Lighten Up

By Joe Schofield

Visions of doom with the advent of the year 2000; The millennial madness associated with the year 2000 does not occur with the year 2000

Countdown to year 2000! Collision with year 2000! Closing in on us: the year 2000! The challenge of the year 2000. The immovable deadline, the unreachable goal, the unmanageable task.

Sound familiar? Are you rattled yet about your software, or are you busy making your reservations for the New Year 2000 worldwide grand tour?

It's only 1997, and most folks are already tired of hearing about the year 2000 problem. Those who have not yet been frightened into salvaging their computer systems within the next three years probably won't be doing so. (For readers who are interested, Figure 1 is a chart showing how many programs will have to be fixed per week in the time that is left. -Ed.)

There is no shortage of scare tactics in industry publications. To conjure up visions of doom with the advent of the year 2000, they use words and phrases like risk; host of problems; big computing problems; skills shortage; time bomb; rising salaries; "bleak [future];" crisis; life threatening; disasters; and survival, like playing "Russian Roulette," or having a gun barrel "pressed against your head."

You should beware of the millennial madness syndrome, however, so often contracted within days of reading in-flight magazines. To turn this terrifying tide of tribulation, I'll offer three quick relief points. First, the millennial madness associated with the year 2000 does not occur with the year 2000.

Instead, a quick review of history indicates that year 1, that is, 1 AD, occurred *after* year 1, that is, 1 BC. History has recorded no year 0. This often-forgotten fact means that a thousand-year millennial period must end after 1000 years; in this case at the conclusion of the year 1000. Necessarily, then, the end of the "second" millennium cannot occur until the *end* of the year 2000; not at its genesis.

Second, it's important to check your calendar. Year 2000, not quite the third millennium, will be ushered in on Saturday, January 1st. Most of the consultants who are profiting from year 2000 hype fail to disclose that companies have an entire extra weekend to deal with the dilemma. While I'm not prescribing that your conversion be postponed until that time, I have seen some real miracles occur over weekends.

Third, consider that some of the same folks raising the flag on the year 2000, and warning of escalating salaries for "legacy programmers" approaching $150 per hour, are some of the same folks who were telling you to get rid of your COBOL programmers just a few years ago. By the way, $150 per line of code change is being suggested by one consulting firm as the going rate for the cost of upgrading [to year 2000 compliance]. [Ed: Others have put the rate at $110 per line.]

Most software engineers would love to work for those wages! Even at the rate of 15 lines of code per day, most programmers could get by on $2,250 per day, not including overtime. (If you see this figure in the next in-flight magazine, you'll know where it came from.)

Feeling better yet? What if you were reassured that many of the forthcoming "catastrophes" will have no impact on your operation in 2000? Here are some of the problems being discussed at conferences, circulating among corporate intranets, and cluttering company E-mail systems.

• ***Existing personal computers using certain BIOS chips will not work*** - I, like many others, just replaced my 486-based home computer at the end of 1996. Do we really think that most of those relics will still be used in critical operations by 2000? How much work will one be able to accomplish in the year 2000 with a 486-chip-based ma-

chine? Will any year 2000 software be able to utilize these machines? Nonetheless, some of these machines are upgradable, ameliorating the impact of advancing calendars.

But take this same set of questions and apply them to personal computers one generation older, the 386-chip-based machines. Sharing the same BIOS dilemma, these too will be unable to accurately execute software in the year 2000. Are you planning on keeping your 386 machines running critical applications in 2000?

Although "solutions" have been suggested, another option might be to buy a real machine within the next three years. We are pleased to report that further insult was avoided when the team that released this data decided not to describe the impact on 8088-chip-based machines.

A couple of popular groupware, netware, and E-mail systems have been identified as having year 2000 limitations. Let's just suppose that you are one of these software suppliers. Does it seem reasonable for you to provide your clients with a year 2000 compliant update? Instead, why not just let all those applications fail; maybe as a marketing ploy, a "we told you so" strategy, or perhaps as an "extortion scheme."

But users should discount the fear of all their off-the-shelf software turning bits to blurs on January 1, 2000. Most of these vendors have upgrades already promised before the turn of the century (which may, however, be a little too close for comfort for those preferring a year's worth of parallel testing).

• ***Windows 3.1's File Manager has been described as misrepresenting 21st century dates*** - Won't we be on Windows 99 by then? Or is that Windows 00?

Inconsistencies in how Microsoft Office Suite products handle dates beyond 2000 have prompted the industry giant to recommend upgrading to Office 97 prior to 2000. Maybe Windows and Office will be combined as WinIce 99 by then.

• ***The inability to sustain legacy software is probably one of the more appropriate areas of trepidation*** - The phrase "legacy software" continues to provoke curiosity. It seems safe to call any software that has been around for more than a decade, legacy software. More recently, the word "legacy" has been applied to anything that was built by the group using the "L" word or, that's not on the "current" platform. However, it may be wise to remind those new to the profession that the code they write today is tomorrow's legacy code. How long will it be before we describe VRML, C++, and Java code as "legacy" software?

As far back as 1982 one of the members of our software development group thrust the year 2000 compliance issue upon the team. None of us really expected the software to be in use by the year 2000 (the software has been migrated and redeveloped at least twice since that time), but we accommodated the year 2000 because we wanted the software to be functional in the event that it was still being used.

More recent systems have addressed the issue, and are year 2000 compliant. With most applications being retired, rewritten, or replaced every five to seven years, how does the year 2000 remain an issue? The reality is that it is not a problem for most systems, but it will be devastating to some that truly deserve the "legacy" description, in that they are outdated, undocumented, and poorly maintained.

Fortunately, many of the problems associated with the year 2000 "problem" will have NO IMPACT on today's information systems. For example, Wal-Mart reportedly acknowledged the year 2000 problem in 1991, and plans to have all its interventions completed this year. Will Wal-Mart's actions further propel the discount retailer's competitive edge in the retail industry?

Also noteworthy are the efforts of those like Kathleen Adams of the U.S. Social Security Administration. The SSA has many date-critical functions, and, in her response to the year 2000 on the Internet, she acknowledges the rewriting of their data bases every two years. Further, Kathleen's teams began the migration to 2000 compliance in 1991! Often, our aged software really isn't.

Much of the information circulating on the year 2000 is distorted, misleading, unfocused, and almost irresponsible. Examples have already been given.

Perhaps one of the hidden dangers of year 2000 has not yet surfaced; at least not until now. Many organizations will lose focus of other significant computing issues as we march toward year 2000. These include reducing the costs of information systems, improving the value for dollars invested, increasing software productivity and quality beyond recent mediocre results, and participating in the electronic commercial usage of the World Wide Web (approaching a $5B industry by the turn of the century).

Other important issues include simplifying, yet enhancing networks; exploiting the potential use of object technology; extending the life of relational and pre-relational data base management systems; planning for the return to mainframe computing (maybe, maybe not); and, of course, planning for Windows 02. (Early rumors hint at an intelligent "undo" with this release which undoes the last thing you didn't mean to do as opposed to the last thing you did.)

Some companies are so prepared for the year 2000 that they will have nothing to do by the time it arrives. With this extra time they might want to consider further distanc-

Years Left to Fix

Number of Programs	3	2.5	2	1.5	1	0.5
2000	13	16	20	27	40	80
5000	33	40	50	67	100	200
10000	67	80	100	133	200	400
15000	100	120	150	200	300	600
25000	167	200	250	333	500	1000
50000	333	400	500	667	1000	2000
75000	500	600	750	1000	1500	3000
100000	667	800	1000	1333	2000	4000

Source: CAP GEMINI AMERICA and Rubin Systems Inc.

Figure 1. Programs to Fix Per Week

ing themselves from the competition. For instance, why not begin working the year 2100 problem: just as a heads up, it's not a leap year. (That means that by 2096, workers will be able to avoid an extra work day for eight years!) And it's not too early to start planning for the following millennium; that's right, beginning with 3001.

For those of you still baffled with what to do as we approach the year 2000, fear not! One popular Internet site has more than 100 consultants to lend you a hand with your assessment and planning. (I'm beginning to think these words mean, "Watch out; cost add-ons ahead.") I'll prognosticate that more time and dollars will be spent over the next three years talking about the year 2000 than correcting remaining problems. (Please note that your reading of this article has just contributed to my point!)

By this year's Fourth of July, about 900 days will remain before the year 2000 arrives. But at $2,250 per day (only qualified legacy programmers need apply), we're sure some folks will be willing to work overtime. No wonder this is a $600B consulting opportunity, a sizable nine percent of the U.S.'s entire Gross Domestic Product. Even at the new going rate for COBOL programmers that's still 6,000,000,000 labor hours, or 2.8M person years.

If you're not feeling a whole lot better yet, consider this closing thought. One source reports that 90 percent of European firms are either unaware of the year 2000 issue, or are doing nothing about it. In Asia only 10 percent of institutions are addressing the issue.

Maybe a two-day weekend won't be enough time for those continents. Maybe even a one-year delay of the entrance of the third millennium will not rescue those who lag far behind. Those who are addressing the year 2000, should retain and sharpen your focus on other initiatives and technologies as well in preparation for prosperity in the coming millennium. The year 2000 may have an immovable deadline, but the opportunities beyond it are still immeasurable.

Joe Schofield enjoys investigating and integrating new technologies into business practices at Sandia National Laboratory. He has made numerous presentations at industry conferences and has authored several articles in information systems journals. He develops and teaches IS courses in the graduate program at the College of Santa Fe.