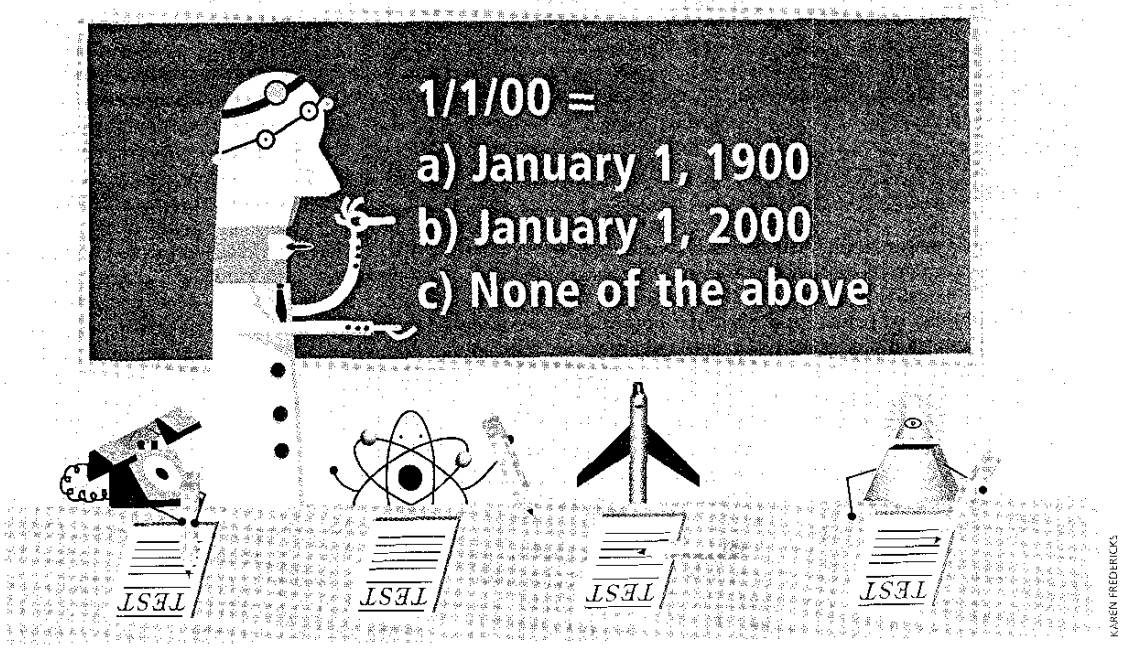
Y2K testing: does success matter?



Even though companies are going all out to ensure their software satisfies Y2K tests, passing grades may not guarantee a secure future

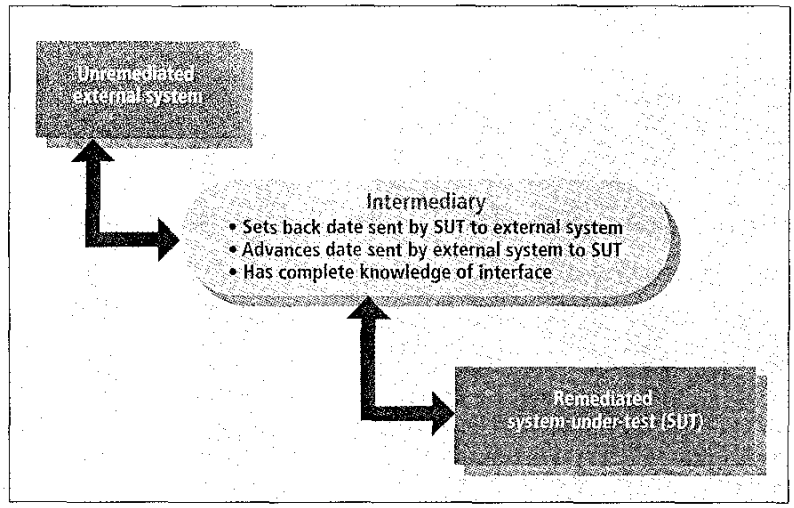
RICHARD COMERFORD
Senior Editor

At the end of April, the Securities Industry Association announced that investors need not worry about Y2K problems. The statement was based on the fact that stock exchanges, clearing houses, and almost 400 brokerages had just completed the largest test ever undertaken by Wall Street.

For six weekends, the group had simulated trading in an effort to uncover any computer flaws that might occur as a result of 1999-to-2000 rollovers [see "Checking up on Y2K," pp. 61–70]. The source of the Y2K problem is software or firmware that uses only the last two digits of the year (for example, '99) to store dates or for calculations. In other words, it assumes that the first two, or century, digits are always 19, which is so only for this century.

In the securities testing, the financiers performed 260 000 mock trades, only four of which were affected by Y2K problems. Indeed, more errors occurred (in 2.5 percent, or about 6500, of the trades) as a result of data entry

0018-9235/99/$10.00©1999 IEEE



[1] Not all computers are ready for Y2K testing at the same time. So to check out the performance of a Y2K-compliant computer in a network, a so-called intermediary is installed between it and any unremediated system(s).

The intermediary's main function is to translate Y2K dates from the system under test into others acceptable to the unremediated system, which it does by shifting the dates back a fixed number of years. Similarly, it takes dates sent from the unremediated system to the one under test and shifts them ahead by the same fixed number. The intermediary must also know how to interpret the formats in which dates are communicated, and adhere to the rules governing the interfaces between systems.

mistakes, miscommunications between traders, or other system problems that had nothing to do with Y2K.

In other industries, U.S. banks, electric utilities, phone companies, and air-traffic controllers have also completed similar tests. While the industrywide tests are attracting a lot of attention now—and are certainly needed to help prevent Y2K problems from affecting key computer-based industries—they are only a part of the total solution. For one thing, these broad tests, sometimes referred to as validation or production tests, are only the last in a series that make up a complicated, multi-step process.

But before any testing can begin, Y2K problems must be fixed. To do this, every single line of a program must be examined to see if it uses a date or dates, and if so, how the information is handled. For example, a 20 000-line accounting program may be known to use dates several times, but exactly where may be unknown.

Source code that has been annotated by its creator to explain its actions is the easiest to work with. But if only compiled code is available, programmers must interpret its machine-readable 1s and 0s. Or they might try converting it back to source code—a difficult process.

To understand source code, of course, programmers must be familiar with the language in which it was written. The older the code, the more likely that it is written in a language that is no longer in use—say, ancient Comtran, or Fortran 1, or even early versions of Cobol. The result has been a windfall for old-timers—programmers who had retired but because of their expertise have been called back and offered lucrative short-term contracts to help in remediation efforts.

Remediation, or plain old fixing of the code, involves any of several techniques. One is "windowing," in which dates are shifted so that years well into the 21st century appear to the computer as being in the 20th century (or 19xx). Another approach changes the code so that it looks for and accepts only four-digit years; in which case changes are also made to all database records and data entry forms so that they include only four-digit years. The International Organization for Standardization suggests using the Y2K dilemma to fix still other dating inconsistencies, including those that occur because of different national styles of writing dates. For instance, depending upon which side of the Atlantic Ocean one is on, 6/6 to 6/9 can be read as a time period of three days or three months. [See *IEEE Spectrum*, "Bad days for software," September 1998, pp. 47–52.]

Only after potential Y2K-problem sources have been identified, problems located, and remedial action taken—all horrendous tasks, especially when huge amounts of undocumented code are involved—can validation testing begin. No wonder the cost of fixing Y2K problems has not been cheap. In the case of stock brokerages, for example, remediation costs are estimated to run as high as US $5 billion for the whole industry; the cost of the six-weekend test is said to have been $100 million.

Further, the time element makes the expenses balloon. Whereas most companies spread the scheduling and expense of upgrading and maintaining their programs over many years, Capers Jones, founder and chief scientist at Software Productivity Research Inc., Burlington, Mass., points out that the cost of Y2K activity is showing up mostly in a single year—this one. For companies that have been ignoring the problem in hopes of finding a "silver bullet" fix, the bother is hitting them all at once.

Layering the test plan

In determining if the remedies have worked, the first requirement is a set of test cases. The basic set should be able to confirm not only that dates are properly processed—that, say, a mortgage program correctly calculates the elapsed time between two dates—but also that the software properly handles the entering of dates, through a keyboard or other means, and their display. These test cases, as well as test procedures, are described in a report titled "Evaluating Systems for Year 2000 Compliance," by Thomas C. Royer, a member of the Y2K compliance team at Mitre Corp., Bedford, Mass.

For instance, as Royer explains, a test case that involves entering a date in the format 01/01/00 should result in the correct four-digit year being displayed. (In

some cases, it may be necessary to enter dates with two-digit years for both 19xx and 20xx without mistakes.)

Another test case is needed for the accurate tracking of leap years. In this instance, the software must ensure that 29 February 1900 not be set or displayed as 29/2/00 because 1900 was not a leap year. But, because 2000 will be a leap year, 29 February 2000 should be displayed. Other valid and invalid dates, such as 30 November 1999 and 31 November 1999, respectively, should also be part of the sample test cases to make certain the system will accept no erroneous data.

To test for correct processing, these same test cases should be employed in two ways: with the system clock set to the current date and with it set to the year-2000 rollover date, 1 January 2000. With the system in the current-date mode, dates before and after that date should be tried. Then, with the clock set ahead to 1 January 2000, the system should again be tested with dates before and after that date.

Overall, the best approach to testing is to start small and grow to a full-system test. A layered test plan is therefore needed to build confidence each step of the way and to find problems when they are easiest to fix. Generally a layered plan consists of six software-testing stages, referred to as unit, component, system, system-regression, pre-production, and production testing.

In the unit testing stage, the goal is to verify that each software unit, or module, that makes up an application has been fixed correctly and performs the function it was meant to. If each module works properly, then the whole component, or application, is checked to see that it meets its specifications. Then, if the application is up to spec, system testing is performed to ensure that that application will work, and be compatible, with other software in the system.

Comparison with past programs

Up to this point, the purpose of testing has been to establish essential functionality. Regression testing entails running the application over time with several sets of data to make certain that the application does not cause system failure. To this end, the data used is taken from past transactions (hence the term regression testing), so current results can be compared with those that were known to be correct. If regression testing succeeds, pre-production testing can begin.

In pre-production testing, the goal is to simulate as fully as possible how the system will operate in the real world, even though the rest of the world may not be ready to simulate future performance. In such instances, an intermediary to simulate real-world systems must be used. The intermediary is a computer that checks out the performance of a Y2K-compliant computer, verifying that it will function normally [Fig. 1].

For production tests, real-world systems—that is, those that have been configured to act as they would in everyday operation, without intermediaries—are set to perform routine tasks as they would in the 21st century. These tests, the final phase in testing, are typically conducted with much fanfare because people who perform them have a good deal of confidence in their success. Usually the tests are announced beforehand and their expected good results trumpeted.

Production testing is the tip of the Y2K iceberg. Highly visible, these tests give a sense that all is well while masking the tremendous amount of test work that lies beneath their success. Indeed, companies and organizations that are not already in the first phases of unit and component testing may not be able to reach this final phase before that irrevocable deadline.

Home safe?

Still, beneath that visible tip lurks danger. Even spending large amounts of money and exerting massive efforts to fix and test all that program source code is no guarantee that a company's software is free of Y2K problems. Software Productivity Research's Jones points out that, historically, "about 30 percent of the problems with software aren't found until after the software is released" for production use. "There's no reason to suppose that the case will be any different for Y2K [-remedied software]," he told *Spectrum*. Thus, while the successful results of final tests may allay fears, in reality faults are still likely to occur with the date change.

A fear expressed by Ed Yourdon, chairman of software consultant Cutter Consortium, Arlington, Mass., is that the jubilation surrounding the successful completion of production testing may distract people from the work that still lies ahead. "The steady drumbeat of upbeat reports and predictions from industry associations and the government," Yourdon said, "has created a state of near euphoria....My concern is that the companies and government agencies issuing the euphoric press releases are tempted to believe their own messages so completely that they abandon much of their risk management and contingency planning perspective."

Risk management, or minimizing the risk that something could go wrong, was on the minds of the powers that be in the Securities Industry Association. Though its tests were successfully completed, the association issued a report proposing additional steps—such as having mutual funds make their customary year-end payouts before Christmas (rather than at the year's end) and having all U.S. financial markets close in the early afternoon on New Year's Eve [see To Probe Further, below]. E. Gerald Corrigan, managing director of Goldman, Sachs & Co., New York City, and head of the committee that wrote the report, noted that such risk management was "an investment in the future with a very low cost and a very high payback."

Contingency planning—that is, planning to minimize the bad effects when something does go wrong—must also get its share of attention. Banks, for instance, could just plan to have extra tellers on call to work in branch offices where good supplies of cash are on hand, just in case those ubiquitous automated teller machines fail to respond on not-so-far-off New Year's Day. ◆

To probe further

The World Wide Web is replete with Y2K sites. That of Mitre Corp. (http://www.mitre.org/technology/y2k/) is extremely well organized and covers the whole range of Y2K issues that must be addressed. It even has a section called "Steps to take NOW," which is frequently updated. The layered test plan referred to in this article, "Evaluating Systems for Year 2000 Compliance," can be found on the Web at www.mitre.org/technology/y2k/docs/TEST_EVAL.html.

More information about the Securities Industry Association's suggestions for contingency plans can be found on the Web at www.sia.com/html/year_2000_intro.html; clicking to accept the agreement posted on that Web page gives access to Y2K information and the report, "Securities Industry Association Report of the Ad-Hoc Committee on Y2K Contingency Planning," referred to above.