How Excited Will Murphy Be?

Varun Grover

"Y2K is not a problem that can be fixed with a weekend of programming. But neither should we expect our banking system to collapse."

Dr. Varun Grover is Professor of Information Systems and a USC-Business Partnership Foundation Fellow in The Darla Moore School of Business at the University of South Carolina. His areas of interest include the effective deployment and impact of Information Systems in organizations.

Why Did It Happen?

The reasons for the Y2K problem have been widely discussed. Main memory, disk space, and even punched card space were at a premium in the 1960s: it was over 10,000 times the cost today! Whatever could be squeezed to save space and money WAS squeezed to save space and money.

For this reason, applications were written with a focus on efficiency, minimizing the workload on manual data entry by using only two digits for years. No one really thought of the potential problem it could create at the turn of the century. No one paid serious attention to the fact that 00 would be interpreted as the year 1900, or that dividing by zero could lead to unpredictable outcomes. If it was a fleeting thought, it was dismissed under pressures of expediency and cost, or it was argued that applications wouldn't last that long anyway.

But many applications have lasted that long, subject to incremental changes and short-term fixes. Even newer applications have been created to work with older legacy systems, as the market has demanded backward compatibility. The situation is compounded as newer applications re-use older code, spreading the problem like a virus through a network. To make things worse, this space-saving, two-digit-year practice is embedded not only in the software, but also in the chips, that run manufacturing plants, telephone systems, nuclear plants, heating systems, military weapons systems, water and sewer systems, oil refineries, and other manifestations of this age of computer dependency.

And, while the dilemma may have been in the minds of some Information Systems types for years, there was no pressure to *act*. Why deal with a long-term problem, when there are tremendous present-day demands of data integration, strategic systems, reengineering, user expectations, and changing technology — all of which have to be resolved now? That is, until the problem becomes perilously close to "now," which is where we are amid the Y2K confusion.

There Are Both Myths and Realities of Y2K

It's the hype that's fueling the problem. Commentators fall on both ends of the spectrum. On one extreme, we have the doomsayers who predict The Millennium Bug will trigger the end of modern society. They create the cults that give credence to events like the crash of the stock market, the failure of power and energy infrastructures, the collapse of the banking system, and a host of other horrors (see **http://www.webleyweb.com/y2k/y2k.html**).

At the opposite extreme we have an equally

vociferous group of commentators who suggest that with each passing day there is more evidence to suggest the Y2K problem is a dud, and January 1, 2000, will be the biggest anticlimax in history. Proponents of these divergent viewpoints often institutionalize their positions with limited facts, overemphasizing criteria that reinforce their innate beliefs. Hence the hype.

As is the case with most predictions, however, the reality will, in all probability, lie somewhere in between. Here's what we can reasonably assess about Y2K.

We Don't Really Know the True Extent of the Problem

Even the most informed person would have a hard time predicting the true impact of Y2K. The reasons for this are manifold.

First, much of the problem is lost in mounds of COBOL code. This code has been altered and upgraded so many times that countless date locations may no longer be available. Further, in numerous cases, the original COBOL code may be lost, since old compilers converted it into machine language understood by computers.

Second, the two-digit issue may emerge in places that are unexpected and unique for every application. Imagine, for example, a system that automatically injects medicine. If it computes the dosage for a newborn on 01/1/00 for that of a 100-year-old patient, the results may be deadly.

Third, much of today's computing is mired in networks of dependencies and linkages, between customers and suppliers, business and government, and old and new systems. The problem is, none of these external dependencies is obvious in the millions of lines of code a business tests, and many can have unpredictable consequences. *But even one dependency gone awry can jeopardize the stability of your firm by resulting in a rapidly cascading effect on many other stakeholders.*

Fourth, there is the problem of embedded chips, where program instructions are hardwired into the chip. These programs cannot be revised by software: the chips have to be replaced. Clearly, this is impractical for the billions of chips present in machines, appliances, microcomputers, etc. Even if 1 percent are not Y2K compliant, we could have a major problem.

The fact is, no one on the planet really knows what will happen when 01-01-00 rolls around. In a century in which humans measured the speed of light, landed on the moon, and cloned sheep, the Y2K saga — how we ignored it for 30 years and then panicked — will not be remembered as our finest hour.

Critical Systems Should Be Fine

Despite our inability to predict the true magnitude of the problem, considerable progress has been made over the past year. A fortunate aspect of the frenzy has been the impetus to organize and do something, given the immutable deadline. Wall Street recently simulated its largest test yet of the readiness of its computerized trading systems to meet the millennium. It went off without a hitch for all 400 firms participating.

Ontario Hydro, North America's largest public utility, rolled some of its internal clocks ahead to December 31, 1999, and waited. It was a crucial test for part of a massive electrical grid that supplies power to the most populated area of Canada and parts of the United States. The nuclear plants stayed on line, the thermal generating stations kept generating — the streetlights didn't even flicker.

The point person for millennium bug fixes in 24 federal agencies, the Director of the Office of Management and Budget for the President, has indicated that most federal systems are also on track for Y2K, and there is no indication that many — or any — government systems will fail. Even Consultant Peter de Jager, considered one of the world's leading pundits and pessimists on the subject, wrote an essay this March titled "Doomsday Avoided," in which he concluded, "We've finally broken the back of the Y2K problem."

This is not to say there won't be dilemmas. Given our difficulty assessing and addressing the entire scope of the problem, *we will see failures, some of them serious*. But the activity of corporations and governments in addressing this issue gives us room to moderate the tone of doomsday scenarios.

It's More Than a Technical Problem

The technologists clearly understand what they need to do to correct the date problem. It has been the subject of virtually every Year 2000 seminar for some time. The generic steps for businesses are well defined.

- Identify the scope of the problem by evaluating existing software and hardware inventory.
- Focus on software-software and hardware-software interdependencies.
- Determine approach: retire, replace, or fix.
- Organize for action through in-house or outside contractors.
- Create a mirror image test site of your environment.
- Keep testing!

While many organizations are spending more time fixing Y2K than they originally expected, most of that time is devoted to determining the scope of the problem, and then organizing, planning, and developing tests to solve it. Changing the code is the easy part: it can be done with current practices and tools.

There Is No Quick Fix

Some companies are trying to sell quick solutions to Y2K. These tools claim to automatically find every date field in a source code and change it. That's nearly impossible! Some of the reasons for this are discussed above. Others include the necessity to involve human

logic to figure out the less conspicuous manifestations of Y2K. Changing the field from 2 to 4 digits necessitates also changing the layouts of certain reports, forms, and screen displays to accommodate the extra digits. A larger complication is that software applications referring to the expanded data would also have to be modified. Also, many early programmers did not label date fields logically — or at all. An automated tool cannot possibly catch every single date field when there is no single method everyone uses to label them.

Regardless of the Outcome, Lawyers Are Going to Make Money on This

Given the scope of Y2K problems, something is bound to fall through the cracks and fail. In fact, lawyers are counting on it. They are already lining up for litigation and hope to cash in on the business of finding those responsible for failed systems and businesses and prosecuting them to the fullest extent. While the Gartner Group (a market research company) has estimated the cost of Year 2000 remediation at $300-600 billion, experts say the cost of litigation will be close to $1 trillion. In fact, it's already started. A New York law firm has launched a class action suit against AT&T and its spin-off Lucent Technologies on behalf of individuals and organizations who use the companies' telecommunications equipment that isn't Year 2000-compliant.

It is ironic the Information Systems folks are the ones under the most pressure for Y2K compliance by the deadline, because these are the same folks, particularly the low-level technical experts, who were trying to get management to address the approaching problem well before the current crisis!

These employees and consultants also present a litigation danger because they are likely to express their unhappiness by blaming management in internal memoranda or E-mail messages — documents that will later be key evidence against the company in Year 2000 litigation. Almost surely these documents are already being created. Therefore, it is important for companies to educate and manage their employees and consultants throughout the process the company undertakes to become Year 2000 compliant.

So what's really at risk? No one knows for sure. There's little chance everything can be fixed in time, given the non-negotiable deadline and the finite number of resources available, especially skilled labor, dollars, and time. Nor is there any good economic or social reason to fix everything.

Murphy Will Pay Us a Visit Soon

Y2K is not a problem that can be fixed with a weekend of programming. But neither should we expect our banking system to collapse. Both these extremes are equally naive. Huge efforts are under way, and many of the problems will be fixed.

What we do know, however, is the scope of this problem cannot be defined accurately, most complex software endeavors rarely work perfectly the first time, and if we have learned anything about large software projects it is that most of them miss their deadlines. The problem is that there is absolutely no scope for lobbying for a new deadline. Hence, Murphy will pay us a visit soon. The question, is how excited will he be? □