Managing the Millennium Bug

Ali Reza Montazemi

As the year 2000 approaches, software companies around the world are grappling with a new challenge: the millennium bug. This paper by Montazemi discusses the background and the issues involved and suggests a plan of action for surviving the year 2000 computing crisis.

Ali Reza Montazemi is Associate Professor of Information Systems at the Michael G DeGroote School of Business, McMaster University, Hamilton, Canada.

References are at the end of the article.
of an internationally accepted date standard [2]. For example, if a date is seen out of its context, it is not clear whether 2/6/97 means the 2nd June, 1997, or the 6th of February, 1997. Although an ISO-8601 standard advocated use of the yyyyMMdd system, the standard has been ignored by many system developers. Inclusion of the date in a program has been left to the discretion of individual information system analysts and programmers, which makes it impossible to develop a standard operating procedure to make a date change across the board.

The second issue is that a year is represented only by its two last digits. This convention was adopted during the 60s and 70s to save on computer memory space. During these two decades, main memory (i.e., RAM and ROM) and secondary memory (e.g., disk space, magnetic tapes, and punched cards) was at a premium, resulting in a strong incentive to reduce memory cost by removing redundancies. Although the cost of computer memory has declined significantly during more recent decades, nonetheless, to save on the cost of revamping old database structures, modern convention has generally been ignored by business firms.

The third issue is the black-box effect of application systems. Many application systems have been poorly coded (e.g., spogetti coding) without appropriate documentation. A black-box effect is the result, making it difficult to know the structure of a transaction system — even a system as simple as one that controls inventory. The result is that programmers and analysts must manually work through a program's computer codes to assess the overall effect of date changes on the rest of a system. Considering that a Fortune 500 company could easily have 100 million lines of code, with an estimated workload of 100,000 lines of code per programmer year, it will take 1,000 programmer-years to fix the millennium bug. Add to this the millions of lines of codes developed by end users during the past two decades, and we will begin to realize that the total cost of fixing the millennium bug will be astronomical.

The fourth issue relates to the modern convention in information technology to integrate information systems to improve usability. On the one hand, this practice removes problems associated with "islands of automation" (i.e., stand-alone application systems). On the other hand, it means that a bug in one system can create an unpredictable effect on those systems with which it is integrated (i.e., a chain effect). For example, electronic data interchange (EDI) is used by many businesses to link order processing and production scheduling between different customers and suppliers. A bug in the order processing system of one company could easily affect the production scheduling system of one of its suppliers.

The fifth issue concerns the accounting convention that typically treats expenditure on software as an expense in the period it was incurred. Therefore, making the cost of maintaining an information system over a certain period "year-2000 compatible" would have an immediate negative effect on a company’s balance sheet. As a result, some companies have delayed maintaining their information systems in the hope that a possible breakthrough in fixing the millennium bug might reduce the cost. However, analysts and programmers working on the problem are in short supply, and the cost of maintenance is increasing as we get closer to the year 2000. Delaying action reduces the window of opportunity for firms to become year-2000 compatible, and thereby increases the chance of future liabilities to be incurred because of the millennium bug.

Despite all the negative ramifications of the millennium bug vis-a-vis organizational activities, only a small percentage of businesses has developed a course of action to deal with the problem. It is estimated that less than 35 per cent of businesses in North America, and less than 10 per cent in Europe, have taken steps to deal with the problem. Japanese and Asian businesses are thought to be lagging even behind the Europeans [2]. Lack of action can be attributed to many factors: management ignorance, reluctance or inability to fund a maintenance budget, or even the hope that the problem will disappear by the advent of new software. This head-in-the-sand attitude could result in significant cost to a company, and make a CEO directly liable for negligence.

**Course of Action**

Companies need to develop a course of action. Senior management should create an action plan to identify strategies for dealing with the year-2000 compliance project. It could dedicate a year-2000 working group, composed of a senior systems analyst and knowledgeable business unit personnel who understand the scope and potential criticality of the problem, to work on fixing the millennium bug. This working group should be charged with identifying and assessing risky application systems currently being used in the company: legacy systems, custom made application systems on microcomputers, coding systems developed for shrink-wrapped products, end user-developed applications, operating systems, and BIOS are targets. Additionally, the working group should locate and
identify all licenses and associated agreements for each hardware component and each software program; conduct a legal analysis of each agreement, including its scope, warranties, representations, limitations on liability, and other key provisions. Third party vendors, who may have or share a legal responsibility, should be identified and charged to participate in solving the problem or contribute to defraying the cost of fixing the problem by appropriate legal notice in writing. Ideally, the end deliverable of the working group should be a document that outlines the risk profile of systems (i.e., possible confounding errors, extent of future liability arising from erroneous execution of software packages) as well as a cost estimate of ensuring systems year-2000 compliance. This document would be invaluable to senior management, who could use it to develop strategies for project scheduling, budget allocation, and to determine the legal ramifications of possible failures to become 100 per cent year-2000 compliant. The outcomes described in the document could also be of great use to the Information Technology (IT) unit when it performs system maintenance. Additionally, based on the document, training programmes could be developed to educate end-users about the millennium bug and to describe means of rectifying the bug in end-user computing environments.

It is the IT unit that should be charged with identifying a strategy for system maintenance: should it be in-house sourced or out sourced. Out sourcing can be an attractive alternative, especially if the IT unit has committed its resources to ongoing operations and new systems development. If, however, a decision is made to do the job internally, software tools can assist. Tools fall into four functional groups: analysis tools, conversion tools, testing tools, and integration tools. Analysis tools are programs that input source codes and attempt to identify the degree and complexity of date dependence for each module [3-8]. The results of the analysis are the raw data necessary for planning and prioritizing the maintenance operation. Conversion tools partially convert existing source codes automatically [7-12]. For example, they can process one million lines of source code in two days with an 80 per cent hit rate. Analysis tools can be called upon to effectively schedule programmer time for fixing the rest of the conversion. Testing tools [It-14] will consume 30 per cent to 50 per cent of the millennium year-2000 project resources because testing is a crucial step in preparation of year-2000 system compatibility. After testing, the company will be left with two versions of the same system (which probably cannot be intermixed, communicate with each other, or share the same database). These two systems can best communicate through bridging communications written specifically for that purpose. At this stage, integration tools [10, 13, 15] can be used for version control, change-tracking, and audit trail.

References

Further Reading