineering is no different from any other engineering specialty in this regard.

Patterson defined the root cause of the poor fit between today’s CS graduates and the needs of industry, writing “There is a huge disconnect between the experience of most professors, who have never worked as professional programmers … and the way in which cutting-edge software is written today.” Perhaps the profession would help itself eliminate the inbreeding sometimes seen in universities, insisting instead that new faculty be drawn exclusively from industry.

Richard L. Lozes
Dublin, CA

In a recent effort to recruit interns and new grads at Apple Computer, I found the number of them familiar with parallel programming issues (such as threading and locking) was rather low compared to those familiar with, say, popular CS algorithms. While recognizing that algorithms and data structures are important, the 21st century, as David A. Patterson pointed out (Mar. 2006), is going to be a tale of reworking virtually all software to perform more efficiently in parallel. We need CS graduates who understand these techniques.

Andy Belk
Menlo Park, CA

To Make CS Relevant, Give It an Industrial Focus

The Tiger Means Healthy Competition

In his “Practical Programmer” column (“Is the Crouching Tiger a Threat?,” Mar. 2006), Robert L. Glass raised five specific concerns. None of them, however, will turn out to be a serious threat to the U.S. position in computer technology. First, quantity does not equal quality. The combined population of China and India is nearly 20 times the population of the U.S.; meanwhile Japan’s population is almost 128 million and Indonesia’s almost 246 million. When more families in these countries have enough money to send their children to college, whether at home or abroad, the number of Asian students enrolled in CS (and everything else) will be greater than the number of native U.S. enrollment. Similarly, there are ultimately likely to be more IT practitioners in Asia than in the U.S., along with more IT-related research papers written and published by Asian institutions.

In terms of quality, Asian computing still has a long way to go. How many top CS scholars and researchers today are from the Asian computing industry? And how many top Asian software companies have their own branded products—the equivalent of Windows, Oracle, and SAP—that dominate a particular software market?
Glass was concerned that “U.S. dominance is going to be overthrown.” Even if Asian computing would soon challenge the U.S.’s dominant position, it represents healthy competition and the promise of a better computing community for everyone worldwide, not just in the U.S.

U.S. success is largely the result of a combination of the country’s scholarship and practice. Both are motivated by the challenge posed by other countries, including those in Asia, trying to improve their own scholarship and practice, not because the U.S. has figured out a way to “get rid of threats.”

Finally, why didn’t Glass express similar concern about the threat posed by the European computing community against U.S. computing dominance?

WAYNE HUANG
Athens, OH

ROBERT L. GLASS (MAR. 2006) wrote that the U.S. has dominated the IT industry thanks to an ecosystem involving industry, universities, and government. But today U.S. government investment in IT research may be falling off, leading to reduced interest by students in university CS programs. The ecosystem may finally be losing energy on its own.

U.S. IT dominance is characterized by two main factors: The first is U.S. companies, which lead because they are able to tap talent regardless of geographical location; relatively few non-U.S. companies produce equivalent world-class software products. The second is programmers, but offshoring is likely to erode U.S. dominance.

The real danger to the U.S. is not the loss of IT jobs but deflation of the ecosystem. Software leaders in U.S. universities and the U.S. government must therefore keep funding the research. When funding is available, students follow. Meanwhile, business leaders are less likely to focus on the U.S. software industry, as their cost-benefit analyses no longer reflect U.S. programmer dominance.

UMESH PANCHAKSHARAIH
Richmond, CA

AS YOU MAY GUESS FROM MY name, I’m originally from one of the countries (Korea) cited by Robert L. Glass (Mar. 2006). I’d like to share what I think are the implications of the facts Glass explored. For example, he wrote about the increasing number of international students and about Asian research publications. But in fact the best international students stay in the U.S. after school and represent a serious “brain drain” for their home Asian countries. Even though many other students return home to teach and try to make their home universities more competitive, U.S. universities have maintained their dominance.

Some Asian universities evaluate academic performance based on number (not quality) of journal publications, but their researchers tend to publish more marginal-quality articles than what could be called high quality. I am an alumnus of the Korea Advanced Institute of Science and Technology but admit such universities are nowhere near the top 10 U.S. universities (at least for now).

The number of students majoring in CS is declining, even as the number of programmers is increasing in Asia. The reason in the U.S., as well as in some Asian countries (such as Korea), is that top students can earn much more by becoming lawyers or physicians than by becoming computer scientists. Bill Gates and the dot-com bubble made some people re-think this proposition, but their reflection didn’t last long.

In Asian countries, programming jobs are considered (relatively) desirable and stable. They are seen as much better than, say, sales/marketing jobs and may be a reason for the increasing numbers of professional software developers in Asian countries.

The real problem is globalization. If you (perhaps the majority in the U.S.) value globalization you would be unlikely to view globalization as a threat. Outsourcing programmer jobs is like buying made-in-China goods at Wal-Mart. Globalization delivers cheaper goods and cheaper software. Anyone unhappy about that should join me (and many others) in opposing globalization.

ZU KIM
Berkeley, CA

THE ASIAN TIGER IS INDEED trying to take over the IT world. Apply the common-sense test to radios, TVs, cameras, and cars. How long did Europe and the U.S. dominate the world of RF communications, CRTs, photography, and automotive engineering, not to mention DRAMs and LCDs? And how quickly did they lose their hegemony?

Wait, you might say, these are manufacturing issues. How exactly will software spare us from the
same forces that moved dominant automotive engineering from General Motors to Toyota? Cost of manufacturing alone does not explain why the market for video game consoles isn’t dominated by U.S. vendors.

Meanwhile, leading scientists worldwide come to the U.S. after being forced to retire back home, then produce the most advanced research of their lives. This is an interesting point and would hold sway if we were talking about scientific discovery. This counter-argument didn’t do much for American Motors.

In The Rise and Fall of the Great Powers [1987], Paul Kennedy, a history professor at Yale University, wrote that a society’s commitment to education is a leading indicator of its economic and political power. Without a commitment to education, a dominant society (think England in the first half of the 20th century) is likely to see its influence in the world reduced to a level commensurate with its share of the world’s population and natural resources. The decline of U.S. hegemony in the IT arena appears inevitable.

Collin Park
Redwood City, CA

Shocked By Email Color

I was shocked by how the article “Does Color in Email Make a Difference?” by Moshe Zviran et al. (Apr. 2006) encouraged spam. It described an experiment in which the authors sent junk email to a mailing list of 1.4 million addresses and studied how the color of the email affected the number of responses. I’m not enough of a lawyer to know whether or not this experiment violates any laws or conventions about human-subject experiments, but it certainly does not seem like the type of work Communications should seem to be encouraging. It has value only to spammers and was done by enlisting the involuntary cooperation of 1.4 million people.

David Evans
Charlottesville, VA

Authors Respond:

While the Internet is often misused and many commercial email messages are spam in the sense they are unsolicited, inappropriate, and of extremely low value, the email in our study was certainly appropriate. It was sent to active users of IncrediMail who had voluntarily agreed to receive periodic email as part of IncrediMail’s particular business model of e-commerce. IncrediMail’s messages are functional messages that inform its users of new technological developments and system enhancements. Any medium can be misused, but the more we learn about media use, the better we are able to recommend how to do so in effective and just ways.

Moshe Zviran
Dov Te’eni
Yuval Gross
Tel Aviv, Israel

Pursue Reality As a Competitive Advantage

Amy S. Bruckman’s response to Curtis Rhodes’s “Forum” comment (“Be Skeptical of Rhetorical Slight of Hand,” Mar. 2006) is even more objectionable than her “Viewpoint” “Student Research and the Internet” (Dec. 2005), which prompted Rhodes’s comment in the first place. She claimed “[An objective view of reality] is but one of many competing views about the nature of truth in a spectrum from objective to subjective.”

Subjective views of reality are a luxury only stable or slowly changing societies can afford. That is because anything other than an objective view of reality always involves some amount of denial. In a quickly changing environment, denial is destructive to the society. People (and societies) who see past it to objective reality have a competitive advantage over those who hold to other ideas.

Bruckman does her students a grave disservice by suggesting that the many different views of reality are equivalent and can be selected like one selects articles of clothing to wear for the day.

David Randolph
Plano, TX

Author Responds:

Recognizing that multiple views of reality exist and other people’s epistemologies may differ from one’s own is essential for understanding others, whether you agree with them or not. However, the fact that the views exist does not imply that all views are equivalent. In our class at Georgia Tech, we teach students basic ethical theories (such as Kantianism, utilitarianism, and social contract), trying to prepare them to make ethical decisions as computing professionals.

Amy S. Bruckman
Atlanta

Please address all Forum correspondence to the Editor, Communications, 1515 Broadway, New York, NY 10036; email: crawfordd@acm.org.
Let me start by paraphrasing another President’s farewell address [1]:

My fellow members of ACM:

A few days from now, after two years in the service of ACM, I shall lay down the responsibilities of office as, in traditional and solemn ceremony, the authority of the Presidency is vested in my successor.

I come to you with a message of leave-taking and farewell, and to share a few final thoughts with you, my fellow ACM members.

Like every other member, I wish the new President, and all who will labor with him or her ... I pray that the coming years will be blessed with peace and prosperity for all.

Our members expect their President and the rest of ACM leadership to find essential agreement on issues of great moment, the wise resolution of which will better shape the future of ACM.

While my two issues—the growing and graying of ACM—are not as great a challenge as the Military-Industrial Complex was in Eisenhower’s 1961 speech, the importance of these two issues to ACM and its members won’t lessen if they’re not taken seriously and not addressed soon.

THE GROWING OF ACM

The graph here shows the path of ACM’s professional membership over the last 25 years. Most organizations like ACM have seen member levels drop since the 1990s. ACM is unusual in that we are turning this trend around, and we’ve recently enjoyed four years of growth.

I believe the following five reasons present a great opportunity to grow ACM membership significantly, and I strongly encourage this move.

- Better recruiting of practitioners. This has been a long-standing problem for ACM since it expanded from being a research organization to
Most organizations like ACM have seen member levels drop since the 1990s. ACM is unusual in that we are turning this trend around, and we’ve recently enjoyed four years of growth.

embrace practitioners as well in the 1960s. Indeed, helping practitioners was a main reason that CACM’s editorial direction was changed in the mid-1980s. I believe the Professional Development Centre (PDC) and Queue magazine have recently helped this cause and I’m hopeful the new Professions Board will lay out a roadmap that will make ACM even more attractive to practitioners.

• Better recruiting of ACM conference attendees. ACM sponsored or co-sponsored about 150 conferences and workshops last year. The total attendance was about 25,000 technical conference attendees at ACM events and about 20,000 technical conference attendees at ACM co-sponsored events. By the end of my term, I will have given welcoming addresses at a half-dozen large conferences. In addition to explaining everything that ACM does and why it’s good economics to join [2], I always ask the audience how many are members. About 30% to 40% are ACM members, 20% to 30% are students, and so 30% to 50% are neither. Non-members pay a higher conference fees, which in some cases is higher than an annual ACM membership. Hence, this group, which may be 10,000 to 20,000 strong, seems ripe for ACM membership.

• Leveraging the rapid growth of the professional IT industry in China, India, and other countries relatively new to IT. It certainly looks like there will eventually be millions of IT workers in China and India alone. I have the strong sense that things are changing fast in such environs, and I hope we are not missing the boat as it is taking us longer to address the issue than one would hope. We are just starting two ad hoc committees to advise ACM on how to become the professional society of choice for these nations. Perhaps we can leverage the popular ACM International Collegiate Programming Contest. It just completed its 30th year with more than 5,800 teams from 1,700 universities in 84 countries on six continents. It would seem that the thousands of former participants from these countries who have either competed or helped organize the events might be a base on which we could build.

• Better recruiting of ACM student members. ACM has another 20,000 student members, but we only succeed in graduating about 20% into professional members. The Membership Services Board is examining this issue. I believe changes to our flagship publication, as well as becoming more attractive to practitioners, could increase the yield of our student members.

• Better recruiting of students to study our field. As I’ve mentioned a few times in this space, we chartered the Computer Science Teachers Association to help those teaching computing in pre-college institutions. Given the increasing competitiveness of getting into a good college, many students do not have the time to take a
computing course, even if there were well-qualified teachers available to teach them. Hence, we should look for ways to supplement textbooks and course material in math, biology, and physics that present exciting and intellectually challenging problems that computing can address, so that such students might be more willing to consider our courses when they get to college and have more flexibility in their schedules [4].

In talking to literally thousands of people about why they do or do not join ACM or other professional organizations, I've come away with a few strong opinions. First, everyone loves the ACM Digital Library, but most can access it through their institution without having to join ACM. Second, many are willing to believe the PDC is valuable for their continuing development, but they have yet to use it. Perhaps 80% of our members have never used the PDC. The PDC is a strong argument for joining ACM since it's a resource that is not available through institutions, so publicizing it and improving it are important for membership growth. Member services that were once valuable, like ACM's email service, have little appreciation in today's world. The recently mailed blue folder from John White and the Membership Services Board has helped raise awareness of the benefits of membership.

The clearest value to members is ACM's flagship publication. Many see joining ACM as subscribing to Communications of the ACM; if they don’t like CACM, they see no reason to join ACM. Hence, if CACM became a favorite magazine of people in our field, it would aid recruiting in all the groups noted here. For example, we could offer free trial subscriptions to conference attendees who are not ACM members to try to interest them into joining. Students could become accustomed to reading CACM, and want to continue reading it after graduation. If it had articles relevant to IT practitioners in China and India, it could help establish ACM in such countries.

I discussed ideas on revising CACM along the lines of Science magazine recently in this space [3]. Suggestions include harvesting the best of conference results, including one-page perspectives to introduce those research contributions, adding news and policy articles from around the world, inviting articles and interviews relevant to managers and practitioners à la Queue, and so on. I received about 50 email messages about these suggestions—a record for my column. Since many people who don’t like CACM don’t bother to read it, I was pleasantly surprised to see that an overwhelming majority of these email messages supported such a change. No more than 10% of the respondents liked the current CACM, and after exchanging a series of email messages with those who wrote back, I'd say 80% liked the new model.

Why is membership growth important? Growth is one measure of feedback on how good a job ACM is doing at serving the field. A second reason is it increases our clout when we speak out on an issue. A third reason is financial. In addition to more annual dues, increasing membership has the added bonus of increasing the value of the advertisements in our publications (although we may need to do a little more work to quantify the purchasing power of our members to attract advertisers as well).

Financial growth makes life simpler for organizations, making it possible to keep up with the costs of inflation and to expand offerings to their members. ACM hasn’t changed membership dues since 2000, despite a temporary drop in membership, primarily because of the growth of income from institutions joining our Digital Library. We are near market saturation as the DL is now in 50% of the libraries around the world, so financial growth will soon need to come from other places. Moreover, there are discussions going on in some circles that research papers funded by government grants should be freely available. Such a change would have a large negative impact on ACM’s current financing.

Hence, growing membership in these good times is wise for many reasons, and the opportunity for growth is out there. Carpe diem.

The Graying of ACM
Along with adding new members, another important challenge is persuading younger members to get involved in ACM activities. I’ve long believed that a fast-changing field like IT is a young person’s
field, as there is much greater value to the present than to the past.

Those of us who grew up in 1960s at least once believed that young people have important new ideas to contribute to older generations. Finally, besides a lack of new ideas, it’s unwise to wait until the Baby Boom generation retires to start recruiting new volunteers.

Hence, I was startled when I became president to see just how few young people were involved in ACM’s volunteer activities. While I’ve been beating this drum my whole presidency, things are changing at a geologic pace. The good news is that relatively new boards like the Professions Board and Membership Services Board do have age diversity. The Publications Board and USACM have moved slowly toward age diversity, but the Education Board has had the lowest turnover of members. SIGs vary widely; some like SIGGRAPH have publicized paths for recruiting of new members, but in many SIGs the average age of the volunteers is growing one year every year.

Although my observation was based on the volunteers, it relates to the ACM headquarters staff as well. Leadership stability is a positive sign of any organization, and ACM has enjoyed continuity in these good times. However, we also need to develop a transition plan as staff heads toward retirement and to groom successors so that ACM can continue as an organization that punches above its weight class. I have seen other volunteer-based organizations decline significantly once a key staff person leaves, and we can’t let that happen to ACM. Given the small size of ACM’s staff, it might be worthwhile to hire an outside consulting firm to assess our staff and make recommendations on how to prepare for such a transition.

My advice is that both the volunteer and professional staffs develop plans to attract new people and groom the next generation of leaders.

CONCLUSION
The good news is while organizations similar to ACM are facing more formidable challenges, ACM is in excellent shape to face its challenges. We can solve them if we address them seriously and promptly.

Let me close by paraphrasing the conclusion of the prior speech [1]:

So—in this my last good night to you as your President—I thank you for the many opportunities you have given me for ACM service. I trust that in that service you find some things worthy; as for the rest of it, I know you will find ways to improve performance in the future...

Now, on Friday midnight, I am to become a private citizen. I am proud to do so. I look forward to it.

Thank you, and good night.

REFERENCES
4. Purdue University. Linking Mathematics and Computer Science; www.cs.purdue.edu/external_relations/k-12_outreach/summer_camps/linking/linking06.shtml

David A. Patterson (pattrsn@eecs.berkeley.edu) is president of ACM and the Pardee Professor of Computer Science at the University of California at Berkeley.
In his ground-breaking book, subtitled *The Psychology of Optimal Experience* Mihaly Csikszentmihalyi of the University of Chicago described the concept of “Flow” [1]. Flow, as described by the professor, is a state of altered consciousness in which our ability to concentrate and perform is enormously enhanced. People who achieve this state also report a tremendous increase in their sense of achievement and satisfaction. There are many disciplines in which Flow may be exhibited. Athletes participating in professional sports sometimes display it, demonstrating astounding feats of physical ability when they do. One only had to watch Michael Jordan when he was on form (anytime between 1987 and 1998) to see the results of this kind of concentration. But the ability to attain this highly effective and rewarding condition is not limited to those engaged in sports. In fact, the good professor described many examples of very different people engaged in very different activities, from parents playing with their children to an ancient Chinese cook chopping up an ox, who attained this state.

**ANXIETY AND BOREDOM**

In our professional and personal lives, we sometimes navigate between two opposing conditions. When we tackle routine tasks that we can easily do, we find our minds wandering and we may become bored with the monotony of doing the same old thing. At the other end of the difficulty spectrum, when we tackle brand-new tasks where our competence is low, we may feel overwhelmed and out of our league. This is particularly true if there is significant pressure to perform well and to be “successful” and, at the same time, our lack of ability and experience is made public.

**THE COMPETENCY ZONE**

Between these two states lies the “Competency Zone” (see Figure 1). This is the area where our abilities and the demands of the task are in sync. The task is not so routine and mundane that it does not require effort and attention, and it is not so difficult that it induces anxiety. Too far in the boredom direction and we tend to become less effective, simply because the task is boring and our attention wanders. Too far into the anxiety region and we become less effective due to the drain on energy generated by our unease at our inability to perform.

Csikszentmihalyi pointed out
that we tend to arrange our work and social activities to remain within the Competency Zone. Of course, we do not stay there all the time, or we would never learn anything new. Tackling any new activity will tend to position us more toward the anxiety region. Too far into that region and people may shut down and move away from the psychic, or even physical, pain caused by the lack of ability. Assuming we stick to the task, and begin to learn, our competency increases (see Figure 2). We become more competent until we are able to successfully complete the task. This is often the point where people experience the greatest sense of reward as they flex and explore their new skill. Continuing to apply the skill makes the task easier and easier, until finally it becomes boring and learning stops.

For the task to continue to be rewarding, we must increase its difficulty. Doing so too much, of course, may put us back into the anxiety region. Music teachers are very familiar with this challenge; music students often “plateau” in their skills where they become increasingly capable at their current level of practice. The challenge for a good teacher is to increase the difficulty to help the student transition to the next level without making the task so difficult that the student retreats from the challenge.

**SOFTWARE FLOW CHART**

Software development, being primarily a learning activity, is subject to the laws of Flow. Tackling tasks that are way beyond our competencies tends to be anxiety-inducing and ineffective for both individuals and organizations. Our lack of expertise in the subject, and our apprehension over performing inadequately, combine to generate lower performance and more defects. The lower quality and the additional effort we have to throw at the task to compensate for our lack of experience also tend to increase our anxiety level, which further compounds the problem.

However, if we play it safe and operate only at a comfortable level of competency it means that we, or our company, are only doing things we know well. It also means we are probably doing what other companies can also do well. This generates serious competitive and price pressure and the business will usually go to the lowest bidder.

At its core, the purpose of a software project is always to do something new. “New” implies learning, and learning implies that we are not quite operating within the competency zone. This can be a problem for some individuals and some companies.

**THE COMFORT ZONE**

Most of us are most comfortable in the “Comfort Zone,” which occupies the lower portion of the Competency Zone. It is in this area that things are easiest for us—we can effortlessly show our skills and flawlessly produce our products. It is called the Comfort Zone because it is the low energy state; it is the area where we are...
most effective at the lowest output of effort. Any further into the boredom region and tedium will likely overwhelm our mindless effectiveness and automatic competence.

While the Comfort Zone is the low energy state and is the “easiest,” it is not the most rewarding. Csikszentmihalyi found that many of us spend large quantities of time in this zone, but we don’t find it rewarding at all. Watching television from the safety of the couch might qualify under this category and, perhaps surprisingly, we seem to spend a lot of our life doing such low-value things.

Operating within the Comfort Zone while we are building software allows us to easily display competence and easily build systems. But it may also be unrewarding and ineffective. The reason is that at the low end of the Competency Zone we don’t learn much. While the business of software does involve some translation of what we already know into an executable form, it also involves the discovery of what we don’t know. The application of our present knowledge does not usually involve much effort, especially when compared to the resolution of our ignorance. In fact, if software development were entirely the application of existing knowledge it would be a manufacturing activity and we would automate it.

Another name for the discovery of knowledge and the reduction of ignorance is “learning.” The Comfort Zone is not very rewarding in the business of software because in this zone we don’t learn much. When learning always seek to learn new ways of doing things, even ones that they can already do. Most of us, once competent, will seek to fine-tune that competence by learning ways to optimize our performance. But this has more to do with reducing effort than it has to do with increasing learning—there just isn’t the same pressure to innovate that exists when it is obvious that we cannot do our job at our current level of knowledge.

Also, our learning within the Competency Zone tends to be incremental, interpolating what we already know and marginally enhancing our existing skills. There are few among us who will voluntarily replace entire bodies of knowledge and skills with completely new capabilities and concepts unless we are absolutely forced to. This is what Thomas Kuhn called the “paradigm shift.” The history of science is dotted with instances where very, very smart people simply couldn’t replace the existing mental models that had served them so well, even when the scientific evidence clearly showed the limitations of that knowledge [2].

Finally, we could assert that new knowledge is the only true differentiator in software development, since new functionality...
The ability of a person or an organization to get into and stay at the Learning Edge and in the state of Flow is critical.

is the primary reason for running a project in the first place. Even if it is not some ground-breaking revolution in software thought, each system is (or should be) at least some new assembly of existing knowledge. Our focus and efficiency at acquiring this new knowledge is what we call “productivity” and it is highly sensitive to our position with respect to the Competency Zone. Therefore, the ability of a person or an organization to get into and stay at the Learning Edge and in the state of Flow is critical.

ACCELERATED LEARNING
Accelerated learning occurs on the outside edge of the Learning Edge (see Figure 3). This is where the most is learned most effectively in the least time.

The trick with managing accelerated learning is getting people as far as possible into the anxiety region without them shutting down. Some organizations are very good at doing this. The U.S. Marine Corps brings its recruits into the Parris Island and San Diego depots at 2:30 in the morning for a reason. They intentionally stress the recruits and push them way into the anxiety region because it is there that they learn the most. Specifically, it is there that they learn how to be U.S. Marines. The Outward Bound school uses a similar, albeit less extreme, approach when it takes at-risk teenagers out into the backwoods and wilderness and places them in a challenging and stressful environment where they can learn the self-respect and self-confidence they often lack. Both organizations freely acknowledge that learning, particularly accelerated learning, is not a “comfortable” activity.

Given the potential for the enormous increases in productivity that have been shown when people and groups attain the state of Flow, it would seem this should be a primary goal for any organization engaged in the business of software.

It is interesting that few companies seem to think this way and seem to be willing to step up to the challenge of truly generating and perpetuating a sense of Flow in their projects and with their people. It may be that, as companies, they simply prefer to flop down in the business Barcalounger or vegetate in the corporate Comfort Zone. If so, this is sad, because such companies deprive themselves of discovery and innovation, productivity and profits, and truly satisfied and fulfilled employees. And, according to Csikszentmihalyi, they also deprive themselves of meaning and purpose.

References

Phillip G. Armour (armour@corvusintl.com) is a senior consultant at Corvus International Inc., Deer Park, IL, and a research director in the Center for Software Development Innovation at Number Six Software, Vienna, VA.

© 2006 ACM 0001-0782/06/0600 $5.00
Incidents of plagiarism are escalating in computer science and engineering. While plagiarism cases were very rare during ACM’s first 40 years in the publishing business, several cases have been uncovered annually in recent years. Most of these cases have been extreme, blatant violations of ethical practice. ACM has dealt with papers published in conference proceedings in which very little change was made in the copyrighted plagiarized article except for a new list of authors. We’ve seen other cases in which two articles differed completely in their wording, but placed side-by-side we discovered that corresponding sentences said exactly the same thing throughout the two articles. In at least one case, a pattern of plagiarism was uncovered that ultimately led to someone losing their job.

Plagiarism—the verbatim copying, near-verbatim copying, or purposely paraphrasing portions of another author’s paper—is a clear violation of ACM’s longstanding Code of Ethics. However, because of the recent rise in the number of incidents, the ACM Publications Board felt the need to further codify ACM’s existing practice in dealing with plagiarism cases, as well as to raise awareness of the issues within the community. As a first step, we issued ACM’s first policy on plagiarism in November 2005.

Why are we seeing a rise in plagiarism? Certainly, pressure on young faculty to publish as many articles as possible continues unabated in academia. We now have a generation of youth who widely disregard intellectual property rights, and who may not have been schooled in the ethical issues surrounding plagiarism. Cultural differences also play a factor in our shrinking world. In some places, copying the work of a master in, say, art, is a sincere expression of respect and flattery. At the same time, tools that enable plagiarism are more readily available than ever before. Document processing systems have become easily interoperable. The cutting-and-pasting of text is so natural that it’s easy to simply forget whose text one is actually manipulating.

Of course, plagiarism is not a new phenomenon. Part of the reason we are discovering more cases may also be the same technology that makes plagiarism easier to do also makes it somewhat easier to detect. Nearly all technical articles are now literally at our fingertips and full-text searchable. So, it’s more common today to run into a plagiarized article that previously might have wallowed in the happy obscurity of a minor conference proceeding.

The ACM Plagiarism Policy defines plagiarism, its various levels of offense, specifies an investigation

---

1 See www.acm.org/constitution/code.html.

2 See www.acm.org/pubs/plagiarism%20policy.html.
process, sets penalties, as well as a process of appeal. Here, we outline a few major policy provisions.

Penalties for plagiarism depend upon the extent of the copying and the stage at which it is discovered. In the case of substantially plagiarized articles discovered in the ACM Digital Library after publication, ACM removes access to the full text of the article from the DL, replacing it with a notice that the article was plagiarized and pointing readers to the original article. A letter is written to the offender’s Department Chair or supervisor informing them of the infraction. The offending author must write a letter of apology to ACM and the original author. If there are subsequent offenses, the author can be barred from further publication with ACM. For copying smaller amounts of text without quotes and a citation, a note may be affixed to the article in the DL clarifying the original source of the text; the author would also be required to write a letter of apology. For plagiarism discovered before publication, say, during the review process, the article can be simply corrected or immediately rejected depending upon the extent of the infraction.

ACM is committed to performing careful and impartial investigations of allegations of plagiarism. For cases discovered after publication, the ACM Director of Publications makes contact with all parties to gather the facts of the case and makes a recommendation on a course of action to the ACM Publications Board. The Board, a group representing your peers, determines the outcome. The Board will strive to take action commensurate with any identified offense, and will seek to protect the interests of innocent parties. Appeals can be made to the ACM President.

ACM’s Plagiarism Policy also deals with the issue of self-plagiarism. This is a much more delicate issue about which there is currently much debate within the community. The ACM Publications Board has focused on one aspect of this phenomenon that it believes most professionals will agree is not acceptable practice—reuse without attribution. Repurposing one’s own words is not in itself a violation of ACM policy. In fact, the ACM Copyright Policy is rather liberal in granting authors the right to reuse any portion of their ACM copyrighted works in other works of their own. However, what is not ethical is the practice of reusing one’s own work in a way that portrays it as new when, in fact, it is not. Thus, reusing significant portions of a previously published work in another work of your own is acceptable, provided you cite the previous work and include a disclaimer stating a portion of the current work was previously published. Of course, self-plagiarism does not apply to reuse of items such as well-known definitions, equations, and so forth.

Respect for intellectual property is a cornerstone of the ACM Code of Ethics. All authors must take responsibility for the integrity of works published under their name. To this end, we urge authors to become familiar with the provisions of the ACM Plagiarism Policy. Educators have a particular responsibility to make their students understand the issues and be aware of the penalties and potential damage to their reputation. Please help us maintain ACM’s high standards.

Ronald F. Boisvert
Mary Jane Irwin
Co-Chairs, ACM Publications Board

Inquiries regarding ACM’s Plagiarism Policy should be sent to ACM’s Director of Publications Mark Mandelbaum (mandelbaum@hq.acm.org).

© 2006 ACM 0001-0782/06/0600
Hot Links

Top 10 Downloads from ACM’s Digital Library

The Top 10 Most Popular Papers from ACM’s Refereed Journals and Conference Proceedings Downloaded in March 2006

<table>
<thead>
<tr>
<th>Month Rank</th>
<th>Year Rank</th>
<th>Title, Author, Publication, Date</th>
</tr>
</thead>
</table>

*tied for number of downloads.

The 10 Most Popular Courses and Books at the ACM Professional Development Centre for March 2006

<table>
<thead>
<tr>
<th>Month Rank</th>
<th>Year Rank</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>OOAD: Introduction to Object-Oriented Concepts</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Java 2 Programming for SDK 1.4—Part 1: Language Fundamentals</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>C Programming—Part 1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>C# Programming for the Microsoft .NET Platform—Part 1: Introduction to C#</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>OOAD: Unified Modeling Language (UML) 2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Month Rank</th>
<th>Year Rank</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>Managing IT Projects: Project Initiation and Fundamentals</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Enterprise Connectivity with J2EE V1.4 Overview</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>UNIX Shell Programming—Part 1: Bourne Shell (Bash)</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Database Technologies—Part 1: Relational Database Fundamentals</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>C++ Programming—Part 1</td>
</tr>
</tbody>
</table>

Statistics for these tables were compiled Apr. 5, 2006.
Hot Links

Top 10 Most Popular Magazine and Computing Surveys Articles Downloaded in March 2006

<table>
<thead>
<tr>
<th>Month Rank</th>
<th>Year Rank</th>
<th>Title, Author, Publication, Date</th>
</tr>
</thead>
</table>

This list does not include downloads for the February and March issues as those statistics tend to reflect e-subscribers downloading current issues.

Coming Next Month in COMMUNICATIONS

July

SERVICES SCIENCE

The services sector has grown to dominate economic activity in most advanced economies over the last 50 years. Yet, experts in this field contend the scientific understanding of services is in a rather rudimentary state. This section will examine the progress made and challenges ahead for the services (as opposed to product-based) economy as well as offer valuable suggestions for future research. The articles will expound the need for a deeper understanding of services for enhancing economic productivity and, in turn, for the potential benefits to be gained.

Also in July:
- Trust Beyond Security
- Developing Quality Conceptual Models with UML
- The Phenomenon of Managerial IT Unconsciousness
- Pair Programming Improves Student Retention and Confidence
- 12i Trust in Videoconferencing
Copyrighting Standards

Should standards be eligible for copyright protection?

Controversies over intellectual property (IP) rights in standards have been common in the past decade. Thus far, IP and standards disputes have mainly been about patents. A new wave of controversies may be on the horizon, this time focused on copyrights claimed by standard-setting organizations (SSOs) as to standards produced by committees formed by or under their aegis, especially when governments mandate use of these standards. These disputes affect and constrain computing professionals when they design software applications for healthcare, e-commerce, or any other domain where standard classifications, identifiers, or code sets are required or desirable for compliance or interoperability.

This column will question whether standards, especially those mandated by the government, should be eligible for copyright protection. It will review several lawsuits that have challenged copyrights in standards with mixed success. Two 1997 appellate court decisions upheld copyright protection in standards. The reasoning of more recent appellate court rulings, as well as policy considerations, cast doubt on the soundness of recognizing copyrights in standards.

Challenge to AMA Code

The American Medical Association (AMA) and the American Dental Association (ADA) claim copyrights in codes of standard terminology for medical and dental procedures. The codes consist of three elements—a standard name for a procedure, a number assigned to represent it, and a brief description of the procedure—organized into logical groups so that users can locate the appropriate code quickly and easily. The associations periodically update these codes to take into account developments in the medical and dental fields. Use of the AMA code has been mandated by the U.S. government's Health Care Financing Administration (HCFA) when filing claims for Medicare and Medicaid reimbursement.

Practice Management Information Corp. (PMIC) wanted to publish the AMA code. The AMA threatened legal action, so PMIC asked a court to declare that the AMA code had become uncopyrightable after HCFA mandated its use, or alternatively, that the AMA misused its copyright by an exclusive license that HCFA forbade the agency to use “any other system of procedure.”
nomenclature...for reporting physicians’ services.” A trial judge issued a preliminary injunction against PMIC’s publication of the AMA code. The Ninth Circuit Court of Appeals affirmed in part and reversed in part.

PMIC’s invalidity argument rested mainly on Supreme Court case law about the uncopyrightability of judicial opinions and statutes. In *Banks v. Manchester*, the Supreme Court decided that judicial opinions could not be copyrighted. The Ninth Circuit distinguished *Banks* from *PMIC v. AMA*, saying that *Banks* involved government employees who didn’t need copyright incentives to write opinions. AMA, by contrast, claimed copyright incentives were important to it. *Banks* also rejected copyright claims in judicial opinions on due process grounds (that is, on a theory that people should have unfettered access to the law). There was, however, “no evidence that anyone wishing to use the [AMA] code has any difficulty getting access to it” and AMA has “no incentive to limit or forego publication” of the code. Moreover, PMIC pointed to cases that had held industry standards to be ineligible for copyright protection. The AMA itself had characterized its code as a “system of procedure nomenclature.”

On appeal, PMIC belatedly made an argument that the AMA code was uncopyrightable as an industry standard and systematic organization of information. *Section 102(b)* of U.S. copyright law states: “In no case does copyright protection...extend to any idea, procedure, process, system, method of operation, principle, concept or discovery, regardless of the form in which it is...embodied in such work.”

PMIC pointed to cases that had held industry standards to be ineligible for copyright protection. The AMA itself had characterized its code as a “system of procedure nomenclature.”

Although PMIC’s belated arguments have merit (explained later in this column), the appellate court rejected them without explanation in a footnote. (Courts tend to perceive late-made arguments as acts of desperation akin to a “Hail Mary” pass in football.)

**Challenge to ADA’s Code**

Shortly after *PMIC*, the Seventh Circuit Court of Appeals reviewed a challenge to the copyright in ADA’s Code of Dental Procedures and Nomenclatures. Delta Dental published a book containing most of the ADA nomenclature and associated numbers, although not the procedure descriptions. ADA asked the court for an injunction to stop Delta from publishing the code and money damages for past infringements.

The trial judge ruled against the copyrightability of the ADA code, although recognizing that compilations generally qualify for copyright protection if there is creativity in the selection and/or arrangement of information. ADA’s compilation was comprehensive, however, precluding creativity in selection. Although the code’s arrangement was creative, it was systematic and highly useful, thereby unprotected under section 102(b). The code was, moreover, the work product of a committee, not an expression of the judgment of an author, and Delta had participated in the drafting of the ADA standard, which further supported its right to use the ADA code.

The Court of Appeals disagreed. In its view, ADA’s classification of dental procedures was creative enough to qualify for copyright protection. “Creativity marks the expression even after the fundamental scheme has been devised.” Because there are many different ways to organize types of dental procedures—“by complexity, or by the tools necessary to perform them, or by the anesthesia employed, or in any of a dozen different ways”—the way chosen by ADA was a creative expression not dictated by functional considerations.

The appellate court thought
ADA’s “numbering system” (its term) was creative because ADA assigned numbers to procedures that were five digits long (when it could have made them four or six digits long). The first number was always a zero, in order to leave room for expansion of the code as more procedures were developed or discovered. The second and third numbers represented a particular grouping of procedures, and the remaining two digits identified the specific procedure associated with that number. To Delta’s argument that section 102(b) rendered this “system” unprotectable, the court responded that the code “is a taxonomy that can be put to many uses. These uses may or may not be or include systems,” but the code itself was not a system.

The appellate court was so taken with the creativity of the ADA code that it decided the name of each procedure and the number assigned to it were original works of authorship entitled to copyright protection. The court distinguished between “discovered facts” and “original facts.” The former were unprotectable under the Supreme Court’s decision in *Feist Publications v. Rural Telephone Service* wherein the Court ruled that facts were unprotectable “discoveries” under section 102(b). The Seventh Circuit decided that original facts should be treated differently.

**Challenge to the Southco Numbering System**

Southco, a manufacturer of products such as latches, handles, and rivets, sued its competitor, Kanebridge, for copyright infringement because the latter’s catalog reproduced product names and numbers from Southco’s copyrighted catalog. Southco claimed these names and numbers were copyrightable as the product of skilled judgment, and since there were many different ways to organize such a taxonomy, there was no “merger” of idea and expression to disqualify its work from protection.

Writing for the Third Circuit Court of Appeals, Judge Samuel Alito (now a Justice of the U.S. Supreme Court) held that Southco’s compilation was an unprotectable systematic compilation of information under section 102(b). Insofar as Southco claimed copyright in the individual product names and numbers, the court found them too small in grain size to be protectable by copyright, citing the longstanding practice of excluding short phrases and titles from copyright protection.

In developing this system, Southco “had to identify the relevant characteristics of the products in the class (that is, characteristics that would interest prospective purchasers); it had to assign one or more digits to express each characteristic; and it had to assign a number or other symbol to represent each of the relevant values of each characteristic.” This did require judgment, but “[o]nce these decisions were made, the system was in place and all of the products in the class could be numbered without the slightest bit of creativity.”

The unprotectability of systems was first announced by the Supreme Court in *Baker v. Selden*. Selden’s widow charged Baker with copyright infringement for copying sample ledger sheets from Selden’s book. The Supreme Court ruled that copyright protected Selden’s explanation of the bookkeeping system, but not the system itself or ledger sheets implementing it.

Judge Becker concurred in *Southco*, pointing out that industry standards are often excluded from copyright protection, such as in *Mitel v. Iqtel* decided by the Tenth Circuit Court of Appeals. Mitel manufactured call controllers, “computer hardware that enhances the utility of a telephone system by automating the selection of a particular long distance carrier and activating optional features such as speed dialing.” It

Industry standards serve an important function in society by allowing everyone in that industry or field to use the standard for effective communication.

and the remaining two digits identified the specific procedure associated with that number. To Delta’s argument that section 102(b) rendered this “system” unprotectable, the court responded that the code “is a taxonomy that can be put to many uses. These uses may or may not be or include systems,” but the code itself was not a system.

The appellate court was so taken with the creativity of the ADA code that it decided the name of each procedure and the number assigned to it were original works of authorship entitled to copyright protection. The court distinguished between “discovered facts” and “original facts.” The former were unprotectable under the Supreme Court’s decision in *Feist Publications v. Rural Telephone Service* wherein the Court ruled that facts were unprotectable “discoveries” under section 102(b). The Seventh Circuit decided that original facts should be treated differently.

**Challenge to the Southco Numbering System**

Southco, a manufacturer of products such as latches, handles, and rivets, sued its competitor, Kanebridge, for copyright infringement because the latter’s catalog reproduced product names and numbers from Southco’s copyrighted catalog. Southco claimed these names and numbers were copyrightable as the product of skilled judgment, and since there were many different ways to organize such a taxonomy, there was no “merger” of idea and expression to disqualify its work from protection.

Writing for the Third Circuit Court of Appeals, Judge Samuel Alito (now a Justice of the U.S. Supreme Court) held that Southco’s compilation was an unprotectable systematic compilation of information under section 102(b). Insofar as Southco claimed copyright in the individual product names and numbers, the court found them too small in grain size to be protectable by copyright, citing the longstanding practice of excluding short phrases and titles from copyright protection.

In developing this system, Southco “had to identify the relevant characteristics of the products in the class (that is, characteristics that would interest prospective purchasers); it had to assign one or more digits to express each characteristic; and it had to assign a number or other symbol to represent each of the relevant values of each characteristic.” This did require judgment, but “[o]nce these decisions were made, the system was in place and all of the products in the class could be numbered without the slightest bit of creativity.”

The unprotectability of systems was first announced by the Supreme Court in *Baker v. Selden*. Selden’s widow charged Baker with copyright infringement for copying sample ledger sheets from Selden’s book. The Supreme Court ruled that copyright protected Selden’s explanation of the bookkeeping system, but not the system itself or ledger sheets implementing it.

Judge Becker concurred in *Southco*, pointing out that industry standards are often excluded from copyright protection, such as in *Mitel v. Iqtel* decided by the Tenth Circuit Court of Appeals. Mitel manufactured call controllers, “computer hardware that enhances the utility of a telephone system by automating the selection of a particular long distance carrier and activating optional features such as speed dialing.” It

Industry standards serve an important function in society by allowing everyone in that industry or field to use the standard for effective communication.
developed and published a set of four-digit command codes for these call controllers.

Mitel claimed copyright in the command codes as creative work. Iqtel copied the Mitel command codes after concluding that “technicians who install call controllers would be unwilling to learn Iqtel’s new set of instructions in addition to the Mitel command codes and the technicians’ employers would be unwilling to bear the cost of additional training.” The Tenth Circuit concluded that the Mitel codes were unprotectable by copyright law because they had become industry standards.

Analysis
In all of the four cases discussed here, the copyright claim rested on the creativity in deciding to pair a number with the name of a phenomenon in accordance with an overall system for efficiently organizing information for a specific purpose. Systems, by their nature, consist of interdependent, interrelated parts that are integrated into a whole scheme. Taxonomies, by their nature, are systematic classifications of information that group subcomponents into logical categories based on similarities of this cluster of phenomena as compared with that. Section 102(b) excludes systems and their component parts from copyright protection.

AMA and ADA codes were designed to facilitate efficient record-keeping, billing, and other information-processing functions (for example, computerized processing of insurance claims). The AMA’s stated purpose in developing the code at issue in PMIC was “to provide a uniform language that accurately describes medical, surgical, and diagnostic services, and thereby serve as an effective means of reliable nationwide communication among physicians, and other healthcare providers, patients, and third parties.” The ADA decision states that “the standardization of language promotes interchange among professionals.” The Southco and Mitel codes were also designed to facilitate effective communication.

Industry standards serve an important function in society by allowing everyone in that industry or field to use the standard for effective communication. The interoperability case law, of which Mitel is one instance, has consistently ruled that the design of computer program interfaces may be the product of human skill and judgment, and thus might seem to qualify for copyright protection. However, once an interface has been developed, the parameters it establishes for the effective communication of information from one program to another constrain the design choices of subsequent programmers. The interface thus becomes an unprotectable functional design.

User investments in the standard facilitate communication. The First Circuit Court of Appeals dismissed Lotus’s “look and feel” lawsuit against Borland in part because users had made significant investments in Lotus’s macro command language, making it a de facto industry standard:

“Users employ the Lotus menu command hierarchy in writing macros. Under the district court’s holding, if the user wrote a macro to shorten the time needed to perform a certain operation in Lotus 1-2-3, the user would be unable to use that macro to shorten the time needed to perform the same operation in another program. Rather, the user would have to rewrite his or her macro using that other program’s menu command hierarchy. This is despite the fact that the macro is clearly the user’s own work product. We think that forcing the user to cause the computer to perform the same operation in a different way ignores Congress’s direction in section 102(b) that ‘methods of operation’...”
are not copyrightable. That pro-
gress can offer users the ability to
write macros in many different ways
does not change the fact that, once
written, the macro allows the user to
perform an operation automatically.
Judge Boudin concurred:

Requests for the protection of
counter menus present the concern
with fencing off access to the com-
mons in an acute form. A new
menu may be a creative work, but
over time its importance may come
to reside more in the investment
that has been made by users in
learning the menu and in building
their own mini-programs—
macros—in reliance upon the
menu. Better typewriter keyboard
layouts may exist, but the familiar
QWERTY keyboard dominates the
market because that is what every-
one has learned to use.

User investments in a standard
constrain the design choices of sub-
sequent users in much the same
way that Iqtel felt constrained by
Mitel’s command codes.

Other Considerations
Reinforcing the conclusion that
copyright law should not protect
standards such as the AMA and
ADA codes are several other con-
siderations. First, SSOs them-
selves generally do not develop
the standards in which they claim
copyright. Rather, they generally
rely upon volunteer service of
professionals in the field to
develop standards and require
these volunteers to assign copy-
right interests to the SSOs. The
community development of a
standard is a reason to treat the
standard itself as a shared
resource.

Second, SSOs generally have
ample incentives to develop stan-
dards for use by professionals in
their field. While theAMA and
ADA certainly charge substantial
sums for the standards they pro-
mulgate and use the funds to sup-
port other activities, it is simply
not credible to claim they would
stop developing standard nomen-
clature without copyright. The
fields they serve need the standards
for effective communication.

Third, the Internet and the
Web now make it very inexpen-
sive and easy to disseminate stan-
dards. Some users of a successful
standard would be willing to
invest in putting the standards
online for all to use.

Fourth, once a standard has
achieved success by widespread
adoption, this success provides an
opportunity for the SSO to charge
monopoly rents for use or access
to the code. In 2003, for example,
the W3C, among others, objected
to anISO policy requiring soft-
ware developers to pay fees for
incorporatingISO country codes
into their products.

Fifth, SSOs have incentives to
invest in persuading governments
to mandate use of their standards.
Another legal decision illustrates
this temptation. The Southern
Building Code Congress (SBCC)
drafted a model building code
and promoted its adoption by
local governments. Under the deal
SBCC offered, local governments
got royalty-free rights to use the
code and a small number of
copies of the code to enable pub-
lic access. SBCC charged anyone
else who wanted a copy of the
code or access to it a substantial
fee. Peter Veeck purchased a copy
of the SBCC code and posted the
contents on the Web. SBCC sued
for copyright infringement.

The Fifth Circuit Court of
Appeals decided that once certain
small towns in Texas had adopted
the SBCC code as law, there was
no effective way to express that
law except through the literal
words of the SBCC code. The
idea of the code and its expression
had, the court concluded,
“merged” upon its adoption as
law, and no copyright protection
remained in the adopted code
because of the public interest in
access to the law. The Veeck de-
cision calls into question the
PMIC ruling about the validity of the
AMA code after the government
mandated its use.

The credibility of SSOs
depends not only on their being
able to produce sound standards,
but also on producing standards
in which the SSOs do not have
such a strong financial interest
that they succumb to the tempta-
tion to abuse the standards
process by making the standards
into a cash cow that must be pur-
chased by anyone affected by the
standard.

Pamela Samuelson (pam@sims.
berkeley.edu) is the Richard M. Sherman
Professor of Law and Information
Management at the University of California
at Berkeley.

© 2006 ACM 0001-0782/06/0600 $5.00

COMMUNICATIONS OF THE ACM June 2006/Vol. 49, No. 6 31
By GREGORY CONTI, Guest Editor

HACKING AND INNOVATION

Why computer science should pay serious attention to the hacking community and its passion for pushing the limits of technology and its role as a counterbalance to its misuse.

There is a passionate and independent-minded global community of highly skilled technical experts that frequently functions outside the mainstream of computer product development and conventional technology research. Sometimes called the hacker community, these experts are responsible for innovation that pushes the limits of technology, sometimes in unintended or uncomfortable ways, as well as for prescient warnings about the threats of both technology and the government’s technology-related policy and regulations. Computer scientists have much to learn about innovation and nontraditional problem solving by listening to and working with them.
Whether inside or outside the mainstream, hackers are less constrained by conventional thinking, and their work often complements (and sometimes conflicts with) its counterpart in traditional industrial organizations, academic departments, and government agencies. In many cases their research is ahead of what’s being done in these organizations but with results that are unlikely to ever appear in academic journals and conferences due to differing ways of disseminating information.

Their passion is especially noteworthy. From Nguyen Phuoc Huy, a medical doctor from the Mekong Delta region of Vietnam who built his own endoscope out of a low-cost Web camera [5], to the Shmoo collective’s Wired Equivalent Privacy (WEP)-cracking robot (see Figure 1), to Ward Christensen’s and Randy Suess’s construction of the first electronic bulletin board system (see Figure 2) in 1978, the contributions are diverse and significant.

Some computer scientists consider it a high honor to be described as a hacker; to others it’s a base insult. For many computer scientists, as well as the general public, the word hacker has a connotation reflecting the sensationalized stereotype often seen in mainstream media. Objective accounts are rare [1–4]. Perhaps due to this perception, two disjoint, typically mistrustful, technology-focused communities—professional computing and hacking—have emerged. Despite having only infrequent interactions, they are often at odds, ultimately frustrating one another’s efforts. As the world increasingly depends on technology, we all must move beyond the semantics and etymology of the word hacker [6] to address the true risks and needs of humanity, either through our own research or when we serve as technical advisors to legislative and technology policy decision makers. Ultimately, each of our scientific contributions should be weighed on the merit of the related ideas, not on academic credentials, institutional affiliation, or age of the source.

Our goal here is to listen to the authentic and expert voice of hacking—a task more difficult than it might appear. The loose-knit hacker community has no formal leaders. Hacking is diverse and by its nature resists formal definition. We have sought out a sample from among the best and the brightest. To this end, these articles were written by individuals who routinely challenge convention, whether from inside the professional computing community or from within the computer underground. Many have never published in the scientific literature before. This fact does not, however, diminish the value of their words but should instead make us listen even more attentively.

The hacker community possesses an extensive body of work, but instead of lying in repositories (such as ACM and IEEE digital libraries), results are presented at such conferences as Black Hat, CanSecWest, the Chaos Computer Congress, DEFCON, HOPE, Interz0ne, ShmooCon, and Toorcon or published in such magazines as 2600, BinRev, and Phrack (see the sidebar “Hacking Sources”). The fact that the ideas exist in circles less traveled by the academic community does not relieve us of the responsibility of exploring them to research related work. You may be surprised to find that your “new” idea was promulgated years ago at a hacker conference or in a
hacker publication. Almost without exception, these articles, presentations, and other artifacts are freely available online.

I have been profoundly influenced by Orson Scott Card’s portrayal of youthful prodigies in his 1985 science fiction novel *Ender’s Game* in which Ender’s siblings, Peter and Valentine, were prodigies too young, despite their great intelligence, to be accepted by the great thinkers and leaders of their day. Despite this impediment, they nevertheless rose to prominence on the merit of their ideas alone by using anonymous online personas to promulgate their thoughts. Similarly, when seeking appropriate and authentic voices for this section, I sought out deep thinkers and gifted technical experts, who, through the power of their words alone, could describe serious personal experience and insights so compelling that virtually any reader from either community would acknowledge the value of their message, even if that reader does not fully agree. My aim is to close the gap between the communities of computer professionals and computer hacking.

The section has two main components: personal viewpoints and in-depth technical articles. I challenged the viewpoint authors to discuss some of the most significant trends and threats they saw emerging in the worldwide Internet-based environment. Tom Cross, creator of the MemeStreams semantic blogging system, which helps people share information about what’s worth reading on the Web, starts us off by exploring the troubling decline in the right of individual experimenters to freely investigate technology. Steve Bono et al. then address the use of the courts, legislation, and government regulation to prevent discourse about vulnerabilities in software and hardware products.

I challenged the technical article writers to explore three facets of hacking—software, hardware, and networks—and explain their personal methods and thought processes when approaching problems. Joe Grand peels back the covers on hardware to reveal approaches to modifying technology in ways unintended by its designers. Bruce Potter, founder of the Shmoo Group of security professionals, well-known for its annual security conference Shmoocon, describes how wireless hotspots break down the traditional security trust model, leaving the typical end user, as well as many power users and even many global corporations, underprotected from potential malicious attack.

Felix “FX” Lindner examines the similarity of the software engineering and security disciplines, finding that, despite that similarity, different approaches and terminology result in less-secure systems. Finally, Dan Kaminsky explores key aspects of request for comment-compliant Domain Name System (DNS) protocol hacking, by probing DNS servers worldwide in order to notify DNS operators of their vulnerabilities. He also shares his work mapping the global spread of the recent Sony rootkit that put a visible face on the magnitude and location of those infected, helping raise a public outcry against Sony’s intrusion.

Hacking is more about innovation and less about computer security. Hacking and computer science are so intertwined it is a travesty the two communities do not share greater respect for and cooperation with one another. To promote the sharing of common interests the hacking story must be told accurately in all its sometimes contradictory aspects. *Communications of the ACM*
represents the public record for the professional computer science community. This section is our attempt to add to this record a glimpse of the heart and soul of the hacker ethic in its members’ own words.

There is a narrow path for success that will help foster collaboration between the two sides of the divide. Antagonists and critics from both sides are waiting to pounce, but the potential for success makes the risk worthwhile. To move beyond common stereotypes, we may work together to advance the interests of human knowledge. The main message we hope to impart is that you should feel free to challenge convention, explore the work done by these researchers, and seek opportunities to collaborate with the hacking community. I ask that you suspend your preconceived notions, ponder the arguments and expertise, and, perhaps, adjust your personal perspective. I daresay you will be more warmly received in their world than they would in ours. Perhaps we can change that.

**References**

5. Le, H. Vietnam medic makes DIY endoscope. BBC News Online (Aug. 22 2005); news.bbc.co.uk/1/hi/technology/4145984.stm.

**Gregory Conti** (conti@acm.org) is an Academy Professor of Computer Science at the United States Military Academy, West Point, NY, and currently at the Georgia Institute of Technology, Atlanta, on a Department of Defense Fellowship.

© 2006 ACM 0001-0782/06/0600 $5.00
ACADEMIC FREEDOM AND THE HACKER ETHIC

There is a global culture of people who call themselves computer hackers that is driven by a fundamental belief that information should be free and that the pursuit of knowledge is an essential human right. Most hackers seek to do creative things with technology, but the community is often beset by controversy because it centers on forbidden knowledge; in particular, hackers like to think about how computer security fails. The general public often has difficulty drawing a line between hackers who study computer security as a technical interest and criminals who break into computers and deliberately cause damage. Some observers in the media and the academic community have argued that the sort of information hackers discuss at conferences and in journals should never be shared publicly, saying that knowledge is itself dangerous, irrespective of the motives of the people discussing it.

Hackers advocate the free pursuit and sharing of knowledge without restriction, even as they acknowledge that applying it is something else.
Controversy involving the hacker community is analogous to new fears being raised about science in general. In the past decade academic and policy circles have begun discussing the idea that as technology advances and laboratory equipment becomes less costly and easier to access, rogue scientists may be able to use their knowledge to harm people, either through accidents or by intentionally crafting dangerous weapons. Sun Microsystems co-founder Bill Joy is often viewed as leading this charge, following his famous essay “Why the Future Doesn’t Need Us” [3] in which he argued that advancements in biology, nanotechnology, and robotics will soon give rise to technological capabilities beyond our control, threatening the survival of all humanity. He proposed that we “relinquish” the pursuit of entire classes of scientific knowledge in order to avoid a catastrophe. Over time, it has become increasingly evident that a broad policy debate about academic freedom is taking form in which the perspective of the hacker community may represent a critical counterweight to overzealous calls for control of the pursuit of knowledge.

You can see the clouds of this debate gathering in recent controversies in biotechnology. Governments worldwide have passed regulations intended to control advanced biotech products and research deemed risky or inhumane. For example, the European Union banned new genetically modified crops for five years ending in 2003, and six African governments have refused to accept genetically modified food aid. In August 2005, the U.N. issued a declaration banning human cloning. Debate has raged in North America about the moral implications of stem cell research. Biological advances scare some of us because we are unsure of the morality of tampering with the fundamentals of life and because we are worried about the unforeseen consequences of releasing organisms into the natural environment after they’ve been modified by humans.

It seems clear that advances in other fields (such as nanotechnology and artificial intelligence) will eventually bring us self-reproducing machines that involve many of the same problems. Many of the arguments being made today in the context of bioethics are broad enough that they can also encompass these future developments, threatening to produce regulations that deeply affect our personal and professional academic freedom.

Francis Fukuyama, an influential U.S. political economist and a member of the U.S. President’s Council on Bioethics, published an essay in 2002 [1] laying out a set of philosophical arguments for government control of basic research, even when pursued outside the federal grant system. He wrote that “Science itself is just a tool for achieving human ends; the political community must decide which ends to pursue.” This idea strikes at the heart of academic freedom. Our universities have certain institutional structures (such as the tenure system) specifically designed to shield basic research from the sort of political influence Fukuyama advocated. These structures exist to enable human knowledge to expand toward every opportunity for growth and for the discovery of truth without being hampered by fear and special interests. Fukuyama’s perspective represents a fundamental challenge to our society’s overall approach to the advancement of science.

How should the scientific community respond? Should it embrace regulations that prohibit publication of certain kinds of technical information? Should it advocate that national governments require approval for private research projects? Should it pass laws prohibiting scientists from examining certain subjects or developing certain technologies? The hacker community would say no. From grappling with these questions over the years, hackers have developed a nuanced and sophisticated understanding of the line between ideas and actions, as well as the dangers posed by allowing governments to control what people are allowed to think about, particularly with regard to scientific or technical inquiry. Their perspectives on these issues, and the lessons they’ve learned responding to critics and working to resist overzealous legislation are a necessary ingredient when considering these questions.

Hackers believe that ethical questions generally apply to the application of knowledge rather than to the pursuit of knowledge. While ethical questions arise in scientific study, they usually relate to ensuring that people are not harmed by experiments rather than whether the knowledge being sought is harmful in and of itself. Knowing how to do something that might be harmful is not the same as causing harm. Once you have knowledge you still must decide what to do with it. For example, if you know how to pick a lock, you can apply that knowledge as a locksmitl, troubleshooting lock problems and designing better locks, or you can apply that knowledge as a thief. You can witness this distinction in action at hacker conferences like Defcon where attendees draw a line between “white hats” trying to improve the state of computer security and “black hats” trying to upend it. Both groups are interested in the same sort of knowledge. The moral distinction comes from how they apply it.
Hackers reject the notion that ignorance makes you safer. In the 1980s, as computer networks grew and computer security problems grew with them, vendors, government agencies, and university labs kept software vulnerabilities secret from the general public, even as they quietly shared information with one another. Computer criminals developed their own independent techniques they shared within their own networks. Left in between were large numbers of people responsible for operational Internet systems who were not part of either community and were largely in the dark about how to protect themselves.

Most people who openly discussed computer vulnerabilities at the time belonged to the hacker community. Partly out of frustration with the status quo, they began “full disclosure” email lists where vulnerability details were discussed in plain view of the general public, usually after software patches had been released. Today, these lists represent a cornerstone of the professional computer security world. This open dialogue has been positive for computer security; the ability to understand and share research findings and learn from the mistakes of others makes security practitioners smarter. Having a smart worldwide community of security practitioners makes end users safer and is well worth the advantage that disclosure might offer unsophisticated attackers unable to develop their own techniques.

Hackers also believe that valuable new ideas do not always come from established institutions. When governments get involved in regulating dangerous knowledge they often overreact by erecting barriers to “amateurs”; for example, the U.S. Digital Millennium Copyright Act of 1998 prohibits research into certain classes of computer security vulnerabilities unless “the person is engaged in a legitimate course of study, is employed, or is appropriately trained or experienced in the field of encryption technology.” These restrictions seem to have been aimed specifically at hackers, as if security research that does not occur in traditional institutions is not worthy of legal protection. This restriction indeed affects the entire computer science community as its members pursue research both personally and professionally.

It is widely understood that critical developments in computer science have come from garages and hobby clubs (most notably the Homebrew Computer Club, an early crucible of the personal computer revolution). Consider the plethora of free and open source software projects that glue the Internet together, many written by amateurs and students. This font of innovation happens in the field of computer security as well. Ask security professionals where they would be without free tools (such as the Nmap port scanner, the NetCat networking utility, and the OllyDbg debugger). All of us would certainly lose a great deal if we deliberately limited science to only a few select laboratories and research institutions.

Most important, hackers believe the pursuit of knowledge is an inalienable right, tied directly to freedom of speech. Individual rights are not important simply because of some idealistic notion of freedom. If government regulators are given the legal authority to decide what questions may be asked and answered, they will have the power to prevent us from discovering the truths that are critical to our interests.

Paul Graham, the creator of bayesian spam filtering, illustrated the connection between civil liberties and technology in a 2004 essay [2] in which he wrote “A society in which people can do and say what they want will also tend to be one in which the most effi-
cient solutions win, rather than those sponsored by the most influential people. Authoritarian countries become corrupt; corrupt countries become poor; and poor countries are weak... This is why hackers worry. The government spying on people doesn’t literally make programmers write worse code; it just leads eventually to a world in which bad ideas will win.”

In 2005, Bill Joy and Raymond Kurzweil wrote an essay [4] protesting the academic publication of the genome for the 1918 flu, which killed tens of millions worldwide. Fearing that terrorists might use this information to craft biological weapons, they called the publication “extremely foolish” and suggested that the precise genome be shared only “with scientists with suitable security assurances.” This sort of argument is too familiar. While the balance of interests and risks in the full disclosure of pathogen genomes is different from those inherent in software vulnerabilities, we cannot give in to fear and presume that the right answer is always as simple as sweeping dangerous information under the rug.

The history of the hacker community is filled with people who have faced significant personal consequences for revealing truths powerful interests sought to suppress. A future in which scientists of all stripes face such pressures and the bulk of human knowledge is kept under lock and key is not the sort of future I want to live in. Government policy makers should manage these risks by controlling access to certain raw materials and regulating practical applications rather than censoring ideas and information. Academic freedom should be restricted only as an absolute last resort, not as the fundamental basis of our national strategies for security and technological development in the 21st century.

References
1. Fukuyama, F. How to regulate science. The Public Interest 146 (Winter 2002); www.sais-jhu.edu/faculty/fukuyama/articles/pi.pdf.

Tom Cross (tom@memestreams.net) lives in the U.S. where he works in the computer security industry as a vulnerability researcher. He wears a white hat.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SECURITY THROUGH LEGALITY

The law alone won’t prevent an unauthorized visit or even a deliberate attack. Security depends on being able to think like an attacker.

That would work, but no attacker would try it,” said the chief designer of a wireless security system we had been contracted to evaluate. After several questions from across the table, we had determined why this particular security product would fail even against an adversary who was only modestly innovative. The security designers, along with a corporate manager and an electrical engineer, had made the common mistake of misunderstanding the potential adversary’s state of mind and unnecessarily bounded the scope of attacks that might be employed. We reminded them that the threat model for any system shouldn’t be based on what an attacker would probably do but on every possible thing an attacker could do.
MISUNDERSTANDING THE ADVERSARY AND RELYING ON THE LAW AS A CRUTCH HAS YIELDED INEFFECTIVE AND DAMAGING RESULTS.

Misunderstanding an adversary’s mind-set and possible intentions has led numerous commercial security products to rely on the threat of legal action as a way to provide security, rather than on tried-and-true security measures. “Security through legality” has been a crutch, as well as as scapegoat, for justifying lax security provisions. As typically found with its better-known counterpart “security through obscurity,” adopting this hopelessly flawed methodology can lead to systemwide compromise and cost exorbitant amounts in damages. Relying on legislation for protection has proved ineffective in the face of widespread abuse, illogical when incorporating a criminal’s apathy toward the law, and to be a way to limit research by law-abiding professors and citizens who seek to improve security rather than subvert it. This is not to say that criminal prosecution and the law don’t play a significant role in providing a secure information landscape. But the unlawfulness of any action does not explicitly prevent it from occurring, suggesting only that it be avoided.

Treat ing the secrecy of a system’s design as a security measure is commonly referred to by experts as security through obscurity. This faulty methodology is widely known among security experts for yielding insecure systems, as simple disclosure of the design can lead to catastrophic security failure. Still, many modern security devices and applications rely solely on a criminal’s inability to figure out how a system works or obtain design documents rather than tried-and-true security methods.

In the same way security through obscurity has inspired insecure systems based on the fallacious assumption that an adversary probably wouldn’t do something, we have noticed an unfortunate trend in commercial security system design of making faulty security assumptions based on what an adversary is legally allowed to do. Security through legality is the misconception that an adversary will not pursue some avenue of attack just because doing so is unlawful.

In lieu of developing preventive measures, designers occasionally argue that criminal prosecution is a sufficient deterrent to system compromise. On the contrary, criminals do not generally let laws stand in the way of breaking laws. A burglar intent on acquiring thousands of dollars in jewelry from a potential mark’s master bedroom is well aware of the penalty if caught and is hardly concerned with the vandalism charge resulting from a broken window.

The U.S. Digital Millennium Copyright Act (DMCA) of 1998, as well as other copyright laws around the world, provide a more than adequate example of how criminalizing certain activities is not sufficient for preventing crimes from being perpetrated. During the past decade, the growing popularity of file-sharing networks has made piracy a major concern of the movie, music, and software industries. Though awareness of piracy as a crime and the fear of prosecution has attracted the public’s attention, no significant evidence has emerged that the unlawfulness of piracy has actually slowed its spread. To be fair, no digital rights management (DRM) system has yet prevented the widespread piracy of movies or music, but preventive measures, rather than deterrents, must always take precedence when designing a security system.

The movie industry once relied on the DVD Content Scrambling System (CSS) to prevent piracy.
Between 1996 and 1999, DVD copy protection depended on the inability of pirates to copy DVD content in “raw,” or unencrypted, form. In October 1999, source code for circumventing this technology, known as deCSS, was released by a group of Norwegian hackers, giving DVD pirates an elegant workaround for defeating CSS. The movie industry asserted that the act of releasing, using, or sharing the deCSS source code was a violation of the DMCA, but doing so did little to prevent consumption of deCSS by the piracy community where recipients cared little that they were in violation of the DMCA, as they intended to break the law anyway. Since then the motion picture industry has funded development of stronger DRM products to directly prevent the unlawful copying of DVDs; examples include the Advanced Access Content System, which claims to allow authorized copies of digital media to be produced while preventing unauthorized copies, and Self-Protecting Digital Content, which offers renewable security as an alternative to revoking players when security vulnerabilities are discovered. This dramatic shift illustrates a budding awareness that concrete piracy-prevention methods are necessary where reliance on the law alone as a deterrent has proved itself ineffective.

Though the DMCA is the best-known legal pillar, circumventing anti-piracy measures is not the only area of law being used as a scapegoat. In 2005, students at The Johns Hopkins University showed that some vehicle immobilizers and the ExxonMobile Speedpass payment system utilizing cryptographically enabled RFID chips are susceptible to cloning attacks that allow thieves to make working copies of each device. A thief could, for example, duplicate car keys with little effort or make fraudulent purchases billed directly to a victim’s credit card. Both systems relied on a secret encryption algorithm developed by Texas Instruments that, once discovered, allows easy and inexpensive replication of these RFID devices.

Aside from committing the security of the entire system to the secrecy of the encryption algorithm (security through obscurity), upon its disclosure numerous arguments have been made suggesting it is still illegal to create or obtain key-cloning devices that use the Texas Instruments’ algorithm, and copyright and patent legislation make it cumbersome to develop commercially available equipment to do so. True as it may be that there are legal and cost hurdles to overcome before commercial cloning kits are available, these hurdles are orthogonal to the minimal legal and economic barriers criminals face in producing such equipment on their own, as disregard for the law is already the norm.

In some cases, rules and regulations have the adverse effect of limiting public investigation and research into the security designs of some products. Massively deployed RFID-based toll-collection systems (such as EZ-Pass in North America) have enormous potential for security and privacy violations associated with tracking and correlating groups of people. These systems employ minimal (if any) security features; empirically exploring the depth of security and privacy concerns associated with them would benefit the industry, as well as the public. However, U.S. Federal Communications Commission regulations prevent the use of uncertified equipment operating in the appropriate frequency ranges at an acceptable power level. The cost of performing such a test is small, but the bureaucratic and cost hurdles of legally doing so make such experiments unreasonable for anyone wishing to operate within the law. However, it is likely that criminals would be able to design, build, and maintain the necessary equipment to launch attacks to an extent that is not yet known.

The law has always given the industrialized world the ability to prosecute criminals and in some cases yielded strong deterrents, but even the most aggressive ones cannot prevent a crime. System designers should never assume security through legality and instead take all necessary steps toward preventing any possible attack, legal or otherwise, against the system.

Misunderstanding the adversary and relying on the law as a crutch has yielded ineffective and damaging results. To avoid such mistakes in the future, today’s security system designers must be educated to think more like the adversary and understand the fault in assuming the adversary would be dissuaded by the unlawfulness of launching an attack.
RESEARCH LESSONS FROM HARDWARE HACKING

Want to know how something works? Tear it apart. Along the way, you might learn to improve it or make it do something it was never intended to do.

Hardware hacking—modifying a product to do something it was never intended to do by its original designers—and reverse engineering—the art of learning from practical examples—are both important facets in the training of any technology professional. Hardware hackers have a natural curiosity about opening things up and seeing how they work. They then go beyond their glossy, finely manicured shells to figure out what else can be done with or to improve them.

Although unconventional in the typical educational environment, such activities provide hands-on experience and a look into product design that cannot be learned from a textbook. The hardware hacking community represents an example of nontraditional learning and how the world can be changed for the better by unorthodox thinkers and their experiments (see the sidebar “A Brief History of Hardware Hacking”).
The thought processes of hackers and academic researchers are almost mirror images of each other. Hackers tend to learn by looking at a completed technology, taking it apart, working backward by breaking down the system into subsystems, and picking up the theory as needed. This style of learning allows them to discover previous methods of solving particular problems, seeing the solution used in a practical, real-world environment. Along the way, they frequently come up with unexpected uses for or improvements in the technology, depending on the goal of the exercise.

Academic researchers tend to work in the other direction. In a typical engineering program, theory, including math, physics, and basic electronics, comes first in a classroom setting. Then, only after they have proved themselves proficient in theory, are the students required to use their skills to design and construct some object or project in their final year.

An entertaining hardware hack-turned-academic study is available at the Hacking Microsoft Barney Web site [1]. This group effort by researchers at the Xerox Palo Alto Research Center and Massachusetts Institute of Technology aimed to reverse-engineer the control mechanism of the Microsoft ActiMates Barney plush toy and create a set of software com-
ponents to allow a user to programmatically control the toy. The resultant “Barney Protocol Stack” allows a user to drive Barney directly via an on-screen control panel to move it around, play sound samples, and read its sensors. Alternatively, the Stack can listen on a network socket for remote-control connections. The remote interface makes it possible for anyone to write applications that talk to a remote Barney server [7].

It appears the desire and opportunity to reverse-engineer some interesting technology—ActiMates Barney—and hack it to do something it was not intended to do by its original developers occurred first; the interesting outcome related to human factors interfaces were a result. This work exemplifies the methodology of combining nontraditional approaches to hardware hacking and a more structured academic environment.

People learn in many different ways, but a hands-on approach helps apply learned academic knowledge to practical problems. Since 2004, I have been leading a two-day training course called “Hands-On Hardware Hacking and Reverse Engineering Techniques” (www.grandideastudio.com/portfolio/index.php?id=1&prod=38) to teach basic electronics, hardware hacking, and reverse engineering skills. I conclude with a full-scale, hands-on hardware hacking challenge in which the students, individually or in small groups, try to defeat the security mechanisms of a custom-designed circuit board and unlock a hidden version of the popular memory game Simon (see Figure 1). Participants report they appreciate being able to experiment with hardware using their own methods, instead of being forced to learn through the more common, theory-based style of academic education.

Hardware hacking and do-it-yourself projects by hobbyists and tinkerers have been popular in recent years primarily due to their free-spirit, unconventional nature, even as the number of graduating engineering students in the U.S. has dropped steadily. The best learning methods depend on individuals’ ways of thinking.

**Microsoft Xbox**

When introduced in 2001, the Xbox was Microsoft’s first video game console, designed to compete directly with the established Sony PlayStation 2 and Nintendo GameCube. The original Xbox employed a PC architecture, making it an attractive target for hardware hackers. The follow-up Xbox 360 was released in the U.S. last November. Although serious hardware hackers began to disassemble the Xbox 360 the day it was released, the copyright protection and authentication schemes have yet to be broken. However, getting past them is likely only a matter of time, as no system is 100% secure, and any system can be broken by sufficiently competent, determined attackers.

The original Xbox platform consisted of several core hardware components: Intel Celeron-class 733MHz processor, nVidia GeForce graphics processing unit (also known as the Northbridge), nVidia media communications processor (also known as the Southbridge), 64MB RAM, 10GB hard drive, DVD drive, and 10/100Mbps RJ45 Ethernet network port. The machine was not dramatically different from a PC circa late 2001 or early 2002, though some components were custom-made for the Xbox (and are comparable to those in PCs).

Andrew “bunnie” Huang’s research with the Xbox began in late 2001 as a simple examination of the console while working on his Ph.D. thesis on supercomputer architecture at the Massachusetts Institute of Technology. One of his early interests was learning how Microsoft was authenticating the software running on the machine to prevent execution of unauthorized software, including pirated and homebrew games created by hobbyists and enthusiasts.

Through an extremely detailed and methodological approach, not unlike what you’d see in an academic laboratory, Huang managed to retrieve the sole symmetric 128b encryption key used for RC-4-based protection of a secret boot loader, ultimately allowing him to execute untrusted code on the Xbox. He created a custom tap circuit to intercept data transfer...
over the Xbox’s HyperTransport bus (an industry-standard, high-speed signal bus between the Northbridge and Southbridge chips). The tap board (see Figure 2) consisted of a single low-voltage differential signaling-to-CMOS logic converter interfaced to a Xilinx Virtex-E field-programmable gate array development board. Xbox security was thus defeated, as it relied on the secrecy of a key Huang figured out how to extract from the hardware.

He then wrote a technical paper at MIT’s Artificial Intelligence laboratory [5], made a presentation at the 2002 Cryptographic Hardware and Embedded Systems Conference, created a Web page/blog documenting his technical triumphs and frustrations (Adventures Hacking the Xbox, www.bunniestudios.com/?page_id=5), and ultimately created a seminal book on hardware hacking [6].

Although Microsoft and nVidia (manufacturer of the Northbridge and Southbridge chips whose communications Huang targeted on the HyperTransport bus) were initially displeased with his published work on defeating Xbox security, his results helped both companies understand the motives and threat vectors of potential attackers, leading to a more secure hardware design and authentication scheme on the later Xbox 360 console. The lesson is valuable; a hacker identified a major problem in a mass-market product and, by informing Microsoft and nVidia, likely saved both companies from repeating their mistakes in future products.

P

Perhaps the only way to stop an attacker is to think like one (see the article “Security Through Legality” by Stephen Bono et al. in this section). Being aware of the latest attack methodologies and trends enables a system’s designers to choose the proper means of protection and helps them understand the potential threats and risks against their products. The designers should try to break the security mechanisms of those products, then fix them and try to break them again. Time should be scheduled for this iterative process during the design cycle. Even more appropriate would be to add an objective third-party viewpoint by hiring a hardware hacker to attempt to defeat the product’s security or, as in open source software, release the product into the community before releasing it to the public, letting it be analyzed, taken apart, and modified, aiming to squash bugs or find previously unforeseen faults or limitations.

LEGAL CONCERNS

Recently enacted laws in the U.S. (such as the Digital Millennium Copyright Act of 1998) have sought to prevent reverse engineering by enabling large corporations to flex their legal and financial muscle against potential competitive threats, as well as against the merely curious. The DMCA was originally intended to stop pirates (selling bootlegged copyright material) from violating copyright laws and defeating anti-piracy protections on copyrighted material. Although used for this purpose, the DMCA’s anti-circumvention provisions have also been used to stifle an array of legitimate activities (such as reverse engineering, hardware modification, and public disclosure of potentially dangerous security vulnerabilities in products). Shrink-wrap and other explicit agreements are also used to get users to agree to waive their rights to reverse-engineer, analyze,
or modify products. More commonly used on computer software, these tactics have also been used on hardware. Suddenly, you don’t actually own what you’ve paid for and are reverse engineering.

A classic, fairly mainstream example of when hardware hacking locked horns with unscrupulous legal tactics can be found in the CueCat (see Figure 3) from DigitalConvergence (now defunct), which aimed to integrate classical push media (such as newspapers, magazines, and television) and the Internet. Beginning in 1999, venture capitalists invested more than $100 million into CueCat technology. The company gave away more than three million CueCat devices free at RadioShack stores throughout the U.S., along with 400,000 to subscribers of Wired Magazine and 850,000 to subscribers of Forbes Magazine in the mail worldwide. Digital Convergence planned to eventually put 10 million scanners into the public’s hands. Using proprietary “cues”—in this case special barcodes—users could swipe cues listed next to products in catalogs they were interested in. When a cue was received, CueCat software on the PC would bring the user to relevant Web pages with more information, usually the Web site of the product or technology.

In August and September 2000, lawyers representing DigitalConvergence sought to use the courts to begin a crackdown on all Web sites that described modifications to CueCat hardware or software, most involving a simple hack to convert the CueCat to read industry-standard barcodes instead of the special CueCat cues. DigitalConvergence claimed that by opening up and modifying CueCat hardware, hackers were violating the company’s intellectual property rights. It also claimed that the End User License Agreement included with the CueCat hardware and software stated that CueCat hardware was only on loan to end users who did not own the hardware. DigitalConvergence’s claim was challenged by the fact that thousands of end users were sent the CueCat in the form of unsolicited mail, classifying the product as a gift the recipient now legally owned.

Word spread quickly via mainstream media of DigitalConvergence’s methods of market research—collecting and analyzing the CueCat activities of individual end users (unknwonst to them), including their personal demographics and buying habits, and attempts to squelch the hardware hacking community. Not only was the company’s reputation tarnished by the bad press, the technology itself quickly became outdated due to the quick evolution of the Internet. By 2001, the CueCat had been relegated to a historical footnote, and DigitalConvergence was out of business.

Despite these and other legal precedents explicitly protecting and allowing certain types of reverse engineering for personal use or educational/academic purposes, please consult an attorney if you have questions or concerns.

CONCLUSION

I look forward to a future of hardware hacking that stretches the bounds of technology. I also look forward to more academic engineering programs incorporating aspects of hardware hacking, possibly in the form of hands-on, loosely structured exercises. Hardware hacking is an interesting direction professors and developers can incorporate into their daily routines. The do-it-yourself ethos of the hardware hacking community, coupled with the more structured approach of standard academic thinking will continue to lead to new and novel technology.

REFERENCES


Several magazines and Web sites also provide useful information on hardware hacking and its colorful community:

A BRIEF HISTORY OF HARDWARE HACKING

Hardware hacking dates back almost 200 years. Charles Babbage’s Difference Engine in the early 1800s was a mechanical form of hardware hacking. The method William Crookes used to discover the electron in the mid-1800s might have been the first form of electronics-related hardware hacking. Hardware hackers have since been involved in the development of wireless telegraphy, vacuum tubes, radio, television, and transistors. Benjamin Franklin, Thomas Edison, Nikola Tesla, and Alexander Graham Bell were all hardware hackers. So were William Hewlett and David Packard and Steve Jobs and Steve Wozniak.

Contrary to how the word hacker is sometimes used to describe criminals breaking into computer systems, a hacker can be defined more simply as someone involved in the exploration of technology. A “hack” in the technology world usually defines a new and novel creation or method of solving a problem, typically in an unorthodox fashion [2–4]. Here, I focus on an idealistic vision of hardware hackers—the good guys—even though some people, corporations, and agencies may use the same techniques for illegal, illegitimate, or unethical purposes, seeking some sort of financial gain or market advantage.

Hardware hacking means different things to different people, coming down to personal preferences, as in art or music. Someone can be taught, to a point, to have the hacker mindset and break the mold of conventional thinking, but hacking goes much further. It is a passion, a drive, something that stems from some amount of anti-establishment and anti-authority sentiment coupled with the desire to do things on one’s own agenda and with one’s own hands. Hardware hacking is the technologists’ version of the classic phrase “Don’t judge a book by its cover.” Hackers are driven by a variety of motivations:

- **Curiosity and fun.** See how things work, scratch the curiosity itch, and have fun experimenting with and modifying products;
- **Education.** Learn by doing;
- **Improvement and innovation.** Build a better mousetrap;
- **Consumer protection.** Ensure a product does what its marketing pitch claims it to do. Often distrustful of marketing or sales literature, hackers want to find out for themselves whether certain claims are true and how they can make a particular product do more; and
- **Security.** Test whether hardware devices are secure, identifying failures or weaknesses. Beyond strengthening the perceived value of a product, it allows users to mitigate the risk of an attack by updating, fixing, or discarding the product.

Most hardware hacks fall into four categories:

- **Personalizing and customizing.** Often called “hotrodding for geeks” it includes modifications, custom skins, and even art projects (such as creating an aquarium out of a vintage computer);
- **Adding functionality.** Making the system or product do something it wasn’t intended to do (such as converting an iPod to run Linux or modifying a classic Atari 2600 video game console to support stereo sound and composite video output);
- **Improving capacity or performance.** Enhancing or otherwise upgrading a product (such as expanding the recording capacity of a TiVo box by adding a larger hard drive, modifying a wireless network card to support an external antenna, or overclocking a PC’s motherboard); and
- **Defeating protection and security mechanisms.** Included are finding “Easter eggs,” hidden menus, and backdoors in DVD players or video game consoles or creating a custom cable to unlock the secrets of a cell phone.

Reverse engineering, generally viewed as a subset of hardware hacking, is essentially the art of learning from practical examples and experience. Examining technologies or any kind of product to see how they work is an integral part of the hardware hacking process and is a great way to learn the state of the art. I use reverse engineering to add to my mental toolbox of circuit designs, manufacturing techniques, and printed circuit board layout tricks, all of which improve my knowledge of the product development process. Reverse engineering and hardware hacking represent continuing education, interconnected with developing new products and technologies.
Achieving a truly secure connection at a public wireless hotspot is an impossible proposition. Despite the lack of security, wireless hotspots using 802.11-based wireless technology have popped up in coffeehouses, bookstores, and restaurants worldwide. The wireless protocol 802.11, better known by the marketing term WiFi, has become the mobile connectivity mechanism of choice for businesspeople, students, and everyone else. Unfortunately, even with the protocol’s ease of use and accessibility, WiFi security options remain limited. The threats against wireless networks reflect the variety of users on the network; the only proven tools for adding wireless security are geared toward large-scale enterprise deployments. Smaller networks lack access to the infrastructure.
needed to secure their transmissions. For all intents and purposes, security at a hotspot is unachievable, given the current state of the technology. Fundamental changes to WiFi security protocols are needed to bring effective security to hotspot users.

In order to understand how dangerous the situation is for a typical user, we must first understand the nature of the networks and the related hotspot attacks. Here, I examine the different types of WiFi networks to determine how they are built and maintained, weighing the state of the art in WiFi attacks and defense technology to understand how troubling the situation is for tens of millions of users worldwide. I also examine possible next steps for users, vendors, and WiFi service providers.

Large and complex corporate networks are a great place to begin to understand the inner workings of WiFi security. Corporate networks tend to take security seriously and as such involve some of the best tools and procedures for securing the technology and related communications. Corporations also represent a single point of trust for their employees—a critical point missing from public hotspots. Corporations can exert control over all aspects of their wireless networks—from client software, to users, to system operators—including most of the variables in the wireless security equation.

WiFi networks consist of two major components: access points and clients. Both exist in what can be thought of as a hub-and-spoke architecture. A client “associates” to an access point and sends all its traffic to it. In wired networks an association is analogous to plugging a cable from a computer into a switch or hub. Just as a switch can connect many computers, an access point can have many associated clients. An association is basically a connection at layer 2 of the OSI model (see Figure 1).

A corporate WiFi network consists of many access points, each representing an ingress point into an internal protected network and can be viewed as a switch waiting for a hacker to connect to in order to attack internal resources. Each one may also have other corporate resources (that is, other wireless users) attached to it that also need protection. Protecting access points and clients spread throughout an enterprise is an important and difficult task.

Networks and their clients can be protected in many different ways. The primary focus for WiFi security is protecting the confidentiality of the data while it is in the air and providing authentication for the client and the infrastructure, so each knows the other is a trusted entity. The first attempt in the original 802.11 protocols (circa 1999) at securing WiFi proved weak in both respects. Developed by the IEEE, the Wired Equivalent Privacy (WEP) protocol sought to do exactly as its name suggests—provide users the same level of security they would have when plugging into a wired network. Unfortunately, both authentication and encryption provided by WEP proved ineffective in the face of even novice attackers.

Newer wireless security standards offer much better security if set up and used properly. For example, IEEE 802.11i, a suite of three different security mechanisms, describes enhanced authentication and encryption mechanisms for WiFi networks using an authentication mechanism called the Extensible Authentication Protocol (EAP). EAP allows system designers to use whatever manner of authentication they need to secure their system. For some, this may be a simple user-name-and-password combination. Others may need much more assurance of the identity of the actors on the network; bidirectional certificate-based authentication is an option and a key method for creating secure wireless networks (see Figure 2).

With both the client verifying the identity of the access point and the access point verifying the client, attackers have difficulty pretending to be legitimate actors in the network. When a strong-enough signature algorithm and key length are used, attackers find
it almost impossible to impersonate a legitimate device, assuming the software that implements the certificate checking is properly coded.

The drawback of this bidirectional certificate-based authentication architecture is the difficulty of assembling and maintaining it. Clients and access points must have software that understands how to perform certificate-based authentication. An enterprise must run a public key infrastructure (or access an outsourced PKI) to handle the issuance, maintenance, and revocation of certificates. The enterprise must have a server for remote authentication dial-in user service to mediate authentication requests. If all parts are assembled correctly, the end result is that both the infrastructure and the clients are protected. However, it takes a great deal of central control and capital to make this happen. Unfortunately, even when technical expertise and money are available, the hotspot situation ends up being dramatically different.

Many enterprises employ operational procedures and tools that help protect their wireless networks. Since all components of an enterprise wireless network are part of the organization’s infrastructure, the operations staff must monitor it all. This important part of enterprise networks does not exist in hotspot networks. Mechanisms like wireless intrusion-detection systems and traffic analysis continuously look for attacks against the network and associated systems. Corporations can impose configuration requirements for laptops connecting to wireless networks, ensuring that wireless clients are not attractive targets for attackers.

HOTSPOT NETWORKS

Unlike the centrally controlled model of an enterprise wireless network, hotspots are much more of a free-for-all. While hotspots use the same basic WiFi technology as enterprise wireless networks, the usage scenarios and security ramifications are completely different. Users of hotspot networks are generally looking for “any port in a storm,” or access to any network that will get them to the Internet. Such open-ended availability and access can lead to problems for both the users and the networks they join.

Unlike the cellular network, there is no common mechanism for users to access networks controlled by different service providers. For example, T-Mobile provides access for a fee to WiFi users in Starbucks coffee shops throughout the U.S. T-Mobile has its own authentication and security infrastructure, as well as its own way of assembling networks. T-Mobile’s network and authentication have nothing to do with the small-town independent coffee shops that have set up hotspots for their customers. Any security mechanisms used to protect the network of one provider generally mean nothing to users of other providers.

Only in so much as it helps keep their networks secure are service providers motivated to focus on the security of their users’ systems. The first and foremost concern for providers is to protect their own infrastructure and systems.

They ensure these systems are usable to their customers by employing firewalls, rate-limiting devices, and some monitoring. They may block incoming connections from the Internet to wireless clients in an effort to keep worms and malware from affecting their customers. While this keeps malicious actors from attacking users’ machines, it also helps preserve the networks of the service providers by minimizing the amount of traffic they deal with and the potential for malicious activity.

Another central issue for ensuring hotspot security is how to manage layer-2 cryptographic data. In an enterprise environment, the clients and access points are controlled by a central authority. With a hotspot, there is no central point of trust that allows for cryptographic data to be given out in a secure and scalable fashion. The configurations used by the clients connecting to hotspots are variable. Some users may have four-year-old PC laptops, some may have cutting-edge MacBook Pros. Others may be running Linux. There is no guarantee that any of these users will be able to support the most current wireless security.

Some users may have systems infected with malware, including viruses and spyware. Rather than sacrifice revenue for the sake of secure users, many service providers recommend using a virtual private network...
(VPN) to keep traffic secure. Unfortunately, VPNs represent layer-3 solutions to layer-2 problems. A layer-3 solution (such as a firewall or VPN) does not necessarily mitigate attacks against layer 2.

Users of wireless hotspots are ultimately responsible for their own security. Although some tools give them enhanced security functionality when using a hotspot, the situation remains unpredictable and insecure for the majority of users. Operating system vendors and third-party software developers do not provide enough information and direction to users regarding the threats on wireless networks. The attacks against WiFi are not terribly complicated, but without tools for triggering alerts and defensive measures aimed at hotspot users, there’s little these users are able to do to protect themselves.

**WiFi Threats**

When users use a wireless network, they give up a foundational piece of information security: the physical layer. As outlined in Figure 1, the physical layer forms the base of the OSI model. Similarly, physical access is the core of many of the threat models used in information security. In wired networks lacking physical access to a network, the possible attacks are limited to layer 3 and above. For instance, in a normal office environment using a wired Ethernet network with IP, an attacker is not likely to be able to plug in without risk. So rather than plug in, the attacker might work through the Internet to attempt to gain access to internal systems. However, the attacker cannot execute attacks against layer 2. Even attacks that start at layer 3 (IP-based traffic) are limited due to the use of firewalls and intrusion-detection systems.

Wireless connectivity throws all this architecture out the window. An attacker has access to layer 1 from beyond the control of a user’s physical environment. Using relatively inexpensive antennas, free software, and personal motivation, an attacker might stealthily access a WiFi network from miles away. What this means in a corporate network is that a hacker could be in the company’s parking lot, a neighbor’s parking lot, or on the hill across the valley launching attacks against the network.

In hotspot environments, while attackers might act from a distance, they might also act very locally. Much of the software used to attack wireless networks runs on Linux and can be deployed on small devices. Linux can be installed on several different PDAs and mated up with wireless cards. Wireless attack tools can be configured to run automatically, giving an attacker the ability to seem to be completely innocuous. A PDA can be hidden in a backpack or jacket pocket or even carried around without being noticed. However, while attackers are ordering venti lattes and blueberry scones, PDAs in their backpacks are busy intercepting data and exploiting wireless clients.

The most dangerous attack against hotspot networks targets the client computers directly by tricking them into connecting to the attacker’s network. An attacker creates a rogue access point that pretends to be a legitimate access point (see Figure 3). When a wireless client attempts to associate with a network, it looks first for all the access points within it. If the client is looking to join the network COFFEE, it sends a probe packet asking to join that network. If multiple access points respond as if they were part of the COFFEE network, the client connects to the access point with the strongest signal.

Attackers wishing to create a rogue access point for the COFFEE network need only ensure that their signal is stronger than the signal of the legitimate access point. This can be accomplished through high-gain...
antennas or through high-power transmitters. In the “PDA in a backpack” scenario, the attacker can use a PC card in the PDA combined with access-point software running Linux to create a stronger signal, even when hidden in a bag or pocket.

Once the client associates to the rogue access point, the attacker then spoofs the rest of the network. The rogue device, having completely subverted the layer-2 connection, can cause undetected havoc at higher layers. The attacker might issue IP addresses to the client, provide bogus DNS responses, and even mask real Web sites with fake ones. Attackers using rogue access points can spoof the login screens of major Web portals (such as Hotmail, Gmail, eBay, and Yahoo) in order to gain usernames and passwords from unwitting victims.

Unfortunately for wireless clients, such an attack is difficult to detect. There is no preestablished trust between the hotspot network and the client. Lacking newer authentication techniques that are not available on many wireless clients, the client has no way to determine if the access point it is being connected to is the right one. Windows (as well as the other operating systems) provide no feedback to the user as to what is happening at layer 2.

The user is not notified if, say, a new access point named COFFEE suddenly pops up and is twice as strong as the previous access point. On the face of it, this scenario should be easy to detect. Unfortunately, few tools are able to identify the situation and send an alert. AirDefense Personal (www.airdefense.net/products/adpersonal/index.php) and other products give users a view of malicious activities at layer 2 and attempt to prevent insecure configurations from being used. Such features need to be included in the operating system to be effective for the vast majority of wireless users.

Several variations on the rogue access point attack are worth mentioning. Most operating systems, including Windows and Mac OS X, keep track of networks to which they’ve previously been connected. When the wireless interface comes online, the client begins looking for these trusted networks. If it finds one of them, the client automatically connects to it. Many users have trusted networks with guessable names (such as “home” and “wireless”). Some users never change their home network name from the default name shipped with their access point. For example, linksys routers use the name linksys by default. Attackers can create rogue access points with the names of common networks (such as “linksys” and “home”) in order to trick clients into connecting automatically. This attack is quite successful and difficult to prevent.

Another problem with using hotspot networks is that attackers can sniff traffic and communicate directly with other client computers. Many users feel that if they are running a VPN connection, they are safe from such attacks. While sniffing and direct communication are more difficult if a user has a VPN, they are not impossible. A VPN can also give a false sense of security if it is used only for communicating with a trusted network (such as a remote office). In this case, known as a “split tunnel,” the client is still sending data across the network in the clear and is likely still accepting connections from other machines.

Modern PCs are very chatty on the network. Attackers sniffing the network are probably able to find usernames, Web sites visited, personal information, and maybe even hashed passwords from instant messaging and mail programs. Rather than sending a password over the network, many applications will create a cryptographic hash of the password and send the hash. Since the hash is a one-way function, application developers feel it is a secure way of transmitting the password across the network. Unfortunately, password hashes are not as resilient to attack as they used to be. Password-guessing programs have become sophisticated, and computers have become powerful enough to quickly guess a great number of passwords. According to [1], a 3.2GHz Xeon processor can sort through more than 9,000 MD5 one-way cryptographic hashed passwords per second and 4.5 million Lanman (Windows authentication) hashes per second. Password guessing is also an easy process to parallelize; it’s so easy that many security experts consider the loss of a password hash equivalent to the loss of the password itself.

Problems with Mitigation
Mitigating these problems is clearly difficult. First and foremost the 802.11 protocol is designed to make layer-2 transitions transparent to the user. While such transparency is great from a usability perspective, it is terrible from a security perspective. To avoid attacks (such as rogue access points) the core protocol must be violated, a preexisting trust relationship must exist in the form of bidirectional certificate-based authentication; otherwise, security software (such as a wireless intrusion detection system) must be added after the fact.

None of these solutions is particularly useful in normal hotspot environments. Worse, even educated users have no way of knowing if something malicious is happening on the network without using specialized wireless security software. Users have been educated over the years that when using a
secure-sockets-layer-protected Web site they must look at the URL to ensure they are at the right site and to “look for the lock” to ensure the traffic is protected. There is no analog for this activity on wireless networks. The network name is the same whether it is the legitimate network or a rogue; moreover, the user has no visual cue to look for.

Applications are unaware of the network environment in which they run. An instant messaging client or Web browser has no way of knowing if the computer it is running on is within a controlled area with a wired network or if it is at a coffee shop with a random wireless network. Attackers who subvert the wireless connection will then probably try to subvert applications running on the client system. Ideally, the applications are able to recognize differing threat environments and reconfigure themselves accordingly. Conventional wisdom with secure software architectures do not account for these situations.

CONCLUSION
For all their utility and ease of use, hotspots are dangerous places. While every coffeehouse and airport lounge may not include an attacker lying in wait for victim hosts, the fact is attackers are likely to be successful. Users in enterprise environments have the luxury of a single point of control and administration that creates “security of scale” for wireless users. In hotspots, users are on their own. Despite the availability of tools and point solutions, most users represent easy prey for sophisticated attackers.

The state of the art with respect to wireless defense is behind the state of the art with respect to wireless attack. As technologies evolve, users will become better armed to deal with the threat posed in hotspots. In the meantime, it may be better to shut the laptop, enjoy the coffee, and keep an eye on the people nearby using PDAs with wireless cards sticking out.

REFERENCE

BRUCE POTTER (potter_bruce@bah.com) is a senior associate at Booz Allen Hamilton, Linthicum MD, and founder of the Shmoo Group of security professionals (www.shmoo.com).
SOFTWARE SECURITY IS SOFTWARE RELIABILITY

By FELIX “FX” LINDNER

Recurrent efforts by academic researchers and the computer security industry have sought to find ways to detect and prevent software vulnerabilities from being exploited. Others have sought to find ways to detect and prevent unauthorized access to computer systems. While attack methods may differ significantly, the underlying security issues (viewed through the prism of academic software reliability research) are called “software faults.” Hackers, however, describe the same issues in different terms. Attempts to identify similarities among faults are biased toward the hacker view, as I discuss here, and often yield incomplete defenses. Missing the fact that reliability and security research addresses the same technical issues leads to inadequate approaches by the academic community [1].

Enlist hacker expertise, but stay with academic fault naming conventions, when defending against the risk of exploitation of vulnerabilities and intrusions.
If software always worked as specified or intended by its makers, only a small subset would be vulnerable to attack, and defenses would be much easier to implement. By resolving the naming issues and looking at hard data (such as vulnerabilities and their fault classes), software defenders and users alike would be able to achieve secure, reliable software.

The science of software testing names software faults according to their root cause. Hackers name security issues according to bug class, a type of software defect they are able to exploit. When a particular type of coding fault is exploited for the first time, it becomes a bug class, and attackers search for instances of the same bug class in all other types of software, targeting similar vulnerabilities. The exploited software fault always belongs to a bug class, while most software faults don’t belong to a particular bug class, since attacking hackers don’t know how to exploit them.

Considering how academic researchers name software defects reveals how many obscure names from the hacker community emerge; for example, in software testing, buffer overflows and integer overflows are called “data reference failures.” Most so-called denial-of-service problems in software are data reference failures, some falsely classified as “denial of service,” rather than “nonexploitable buffer overflows.” This leads to the perception that the fault cannot be used to break into the system, though only the person who classified it is unable to break in. This nomenclature is evidence that naming is based on exploitability rather than cause and thus often produces solutions that ignore entire groups of faults.

So-called format string bugs and several types of race conditions belong to the fault class known as “interface failures.” Directory traversal bugs, illegal directory or file access, and remote command execution almost always turn out to be interface failures. The difference is that the interface failing is so important that these faults deserve their own name—“operating system interface failures.”

Two bug classes that have emerged in the context of Web applications are “SQL injections” and “cross-site scripting vulnerabilities,” each of which can be classified as input/output errors. Academic research in reliable software and software testing has identified several other defect types that hackers have not (yet) been able to exploit and hence have not yet named.

How do hackers distinguish between a software bug that is not security-related and a vulnerability? For example, finding all potential buffer overflows in a given piece of software may yield a number of findings; none, some, or all might be classified as a vulnerability. Classification depends on who causes the condition, what is the condition’s most severe effect, and how the condition might be used to perform an
action that would normally be restricted. If the who question can be answered with “hacker,” the what question with an (at least) “mostly controllable condition,” and the how question with “a known procedure for this bug class,” hackers call it a vulnerability, since they are able to exploit it. A piece of software written to automatically take advantage of the situation is called “the exploit.”

Developers must distinguish among software fault, vulnerability, and exploit to be able to build effective defenses. Protection mechanisms [3, 5, 6] aim to prevent functioning exploits while at least theoretically (and often practically) still allowing a software bug to turn into a vulnerability. In hacker parlance, the bug class is the same and valid; only the exploitation parameters have changed.

To clarify the process of turning nonideal software behavior into an exploit, I offer two simplified, fictionalized examples from entirely different bug classes and explain the thought processes needed to exploit each of them. One is a binary program called “user-agent capability matching,” or uacm, running on Linux. The tool matches the HTTP user-agent string, usually sent by Web browsers, against a growing database of known Web browsers and identifies their capabilities; this way, a Web developer knows exactly which HTML version and features will work and which won’t. Web developers call this command line tool from their common gateway interface scripts on a Web server to generate elegant and functioning HTML based on its output. The tool takes the environment variable HTTP_USER_AGENT as input and returns the result to the standard output.

The second example is a front-end Web application called “Web customer relationship management,” or WebCRM, involving an input form requiring a username and a password for logging-in as a customer. In this application, customers cannot simply register themselves.

As I discussed earlier, a bug must be identified by the hacker before it becomes a vulnerability. The same must happen in the two examples. The uacm binary can be tested on the Linux command line by setting the environment variable HTTP_USER_AGENT to arbitrary values and then running the program. In this case, the hacker chooses to set the environment variable to a long string of characters (typically the character A) and runs the program. There is a fair chance that the attack will not be as straightforward as the test, but this is of no concern to the hacker when identifying a vulnerability.

When running the uacm binary with more than 200 characters, a hacker would observe a crash that produces the message “segmentation fault (core dumped)” as a result of a critical error occurring in the program’s memory space. Identifying a potential issue in WebCRM is different. The hacker inserts a number of nonalphanumeric characters (such as “,” , and %) in an application’s username field and clicks the login button. The application returns an error message stating that the execution of a SQL statement failed due to a syntax error, though the application does not show the failing statement. When trying to login with alphanumeric characters for both username and password, the application presents a “wrong password” Web page.

For the uacm binary, the hacker identifies the addresses of the last successful library calls by using ltrace, which is designed to record calls to library functions and their arguments [2], as in this classic stack buffer overflow scenario [5]:

```
[0x804846d] getenv("HTTP_USER_AGENT") = "AAAAAAAAAA..."
[0x80484a8] strcpy(0xbfb5e7a8, "AAAAAAAAAAA...") = 0xbfb5e7a8
```

Here, the address of the first argument of strcpy is a stack location. Inspection of the disassembled code around the caller's address [0x80484a8] then shows that the destination buffer is approximately 110B, after which saved addresses of the CPU are overwritten. For the WebCRM application, a hacker would test each of the previous nonalphanumeric characters separately. By deducing that only the ’ character causes an error message, the hacker would then make an educated guess about the type of issue—in this case a SQL injection.

The methodologies of the two types of attacks converge at this stage in the process of identifying a vulnerability and turning it into an exploit. The attack on the uacm binary and the attack on the WebCRM application each present the same general challenge. The attacker must build a mental representation of a remote system through educated guessing and intuition. Based on this representation, which a hacker might or might not be able to verify, the hacker would have to deduce a method to influence the remote program. The hacker imagines how the process operates on the remote system when overflowing the buffer on the stack and overwriting the saved return address on the stack to control the remote program. While the WebCRM application suffers from a completely different type of vulnerability, the steps the hacker must take in building a mental representation are the same.
as with the uacm binary. When the user data contains the ' character, the hacker is able to terminate the data and modify the actual SQL statement. The hacker must also make assumptions concerning the nature and structure of the statement to be modified, since it is not visible. The aim is to modify the executed statement and change its meaning in a syntactically correct way, causing the system to falsely identify the hacker as a legitimate user.

In the uacm binary example, the hacker must make assumptions concerning the layout of the stack on the target machine. Overwriting the saved return address of the affected function with an address pointing inside the buffer would cause the CPU to attempt to execute (as code) the data in the user agent string. Exploiting this effect, the hacker can send custom-developed machine instructions in the string instead of a series of capital A letters. If everything works well, execution redirection occurs, the code is executed, and the hacker can run arbitrary functionality of the hacker’s choosing on the remote system.

The WebCRM application is exploited by supplying the specially crafted string ‘ OR ‘1’=‘1 as the username without a password; it is then concatenated to the SQL statement on the server-side (attacker string underlined):

```
SELECT * FROM usertable WHERE username = ' OR '1'='1' AND PASSWORD= '',
```

The string causes the database to return any username with an empty password. If at least one user exists without a password, the hacker gets in.

The respective victims are, however, able to fix the bugs relatively easily. For example, the developer of the uacm program can introduce a length check of the HTTP user-agent string before copying it into a fixed size buffer. And the developer of the WebCRM application software can disallow any character other than alphanumeric ones in usernames and passwords. Unfortunately, it is common knowledge that such selective fixes do not work well in the long run. Where there is one bug, there are others, essentially representing a quality problem in the software.

**Bug Class Evolution**

While not a perfect data source, the Common Vulnerabilities and Exposures database (cve.mitre.org) contains (as of Feb. 6, 2006) 15,024 entries of publicly known security issues. I used the entire database and a simple keyword-matching script to reclassify the vulnerabilities from hacker terms into a number of the software fault classes known by academic researchers.

This remapping yielded several interesting insights concerning the evolution of bug classes (see the figure here). The most interesting is the prominent increase of input/output errors since 2000, likely occurring for two main reasons: ease of testing for faults and a gradual change in development environments. For example, it is easier to test for faults in the category of SQL injections and cross-site scripting. Moreover, the number of Web-based systems has also increased, along with the number of potential targets. More and more potential targets attract more and more hackers to look for this type of vulnerability.

Meanwhile, most programmers don’t write critical software in C anymore, especially in Web environments, (the most widely attacked systems), due to their visibility and Internet-wide accessibility. Dominant in this domain are languages like PHP and Java that are less prone to buffer overflow attacks but are more likely to produce operating system interface- and input/output-error-type faults. Also, along with increased use of modern programming languages has come a steady decrease in the number of data-reference faults, or buffer overflows over time, due to their built-in boundary checks.

An emerging trend among hackers involves going from attacks on the deepest level of software upward on the abstraction level toward attacks involving the application’s own logic. In principle, these attacks do not differ much from buffer overflow attacks, but it’s much easier for attackers to adjust their methods...
from buffer overflows to SQL injections than to modify specialized defenses to prevent new attacks.

An additional problem is identifying the point of prevention. The National Vulnerability Database [7] reported that in 2000, 59% of all published vulnerabilities concerned server software. The picture had reversed in 2005 with 63% of all published vulnerabilities concerning clients (such as Web browsers). This finding does not mean that servers are more secure but that attackers moved on because there is still plenty of vulnerable software around.

Commonly deployed defense mechanisms (such as network and application firewalls) have changed the picture only slightly for attackers while requiring a significant amount of additional development and maintenance work on the part of system administrators. Ways to prevent specific exploitation techniques are likely to be rejected when undergoing thorough cost/benefit analysis.

The anti-hacker technology that probably has the greatest impact on system security is the rule write XOR execute, which declares memory either writable or executable, never both. It is aimed at preventing the execution of code in memory areas that are writable but does not affect legitimate applications and defies exploitation. The rule was developed by hackers who gave it to both the software development and hardware industries.

The only approach that seems to work well for identifying vulnerabilities and protecting all kinds of systems is source code auditing, whereby a skilled third party reevaluates the software design and its implementation. Unfortunately, the availability of skilled third parties is limited, especially with the amount of software worldwide doubling in size approximately every 18 months.

Hackers are for computer security what [8] described as the “intraspecialist level,” or highly motivated experts in their own limited field. Software security and reliability research is very strong at both the interspecialist level and at the pedagogical level and is found in the academic community and in authoring textbooks. Recent breakthroughs (such as binary code-matching algorithms) by hackers and computer security researchers have been possible only because hackers found their way into the academic world [4] yet were still required to be hackers.

The most pressing issue in the cooperation between hackers and academic research is a lack of access to each other’s work methods. The more accomplished hackers are motivated to find solutions for the security challenges faced by the worldwide community of computer-dependent people. They would thus work well as peers in a scientific context. What they need is a team of researchers with the required background, or what Ludwik Fleck, a Polish biologist, calls “textbook science” [9]. Access to such work environments is not available for most hackers due to their lack of academic titles, credentials, and status, but strong demand for qualified security experts signals the value of their expertise.

**CONCLUSION**

Software testing procedures and algorithms have advanced only incrementally since the 1970s and 1980s. Software security is the driving force behind the need for software quality, since a lack of quality is the primary reason for insecure software. Hackers have managed to reinvent what the world of computer science has known for decades, only poorly. On the other hand, they identify a lot of software vulnerabilities through their innovation. The only solution for making systems as secure as we need them to be by eliminating software vulnerabilities is for hackers and academic researchers to unite.

**REFERENCES**


**FELIX “FX” LINDNER (fx@sabre-labs.com) runs SABRE Labs, a computer security consulting company in Berlin, Germany.**

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2006 ACM 0001-0782/06/0600 $5.00
EXPLORATIONS IN NAMESPACE: WHITE-HAT HACKING ACROSS THE DOMAIN NAME SYSTEM

DNS cache scanning across a sample set of more than 500,000 name servers revealed the extent of last year’s Sony rootkit infestation on client machines.

It’s a fact that the larger the data set, the more difficult it is to update any individual entry within it and the more likely an individual record will become out of date. It’s from this observation that the Domain Name System (DNS) was created to cope with the growing lists of domain names (such as www.doxpara.com) that need accurate mappings to Internet Protocol (IP) addresses (such as 209.81.42.254). Originally designed in 1983 by Paul Mockapetris, then of the Information Sciences Institute of the University of Southern California, DNS offers a scalable, hierarchal, distributed approach to name resolution. DNS remains a core component of the Internet, but something has changed over the years. As the Internet has grown, attackers anywhere have become attackers everywhere, in part because of DNS.
My personal interest involves identifying and reporting these security faults on a global scale. The results have been surprising, even to me, considering that I look at every kind of DNS-related traffic I can access on the Internet; DNS is much more complicated than I thought, and name servers most certainly do not always tell the truth.

My scans for these faults have not been trivial. Code is easy, politics is not, particularly when dealing with organizations responsible for network infrastructure. But we cannot defend our networks if we are blind to what they contain. DNS is a massive, distributed, shared worldwide resource. We must not cede knowledge of it to those who wish to destroy it. DNS is one of the few remaining pre-firewall-era applications and may be the only one on the Internet today against which administrators of firewalls refuse to interfere. From a firewall perspective, the configuration “problem” of properly securing DNS is not like the configuration problem with the File Transfer Protocol (FTP). FTP uses a port-selection strategy not used by other protocols; it is therefore only moderately at risk from devices in the middle of networks that seek to censor and alter traffic flows. More important, FTP fragility problems are limited to FTP transmissions.

On the other hand, DNS is a prerequisite service for almost every network-enabled codebase in existence. When something breaks the domain-name-to-
THE INTERNET DOES NOT ROUTE NAMES, AND SONY’S ROOTKIT DID NOT HARD-CODE IP ADDRESSES. THE LACK OF HARD-CODED IP ADDRESSES MEANT (AHA!) DNS LOOKUPS WERE REQUIRED FOR THE ROOTKIT TO CONNECT BACK TO A MASTER SERVER.

IP-address service provided by DNS, the result is that random, highly visible services (such as email and Web browsing) begin to fail. Time and again, it takes so long for network administrators to trace the problem back to DNS—hindered by blocking libraries and poor runtime debugging—that by the time the problem is identified, the firewall vendor, rather than DNS, is often blamed instead. Firewall vendors fear inhibiting DNS in any way, while system administrators tend to feel that DNS is more likely to break if they touch it. So name servers chug on, providing their services as they always have.

Minimal oversight from system administrators and firewalls leads to other problems. For example, in March 2005, a number of name servers started returning bad data for Google servers. Attackers had exposed the servers to a variant of what’s called a Kashpureff attack. Named after the erstwhile Eugene E. Kashpureff, whose zeal for alternative DNS top-level domains in 1997 exceeded his fear of deportation and eventual prosecution by the U.S. government. The Kashpureff attack on Google’s servers involved sending more domain-name-to-IP-address mappings than had originally been requested; the name server would then cache these additional answers alongside what had actually been requested.

What made this variant interesting to security professionals was the fact that name servers are often not set up to operate independently but can be configured to run in fairly complicated trees and graphs that query other nodes before escaping out to the DNS root servers for resolution services. This preference for local nodes, called a “forwarding relationship,” involves trusting that whoever you were speaking to will pass you only appropriate data (see Figure 1).

The new attack involved the Berkeley Internet Domain Name Domain (BIND) (version 8) implementation of the DNS protocols, which didn’t trust Kashpureff-polluted packets but would forward them unfettered anyway. Other name servers (Microsoft DNS was targeted, though others were vulnerable) would trust anything forwarded to them. The attackers had revived an old technique by identifying and exploiting it in new architectures.

Resurrecting old techniques is common in today’s security environment; why find new bugs, when the old ones can be used to locate architectural mistakes programmers are likely to make. Regression testing for security flaws needs more work to ensure that already-identified security flaws do not make their way into new software applications. Luckily, the attackers didn’t hit too many name servers. But how much damage could they have done had they succeeded on a global scale?

CHALLENGING SCAN AND NOTIFY
Two things are not generally recognized by network administrators and governance bodies as being part of the status quo of operating on the Internet: servers will be scanned, and users who detect servers that appear to have been compromised can notify anyone else of their discovery.

Computer security professionals generally know that the average lifespan of an unpatched system on a network without a firewall can be measured in minutes. Less known is that at least one Class A network (16 million IP addresses) is constantly receiving 5MB/sec of traffic. But there’s nothing there, because no machines reside at any of the 16 million addresses, and the addresses should not receive any traffic. From this we can surmise that unknown individuals and organizations are continually scanning the Internet’s entire IP address space. Since “white hat” hackers do not break into systems as a matter of course, and systems are indeed being broken into en masse, we know that the “black hats” are largely doing the scanning. Whereas black hats ought not to have better intelligence than we do, white hats must be able to scan at least as effectively as their black-hat counterparts.

When it comes to notification, things get interesting, because the great advantage white hats have over black hats is they don’t have to hide. Indeed, on my own scanning node (deluvian.doxpara.com), you’ll even find my cell phone number. Because we white hats have no fear of being caught, we don’t need stealth. White hats should push this advantage as
much as possible. Not only would you be notified by them that your scans have been noticed but that your servers and network hardware may have been broken into.

Black hats have long histories of breaking into systems, recruiting them for information gathering and using them as stepping-stones to exploit other third-party systems. A common line of communication involves an administrator who notices a scan from an external network, then informs the network’s administrators that they have a compromised machine; remediation can then begin. Notification is part of the status quo.

My scan-and-notify proposition involves a caveat. While both scans and notifications are already taking place on the Internet, scanners generally do not notify the servers they are scanning, and notifiers are generally not also scanners. I intend to change the role of scanners to help improve large-scale Internet security. The security community’s advisory model—in which recipients are obliged to evaluate their networks against the constraints documented in the advisory—doesn’t work well. It also hasn’t helped that it’s become so difficult to acquire tools capable of verifying vulnerabilities.

Would it help to have site-customized advisories able to state that not only is there a vulnerability but that the scanners can see that your site is affected? Possibly, so I thought it would be worthwhile to find out. In 2005 after setting up a suitably uncloaked scanning host (with custom whois entries, reverse DNS, and my cell phone number being available via HTTP) I began trying to document the extent that other DNS servers across the Internet were vulnerable to a Kashpureff attack.

**Shedding Light**

My first step in the scan-and-notify process was to sweep the IP space. I modified my high-speed TCP service scanner, Scanrand (www.doxpara.com, part of the Paketto Keiretsu suite of tools for manipulating TCP/IP networks), to emit User Datagram Protocol (UDP) packets on the standard DNS port—53. Although TCP scanning could have detected servers, many name servers fail to respond on TCP/53, suggesting a UDP approach was warranted. But which packets should be sent? Unlike TCP-based services, applications that expose their services over UDP must participate in announcing their presence. The message delivered to them must be understood by the DNS server, and the server must emit a response. So a DNS query is required, one that would be understood and responded to by any host receiving the request, preferably without having to travel elsewhere in the DNS hierarchy. I then considered how to construct this query, ultimately using localhost (127.0.0.1), a special IP address machines use to refer to themselves.

I constructed a DNS PTR (pointer) lookup query for 1.0.0.127.in-addr.arpa. It asked the DNS server to return the domain name associated with a given IP address, a process also known as a reverse lookup. In theory, each server that receives the request should return “localhost” as a reply. In most cases, this was exactly what occurred in response to my query, but in a surprising number of cases the servers leaked additional information by announcing themselves as localhost.foo.com, where foo is the name of the domain associated with the DNS server. I detected other response patterns as well. One conclusion I made while doing these scans was that when scanning for a particular Internetworked application, domain-specific probes can yield more data than might otherwise be expected.

Another result of my scans was that I witnessed a second class of DNS traffic. I configured my DNS server to allow anyone attempting to lookup the IP address for my scanning computer (deluvian.doxpara.com) to receive a name directing this person to more information, including scanning. Please-browse-to.http.deluvian.doxpara.com. Approximately 35,000 individual servers took me up on this request, many immediately after I probed their related networks. It is likely that automated systems—some designed for forensic logging—were announcing their presence to me, despite my being an unauthenticated party. I saw at least one packet from a Class A network from which I had never seen anything else. But more probable was that some of this traffic was coming from active forensics personnel trying to determine what I was up to.

I initially found nine million hosts, not all willing to keep talking to me. As my work progressed, I selected a 2.5-million-node subset for manageability purposes and extracted name-server version data from each host using Roy Arends’ excellent DNS fingerprinting software (www.rfc.se/fpdns/). What remained was the difficult part of my search for servers still vulnerable to a Kashpureff attack—detecting and documenting forwarding interrelationships among DNS servers.

**Peerage**

All known mechanisms for reasonably detecting interrelationships among name servers rely on the fact that when one name server forwards a request to another name server, this other server ends up with the response in its cache. Luis Grangeia, a systems
and network auditor, wrote in [1] that an obscure mode of DNS, accessible via the Domain Internet Groper tool, allows cache contents to be probed nondestructively. Therefore, cache contents can be, at least to some degree, publicly monitored.

This mode of DNS offered three approaches for identifying interrelationships. First, I could issue a normal (but unique) query to one host, then nondestructively see if the results from that query showed up anywhere else. If, for example, I had asked Alice to retrieve data for me, and Bob ended up with that data, I could be pretty sure Alice and Bob were talking to each other. The problem was we weren’t just talking about Alice and Bob; I wished to test millions of name servers on the Internet—an O(N²) solution—meaning my approach would quickly become computationally infeasible.

An interesting variant of this approach is to recognize that old data can be differentiated from new data; it’s possible to know to the second when a given record entered a particular host. We are able to determine such information because each record is associated with a Time To Live (TTL) value describing the number of seconds until the data is considered stale. Each node in the DNS is specified to decrement TTL, so it’s trivial to subtract the original TTL from the retrieved value and determine the precise time a given value entered the cache.

This approach also had trouble dealing with complexity; for example, to scan more than a single node per second, the scanning machine would have to constantly shuffle its scan space, then identify relationships that show up at matching offsets, no matter what the order of the scan may be. This processing overhead is doable but can be prohibitively complicated.

I eventually implemented a more straightforward solution. In DNS, I can control parts of the namespace; if anyone in the world tries to look up something inside the doxpara.com domain, the packets come to me. So, I began probing DNS servers by asking them to look up a unique name, then monitored who came to me to service that request. If, for example, I asked Alice to look up something, and Bob came to my server hat in hand with a lookup in tow, that would mean Alice was talking to Bob. I used this simple solution to detect several forwarding relationships:

- 13,000 Microsoft DNS servers forwarding to BIND8;
- 18,000 BIND9 forwarding to BIND8; and
- 230,000 total name servers (almost 10% of the detected sample set) forwarding to BIND8, in direct contravention of the security disclaimer on the BIND home page. (Apparently, a very visible security advisory on BIND’s Web site wasn’t enough for a number of people to properly configure their machines.)

This probing produced yet another inexplicable result. After generating two kinds of information—the type of nameserver software run by each node and edge information, or which nameservers were forwarding to which other nameservers—I tried to generate the full graph for the analyzed subset of the global DNS architecture. Building the graph, I mainly expected to find clusters of four or five nodes in some sort of cooperative arrangement for DNS lookups. This was indeed the case for the 40,000 networks I had probed, but I then found an anomaly consisting of 220,000 nodes, 330,000 edges, and a depth of 20. In one case, a DNS request sent to one particularly interesting server yielded traffic from one of a thousand other systems. I am still analyzing this result.

Live in Your World; Get Pwned1 in Ours

During the late 1990s, the term “Halloween documents” referred to embarrassing leaks from Microsoft. More recently, during Halloween 2005, Mark Russinovich, author of a number of advanced Windows utilities, highly regarded technology blogger at www.sysinternals.com, and probably one of the top five Win32 hackers in the world not working for Microsoft, documented how in the development of his RootkitRevealer tool, he’d found that his own system was infected by a rootkit. A rootkit is software that allows malicious applications to operate stealthily. He’d traced the matter back to a Sony CD he’d purchased; using “sterile CD” software from the U.K.’s First4Internet, a commercial provider of digital rights management (DRM) solutions (www.first4internet.com/), Sony had caused an indeterminate number of PCs to lose some of their ability to rip CDs into applications like Apple’s iTunes or burn their own mix CDs.

This software cloaked itself using mechanisms so deep in the black-hat playbook it could be referred to as a rootkit. Indeed, black hats soon began borrowing access to its cloaking functionality to evade antivirus and antispy software. Worse, the initial uninstaller provided by Sony removed only the cloaking and not the DRM software—hardly the total removal users were looking for. It became clear to me that this code was designed to expect the loss of consent to execute. At some point, the rootkit developer must have asked himself whether users would want to get rid of it. The

---

1Pwned (Owned) is Internet slang for gaining root privileges on a computer by exploit- ing a security vulnerability (see en.wikipedia.org/wiki/Pwned for more information).
answer, it seems, was “not if they don’t know it’s running.”

Such behavior from a company like Sony posed a problem for the security community and everyone else. It’s difficult to fight skilled hackers out for fun. It’s difficult to fight experienced, financially motivated criminal operations. But fighting a billion-dollar, corporate-funded hacking operation is impossible. Even the most experienced infantry will fall to aerial bombardment, and with four million CDs being sold directly to consumers, these CDs were a bombardment we (security experts throughout the Internet) hadn’t yet noticed; it took six months and Mark Russinovich to detect the presence of the rootkit. Where were the security vendors? Only Sony knew how many systems were infected. How could the security industry respond to something it could not measure?

I needed hard data that would allow me to measure the impact of the infection. Traffic dumps from rootkit-infected systems showed network connectivity to connected.sonymusic.com. While I did not expect traffic to this domain to be only rootkit-related, other researchers told me that updates.xcp-aurora.com had been identified as another suspicious domain.

The Internet does not route names, and Sony’s rootkit did not hard-code IP addresses. The lack of hard-coded IP addresses meant (aha!) DNS lookups were required for the rootkit to connect back to a master server. The lookups would execute, the responses would cache, and I could use the DNS cache snooping method [1] across my entire sample set to estimate penetration levels by the Sony rootkit. However, using this technique, I was unable to get per-host results, but the per-network results I could get would give me an idea (at least) of the scale of the rootkit infections. My analysis showed more than 500,000 name servers responding to nonrecursive queries with one of the two monitored names. This was rather more than I expected.

Over the next few days, I weeded out false positives through two approaches:

- Many servers look up data recursively even if they’ve been instructed not to. Originally, I found almost a million hosts responding; due to the fault, I had to eliminate 350,000; and
- Some servers lie. Captive portals, commonly used in wireless portal applications, intercept all incoming DNS queries and return with links to themselves. Along with many recursive lookers, these portals were easy to detect, as their TTLs were always equal to some value divisible by 100.

I ultimately found that more than 556,000 name servers had returned names related to the Sony rootkit, with 76% of the servers returning both connected.sonymusic.com and one of the First4Internet addresses. Geolocation allowed me to identify infected nodes in 165 countries around the world (see Figure 2). It was clear that security researchers had a
new weapon—DNS—in the war against malware and that Sony had been its first target.

Even so, matters were still not clear. For example, connected.sonymusic.com was in fact an address used for almost eight years by Sony Enhanced CDs (but not rootkitted). Only gawkers and uninstallers went to the First4Internet sites; the rootkit itself would not natively go to that address. Security experts around the world were concerned (despite the CDs supposedly being sold only in the U.S. and Canada). Luckily, a breakthrough emerged that allowed new, much more accurate scans. In it, connected.sonymusic.com would issue an HTTP redirect to another site—xcpimages.sonybmg.com—but only in situations where an infected CD was inserted into a computer. Following these redirections yielded far more accurate data; for example, I found that more than 350,000 sites in 135 countries were linking to the updates.xcp-aurora.com site. While smaller than the original data set, I completed these new scans a month later (December 2005) after significant press and Sony’s own genuine efforts to manage the repercussions of the rootkit infections—for which it deserves credit and respect.

I still wanted to know whether the rootkit was a small-scale event or one of, perhaps, global magnitude. For 350,000 or 550,000 name servers to be so affected, the best available data suggested that Sony had indeed caused an event of worldwide proportions.

CONCLUSION
Hacking requires a mind-set that seeks to know not what a system is intended for but what it is capable of doing. Hacking is not limited to networks or to those who call themselves hackers; every programmer finds ways to repurpose old tools. White hats will never cede the creative ground to black hats. The greatest white-hat advantage—no need for stealth—exposes creative territory their adversaries cannot match. The results of using DNS as a global measurement platform can now be explored for the first time.

REFERENCES

Dan Kaminsky (dan@doxpara.com) is a senior security consultant at DoxPara Research, Seattle, WA.

© 2006 ACM 0001-0782/06/0600 $5.00
Software piracy is a worldwide phenomenon. In 2001, the Business Software Alliance (BSA) [1] found there are no nations with less than a 20% piracy rate, and two that exceeded 90% (see Figure 1). It is estimated that in 2001 software manufacturers lost $10.97 billion globally as a result of software piracy [1]. Furthermore, in most regions of the world, piracy increased by 37% in 2000 and by 40% in 2001 (see Figure 2). After showing a decrease for several years, software piracy appears to be on the rise.

The literature offers few explanations for the growth in software piracy. While a Software Publisher’s Association (SPA) study [1] found that per-capita income is not an issue in software piracy growth, Shin et al. [9] found evidence that the high price of software is a driving force in piracy in low GNP nations where residents cannot afford high-priced software. They noted that an increase in per-capita GNP is associated with a decrease in piracy rates. Other factors have also been suggested, including availability of pirated software, less stringent implementation of copyright regulations [8], moral problems with the copyright enforcement, the degree of IT infrastructure, and access to the Internet [3].

In this article, we introduce an approach consistent with previous research [4, 8, 9]. We also suggest some additional factors that may help explain software piracy growth and examine what changes in piracy rate determinants, if any, have taken place over the years.

**Why Do Piracy Rates Differ?**

Commonly assumed factors to explain piracy were grouped into four categories:

**Economic Factors.** Researchers have long recognized the importance of software price in piracy. Shin et al. [9] suggest that Gross Domestic Product (GDP) per capita is inversely related to software piracy level (more affluent nations have less need for piracy). Given the decrease in software prices, it may...
be that a country’s GDP has less of an effect on piracy than it did some years ago.

**Technical Factors.** Software piracy is assumed to be more prevalent in nations with low IT infrastructure since the quality (older editions with fewer features) of available software is lower. Thus, people often copy and work with pirated software. The problem is compounded as software revisions occur. While the need to remain competitive through upgrades is important, it is worthwhile only if affordable [8]. Many researchers also believe that software piracy has also flourished with the advent of the Internet [3] since some Web sites provide free software or at prices cheaper than the manufacturer’s [9].

**Regulatory Factors.** By imposing high tariffs, hence increasing the cost of software, governments may unwittingly encourage piracy. Developing nations often impose high tariffs on computer products and thus their piracy rates are higher. It is also believed that low censures for buying and high availability of pirated software are also reasons for piracy growth [8].

Copyright rules are considered foreign-induced, not easily understood, and far less rigidly enforced. Meso et al. [7] have identified enforcement of copyright laws as a salient issue in the development of a sound national IT policy. However, governments do not consistently implement such rules, partly because of lax law-enforcement facilities and institutional traditions that tend to ignore corruption. Thus, while enforcement of intellectual property rights and proper education may alleviate the problem, in reality few offenders are caught or prosecuted. Occasional raids from law-enforcement authorities have seldom disturbed the flow of pirated software. In nations that allow pirated software to be sold at lower prices, black markets have flourished. Individuals and firms in these nations may not even be aware that buying pirated software is illegal since the software has an aura of legitimacy. The overall effect of low censure and easy availability can be captured by the indicator of corruption. Corruption can be defined as “the cost of obtaining privileges that only the State can ‘legally’ grant, such as favoritism in taxation, tariffs, subsidies, loans, government contracting, and regulation” [5].

**Social/Cultural Factors.** These factors refer to the prevailing social structure of a country and the attitudes shared by members of that society. One measure of social structure is the distinction between individualism, a loosely coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves, and collectivism, a tightly coupled social network where people take care of themselves. Piracy is popular in collectivistic societies where people tend to create a psychological distance between members of the in-group and the out-group. Obligations or loyalty to in-groups are considered extremely important, and in exchange people expect that members of in-groups will look after them [6]. Out-groups, on the other hand, are not deemed worthy of respect, since they do not contribute to the general well-being of the in-group. In such societies, software purchased by an individual is expected to be shared among members of the in-group. Since most collectivistic societies tend to be third-world or developing countries, software producers in foreign nations are viewed as out-groups.

Piracy may also be more acceptable in low uncertainty-avoiding nations (that is, high risk-taking nations). Low uncertainty traits include less resistance to changes and breaking of rules, if needed. In high uncertainty-avoiding nations, there will be concern and anxiety in illegal use of software. People in high uncertainty-avoiding nations may feel more secure and

**The implication is that in order for piracy rates to decrease there must not only be a change in economic conditions, but also in societal structure and attitudes.**
comfortable with software obtained via legal means.

In a study of 53 countries (later extended to 69 nations), Hofstede [6] developed an index for rating Individualism/Collectivism (IC) and Uncertainty Avoidance (UA). This index remains the most highly regarded index for measuring social dimensions. For example, Japan has an uncertainty avoidance value of 92 (out of a maximum 100), The U.S. (91), the U.K. (89), and the Netherlands (80) and are considered very individualistic nations. Nations such as Ecuador (8) and Pakistan (14) and Sweden (29) are collectivist societies.

Attitudes about copying also play a major role in piracy. Unlike countries such as the U.S., where copying is generally equated with cheating, copying is often viewed a cultural exercise, not an immoral practice, in many Asian nations. Asian art students are required to diligently copy the work of masters as an exercise in learning. Success is measured by the similarity of the copy to the original. Copyright laws are not well understood or even recognized in many of these nations [10].

Even in developed Western nations, some people believe that pirating software is not wrong. The example of piracy in the music industry is worth noting in this context; many U.S. citizens who could otherwise buy a copy of a popular music CD regularly downloaded pirated music from sites such as Napster and Kaaza [3]. Since the price of a CD is frequently considered exorbitant relative to the cost of producing one, individuals in many uncertainty-embracing nations may feel that the injustices inflicted by the manufacturers justify piracy.

**Methodology and Assumptions**

The dependent variable, software piracy rate, was obtained from the BSA/SPA piracy study [1] which compared the difference between the demand for new software applications, and the legal supply of new software applications. PC shipments to major countries were estimated from proprietary and confidential data supplied by BSA/SPA member companies. The number of software packages installed per PC shipment was viewed as product demand. The number of software applications legally shipped formed the supply side. The difference between the demand and supply formed an estimate of software pirated. The piracy rate was defined as the amount of software pirated/total amount of software installed. The data was compared and combined to form a consensus estimate.1

To examine the impact of each of the explanatory variables on piracy, a partial least squares regression [2] on 37 nations was conducted for three periods of time: 1996, 2001, and 2003 (The results are similar with ordinary least square regression). This data constitutes the complete set of data available (see Table 1 for sources and expected relationships). The usual regression conditions were met using this scheme.

<table>
<thead>
<tr>
<th>Factors/Dependent Variable</th>
<th>Definition</th>
<th>Data Source</th>
<th>Expected Relationship to Piracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economic</td>
<td>GDP per Capita</td>
<td>World Bank</td>
<td>Inverse (-)</td>
</tr>
<tr>
<td>Technical</td>
<td>IT Infrastructure Internet Access</td>
<td>World Bank</td>
<td>Inverse (-) Positive (+)</td>
</tr>
<tr>
<td>Regulatory</td>
<td>Trade Regulations Enforcement (Corruption) IT Law</td>
<td>World Bank</td>
<td>Positive (+) Inverse (-) Inverse (-)</td>
</tr>
<tr>
<td>Social/Cultural</td>
<td>Social Structure Social Conservatism</td>
<td>Hofstede’s Index</td>
<td>Inverse (-) Inverse (-)</td>
</tr>
</tbody>
</table>

**Economic Factors.** GDP data is readily available from the World Bank (www.devdata.worldbank.org) and is generally assumed to be inversely related to piracy rates [4, 9], although other studies [1] found no relationships. We believe that GDP is an important factor, although its impact may be lessening.

**Technical Factors.** Determination of IT infrastructure was based on the availability of two main information technologies: PCs and the telephone.2 Internet access was determined by the number of Internet Ser-

---

1This procedure was verified and validated by several experts.

2A Principal Component Analysis for data reduction method was used for this purpose. The component extraction score was significant (98%).
vice Providers (ISPs). While software availability over the Internet might reduce the need for piracy, it could simply facilitate the ability to acquire legitimate software at reduced prices, and consequently, no assumptions can be made.\(^3\)

**Regulatory Factors.** Tariff rates can be captured by the trade policy index developed by the Heritage Foundation [5]. The Trade Policy Index (TPI) is based on a country’s average tariff rate. Since tariff rates and TPI are correlated and the TPI is a more general indicator, we decided to drop tariff rate in favor of the TPI to keep the regression parsimonious. We assume that piracy will be directly related to the TPI: the higher the TPI, the lower is the software piracy.

Transparency International/Göttingen University have generated a Corruption Index [11], or Corruption Perceptions Index (CPI), which reports perceptions of corruption (as seen by business people, risk analysts, and the general public) within a range between 10 (highly uncorrupt) and 0 (highly corrupt). The Black Market Index [5], based on the CPI, measures the prevalence of black market activities. Because of the correlation between the Black Market Index and the CPI, we decided to drop the Black Market Index from our analysis. Nonetheless, we assume the CPI will be directly related to piracy. We did, however, include ITLAW [12], (subject to the availability of data) as it might measure slightly different aspects of regulatory law.

**Social/Cultural Factors.** Using Hofstede’s [6] cultural dimensions, we postulated that piracy rates will be negatively related to a nation’s IC rating and its UA rating (a high IC rating indicates high Individualism; a high value of UA index denotes high uncertainty avoidance).

### Findings and Discussion

Table 2 provides a summary; prior to any discussion, it should be noted that while all available data was used, the sample size is limited. Consequently, those relationships that were not found to be significant at generally applied levels, may become significant should more data become available.

**Economic Factors.** GDP was found to be a significant indicator of piracy in 1996 but not in 2001 or in 2003. Further, GDP alone could explain only 62%–63% of the variance in piracy. These findings may help explain why different studies have arrived at different conclusions about the role of GDP. It appears that the decrease in relative cost of software (and in some cases the absolute price of software), has lessened the need for piracy. Given the increase in piracy over this period, however, it follows that other factors are involved, and further that these factors are increasing in impact.

**Technical Factors.** Corresponding to the impact of GDP on piracy, IT infrastructure was inversely related to piracy (the greater the IT structure, the less the piracy rate) in 1996, but not in 2001 nor 2003. Regardless of period, Internet growth rate was not found to be a significant predictor of piracy. Internet use, however, emerged as significant in 2003.

In order to examine whether the relationship between Internet use and piracy rates differs in nations with high piracy rates, a separate analysis was performed using only those nations with high piracy rates (piracy rate \(\geq 50\%\)). The findings were insignificant, which intuitively follows, since Web site content is the same regardless of point of access.

These findings offer a few possible explanations. It may be that the increased availability of legal software over the Internet, especially given the decrease in software prices, lessens the need for piracy. This corresponds to the relationship between GDP and piracy.

---

\(^3\) Software purchased over the Internet may or may not be legitimate; we assume that it is, although we concede the possibility of pirated software being sold.

---

**Table 2. Regression results between piracy rate and explanatory variables.**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GDP</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>Significant</td>
<td>NS</td>
<td>NS</td>
<td>Partly significant</td>
</tr>
<tr>
<td>IT Infrastructure</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Significant</td>
<td>NS</td>
<td>NS</td>
<td>Partly significant</td>
</tr>
<tr>
<td>Internet use</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>Trade Regulations</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Significant</td>
<td>NS</td>
<td>NS</td>
<td>Partly significant</td>
</tr>
<tr>
<td>ITLAW</td>
<td>-</td>
<td>N/A</td>
<td>N/A</td>
<td>-</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Significant</td>
</tr>
<tr>
<td>Corruption</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Significant</td>
<td>Significant</td>
<td>Significant</td>
<td>Significant</td>
</tr>
<tr>
<td>Cultural Collectivism (IC)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Significant</td>
<td>Significant</td>
<td>Significant</td>
<td>Significant</td>
</tr>
<tr>
<td>Cultural Uncertainty avoidance (UA)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NS</td>
<td>Significant</td>
<td>Significant</td>
<td>Partly significant</td>
</tr>
</tbody>
</table>

\(^a\) Number of nations available \(=37\); NS=not significant; --Not included

\(^b\) If significant, significance level is 0.10 or better
Regulatory Factors. Once again, trade regulation was found to have a significant negative impact on piracy in 1996, but not in 2001 nor in 2003. There is no easy explanation for this finding. It may be that the regulations themselves have changed. It may be that the increased availability and the decreased costs of software have lessened the need for regulations. Additional investigation is needed.

Consistent with our assumptions, a high degree of corruption (low corruption index value) was positively correlated with a high rate of piracy and was significant in both the initial and in the final two regressions. Not only are rules and regulations less rigorously enforced in highly corrupted societies, but there appears to be a preference for pirated software.

ITLAW was significant in 2003 (the Pearson’s correlation coefficient with software piracy was also significant: r=-0.68, p<.000). This may be due to better law enforcement in recent years in many of the nations studied.

Social/Cultural Factors. A high collectivism rating was found to be positively related to piracy. This was anticipated since collectivistic societies tend to regard foreigners as outsiders [6] and most of these programs are exported from the U.S. Interestingly, piracy rates were higher in high-uncertainty avoiding nations in years 2001 and 2003, but not in 1996. In the final regression, the additional variance explained by these indicators amounted to 13%–18%, after controlling for GDP.

CONCLUSION
This preliminary study provides some empirical results on the causes of global software piracy. Indicators in four categories of factors (economic, technical, regulatory, and social/cultural) were examined and a combination of a few indicators together explained the vast majority of variance in software piracy data. Nations with less corruption and weak collectivism had less piracy. These factors remained significant over time. Factors such as strong economic growth, low trade regulations, high uncertainty avoidance, low Internet use, better ICT laws, and strong IT infrastructure had partial influence on low piracy, within the timeframe of the study.

The study indicators together explained 78% (1996) to 84% (2001) to 83% (2003) of the variance in software piracy data. It is obvious that more than mere economic factors are involved, and it would appear that non-economic factors, or a combination of such factors, are much more relevant in predicting piracy. This might help to explain why piracy is prevalent even in developed nations. The implication is that in order for piracy rates to decrease there must not only be a change in economic conditions, but also in societal structure and attitudes.

Nonetheless, additional investigation is needed. As noted previously, the data available is limited, although it is expected that additional data should be available in the next few years. There may also be additional social/cultural factors that impact on piracy (such as trust), and variations in growth and product elasticity across nations could be significant. Combinatory interactions between factors also need to be investigated. For example, it may be that Internet usage and collectivism together are better predictors of piracy than either one alone.

This study was also limited to a five- to seven-year observation period: as more data becomes available, additional observations may offer new insights. It also seems reasonable to assume that as a result of changing legislation and shifts in attitudes toward piracy, we can expect the factors affecting piracy to change. Longitudinal studies mapping the changes over time might prove enlightening with respect to constructing policies to discourage piracy.

References

KALLOL BAGCHI (kbagchi@utep.edu) is an assistant professor in the Department of Information and Decision Sciences in the College of Business at the University of Texas at El Paso.
PEETER KIRS (pkirs@utep.edu) is an associate professor in the Department of Information and Decision Sciences in the College of Business at the University of Texas at El Paso.
ROBERT CERVERNY (cerveny@fau.edu) is a professor in the Department of Information and Decision Sciences in the College of Business at the Florida Atlantic University at Boca Raton.
Seeking a mutually beneficial relationship between IS departments and users.

Do unto others as you would have them do unto you: This maxim is often referred to as the Golden Rule. Simply put, in relationships, we get what we give. Relationships are important because they possess the advantage of maintaining a bond over time and providing continuity through the ups and downs of individual good and not-so-good events. The quality of the relationship is determined by beliefs held by the parties in the relationship and has an effect upon the future actions taken by each party to the relationship. In this article, I examine the relationship between information systems (IS) departments and IS users with the goal of determining some of the beliefs affecting that relationship and how they might be managed to the mutual benefit of IS users and IS departments.

Data from 22 IS user and 22 IS manager interviews I’ve conducted in 11 large organizations over a one-year period suggests there is a bias toward the “hard” functional and away from the “soft” relational goals in user/IS department employee interactions. The typical IS department often favors the immediate “technological” aspects of each project, ignoring the longer-term “psychological” relational factors. Relational factors (such as trust, satisfaction, and user commitment) provide the glue holding together the relationship between IS users and the IS department. Neglecting the cultivation of positive relational factors can create tension between the IS department and users, resulting in some instances in negative user behaviors vis-à-vis the IS department.

Interview data I’ve collected suggests that in some cases it is possible that neglecting the relational component can result in user distrust, dissatisfaction, defection, and if severe and widespread enough, the dissolution of the IS department in favor of outsourcing. In order to prevent these negative user reactions from harming the IS department, IS managers should detect, understand, measure, and manage factors that influence IS user-depart-
mental relationship quality. Recognizing, monitoring, and working to improve these factors, once they are discovered, places the IS department in a much stronger position capable of managing IS user/IS department relationship quality and thereby managing the quality of reciprocal user behaviors and perhaps, ultimately, its own destiny. This study provides insight into the nature of some relationship quality enhancing/reducing factors and how they can be managed.

In general, the relationship concept is based on the principle or norm of reciprocity. The principle of reciprocity states that people often match behaviors experienced from others with actions performed for others, giving in proportion to what they receive. According to the reciprocal action theory, actions taken by one party in an exchange relationship will be reciprocated in kind by the other party [2]. In the IS realm this means IS departmental actions will be evaluated and reciprocated by IS users based on evaluations of what they receive from the IS department.

The phrase “relationship marketing” was coined by Leonard Berry [6] and applied to the marketing of services, both external and internal to the organization [5]. Relationship marketing is defined as the ongoing process of engaging in cooperative and collaborative activities and programs with immediate and end-user customers to create or enhance mutual economic value at reduced cost [5].

IS services are those activities and programs undertaken by the IS department to support users in their use of information systems. A conceptual model of the relational evaluation and reciprocation process undertaken by a typical IS user is shown in Figure 1. On the left in Figure 1 are the functional or “hard” IS services performed for users such as systems development, systems configuration, systems maintenance. Also on the left, the relationship building or “soft” activities undertaken by the IS department to build and maintain relationships with IS users are shown. These “soft” behaviors can be as simple as proactively explaining IS departmental goals, getting to know users by name, keeping individual promises, evincing a willingness to help users, and promoting trust with users. Both sets of activities (hard and soft) are evaluated by IS users by comparison with real or imaginary standards. Judgments of the quality of the relational bond between the IS department and IS users are then made by the users.

Finally, based on the perceived quality of the relationship, IS users decide on their beliefs and behaviors vis-à-vis the IS department.

Interview data suggests that if the relationship is positive, the IS user behaviors will be cooperative and supportive such as positive word of mouth, voluntary participation in IS development projects, and even championing the IS department in management decisions. If the relationship is judged to be negative, the IS users will engage in negative behaviors vis-à-vis the IS department, such as complaints, indifference, and in extreme cases, sabotaging the IS department activities, and ultimately requests for outsourcing all or parts the IS department. The question of interest in this study is, does this model hold in the real world? Can delivery of better IS service quality create stronger IS department/user relationships and thus more positive reciprocal behaviors from users? (See the sidebar and Table 1.)

The variables and their associations were tested by fitting the structural equation model shown in Figure 2 to the data. The LISREL 8.54 program was used to

**METHODS AND DISCUSSION**

A survey instrument containing items assessing the variables shown in the table here and modeled in Figure 2 was administered to a random sample of 300 employees of a large Midwestern U.S. health care provider. Some 238 usable responses were obtained from a representative cross-section of the employee population. The first four measured variables in the table represent IS service quality: reliability, responsiveness, assurance, and empathy. These are found in the IS-Adapted SERVQUAL instrument [9]. The next three variables represent relationship quality: satisfaction, trust, and commitment to the IS department [8]. The final two variables represent IS user beliefs and behaviors vis-à-vis the IS department: user identification with the IS department and voluntary participation in IS departmental activities [4].
C onstruct validity and reliability were successfully assessed through the framework for organizational research identified in [3]. This framework involves sequential assessment of construct reliability, discriminant and convergent validity, and nomological validity for each construct or concept in the model. The model demonstrated good reliability, discriminant and convergent validity. Nomological validity was also good as evinced by the positive and significant structural model in Figure 2. The isolated arrows shown in Figure 2 can be thought of as correlations among the variables, whereas the $r^2$ numbers reflect the amount of explained variance in a dependent variable caused by one or more independent variables.

From the model in Figure 2 as fit to the data we can make several inferences. Clearly, IS users base their evaluations of IS service quality in large part on the following service quality factors: The ability to perform the proposed IS service dependably and accurately; the willingness of the IS departmental employees to help IS users and provide prompt IS service; the knowledge and courtesy of IS employees and their ability to inspire trust and confidence; and the level of caring, individualized attention the IS service provider gives its users.

User perceptions of the four service quality variables combine to form user's beliefs in the extent to which they can trust the IS department and the level of overall satisfaction users feel with the IS department. Trust and satisfaction beliefs taken together with user commitment to the IS department represent relationship quality. Relationship quality beliefs are strong predictors of the level of user's reciprocal behaviors toward the IS department. The reciprocal behaviors match the user's feelings of trust in, satisfaction with, and commitment to the IS department. Satisfied, trusting users will identify with and commit to enacting positive, supportive behaviors toward the IS department. Interview data suggests that dissatisfied, distrustful, hostile users will be alienated from the IS department, and may enact negative behaviors toward the IS department such as complaining, retribution, and sabotage. Several interviews confirm that the level of relationship quality and commitment users experience may even have a determinative effect on outsourcing decisions.

It can be concluded from this research that IS departments can do a better job of internal marketing to keep users satisfied, trusting, and committed to their goals. The next question of importance is: How
can IS departments improve the quality of their relationship with IS users? Or, how can IS departments become better internal marketers of their services? These questions are answered in the elaboration of the following best practices for internal marketing of IS services.

**Discussion and an Inventory of Relationship-Building Best Practices**

IS users reciprocate the IS departmental behaviors, good for good, bad for bad. Relationship quality is the glue that binds IS users to IS departments through the thick and thin of IS project success and failure. The following are relationship-building best practices taken from 22 IS manager and 22 IS user interviews in 11 distinct business organizations and placed in a relationship marketing framework developed to describe relationship-building best practices in any service marketing situation. Claycomb and Martin developed the framework from research in which they conducted 83 interviews with managers of service firms focusing on finding the most important relationship-building practices. They then surveyed 205 managers to finalize the framework [7]. This framework, when contextualized, exemplifies an elaborated Golden Rule for IS departments. In the discussion that follows, it should be kept in mind that each of the internal marketing “best practices” involve trade-offs among service relationship quality-building actions, functional actions, and costs in person-hours. Not all of the following relationship best practices will apply in every situation or context.

**Avoiding lengthy gaps in contact between IS departments and IS users.** IS users remember the IS department that remembers them. In other words, the frequency of both formal and informal contact is important. Examples of different types of contacts include IS departmental email newsletters to keep users informed about updated capabilities, new products, new people, technologies trends, and so forth. Other methods of contact include regularly scheduled informational email messages and telephone calls, development and use of focus groups, regularly visiting informally with users, IS departmental Web sites, and quarterly and annual IS user conferences. One of the strongest complaints from users involves the lack of contact between the IS department employees and IS users.

**Improving service performance.** Although promises of quality may attract IS users, IS departments must realize that delivery of service quality is essential to building and maintaining IS user relationships. Delivering on promises is the essence of mutually satisfying service relationships. Service quality refers to the consistency with which IS user’s expectations are met and the general superiority of the service relative to that of possible substitutes. IS departments must take the initiative including the adoption of any practices focused on eliciting and identifying which services and service attributes IS users want (doing the right things). Examples of services include assisting with all the small sub-actions needed in purchasing, configuring, and maintaining user’s information systems or suggesting shortcuts to improve user efficiency with their information systems. These services must be provided to IS users at a level that gives satisfaction, certainly better than potential substitutes (doing things right). Efforts to raise standards and improve service performance, listening to IS user’s preferences, and ensuring that IS user’s requirements are met are included in this category.

**Providing consistent, fair, and reliable service.** Providing friendly, professional, courteous service that is consistent, fair, and reliable is one of the best ways to establish and maintain IS department/IS user relationships. This is exemplified by being on-time, supplying a wide range of systems and services, having a knowledgeable staff, and providing technical competence. Service quality also includes listening to IS users—knowing the user and understanding IS users’ needs.

**Treating each IS user as unique.** Personalization refers to the customization of some aspect of the service or its delivery, treating each IS user as a unique individual with a unique set of service requirements—thereby creating unique fits between IS users and IS services. Personalization initiatives provide direct linkages between IS users and service personnel and between IS users and services themselves.

**Personalization practices occur on three different levels: interpersonally, operationally, and organizationally.** Examples of interpersonal aspects include learning and using IS user’s names, building rapport by encouraging face-to-face contact between IS departmental employees and IS users, “getting to know” IS users informally, and acknowledging IS user’s backgrounds and achievements.

The operational level involves efforts to obtain a detailed knowledge of IS user’s tasks, processes, and requirements. This allows the IS department to provide unique ideas to help users do their jobs more effectively. Additionally, IS departmental employees should be given the flexibility to deviate from rigid procedures when serving IS users who have special needs or unique requests.
At the organizational level, personalization can be enhanced by assigning IS departmental employees the responsibility of serving specific IS users rather than having them assigned the responsibility of performing specific tasks for all users. This encourages close personal relationships between the IS departmental representatives and the IS user. Full implementation of personalization requires top management to structure the IS department in order to assign IS departmental employees the responsibility of serving specific IS users, and to empower IS departmental employees to treat each IS user as a unique individual. There will obviously be a trade-off between the seeming efficiency of assigning specific tasks to IS departmental employees and assigning specific sets of users. The point is raised here to highlight the balancing act involved in boosting relationship quality.

**Evoking emotional responses.** Affective engineering is the label often assigned to a range of efforts designed to evoke IS user’s emotional responses to make them feel “warm and cozy” in their relationship with the IS department. Our survey and interviews suggest that IS user’s affective commitment (emotional attachment to the IS department) is positively related to their willingness to remain in a positive, reciprocal relationship with a IS department. If that is the case, then any activities that promote trust, satisfaction, and commitment to the IS department will be useful in developing IS/user relationships.

**Providing easy and convenient contacts with the IS department.** Systems friendliness refers to practices that make it easy and convenient for IS users to interact with the IS department. This involves making IS department representatives accessible, removing contact barriers, ensuring the IS department’s interfaces with technology are not overwhelming, and not making IS users wait for service unnecessarily or perform tasks (for example, fill out paperwork or Web forms) they would rather avoid. Understandably, IS user relationships can suffer when unfriendly systems leave IS users feeling frustrated, unwelcome, and convinced there must be better substitutes for service providers. Ideally, system interfaces should leave IS users with a sense of looking forward to interacting with the IS department in the future.

**Delivering on promises of service.** Trust is often described as the cornerstone of strong relationships. Trust is an important element of a relationship-building program, because it builds confidence, fosters cooperation, and gives the IS department a second chance when inevitable mishaps occur. Trust is particularly relevant, because IS users often do not obtain services per se. What they obtain are implicit and explicit promises of IS service that must be trusted to

Our interview data suggests that failure to recover from an initial problem creates unhappy IS users who try to ignore the IS department and get help from another source, such as colleagues, knowledgeable friends, or spouses.
be kept when the time comes. For example:

- Promises that the IS departments will honor service-level agreements, must be trusted in that it will occur in the future;
- Promises that IS departments will install, configure, maintain, and upgrade systems once provided in the future;
- Promises that IS departments will provide security for systems and alert users immediately when they are compromised.

Trust-related concepts such as trustworthiness, honesty, integrity, or ethical behavior are important relationship-building practices. More specifically, IS departments build trust by keeping their commitments, not overpromising (such as keeping claims and predictions of product and system success realistic), managing IS user’s expectations (such as clearly clarifying what IS service does and does not entail), maintaining open channels of communication with IS users so that misunderstandings and mishaps can be quickly identified and remedied, and establishing codes of ethics and training to promote trustworthy actions throughout the organization.

**Establishing Long-term Management Commitment**

Many of the continuity, communications, and frequency programs that link IS users to the IS department require a long-term management commitment. Not only do many of the more elaborate programs require establishing and maintaining an extensive database to keep track of each IS user’s status in the program, but also typically the commitment must be sustained over a much longer time period than a few individual transactions. For instance, if a user has a problem that occurs frequently, it is of vital importance to develop a profile of the user and the problem and not simply fix the problem repeatedly each time it occurs. A continued problem may mean the user needs more training in some area of the use of the information system. Alternatively, it could mean an application program or the operating system needs patching. In either case there needs to be an institutional memory (database) of the pattern of problems for each user. These patterns should be able to be tracked across users and larger patterns developed so that standards for solving problems can be constructed.

**Improving Service Recovery**

For most IS service operations, there are simply too many details involved to expect flawless operations at all times. And, unfortunately, when mistakes do

Furthermore, unhappy users will tell other users about their dissatisfaction. In other words, good service recovery promotes good word-of-mouth, bad service recovery promotes bad word-of-mouth among users.
occur, it is not uncommon for IS users to find out before IS departments. While efforts to minimize occasional (but inevitable) mishaps are appropriate, efforts to enable both the IS department and IS users to recover from the errors are needed as well. For example, a user has an issue with a personal information system and calls the help desk. The help desk attempts a solution, is unsuccessful, and then stops working on the problem without solving it. This is an example of poor service recovery. All service recoveries must result in solutions at least to the point that users can continue to work with their systems.

Our interview data suggests that failure to recover from an initial problem creates unhappy IS users who try to ignore the IS department and get help from another source, such as colleagues, knowledgeable friends, or spouses. Furthermore, unhappy users will tell other users about their dissatisfaction. In other words, good service recovery promotes good word-of-mouth, bad service recovery promotes bad word-of-mouth among users. This situation can result in an apparent reduction in user problems or trouble-tickets. The IS department may view this demand reduction as a positive thing. When in fact, the IS department is performing so poorly that users will attempt to get help anywhere else.

Service recovery involves practices that IS departments can use to aggressively correct mistakes when they occur and offset IS user’s inconveniences and other negative consequences caused by these mistakes. Often, through service recovery, service failures can be transformed into positive acts that strengthen customer attitudes toward the service provider [10]. In fact, IS user satisfaction with the process of service recovery may be more important than the initial service attributes in influencing overall IS user satisfaction, user/department relationship quality, loyalty intentions, cooperation, and positive word-of-mouth communication by IS users.

RECOGNIZING THE IMPORTANCE OF RECIPROCITY

Building and maintaining IS-user relationships may mean accepting some risk, trusting IS users, and giving IS users something of value without any certainty of receiving any sort of reciprocal consideration or commitment. Free technology, training, diagnoses, advice, and personal or confidential disclosures are examples of things of value that can be given by IS department employees to individual users to increase user commitment to the IS department. Although somewhat risky, these practices are based on the well-established equity norm or principle [7]. People frequently do feel compelled to reciprocate in some way after someone else has helped them or given them something of value. IS departments must take the difficult first step in this commitment process.

CONCLUSION

The central idea presented in this article is that IS departments must learn from relationship marketing research and take that “extra first step” to initiate, maintain, and enhance IS department/IS user relationships. This will involve changes in mind-set for IS managers who typically emphasize the “hard” technical and short-term project-oriented aspects of the job at the expense of the long-term, relational, “soft” social aspects. IS managers will need to be better internal service marketers in order to develop and maintain strong, committed, supportive users.

REFERENCES


Christopher L. Carr (dclarr@katz.pitt.edu) is an assistant professor of business in the management information systems interest group at the University of Pittsburgh’s Katz Graduate School of Business.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2006 ACM 0001-0782/06/0600 $5.00
Who commits software misuse? Knowing the answer to this question will help organizations protect their information systems.

“Professor, can you help me? I installed software on my computer from my friend’s CD and it doesn’t work anymore. My friend is gone and I don’t have the original CD. What should I do?”

Does this statement sound familiar? If so, you are not alone. Many students openly admit to illegally installing software on home computers or otherwise misusing computer information systems. Other studies have examined characteristics of students (and non-students) who admit to committing information systems misuse, piracy, and computer crime. We used a survey to examine demographic characteristics of students as well as their awareness of university computer usage policies.

Thirty-four percent (34%) of students responding to this survey admit to committing some form of software misuse or piracy and 22% admit to committing data misuse during their lifetimes. Knowing that students commit information systems misuse is not new [10]. However, today’s students are tomorrow’s professionals. As such, an understanding of the demographic factors common to those students who commit misuse could help both university information systems departments and organizations better protect their information systems.

An amazing number of students in this study report committing some form of information systems misuse or computer crime. As mentioned, 34% and 22% of respondents admit committing software misuse and data misuse during their lifetimes, respectively. Software misuse in this study means destroying or copying software, using copied software, or distributing copied software without permission. Data misuse means accessing, modifying, or copying data stored on a computerized information system without authorization. Behaviors characteristic of misuse were located in the literature and condensed into these two areas. This study examines these responses by years of experience with computers, academic classification (underclassmen, upperclassmen), major, gender, and age.
Familiarity with computers. As expected, students who are more familiar with computers report committing more misuse. Underclassmen, students with more experience, and students in computer-related majors all report committing more misuse than others. An interesting trend appears (Table 1) when broken down by academic classification. Underclassmen (freshmen and sophomores) report the least amount of software misuse (18%), while upperclassmen (juniors and seniors) report notably larger amounts (37%). Overall, 34% of respondents report software misuse; 7% report 10 or more occurrences. A similar pattern also is observed for data misuse, although fewer incidents of misuse are reported (underclassmen, 12%; upperclassmen, 25%). Overall, 22% of respondents report data misuse; 3% report 10 or more occurrences.

Further, of greater concern, individuals who indicate reading the computer usage policies also report more software misuse and data misuse. For example, of underclassmen who read the computer usage policies, 39% committed software misuse and 8% admit 10 or more occurrences. Of underclassmen who had not read the policies, 15% committed software misuse and 2% report this level of misuse. This unexpected and troubling result can be observed for both software misuse and data misuse in most academic classifications.

Years of experience with computers also are thought to influence misuse [10]. As seen in Table 2, respondents with greater experience report greater numbers of misuse. For example, all individuals with less than one year experience indicate no software misuse during their lifetimes, whereas 41% of individuals with more than 14 years experience make the same claim. Also, 78% of individuals with less than one year experience indicate never engaging in data misuse. This percentage drops to 61% for individuals with more than 14 years experience.

<table>
<thead>
<tr>
<th>Software Misuse: How many times have you destroyed or copied software, used copied software, or distributed copied software without permission during your lifetime?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage Policy</strong></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>All</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Read Policy</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Did Not Read Policy</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Misuse: How many times have you accessed, modified, or copied data stored on a computerized information system without authorization during your lifetime?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage Policy</strong></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>All</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Read Policy</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Table 1.** Classification and familiarity with the university computer usage policy.

Upperclassmen, students with more experience, and students in computer-related majors all report committing more misuse than others.
Misuse by major is presented in Table 3. As one might expect, computer information systems (CIS) majors report the most software misuse with 24% of CIS majors performing 10 or more instances within their lifetimes. This percentage does not exceed 8% for another major. Examining the percentages of individuals who report no misuse presents a similar view. Forty-nine percent of CIS majors indicate they never committed software misuse, while 57% of arts and science majors and 71% of business and economics majors make the same claim. Further, 73% of CIS majors, 78% of business and economics majors, and 83% of arts and sciences majors deny ever committing data misuse.

**Gender and age.** Other factors examined by this research include gender and age. As anticipated, males commit more misuse than females, while individuals in their twenties and thirties commit more misuse than other age groups. Gender often is associated with increased misuse [10]. Fifty-five percent of males and 76% of females report no instances of software misuse, and 13% of males and only 2% of females report committing 10 or more software misuses. Further, 69% of males and 86% of females never committed data misuse, while 6% of males and less than 1% of females report committing 10 or more data misuses. However, the aforementioned percentages change dramatically when broken down by familiarity with computer usage policies. For example, of respondents who read the policies, the percentage committing 10 or more software misuses increases to 18% of males and 5% of females. Whereas, of those who do not read the policies, only 10% of males and 2% of females report this much software misuse. And, of those who read the policies, the percentage committing 10 or more data misuses increases to 7% of males and 2% of females. Of respondents who do not read the policies, these percentages are 5% of males and less than 1% of females.

The final demographic factor examined in this study is age. Thirty-five percent of respondents under 40 and 39% of respondents 40 and older report committing software misuse during their lifetimes, while 22% of respondents under 40 and 17% of respondents 40 and older report committing data misuse. However, the highest frequency of misuse occurs within the younger groups. Nine percent of respondents under 20 and 8% aged 21 to 29 report committing 10 or more lifetime software misuses, as compared to 4% each of respondents 30 to 39 and 40 and older. (Of the 509 usable responses to this question, only 23 respondents are 40 and older.) For data misuse, 3%, 3%, 2%, and 9% of respondents less than 20, 21 to 29, 30 to 39, and 40 and older, respectively, report 10 or more instances. However, these results must be interpreted with caution as this survey was administered to college students, and thus is biased toward younger respondents.

### A Widespread Problem

Other studies have evaluated the prevalence of information systems misuse and computer crime by university students. A recent study notes 40% of students surveyed at two universities admitted to committing software piracy [3]. Further, none of these students were worried about punishment for their actions [3]. In a survey of 581 students at a southern university, 41% “knowingly used, made, or gave to another person a ‘pirated’ copy of commercially sold computer software” at some time in the past, while 34% did so during the past year [10]. Further, 18% “accessed another’s computer account or files without his or her knowledge or permission just to look at the information or files,” while 7% “added, deleted, changed, or printed” information from another’s files without permission. Finally, 21% guessed passwords in attempting to access another student’s accounts or files. In another study, 10% of respondents committed software misuse during the prior semester [5]. These misuse figures are very close to those generated within the present research, which indicate 34% of respondents committed software misuse during their lifetimes, while 22% committed data misuse sometime during their lifetimes.

The demographic results of the present study are
also very similar to the results of past research. For example, males over 22 years old, enrolled as seniors or graduate students, were most likely to report committing misuse [5]. Further, misuse was especially common among majors dealing with forestry, engineering, business, liberal arts, and the sciences [5]; and misuse was more prevalent among computer science and engineering students, especially those in upper-level classes [3]. As previously noted, this research suggests males commit more misuse than females, as do students majoring in CIS.

Although the three universities discussed within this article publicly post computer usage policies (two of the universities insist students read these policies before email accounts are activated), only 24% of the respondents report having actually read the computer usage policies. Of these, 62% indicate reading the policies more than one year before the survey. Also, respondents who indicate reading the policies report higher levels of misuse.

These findings present an interesting challenge to universities: should additional resources be expended to familiarize all students with the university computer usage policies? The majority of students are unfamiliar with the university computer usage policies; however, students who are familiar with the policies report committing more misuse. Although an explanation of this unexpected result is beyond the scope of the current research, some possible explanations can be identified. For example, students who commit misuse could be more interested in reading the university computer usage policies than students not committing misuse. A second alternative might involve the university computer usage policies acting as a challenge to students and thus increasing the performance of misuse.

Until further research clarifies this matter, university computer security administrators must reconsider the methods used to educate students as to acceptable and unacceptable uses of university computing resources. This research clearly demonstrates that the majority of students are unfamiliar with the rules guiding their usage of university computing equip-

<table>
<thead>
<tr>
<th>Major</th>
<th>Ten or More</th>
<th>Seven to Nine</th>
<th>Four to Six</th>
<th>One to Three</th>
<th>Never</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arts and Science</td>
<td>4.3%</td>
<td>2.2%</td>
<td>8.7%</td>
<td>28.3%</td>
<td>56.5%</td>
<td>46</td>
</tr>
<tr>
<td>Business and Economics</td>
<td>3.9%</td>
<td>3.3%</td>
<td>6.1%</td>
<td>15.6%</td>
<td>71.1%</td>
<td>360</td>
</tr>
<tr>
<td>CIS</td>
<td>24.4%</td>
<td>2.6%</td>
<td>3.8%</td>
<td>20.5%</td>
<td>48.7%</td>
<td>78</td>
</tr>
<tr>
<td>Other</td>
<td>7.7%</td>
<td>7.7%</td>
<td>15.4%</td>
<td>19.2%</td>
<td>50.0%</td>
<td>26</td>
</tr>
<tr>
<td>Undecided</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Software Misuse:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major</th>
<th>Ten or More</th>
<th>Seven to Nine</th>
<th>Four to Six</th>
<th>One to Three</th>
<th>Never</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arts and Science</td>
<td>5.0%</td>
<td>2.5%</td>
<td>10.0%</td>
<td>82.5%</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Business and Economics</td>
<td>1.9%</td>
<td>1.9%</td>
<td>4.4%</td>
<td>14.2%</td>
<td>77.7%</td>
<td>367</td>
</tr>
<tr>
<td>CIS</td>
<td>9.1%</td>
<td>3.9%</td>
<td>2.6%</td>
<td>11.7%</td>
<td>72.7%</td>
<td>77</td>
</tr>
<tr>
<td>Other</td>
<td>3.8%</td>
<td>3.8%</td>
<td>7.7%</td>
<td>84.6%</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>Undecided</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td><strong>Data Misuse:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Misuse by major.

ment. Perhaps repeated exposure would be more effective.

These unexpected results challenge the long-held belief that university computer usage policies prevent or limit the performance of misuse. Since organizations also utilize computer usage policies, the concern generated from these findings must be extended from the university setting to the organizational setting.

Although the use of student samples raises questions of representativeness and generalizability, in this case the students are valid users of the computing resources of these organizations. Users are defined as “individuals who interact with the system regularly” [11]; students utilizing university computers meet this definition of a user. From a technological standpoint, universities and other organizations share the same types of technology and the same risk factors. Universities must utilize the same methods as other organizations to protect themselves. In addition, universities may face even greater threats than the typical business organization. Since the computers in a classroom or lab are open for public use, tracking an instance of misuse usually leads back to the computer rather than the user. Further, university networks are often more vulnerable than corporate networks due to the need for collaboration and easy access to data [8].

The target population for this study is university students. The sample consists of 519 students enrolled in junior- and senior-level business courses at three Midwestern U.S. universities. The universities (and courses) were selected based upon the willingness of colleagues to participate in the study. Although this sample does not represent all students enrolled at these universities, this sample was deliberately chosen to maximize the potential for reported misuse conducted by the subject students. Students from arts and science colleges, business and economics colleges, and engineering colleges commit more misuse than other students [5].

All three universities utilize computer usage policies that outline acceptable and unacceptable use of computer systems. Each university also posts the policies on its Web site; two universities require their stu-
dents to read these policies before email accounts are issued. The use of such policies has been linked to lower levels of misuse, while failing to use them has been linked to misunderstanding of correct use and thus to misuse [12].

The survey questionnaire was constructed by combining Straub’s Computer Security Model Victimization Instrument [12] and items from instruments focusing on Ajzen’s Theory of Planned Behavior [1]. The items based on Ajzen’s Theory of Planned Behavior were customized to two specific areas of interest: software misuse and data misuse.

CONCLUSION

Although concern with information systems misuse and computer crime is not new [10], it is of growing concern to commercial organizations [4] and the military [9]. Moreover, information systems misuse, piracy, and computer crime are international in scope. Reports suggest that the frequency of misuse is increasing rapidly [2]. Further, the cost of misuse is extremely high. A recent survey reports that respondents estimated losses of $141,496,560 during 2004. However, only 269 of 494 respondents were willing to report estimated dollar losses [4]. The actual loss is probably greater than stated since estimates only include recognized losses, and many organizations elect not to report losses for fear of negative publicity [4, 7].

Many organizations are so dependent upon their information systems that disruptions or failures often result in severe consequences that range from inconveniences to catastrophes such as complete organizational failure [6]. In addition, access to organizational information systems through networks and dial-in accounts leads to an extremely vulnerable environment [6]. This same situation may be found in universities around the country. Campus networks are becoming “an alluring target for hackers” and, possibly, terrorists [8].

Several researchers have reported that three-fourths or more of computer security violations by humans could be attributed to insiders or other trusted individuals, although current research suggests this trend may be changing. The 2004 CSI/FBI Computer Crime and Security Survey notes that about half of all reported incidents originate within the company, while half are external [4].

This research confirms past conclusions: students commit misuse and pirate software. Students possessing greater familiarity with computers report committing greater amounts of misuse. Individuals with certain majors, such as CIS, tend to commit more misuse than others. In addition, individuals with more computer experience tend to commit greater amounts of misuse than novices. Also, more misuse occurs by upperclassmen than by underclassmen. Finally, males commit more misuse than females, and individuals in their twenties and thirties report more misuse than other age groups.

However, the results of this research also suggest university computer usage policies are not effective in preventing students from committing misuse. First, the majority of respondents never read the computer usage policies at their universities. Second, students who read the policies report committing more misuses than those who do not read the policies. This unexpected result, which disagrees with past findings, suggests the need for continued research in this area.

Both of these results are particularly concerning as many organizations utilize written policy statements to explain proper and improper use of organizational information systems. It is thought such policies reduce the occurrence of misuse within an organization. Future research should address the issue of familiarity with computer usage policies. Given that the majority of respondents have not read the policies despite being required to do so by their respective universities, a method to enforce exposure to computer usage policies must be found.

Controlling misuse has been a concern in the MIS literature since the early 1960s [10], however, many organizations and critical systems are still vulnerable, especially as the modern computer environment incorporates ever-increasing amounts of networking and Internet connectivity. Existing research suggests

**THE RESULTS OF THIS RESEARCH ALSO SUGGEST UNIVERSITY COMPUTER USAGE POLICIES ARE NOT EFFECTIVE IN PREVENTING STUDENTS FROM COMMITTING MISUSE.**
organizations can defend themselves against such misuse by using computer usage policies. Unfortunately, the results herein, as well as simple observations of ever-increasing amounts of misuse, suggest these policies are ineffective. As a result, organizations need to consider other methods of protecting themselves. The first problem noted in the current research is a lack of familiarity with computer usage policies. Perhaps organizations need to enforce exposure, rather than relying on the user to read the policies. Further, repeated exposure could increase user retention of computer usage policies. The second problem noted in this research is the ineffectiveness of such policies at stopping misuse. While this could be a result of lack of familiarity with organizational computer usage policies, organizations must consider the possibility that such policies are simply ineffective in today’s environment. This suggests other approaches should be explored, especially more active approaches, such as password protection and encryption.

It is clear that additional means are necessary for every member of an organization to develop greater appreciation of, to understand, and to comply with computer usage policies. Unfortunately, simply having a company-wide computer usage policy in place does not correspondingly lead to the practice that the policy will be observed (or even enforced by the organization).

Future research should examine the impact of multiple exposures to those policies and should explore the relationship between repeated exposure to computer usage policies and reported instances of misuse as well as the implementation, communication, and enforcement of such policies. In order to reduce the cost and frequency of information systems misuse, piracy, and computer crime in today’s environment, the authors recommend that an organization’s (university’s) employee (student) orientation program must include discussion of correct and incorrect computer usage, penalties imposed for violations, moral appeals, and methods of enforcement along with tougher enforcement policies.

**References**


**Timothy Paul Cronan** (cronan@uark.edu) is a professor and the M.D. Matthews Chair in Information Systems in the Sam M. Walton College of Business at the University of Arkansas.

**C. Bryan Foltz** (foltzc@utm.edu) is an assistant professor in the computer science and information systems department, College of Business, at the University of Tennessee, Martin, TN.

**Thomas W. Jones** (twjones@uark.edu) is a professor of Information Systems in the Sam M. Walton College of Business at the University of Arkansas.
In current IT practices, the task of managing post-deployment system changes often falls in no-man’s land.

David Kang and Roger Chiang

An information system will change during both system development and use. But while IS change has been systematically managed during system development, it is handled haphazardly at best by IT management during system use, largely due to the static and unproblematic view of IT [4]. Chaotic management of post-deployment system changes often can lead to real work disruptions and user dissatisfaction that tend to increase the risk of IT project failure. Through an in-depth examination of one such case, we found that IT is a dynamic artifact that evolves even during production use as the result of mutual adaptation between IT and its users’ work practice. Our findings suggest that an organization should adopt a systematic and multifaceted approach in managing post-deployment system changes.

Today’s information systems are increasingly deployed to support and transform entire business processes. The transformation role played by IT implies that its deployment will enable users to work differently. Due to great difficulties in foreseeing how users would work differently by using a new IS, there are unavoidable mismatches between IT design visions and their real-world appropriation by users. IT deployments will likely be followed by mutual adaptations between the systems and the business process they support [3]. The consequences of post-deployment system changes necessitated by such mutual adaptations are far more serious than those induced by changes during system development. System changes during production use will affect more than system developers and IT management. They will likely cause real work disruption, user retraining, and reevaluation of policies and procedures. For these reasons, it is not sensible to manage such changes haphazardly.

Compared to the intense interest in system
development that has sprouted more than 1,000 system development methods [2], relatively little attention has been paid to the management of post-deployment system change. A few researchers who contemplated the issue of post-deployment system changes have suggested flexible and adaptive system design as the solution [1, 5]. But no one has systematically investigated this problem from a project management perspective. This perspective is important because even changes made to a flexible and adaptive system must be carefully managed to minimize disruptions to users’ work. To investigate and understand challenges associated with managing system changes in use, we conducted this longitudinal field study to track the project management practice employed by Bank X in managing post-deployment system changes to a commercial banking workflow application. Based on findings, we then articulate a systematic approach to manage post-deployment system changes on multiple fronts.

Bank X is a large West Coast commercial bank that has more than $30 billion in assets and about 3,000 commercial lending clients. The bankers involved in its commercial lending business included about 90 line representatives and 70 staff analysts. The line representatives were stationed in 12 regional sales offices to generate loan prospects and maintain client relationships. The staff analysts worked at three central sites to provide credit analysis support and loan approval. Prior to the use of the new workflow application, the collaboration among these geographically dispersed parties relied on an antiquated document management infrastructure consisting of filing cabinets, interoffice mail, fax, express mail, and file servers. As a result, document turnaround time was long, and management perceived this delay as a serious handicap in a highly competitive market.

Under the sponsorship of the vice chairman in charge of commercial lending, Bank X chartered a project team to develop a new workflow application for its commercial lending business process shown in Figure 1. The project team consisted of an IT manager, a business process manager, and several system analysts. One of the authors served as a special member on the team to be in charge of post-deployment system support. The project team chose Lotus Notes as the application development platform and contracted out system development to a Lotus business partner. The project team secured extensive user participation during the application development process by conducting joint application design (JAD) sessions and adopting a prototyping method. The first version of the application was put through a three-month pilot test by one small commercial banking team before the bankwide deployment.

At the center of the workflow application was a Notes database that contained 27 standard types of commercial lending documents and numerous other document types embedded in them as OLE objects. These electronic documents were managed by a complex scheme that dynamically determined what could be done to a particular document based on the combination of user access privileges, the type of the document, the status of the document, and type of loan packages that the document belonged to. The new workflow application allowed users to create, store, and route major types of credit documents online. Any user, if authorized, could open, print, and edit any document at any time in any of the bank’s offices. The application also enabled co-authoring and electronically “stapled” together documents of a loan package.

As the primary IT platform for the commercial lending business process, the new workflow application was technically complex. It had six Lotus Notes servers distributed across the state of California. Multiple groups of users were assigned to each of the

Relatively little attention has been paid to the management of post-deployment system change.
servers; many of them worked at different sites on different days of the week and therefore had dynamic access privileges on different servers. Another source of the system’s complexity was its connections with other banking applications. The workflow application was connected to the bank’s mainframe-based loan payment system. Existing customers’ loan payment records were periodically downloaded into the workflow application, which also needed to interact with Lotus Office Suite, WordPerfect, and special banking applications. These connections made the workflow application a multi-vendor and multi-product IT platform.

After the bankwide deployment of the new workflow application, the project team was plagued with the needs to handle post-deployment design changes. For example, a drop-down list for a field may not contain a certain value that was invented in handling a special loan package; an approved loan amount was changed and there were no extra fields to keep both the original and the changed loan amount; and so on. These needs for design changes were not necessarily generated by poor original design. The new workflow application was designed to support a business process that was simply too complex and fluid for a few system designers to completely map out all possible use cases during design. Most of these problems did not even surface in production use until the special situations emerged. Only the production use by all the users could be seen as a true, robust, and comprehensive test of a system design.

The project team anticipated the need for post-deployment design changes. But because it could not predict what and how changes would emerge, a commonsense approach shown as Iteration A in Figure 2 was initially adopted to manage emergent design revisions. When a problem was reported to a project team member, the individual would pass it on to the project manager. He would then hold a discussion among related stakeholders. Following that, a decision would be made and communicated to the contractor’s development team. The developers needed to make and test the necessary design changes on the development (Alpha) version of the application hosted at the contractor site. When it was done, the design changes would be replicated to the production (Beta) version of the application, typically overnight.

This iteration met its challenge from day one because most of the early design changes were technical ones that demanded quick action rather than elaborate discussions. The long information delay and discussion time frustrated users whose normal work activities were disrupted. Adapting to the situation, the project team quickly adopted a contingent discussion approach shown as Iteration B in Figure 2. In this new approach, a help desk staff member immediately documented all reported problems in a help event log maintained online as a Notes database. A new task called “problem characterization” was created. This
task was performed by a senior system analyst who constantly monitored the help event log. If a desired design change was of a purely technical nature, the senior analyst would immediately instruct the developers to make the change. Otherwise, he would defer the decision to the project manager.

While mending the approach of technical design changes, an even bigger problem emerged. Testing design changes on the Alpha platform (the development platform) proved to be problematic because the testing data and operating environments were different from those of the production system. A change made to one form could trigger unintended effects on other forms. These unintended effects frequently surfaced only after the design changes were implemented on the production system that maintains the much richer real data. The users were extremely frustrated when work stoppage caused by these “solutions” to their previous problems led to significant new productivity drags. To solve the problem, the project team adopted the realistic testing approach shown as Iteration C in Figure 2.

The most salient feature in this new strategy is the Gamma platform. It ran application codes replicated from the Alpha and real documents replicated from the Beta platform. Any new programming changes developed on the Alpha platform would be replicated first to the Gamma platform so that they could be tested against production data before implementation on the production platform. Since the Gamma platform ran on a server that was part of the bank’s computer network, it also duplicated the operating environment of the production system. Furthermore, the project team members could do the testing. This made the testing more realistic since they were much more familiar with the “weak spots” in the application. The project team nicknamed it a “leapfrog approach” because the Gamma was the would-be future platform. Iteration C has solved most of the technical problems associated with the programming changes.

The evolution of system change management approach.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Problem</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Slow technical trouble shooting</td>
<td>Work disruption</td>
</tr>
<tr>
<td>B</td>
<td>Superficial testing</td>
<td>Implementing problematic solutions</td>
</tr>
<tr>
<td>C</td>
<td>Confusion over the priorities of system enhancements</td>
<td>Trouble in resource allocation and benefits realization</td>
</tr>
</tbody>
</table>

Managing System Enhancements

Learning happens in the process of working. Users suggested many system enhancements once they moved beyond knowing the basic functionality of the new application. For example, users found that the workflow application could also serve as a knowledge repository for sharing expertise. These enhancement suggestions pushed the application along different directions from its initial design. The project team created a public database to register these suggestions. On a rolling basis, this database usually contained between 200–250 suggestions that needed to be dealt with. Because of the resource constraints, prioritizing these suggestions became a serious challenge. Naturally, most users assigned high priorities to their own suggestions. After a month of struggling, the IT project manager realized he could not make the prioritizing decisions because the criteria, other than the resource concern, were essentially business oriented. Per his request, the bank created a user steering committee to preside over the task. This user-centered change prioritization approach is shown as Iteration D in Figure 2.

This shifting of decision power from the IT project to the users and general management focused the debate on prioritizing the merits of the suggestions rather than resource allocations. Later, the project manager realized the additional value of letting the user steering committee members use the Gamma platform to test future system releases. Packaging system changes into new releases had always been a major challenge. Fewer but bigger releases of system changes would incur less overhead in terms of weekend overtime, planning and drafting of user communications, retraining of users, data reconciliation, procedural changes, and system documentations. But more frequent smaller releases would deliver benefits to users faster and reduce risks. It turned out the user steering committee could better assess alternative release packages on their benefits, urgency, potential for work disruption, and retraining needs. The committee would first test different packaging plans and then produce recommendations to the IT project management. Later implementations of system changes became smoother with this management strategy.

Managing Volatility Associated With External Changes

Some major system changes were dictated by external events. For instance, after purchasing another smaller bank, the workflow application has to be adapted to accommodate its credit documents. The forming of an alliance with several large banks also necessitated changes to document sharing and control mechanisms. These types of external events often demanded rigid deadlines that led to wholesale project scheduling changes. The lack of mutual
understanding between IT and general management made it difficult for the project manager to foresee these problems. According to the bank's vice chairman who oversaw the alliance building effort, general management did not understand enough about IT to foresee the alliance-building related IT problems and therefore did not give early warning to the IT project manager. He also said the project team could have done a better job educating the general management about what types of organizational changes might affect the IT project and why the IT people needed to be informed. After several initial mishaps, the bank's CIO pushed for the inclusion of key IT project managers in the bank's strategic planning process.

LESSONS LEARNED

The table summarizes what motivated the bank's first three iterations of system change management approaches before it settled with the last one. Each iteration exhibited problems during execution—thus motivating the next iteration as a solution. As shown in the table, the issues that Bank X encountered in managing post-deployment system change encompassed design changes, testing, and system enhancements. The success in managing these issues hinged upon the establishment of a systematic approach that coordinates activities of multiple stakeholders. The experience of Bank X suggests that managing system changes is not an activity confined to system design and development. The goal of system design is to map out an IT solution based on both the current and the future model of a business process. It is important to recognize this design is frequently built on potentially unstable ground. While the current business model is subject to change, the future business model is only a vision of change, the future business model is only a vision of what the future has not been tested in real-world use.

While the current business model is subject to change, the future business model is only a vision of the future that has not been tested in real-world use. Because of this, system design is really a continuous activity driven both by visions for the future and during production use.

In developing a systematic approach to managing post-deployment system design changes, IT management must realize the battle will be fought on two fronts. On one hand, system changes must be made for technical reasons. The management of this type of system change should emphasize speed as these problems usually can immediately disrupt users' work. The principles in managing this type of system change should include efficient reporting and communication, proper delegation of decision locus for design changes, and a robust testing environment that uses production data. Bank X, through a rather costly learning process, eventually established a systematic approach that embraced these principles. On the other front, the effective management of system changes caused by learning and organizational changes requires participation from users and general management. Design decisions of this type cannot be made by project teams as pure technical changes to a system. The user steering committee established by Bank X demonstrates one mechanism to accommodate such user and management involvement.

CONCLUSION

In current IT practices, the task of managing post-deployment system changes often falls in no-man's land. IT professionals typically focus on system development and implementation while management and users typically focus on the use of IT. The post-deployment involvement of IT professionals has been typically restricted to providing technical support. As the trend toward intertwining IT with business logics accelerates, the current practices of managing system development and use as separate activities will be increasingly challenged. It is time for both IS researchers and practitioners to reexamine this traditional divide. This study described a systematic approach of managing IT changes as one possible solution to this problem. But even in situations where this specific approach does not fit, the principles of contingent discussion of technical changes, realistic testing of system changes, and user-centered change prioritization are still of value for crafting different solutions.

References


David Kang (kang@chapman.edu) is an assistant professor of MIS at Chapman University, Orange, CA.

Roger H.L. Chiang (roger.chiang@uc.edu) is an associate professor of IS at the University of Cincinnati, OH.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2006 ACM 0001-0782/06/0600 $5.00

David Kang (kang@chapman.edu) is an assistant professor of MIS at Chapman University, Orange, CA.

Roger H.L. Chiang (roger.chiang@uc.edu) is an associate professor of IS at the University of Cincinnati, OH.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
As firms become progressively more dependent on Internet-based information systems, they are increasingly vulnerable to defects in those systems. These defects can lead to errors, undetected fraud, and malicious intrusion. Information system errors can be catastrophic, whether they occur in market transactions, banking, air traffic control, and so forth. Resulting damages can include lost revenue, lost data, lost trust, and increased costs [12].

Consequently, effective design of e-business processes is essential to avoid defects that can otherwise lead to errors, fraud, and intrusion. Whereas carefully designed e-business protocols can perform effectively within most expected situations, assuring correct processing under all circumstances is exceptionally complex and difficult. Hidden flaws and errors that occur only under unexpected and hard-to-anticipate circumstances can lead to processing errors and potentially ruinous failures. Continued growth of e-business will, in large part, depend on protocols designed to ensure the information exchanged between trading parties is protected from unauthorized disclosure and modification [8].

By Bonnie Brinton Anderson, James V. Hansen, Paul Benjamin Lowry, and Scott L. Summers
Verifying an e-business protocol is robust against hidden flaws and errors is a formidable task. Manual methods are slow and error prone. Even theorem provers, which provide a formal structure for verifying protocol characteristics, may require human intervention and can be time consuming. Moreover, theorem provers generally do not provide much help in locating failure sources. Finally, simulations offer computational power, but they are ad hoc in nature and there is no guarantee they will explore all important contingencies [12].

Conversely, model checking is an evolving technology that offers a platform for effective and efficient evaluation of e-business protocols. Current model checking technology is based on automated techniques that are considerably faster and more robust than other approaches such as simulation or theorem proving. With today's model checkers, large state spaces can be analyzed in minutes. Additionally, model checkers are able to extend their analysis by supplying counterexamples that identify the precise location where a protocol failure is discovered [7] (see the accompanying table).

Although still relatively new to the analysis of e-business processes [5, 12], model checking has an established record in the practical analysis of complex hardware and software processes [6, 9]. Elsewhere, we have provided more technical presentations of applications of model checking to e-business processes [1, 2]. The purpose of this article is to provide an exposition of model checking in the context of e-business that is accessible to managers, auditors, and system developers. We believe readers can benefit from an accessible introduction to model checking with a discussion of its potential for enhancing the development of reliable e-processes. The e-processes on which we base this discussion are from the work of Ray and Ray [7], which incorporates processes fundamental to a broad class of e-business operations. These processes include distributed processing, parallelism, concurrency, communication uncertainties, and continuous operations.

### Related Work on Model Checking

E-business managers, developers, and auditors require robust tools to assure users that e-business systems are secure and reliable [7]. Designing and implementing highly secure and reliable e-processes is challenging, and requires adherence to several specific criteria to be effective. Adding to the challenge of designing effective e-business protocols, recent research demonstrates that money atomicity, goods atomicity, and validated receipt are critical e-process requirements.

Money atomicity ensures that money is neither created nor destroyed in the course of an e-business transaction. Goods atomicity guarantees a seller receives payment only if the customer receives the product. Validated receipt ensures the buyer is able to verify the contents of the product about to be received before making payment [7].

Heintze et al. [5] use a model checker to examine the non-security characteristics of e-processes to verify the money and goods atomicity properties of two e-business processes—Digicash [3] and NetBill [4]. This seminal work demonstrates how to model e-business processes and their properties of interest in a process algebra language—CSP—the language used in FDR. The model checker determined the NetBill process does achieve money atomicity and goods atomicity. Conversely, the model checker discovered that Digicash failed to ensure money atomicity—an interesting finding in a commercial software package. In the latter case, a detailed counterexample was generated to illustrate a set of actions leading to a state where money atomicity was not realized.

Wang et al. [12] make a persuasive case that model checking can play a valuable role in designing e-processes and can be an effective method for evaluating and auditing existing e-processes. The authors use a ticket sales application to demonstrate the capabilities of two model checkers. While their application is conceptually sound, it abstracts the

### Table 1. Comparison of Formal Verification Methods.

<table>
<thead>
<tr>
<th>Formal Verification Methods</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual proofs</td>
<td>• Flexibility</td>
<td>• Time consuming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Error prone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Not economically viable</td>
</tr>
<tr>
<td>Theorem provers</td>
<td>• Reduce human error</td>
<td>• Require significant expertise</td>
</tr>
<tr>
<td></td>
<td>• Provide formal structure for verifying protocol characteristics</td>
<td>Often poor documentation</td>
</tr>
<tr>
<td></td>
<td>• Prove program specifications</td>
<td>Doesn't produce counterexample upon failure</td>
</tr>
<tr>
<td>Simulations</td>
<td>• Computational power</td>
<td>Ad hoc in nature—must update each time the model changes</td>
</tr>
<tr>
<td>Model Checkers</td>
<td>• Provide effective and efficient evaluation of e-business protocols faster and more robust than other approaches such as simulation or theorem proving</td>
<td>• May be difficult to model business system</td>
</tr>
<tr>
<td></td>
<td>• May supply counterexamples that indicate the precise location where a protocol failure is discovered</td>
<td>Limited expressiveness of formal presentation language</td>
</tr>
<tr>
<td></td>
<td>• Locate subtle but critical flaws that other approaches may not find.</td>
<td>Does not create nor exhaust all possible model specifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Does not externally validate the model itself</td>
</tr>
</tbody>
</table>

Comparison of formal verification methods.
Ray and Ray [7] extend the basic structures examined by Heintze et al. [5], who assume that neither the NetBill server nor the communication links to the NetBill server ever fail. Ray and Ray provide a more comprehensive treatment of system and communication failures by allowing a communication link among a customer, a merchant, or a trusted third party to fail arbitrarily. Additional mechanisms were developed to ensure that desirable properties are preserved despite such failures. The result is a realistic and practical platform for many e-business applications.

**An E-Business Protocol Example**

The figure here outlines a possible e-business protocol [7]. Messages are exchanged between a customer, a merchant, and a trusted third party (TTP). A merchant has several products to sell. The merchant places a description of each digital product in an online catalog service with a TTP along with a copy of the encrypted product. When a customer finds a product of interest in the catalog, the customer downloads the encrypted product and sends a purchase order (PO) to the merchant. The customer cannot use the product until it is decrypted, and the merchant does not send the decrypting key unless the merchant receives a payment token through the PO process. The customer, in turn, does not pay unless he/she is sure the correct and complete product has been received. The TTP provides support for PO validation, payment token approval, and approval of the overall transaction between the customer and the merchant.

At a more detailed level, the process begins as the customer browses the product catalog located at the TTP and chooses a product. The customer then downloads the encrypted product along with the product identifier—a file that contains product information such as its description and identifier. If the identifier of the encrypted product file corresponds to the identifier in the product identifier file, the transaction proceeds. If the identifiers do not match, an advice is sent to the TTP and the customer waits for the correct encrypted product. This process ensures the customer receives the product requested from the catalog. Next, the customer prepares a PO containing the customer’s identity, the merchant identifier, the product identifier, and the product price. A cryptographic checksum is also prepared. The PO along with the cryptographic checksum is then sent to the merchant.

The combination of the PO and cryptographic checksum allows the merchant to ascertain whether the PO received is complete or whether it was altered while in transit. Upon receipt of the PO, the merchant examines its contents. If satisfied with the PO, the merchant endorses the PO and digitally signs the cryptographic checksum of the endorsed PO. This is forwarded to the TTP, which is involved in the process to prevent the merchant from later rejecting the terms and conditions of the transaction. The merchant also sends a single use decrypting key for the product to the TTP. The merchant then sends a copy of the encrypted product to the customer, together with a signed cryptographic checksum. The signed cryptographic checksum establishes origin of the product and also provides a check to signify whether the product has been corrupted during transit.
Upon receipt of the second copy of the encrypted product, the customer verifies the first and second copies of the product are identical. Through this process, customers can be assured they received the product ordered. The customer then requests the decrypting key from the TTP. To do this, the customer forwards the PO and a signed payment token to the TTP, together with its cryptographic checksum. The payment token contains the customer’s identity, the identity of the customer’s financial institution, the customer’s bank account number with the financial institution, and the amount to be debited from the customer’s account.

To verify the transaction, the TTP first compares the digest included in PO from the customer with the digest of the same from the merchant. If the two do not match, the TTP aborts the transaction. Otherwise, the TTP proceeds by validating the payment token with the customer’s financial institution by presenting the token and the sale price. The financial institution validates the token. If the token is not validated, the TTP aborts the transaction and advises the merchant accordingly. If the token is validated, the TTP sends the decrypting key to the customer and the payment token to the merchant, both digitally signed with the TTP’s private key.

Secure channels guarantee the confidentiality of all messages throughout this protocol, which ensures money atomicity if the payment token generated by the customer contains the amount to be debited from the customer’s account and credited to the merchants account. Consequently, no money is created or destroyed in the system by this protocol.

Goods atomicity is guaranteed if the TTP submits the payment token only when the customer acknowledges the receipt of the product. The process also ensures the product is actually available to the customer for use.

Delivery verification is guaranteed if the TTP receives a cryptographic checksum of the product from the merchant. Also, the customer independently generates a checksum of the product received and sends it to the TTP. Using these two copies of the checksums, which are available at the TTP, both the merchant and the consumer demonstrate proof of the contents of the delivered goods.

**Example of Model Checking**

The e-commerce system described in the preceding section can be modeled and tested in a model checker, such as FDR, which has been used in a variety of hardware and software applications, and has been successfully applied to testing for atomicity in two commercial e-commerce payment systems, as noted earlier [12]. Recall that three of the most important requirements of e-commerce trading are:

1. Money is neither created nor destroyed in the course of an e-commerce transaction. A transaction should ensure the transfer of funds from one party to another without the possibility of the creation or destruction of money. No viable e-commerce payment method can exist without supporting this property [10].
2. Both the customer and merchant should receive evidence that the goods sent (or received) are those to which both parties agreed. Particularly when dealing with goods that can be transferred electronically (as we do here), the combination of both is essential [11].
3. The customer is able to verify the contents of the product about to be received before making payment. The customer must be able to verify the product about to be received is the same product that was ordered, before the customer pays for the product.

Model checking provides verification of whether or not our system guarantees these requirements by writing them as specifications in the model checker. If a specification is not satisfied, the model checker returns a counterexample, allowing the analyst to immediately see how the requirement can be violated. This, in turn, aids in modifying the system to resolve the problem. As an example, we consider a specification for requirement (1) here.

**Specification Requirement 1**

\[
\text{IF NOT } \text{transaction\_trace} = \text{STOP} \\
\text{OR} \\
\{ \text{customer sends payment\_Token to TTP;} \\
\text{merchant receives payment\_Token from TTP;} \\
\text{STOP;} \} \\
\text{OR} \\
\{ \text{customer sends payment\_Token to TTP;} \\
\text{customer receives transaction\_Aborted message from TTP;} \\
\text{STOP;} \} \\
\text{THEN} \\
\text{Requirement\_1 is violated;} \\
\text{End requirement\_1;} \\
\]

This specification asserts the following:

1. The customer must send payment to the TTP and the merchant must subsequently receive that payment from the TTP, or
As e-commerce transactions become increasingly complex, coupled with increased regulations and liability exposure, the need for assurance in e-commerce protocols will likely grow. A potentially valuable development would be to adapt model checking to dependability auditing of e-business processes prior to implementation.

2. The customer can send payment to the TTP and (in the case where the payment is invalid) the customer then receives a transaction aborted message from the TTP.

3. If neither (1) nor (2) is satisfied, the requirement is violated.

The following is an example of what the model checker returns as a counterexample when the implementation fails to satisfy specification requirement_1:

1. The customer receives the encrypted goods from the TTP, and then sends a PO to the merchant.
2. The merchant then sends the encrypted goods to the customer and sends a key to the TTP, after which the encrypted goods are received by the customer who sends a payment token to the TTP.
3. The TTP then receives the key from the merchant and the payment token from the customer.
4. The TTP then sends the payment token to the merchant, after which the customer receives the key from the TTP. The process then stops. Since the next step should have been the receipt of the payment token by the merchant (from the TTP), we know where to look to examine the failure.

In this case, the model checker showed the failure occurred in the e-process controlling the sending of an encryption key from the merchant to the TTP. Once this was identified, the problem was rectified and proven in a subsequent run of the model checker.

CONCLUSION

As e-commerce transactions become increasingly complex, coupled with increased regulations and liability exposure, the need for assurance in e-commerce protocols will likely grow. A potentially valuable development would be to adapt model checking to dependability auditing of e-business processes prior to implementation. The assurance afforded by model checking can justify greater confidence in e-processes and thereby increase the acceptance and legitimacy of e-business.

REFERENCES


Bonnie Brinton Anderson (bonnie_anderson@byu.edu) is an assistant professor of information systems in the Information Systems Department of the Marriott School, Brigham Young University, Provo, UT.

James V. Hansen (james_hansen@byu.edu) is the J. Owen Cherrington Professor of information systems in the Information Systems Department of the Marriott School, Brigham Young University, Provo, UT.

Paul Benjamin Lowry (Paul.Lowry@byu.edu) is an assistant professor of information systems in the Information Systems Department of the Marriott School, Brigham Young University, Provo, UT.

Scott L. Summers (summers@byu.edu) is an associate professor in the School of Accountancy of the Marriott School, Brigham Young University, Provo, UT.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2006 ACM 0001-0782/06/0600 $5.00
The goal of enterprise integration—to create a shared information environment that supports the delivery of products or services—becomes an even greater challenge when business units are spread throughout the world, decentralized, and accustomed to working their own way.

ENTERPRISE IN
Across the Globally Disbursed Service Organization

There is a paradox in managing a global service organization. On the one hand, there are pressures to decentralize operations and outsource appropriate business processes. On the other hand, there are competitive pressures to centralize management functions and integrate the dispersed operations of the firm.

Integration is frequently considered the price of entry for running a business [7]. For most organizations, integration is supported through enterprise information systems, capable of sharing information across the extended enterprise from suppliers to customers. They promise reduced costs, tighter control, higher quality, and above all, increased customer service and satisfaction.

Enterprise systems, however, do not always meet their goals. Spending often exceeds budgets, staff frequently resists changes, and projects can be burdened with technical problems [10, 11]. Some fail altogether and others produce only disappointing results [3, 8].

While creating an integrated enterprise environment is certainly challenging for the organization whose business units are geographically concentrated and centrally controlled, it becomes even more challenging when these units are geographically dispersed and individuals from around the globe must work together [1].

This article highlights a strategy for managing an important component of the integration process and illustrates the use of this strategy with a case study of how the largest financing company in the world addressed integration issues and how it succeeded in bringing together a diverse group of business units, business practices, legacy systems, and managers.

Emphasis for Global Integration
Enterprise integration must address technological/business process issues [9], and organizational/management issues [10]. The emphasis given to each varies. When the emphasis is on technological/business process issues, there is an assumption that the type of knowledge transferred—including best business practices, which dominates organizational/management issues—is relatively straightforward and explicit, and that carefully articulated work processes can be implemented regardless of cultural context [2]. For example, implementing an airline reservation system in a new region of the world is considered a relatively straightforward process since only explicit knowledge needs to be transferred. Here, centralized plan-
ning and implementation does work (airlines do it all the time!). When the business processes become more complex, when the type of knowledge transferred is complex and tacit, the challenge increases.

In one Fortune 500 company I studied, a centralized manufacturing and accounting system was implemented at plants across the globe. At headquarters, a map of the world revealed plant locations. Green pins represented a successful implementation; yellow pins represented partial success; red pins represented failure. Pointing to the red pins (one-third of the pins were red), one manager confided, “They just don’t get it.” Offering an explanation, he continued, “We’ve had trouble finding the right managers for the job.” But his response suggests an emphasis on the technological/business process issues. Little was expressed about the challenge of organizational/management issues.

When the emphasis is on organizational/management issues, attention is drawn to the social structure of business units and the challenges these structures impose on the implementation process. Scott and Vessey [10], studying ERP implementation at two companies, concluded the major difference between a successful and failed implementation was related to organizational culture. They suggest that implementation success increases when the culture supports open communication and when the company recognizes that employees are the primary source of ideas.

A wider variation in organizational culture is expected when implementation spans the globe. This can be attributed to differences in national culture. A recent study, Project GLOBE, compared 18,000 middle managers from 62 countries [6]. What they found, confirming earlier studies [5], was that managers behave differently across the world. Examples of cultural dimensions affecting these differences include: the assertiveness of individuals; power distance among peers, subordinates and superiors; the willingness to take risks; and the preference for collective or individual activity. They suggest that as the geographic scope of a project broadens, the social framework within which the project must unfold becomes more complex.

The implications of these studies are that cultural factors may add another layer of complexity to the role of organizational/management issues in the integration process.

**Case Study**

This case study was based on corporate presentations as well as interviews with a senior project leader. It examined the transition from what management referred to as an “old legacy system” designed for “old
business processes” and using “old organization structures” to new business processes featuring state-of-the-art integrated systems.

**COMPANY BACKGROUND**

IBM Global Financing, IGF, offers financial products and services in two areas—asset financing and asset recovery. With over $37 billion in assets under management, IGF is the largest financing company in the world and is equal in size to the 18th largest U.S. bank. The company employs 3,200 people and provides services to over 125,000 customers located in 43 countries.

Asset financing includes such products and services as fixed rate, variable rate, and balloon loans targeted to customer, who have purchased IBM and non-IBM hardware and software products. In most cases these loans represent long-term obligations for technologies subject to high obsolescence. To accommodate these long-term risks, IGF offers options for mid-term upgrades, swaps, and early termination of leases.

The second area of business, asset recovery, provides services that help customers manage end-of-lease decisions. These services include refurbishing equipment for resale, providing resale support, demanufacturing equipment to salvage components, or scraping equipment altogether. With both flexible financing and asset recovery services, IGF maintains that their services help ensure that business needs drive a customer’s end-of-lease decisions rather than end-of-lease decisions driving business needs.

**CRITICAL SUCCESS FACTORS**

Fui-Hoon Nah and Lee-Shang Lau [4] summarized the critical success factors for enterprise integration. They included top-level support, adequate business plan, business process reengineering, collaborative teamwork, effective communication, appointment of a project champion, and capable project management. Here is how IGF addressed these factors as it managed the transfer to an integrated environment.

**Top-level support.** IBM CEO Lou Gerstner stated in 2002 that to really get the benefit of e-business there was a new imperative to integrate all parts of the enterprise, all those core processes, and the applications that support them. He continued that it was important to integrate customer relationships with all the products and services the enterprise created, and all the relationships that connect the enterprise with the outside world.

**Business plan.** With a mandate from the top, IGF developed a business plan that included many strategic goals.

- Integrate customers, suppliers, and partners;
- Improve customer and business partner satisfaction;
- Establish a Global Asset Recovery Services brand;
- Reduce administrative costs;
- Raise productivity;
- Lower IT operations and systems maintenance costs; and
- Enhance security and business controls.

The new system would integrate the extended enterprise and link all offices in all locations. It would transform a geographically dispersed organization with independent systems into one virtual company capable of competing in an “e-business environment through the next decade and beyond.”

At the center of this project would be a common global system. It would be the most on-demand business at IBM and provide the highest level of integration.

**Business process reengineering.** The starting point for the project was the development of an as-is model of the business. Processes were documented worldwide and as expected, uncovered many different approaches to getting the job done. Loan application, credit scoring, loan payment, and asset record-keeping processes had been created to accommodate local or regional preferences. The as-is model provided the data to identify cross business synergies, and common process flows. As work continued key business decisions were defined, followed by flowcharts of the leasing life cycle, and, from this data, recommendations were made for business process improvement.

**Teamwork.** IGF chose a strategy that addressed both the technical/business process and organizational/management challenges. This strategy combined centralized, top-down leadership with bottom-up collaboration. It would be, commented one manager, a “new approach for IBM.” Further, it would be a strategy that would provide the opportunity for business units across the globe to participate collectively in the integration project.

A team of 15 individuals was selected, including six U.S. managers and the remaining representatives from operations in Denmark, France, Sweden, the U.K., Japan, Australia, and Italy. Workshops were held at six different international locations. At these meetings business processes, used by 3,200 employees from 43 countries, were studied and compared. What became clear was that the inability to share data was having a detrimental effect on the ability to compete in an increasingly competitive marketplace. As one manager commented, “It didn’t have to be that way.”

Making the transition from a geographically dispersed and decentralized system is a difficult chal-
challenge, but the team of 15 managers, from across the world, and meeting face-to-face, made progress as a common environment evolved. Indeed, the team approach appeared to provide an effective mechanism through which complex differences of substance and style could be addressed and effectively resolved. Teams, then, became the cultural melting pot.

One observer commented that it was not the particulars of new business processes that seemed to be the biggest challenge. Nor was it the protection of familiar processes these individuals had used for a long time. Nor was it the threat of new software. The biggest challenge, he said, was in “managing a cultural change, not a software change.”

Communication. Participation by individuals from a wide range of national cultures can make effective communication challenging. Only five of 15 team members on the team spoke English as a first language, but language was considered a non-issue. A more significant issue was the challenge of bringing different organizational cultures and management styles together. Using Hofstede’s research [5], it could be expected, for example, that participants from high power distance countries like France and Japan might be reluctant to voice their concerns about a project initiated with a top-down perspective. Or, it could be expected that they might be reluctant to abandon a decentralized system, over which they maintained control, for a centralized/corporate system, over which they could exercise much less control. Uncertainty avoidance could also affect team participation and project outcomes. Those from countries low in uncertainty avoidance like Denmark and the U.S. might be expected to be less resistant to trying new ways of processing loans or communicating with customers than those from high power distance countries like Japan and France.

Indeed, the kind of effective communication that would ensure a successful implementation would be difficult to achieve, but the cultural melting pot, created by the team approach, served to provide the context within which these different management styles could be used to best advantage.

Project champion. Champions are often seen as the motivators behind the success of any project. For global integration projects they must be capable of balancing the technological/business process and management/organizational issues in a cross-cultural environment. The IGF champion, a senior person in the organization, was an individual who seemed to balance these issues quite well and who was capable of addressing the complexities associated with managing a diverse group of 15 individuals from many countries.

Project management. According to one manager, traditional IBM strategy in situations like this would be to introduce changes in bits and pieces where modules would be rolled out one-at-a-time and the first tested before the next began. The IGF project would take a very different approach, one in which the extended enterprise would be integrated at once. This was particularly challenging because the legacy systems to be reengineered included 800 unique business processes used worldwide and with 290 different IT systems. As is true in many integration projects, some technological challenges, especially software development, can be outsourced. The new asset financing system was implemented using SAP standard core modules. They included Finance/Controlling, Fixed Assets, Sales and Distribution, Accounts Receivable, Accounts Payable, and Lease/Loan Accounting. Other modules were implemented including an SAP Customer Resource Management module.

The global asset recovery system was also built on standard SAP core modules. It maintained data on the tracking and receipt of off-lease and other IBM used equipment, surplus, and excess inventory. In addition, it maintained data on refurbished, dismantled, resale, and scrap activities.

Not all applications were integrated in an ERP environment. IGF would still use best of breed applications. For example, a system developed by Fair, Isaac and Company Inc., would support the credit review process.

Realistic project milestones, tight controls, and constant communication were the hallmarks of project control. Often, however, it is the management and organizational issues that challenge the timetable. Here the emphasis on communication seemed to keep activities moving toward a timely completion of the project.
PROJECT RESULTS
When completed, 800 unique business processes were reduced to 19 common processes used in all offices throughout the world. Furthermore, 290 separate IT systems were reduced to 36 including the SAP software and the best of breed applications that were specific to IGF’s business.

Benefits from the system included a reduction in sales cycle time from days to hours, improved inventory turnover, reduced administrative costs, improved customer satisfaction and loyalty, and increased business. In one year the asset-financing systems alone produced $200 million in gross profit, a significant return for a project with a price tag in the range of $60 million. Its success was recognized by the leasing industry when IGF won the 2002 annual Technology Innovation Award.

But the benefits of an integrated system extended to management planning and control. Using data warehousing technology, which capitalized on the accessibility of the transaction data stored in the common database environment, managers now had “views of data that they never had before.” Strategic planning, as a consequence, significantly improved.

LESSONS LEARNED
IGF expressed several lessons learned from the experience:

• Take a holistic or central view.
• Wait for absolute CEO commitment.
• Appoint an organizational champion.
• Outsource what you can ... no one can do it alone.
• Assign the best goal-oriented business and IT builders to the project.
• Maintain tight management control.
• Communicate, communicate, communicate.

These lessons address both technological/business process and organizational/management issues. The last lesson, repeated three times by management, underscores the additional challenge imposed by global integration. Communication in this context is complicated by managers whose national cultures affect their way of thinking and their management style. To facilitate communication among this diverse group, this lesson suggests the creation of international teams supported by face-to-face meetings.

While it is difficult to speculate on the degree to which national differences would have interfered with the IGF project had other methods of development and implementation been used, anecdotal evidence from those involved suggests that face-to-face collaboration within a structured framework that combined a top-down centralized focus with a bottom-up collaborative environment was very effective. Establishing a working team of international managers is therefore a strategy worth considering when an integration project must span geographic and cultural boundaries.

Another lesson worth emphasizing, and closely related to the communication lesson, is that the complexities of global integration underscore the need to choose the right champion. That individual must have the skills necessary to balance technological/business process and managerial/organizational issues including the complexities brought on by a diverse cultural group. Further, this individual must be capable of taking greater organizational risks by opening the process to international collaboration, yet maintaining tight project control. Less social control but more project control is the real paradox in managing the global integration. It is the challenge this paradox expresses that may separate success from failure.

REFERENCES


BARRY SHORE (bshore@hypatia.unh.edu) is a professor of Management Information Systems at the Whittemore School of Business and Economics, University of New Hampshire, Durham, NH.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2006 ACM 0001-0782/06/0600 $5.00

106 June 2006/Vol. 49, No. 6 COMMUNICATIONS OF THE ACM
A Simple Approach to Improving Email Communication

Going back to basics.

Email communication is becoming a burden for many employees and the way email is handled is far from efficient [4]. Employees are overwhelmed by the volume [3], lose important items [5], and feel pressured to respond quickly (often within seconds [2]). The major research in this field is trying to solve these problems by designing and building better email systems through understanding email usage [4]. Although these systems will probably improve email communication, would going back to basics provide, at worst, an interim solution?

It would be a fair assumption that the end users of an email application are the major source of the problem, as they create and receive the email that periodically causes the problems. The back-to-basics approach is based on identifying the major problems users face with email and then administering training on how to become a more effective email communicator. Although the approach sounds simple and has successfully been applied to many other problem areas, will it work for email communication?

Stage 1: Identifying the Problems

The first stage of the study involved developing and distributing a questionnaire to all staff who use email to capture employees’ views on how email was used within their organization (a large U.K. Plc with 2,850 email users). To give the employees added security that their individual responses would not be disclosed to the management at the organization, the questionnaire was securely hosted on a remote server.

The questionnaire was designed to highlight any inefficiencies or defects in the way email is used. It asked employees to specify how many email messages they received on average each day and what proportions of these were irrelevant or unnecessary. Employees also answered questions that related to how they viewed email use within the organization.

Stage 2: Create Specific Training

The second stage of the study involved analyzing the data captured from the questionnaire (875 responses) and creating an email defects training program specific to the large U.K. Plc. The main findings from the questionnaire are:

- 16% of email messages received were copied unnecessarily;
- 13% of email messages received were irrelevant or untargeted;
- 41% of email messages received stated what action is expected;
- 46% of actionable email messages received stated what action is expected;
- 56% of employees agreed email is used too often instead of phone or face-to-face communication; and
- 45% of employees say their email messages are easy to read.

These findings indicate that there is a need for training to improve email communication. The training program was designed to address the specific problems identified in the questionnaire and to provide employees with the skills they need to use email more effectively.
To assess the extent, if any, the training program would have on the organization, a sender-recipient study was undertaken. Both sender and recipient groups (11 employees and 20 employees respectively) received different training on the best practice of email use. The sender training was more comprehensive than the recipient as the recipient only required a basic knowledge of the email defects to complete an evaluation sheet per email message received. The sender training explained the negative effects of email defects and the sessions were interactive with the participants having to highlight the defects of poorly written email messages. The training sessions targeted the following areas for optimization:

- Is an email message necessary?
- Targeting your email;
- Use an effective subject line;
- Getting your message across;
- Sending attachments; and
- Managing your Inbox.

All recipients were asked to mark up to 20 email messages they received from the sender before and after the sender had received training on the best practice of email use. The recipients marked each message against a set of criteria, giving a score depending on how well the message met each criterion. The scores both before and after the training were averaged for each sender and recipient pair. The chosen pairs for the experiment were based on high-volume email senders with a recipient that was likely to receive a high number of the sender’s email messages during the two-week monitoring process (two weeks before and after training).

### DOES GOING BACK TO BASICS WORK?

T-test analysis showed that there was an improvement in the quality of email messages received by the recipients as a result of email training for the senders. Email training also had a significant impact on the following areas:

- 99% significantly better use of the subject line, which makes it easier to prioritize an email message and also to assess the content.
- 95% significantly better written email messages that were easier to read and to the point.
- On average it takes less time (10 seconds) to read and understand an email message as a result of the training.

### IS THERE A FINANCIAL BENEFIT?

To determine any financial savings due to the training, the cost of reading email must be determined. Building on previous work by Jackson into the cost of reading business and non-business email, a cost of reading email formula was constructed [1]. Using the data obtained from the questionnaire, on average, employees received 23 email messages per day.

<table>
<thead>
<tr>
<th>No. of email messages received per day</th>
<th>Employees received training</th>
<th>Remove cc’d /unnecessary email</th>
<th>Application check for new mail (mins.)</th>
<th>Cost (£) per employee</th>
<th>Saving (£) per employee</th>
<th>Saving (%) per employee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>23</td>
<td>No</td>
<td>No</td>
<td>5</td>
<td>3,440</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>No</td>
<td>No</td>
<td>45</td>
<td>2,598</td>
<td>842</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>No</td>
<td>Yes</td>
<td>5</td>
<td>2,442</td>
<td>998</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>No</td>
<td>Yes</td>
<td>45</td>
<td>1,845</td>
<td>1,595</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>Yes</td>
<td>No</td>
<td>5</td>
<td>3,181</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>Yes</td>
<td>No</td>
<td>45</td>
<td>2,339</td>
<td>1,101</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>2,259</td>
<td>1,181</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>Yes</td>
<td>Yes</td>
<td>45</td>
<td>1,661</td>
<td>1,779</td>
</tr>
</tbody>
</table>
messages per day and it takes approximately 76 seconds to read each message. Employees therefore spend on average 29 minutes per day reading email. In addition, there is also an interrupt recovery time associated with email, which is reported to be 64 seconds per message [2]. Assuming an average salary of £16,640 per annum and an assumed overhead of a further £16,640 per year, the total cost per day of reading email for an organization can be calculated using the equation here (assuming each email message is read and has an interrupt recovery time):

\[
\text{Cost of reading email} = (t_1 + t_2) \times w \times n
\]

Where
- \( t_1 \) is the time taken to read all messages received (minutes).
- \( t_2 \) is the total interrupt recovery time (minutes).
- \( w \) is the average employee wage per minute.
- \( n \) is the number of employees within the organization.

The daily cost of reading email for this U.K. Plc with 2,850 email users is £40,848 and the cost per year over £9.8 million (based on 48 weeks a year).

The results from the sender and recipient study, after training, show that the U.K. Plc (2,850 email users) could save £3,071 per day and almost £737,000 per year on time spent reading email as a result of the training. This is the minimum saving of 8% on the total cost of reading email and equates to £259 per employee per annum.

**Targeting Further Email Savings**

As already mentioned, on average 29% of the email an employee receives is of no value to them. Assuming that all these email messages are read and the employees have not received email training, an organization could further reduce the time employees spend reading email. For this U.K. Plc it would equate to almost £12,000 per day and over £2.8 million per year.

With the majority of email applications set to check for new email every five minutes, employees can become more efficient if they change the duration of when their email application checks for new email [2] resulting in a further saving. Time saved per email user as a result of increasing the duration of checking for new email from 5 to 45 minutes is 13.16 minutes per day. This could save this U.K. Plc £10,000 per day. The savings that can be made through the introduction of training, reducing no-value email and interrupt recovery time are detailed in the table.

**Conclusion**

Our research has shown that going back to basics has increased email efficiency and at minimum saved this U.K. Plc 8% financially on the total cost of reading email. Although the results are specific to this large firm, the study indicates how an organization can become more effective, by reducing the cost associated with email use through simple email training. The implication for managers in other organizations is that if their own employees respond to training in the same way, they would also benefit from an increase in employee productivity.

**References**


**Thomas Jackson** (t.w.jackson@lboro.ac.uk) is a senior lecturer in the Research School of Informatics at Loughborough University, Leicestershire, U.K.

**Anthony Burgess** (a.k.burgess@lboro.ac.uk) is a research student at Loughborough University, Leicestershire, U.K.

**Janet Edwards** (j.edwards@lboro.ac.uk) is a lecturer in Computer Science at Loughborough University, Leicestershire, U.K.

© 2006 ACM 0001-0782/06/0600 $5.00
The problem of the rising costs and uneven quality of healthcare is a worldwide concern. Industrialized countries spend on average 10% of their GDP on healthcare, with the U.S. spending nearly 15% in 2005. For comparison, U.S. defense spending is 4% of GDP. Left unchecked, U.S. healthcare costs are projected to rise to 25% of GDP within a generation as the U.S. population ages. Similar cost increases are projected for industrialized countries as well. Over the past decade, several countries such as Australia, the U.K., and the U.S. have started IT initiatives aimed at stemming rising healthcare costs. Central to each of these initiatives is the creation of electronic health record (EHR) systems that enable a patient's EHR to be accessed by an attending healthcare professional from anywhere in the country.

The benefits claimed for EHRs are that by being able to quickly and accurately access a person’s entire health history, deaths due to medical errors (estimated to be 100,000 a year in the U.S. alone) will be drastically cut, billions of dollars in medical costs will be saved annually, and patient care will be significantly improved. Experience at the U.S. Veterans Affairs and Department of Defense (among the largest users of EHRs in the world) supports many of the benefits claimed.

However, the attempts at creating national EHR systems have been encountering difficulties. In Australia, the implementation cost has risen from an estimated AU$500M in 2000 to AU$2B today. In the U.K., the implementation costs have risen from an estimated £2.6B in 2002 to at least £15B today. In the U.S., the “working estimate” for a national EHR system runs between $100B and $150B in implementation costs with $50B per year in operating costs.

The U.K. Connecting for Health initiative calls for everyone in the U.K. to have EHRs by 2008. However, there have been ongoing problems with its implementation that spurred 23 leading U.K. computer scientists to write an open letter to the Parliament’s Health Select Committee in April, recommending an independent assessment of the basic technical viability. In their letter, they ask whether there is a technical architecture, a project plan, a detailed design, assessments of data volumes and traffic loads, adequate resiliency in the design, as well as conformance with data and privacy laws, and so on.

The U.S. approach to creating a national EHR system differs from the U.K. approach. Whereas the U.K. EHR system is publicly funded, the U.S. has decided to adopt a market-based approach, where the government acts as technology coordinator and adoption catalyst. Instead of funding the building of a single, integrated networked system with a central EHR database as in the U.K., the U.S. government is facilitating the definition of standards to allow the interoperability of commercially available EHR systems as well as interoperability certification standards. The U.S. government has high hopes for EHRs and views the development of an EHR system merely as a technological catalyst for changing how healthcare is delivered and paid for.

Some concerns are already arising with the U.S. initiative and whether its objective of providing most U.S. citizens EHRs by 2014 is realistic. For example, the government’s initiative has been chronically underfunded from the start. Medical researchers, pharmaceutical companies, insurance companies, EHR vendors, and other interested parties cannot agree on what functionality, form of information capture, and record access a national EHR system should support. Physicians working in small medical practices worry about the costs involved.

Whereas many of the issues encountered merely reflect specific instances of generic software system development problems, their number, complexity, and potential personal and political impacts magnify their importance. Thus, concerns arise from the absence of both a full business case for a national EHR system and a comprehensive risk assessment and management plan outlining the potential social, economic, and technological issues involved in creating and operating the system. As the U.K. is discovering, focusing on the technology of electronic medical records without considering the myriad socioeconomic consequences is a big mistake.

The implementation of a national EHR system presents an opportunity to constructively transform healthcare in the U.S. Whether it does will depend in large part on how well the relevant benefits and risks are understood and managed.