Towards the close of the current century, all data processing systems in the world are expected to cause havoc by producing inaccurate results while dealing with data involving dates (Jager, 1993,1997). This crisis arises because most of the existing systems, both hardware and software, represent the year number in a date through only two digits, making a program blind to the distinction between the years in two different centuries. At first sight, it is hard for us to believe that the entire computer community has committed this mistake, an act which, as we can see today, was outright careless and short-sighted. However, the choice of two digit years does not seem so thoughtless when we consider the scarcity and cost of computers, particularly the exorbitant cost of computer storage, at the time when the decision was made. The foundation for most of what we call today as legacy applications was laid in the 50's and the 60's, and to save storage cost, it was a common practice for programmers at that time to minimize the length of the data items. As the system designers of those days did not expect their systems to last till the end of the century, they thought it extravagant to use four digits to represent an year, when they could do with just two digits. Thus, the entire world has come to get stuck today with the two digit year systems, and what is worse, there does not seem to be a single stroke solution to this problem. Many users are, of course, still waiting for the arrival of a magic chip, which when pushed into their PCs would immediately make the machines date friendly; but, alas, their dreams are not destined to come true. In this crisis, the only way out is to locate and fix each faulty line of each affected program individually. The sooner an organization accepts this stark reality the better it is for its business.

The problem of a system not being able to correctly process the dates in year 2000 and beyond is called the year 2000 problem, or Y2K for short; the other terms that refer to this are the Year 2000 Crisis, the Time Bomb, and the Millennium Bug. Similarly, modification of an existing system to correct the Y2K problem is referred to as century date change, Y2K conversion, Y2K remediation, and Year 2000 date change. A system which is free from the above bug...
is called as Y2K compliant, and a system which is not as Y2K non-compliant.

The consequences of using Y2K non-compliant systems are truly alarming. A few possibilities are listed below:

- In 1998, a bank automatic teller machine (ATM) rejects a teller card on which the expiry year is recorded as 01. In reality, the expiry year is 2001, which is recorded in the computer as 01 because of the two digit limitation. But, the computer, which is processing the card in the year 1998, concludes mistakenly that, as 01 is less than 98, the card has already expired.

- In the year 2000, a payroll program which is supposed to deduct tax according to the new rules introduced in 1995, mistakenly computes and deducts tax according to the old rules. The reason is that year 2000 is represented as 00 in the system, and as 00 is less than 95, old tax rules are applied by the computer.

- Suppose the date of an invoice, corresponding to a pending receivable, is 01 Jan 1999. Suppose a computer program is scanning all the pending receivables on 01 January 2000, to send penalty notices for the buyers whose payments are long overdue. As the age of an invoice is determined by subtracting the invoice date from the current date, and as the year numbers are represented with two digits, the age of the current invoice would be computed as 00-99=-99 years. This is an absurd value. Depending on how the system is programmed, either the system would stop with an error message, or, ignoring the minus sign, would interpret the age as 99 years, and would impose a ridiculously large penalty on the buyer.

- Suppose, in a country, a new type of currency was introduced in the year 1990. All transactions occurring on or after this date should be processed according to the new currency. However, the year 00, which is actually supposed to stand for 2000, would be interpreted as occurring before 90, and hence the transaction would be wrongly processed using the old currency format. An old lady born in 1894, and still living in 1998, receives an invitation from a county school to join their kindergarten class.

Among the worst hit by this crisis are banks, financial institutions, and insurance companies, which have already begun feeling the tremors of the impending doom (Cohen), because the data processing in these organizations is date intensive. Furthermore, these organizations deal with projections involving dates in far off future. In fact, financial institutions were the first to respond to this crisis by launching efforts to remedy the Y2K problem. Even in India, where the intensity of computerization is much lower, and computer applications are of relatively recent origin, a few banks and insurance companies have already set up Y2K remediation project teams. But, most of the world's Y2K conversion work is related to the software belonging to North American and European corporations and governments. Software export houses of India, Israel, and Ireland are the major contractors for this work.

No doubt, in a crisis like this, doomsayers have their field day, and a certain amount of exaggeration in predictions about the crisis is inevitable. Still, even if we go by only the most optimistic estimates, the devastating power of this crisis, and the expenditure and efforts needed to fight it seem to be staggeringly high. It is estimated that about 90 per cent of the computer applications are going to be affected by the Y2K problem; about 75 per cent of these are going to be legacy systems, mostly written in COBOL, and the remaining 25 per cent client-server. The total cost involved in rectifying the problem all over the world is anywhere between US $ 300 billion to one trillion; an equal amount might have to be spent on related damages and litigation. In COBOL alone, the amount of code that will have to be dealt with was estimated to be 180 billion lines; the total number of COBOL programmers needed for this task would be 1.1 million. It is to be noted with concern that even if we spend adequate resources on this activity, the conversion efficiency would be only about 95 per cent; that is, when the new century arrives, the lurking bugs will come to surface, causing a lot of chaos. Yet, in spite of the urgency of the problem, only about 50 per cent of the corporations seem to be addressing themselves seriously to it. There is a danger that if the Y2K conversion does not take place in time, about 20 per cent of the companies will go out of their business. Still, about 20 per cent of the managers are highly ignorant of the problem1.

The objective of this paper is to help interested managers appreciate the relevant aspects of the Y2K problem and the choices a manager encounters in guiding his or her organization to solve the problem. Most of the paper discusses the different phases of a Y2K conversion project. Additionally, it highlights other relevant aspects of the Y2K crisis such as its

---

1Awareness phase, http:///nfosphere.safb.af.mil/-jwic1/fadl/world/aware.html#deci
legal and human resources problems. Most of the estimates given in the paper are taken from the web sites of various US corporations, US government, and independent IT consultants. Such data is not entirely irrelevant to an Indian reader because most Y2K work being handled by Indian consultants is related to systems belonging to north American and European corporations.

In section 1 of the paper, a Y2K project is divided into five phases and the manager's role in each phase is discussed. Section 2 discusses the role of Y2K tools and their impact on project productivity. Section 3 outlines the way each kind of software and hardware users and vendors are going to be caught in litigation arising out of the Y2K problem. Section 4 describes the human resources constraints being faced by the Y2K solution providers and the common practices adopted by them to overcome these constraints. Section 5 discusses a variety of similar problems slowly surfacing along with the Y2K problem, and the implications of their existence to the business strategy of solution providers. The concluding section lists a few lessons to be learnt by the IT community from the Y2K crisis.

1. Y2K Conversion Project: Different Phases

Y2K bug fixing requires, in addition to technical skills, project management skills of a high order. Because of several reasons, to be discussed later, most organizations of Europe and US are contracting out the conversion work to third party Y2K conversion consultants, also called Y2K solution providers or Y2K conversion vendors, located in countries like India. Such vendors located in a country different from that of the software owner are called off-shore vendors. An off-shore vendor performs the conversion project with two teams, an off-shore team and an on-site team. The off-shore team, usually the larger of the two teams, operates from the vendor's software factory, whereas the on-site team stays with the software owner and acts as an interface between the customer and the off shore team. The software owners themselves have to make critical decisions before the conversion project begins, during the negotiation of the conversion contract with the solution provider; they should also closely monitor the project throughout its duration.

The major phases of a Y2K conversion project are: 1. Building awareness, 2. Listing the inventory of various hardware and software items possessed by the organization, 3. Assessing the work involved and planning for the project, 4. Renovation of hardware and software, 5. Testing and validation. While the relative effort demanded by each phase varies widely between projects, the estimated ranges for the effort are: 1 to 10 per cent for the awareness phase, 5 to 10 per cent for the inventory phase, 15 to 25 per cent for the assessment and planning phase, 10 to 20 per cent for renovation, and 15 to 50 per cent for testing. We describe below the activities performed and decisions taken in each of these phases.

Awareness Phase

Even though a lot is written on the Y2K problem, not only in the technical journals but also in the popular press, the amount of publicity given seems inadequate, because, still, a surprisingly large number of senior managers are said to be either unaware of this problem or are not convinced about its seriousness. Therefore, the first challenge faced by a Y2K champion in an organization is to give the top management adequate exposure to this problem. The Y2K champion has to convey to the top management what this problem is all about, its impact on the organization, and the action needed to solve it. If the top management and the various functional heads are not adequately aware of the problem, there would be no sense of urgency to tackle the problem, action to solve it would be delayed, and the resources sanctioned for the remedial action would be inadequate. Thus, the objective of this phase is to obtain top management's approval to initiate the Y2K remediation project.

This phase is essentially a phase of meetings, seminars, presentations, and committee formation.

During the various presentations, the following facts and possibilities should be impressed on the top managers and key decision-makers:

• Y2K is not merely an information system problem, but is a business problem.

• Time is of utmost essence in deciding to act; wrong results may start occurring much before the turn of the century.

• Not merely one type of application or one type of system, but all types of software and hardware, including embedded systems, are susceptible to misbehaviour because of this problem.

• Dates are also used in non-date related calculations such as encryption and random number generation and a wrong execution of these calculations may lead to security breaches.

Not all the source code that the organization may need for fixing this bug may be readily available; the missing code has to be generated from the corresponding object code/ and this itself might form a significant component of the Y2K project.

Of the several teams that need to be formed for planning, monitoring, and executing the project, the two most important ones are the project team and the organizational steering committee, and they need to be constituted at the earliest possible opportunity during this phase.

Members of the project team should have technical expertise in databases, programming and systems analysis. They should also be possessed with communication skills, organizational knowledge, and knowledge of interaction among the different systems. The steering group should consist of representatives of top management and other key organizational personnel.

At some point, it is wise to let some important business partners like suppliers and customers also participate in some of the Y2K committees, because, the partners, being some of the most affected parties, would be able to help the project and steering committees in spelling out the scope of the Y2K project, and in setting priorities of system conversion.

It is desirable to produce, at the end of the awareness phase, a high level management plan which includes a brief statement of the Y2K problem from the view point of the organization, Y2K project objectives, Y2K conversion management strategy, and assignment of responsibilities for this conversion.

Assessment Phase
Once the top management approves the proposal to initiate Y2K conversion, the next task is to estimate in detail the scope and magnitude of the conversion project (Freeman and Meador, 1997). To be answered right at the beginning of this phase is the question which of the several hundreds of systems owned by the organization need to be converted and how. Along with the conversion of the programs the related data bases, and some times even the embedded systems need to be changed. Hence, a detailed list, called the inventory, of all the programs and the associated utilities has to be first prepared; and then the dependencies between the items of this list have to be analysed. Preparation of such an inventory is quite a tedious, time-consuming, and frustrating task; it usually involves rummaging through the old cupboards, ancient file folders, and discarded tapes, and peering at programs written in outdated languages by obscure authors. Over the years, even a mediumsized company accumulates so many systems that it is quite a daunting task to find out their details. The only solace in performing this task is the possibility of discovering that a lot of code in the inventory is no longer of any use to the organization. Such code is called dead code. In fact, 40 per cent of all the code possessed by organizations is estimated to be dead code, and one should lose no time in discarding it. Many programs with missing source code also would be discovered during this phase.

Once the inventory is available, the next step is to decide which systems need to be converted before the dawn of the new century, and which systems can be taken up for conversion later, and lastly which do not need any conversion. Mission critical nature of the systems needs to be considered in making this classification.

During this phase, the organization has to decide to whom the conversion work would be assigned. As mentioned earlier, several US, European, and Canadian companies tend to award the Y2K projects to offshore conversion vendors. What weighs heavily in favour of the off-shore conversion strategy is the cost factor. The cost of a programmer in the US is estimated to be US $ 65 per hour, and this increases at the rate of 10 per cent per year. This cost in India was only US $ 12 about three years back, and US$ 25 to 35 today. What is, however, against the off-shore strategy is the security aspect of the project. During the conversion, the entire source code of a system has to be laid bare before the conversion vendor, and further, the code is exchanged several times through communication media like telephone and satellite. For this reason, systems of defence departments are rarely given out to off-shore vendors. Even if security is of paramount importance in dealing with a system, its Y2K conversion can still be awarded to off-shore parties, provided the off-shore site has adequate facilities for software and hardware locking and fool proof encryption and decryption.

Another alternative that most organizations would like to consider at this stage is re-developing their systems from scratch instead of converting the existing ones. For even a mediumsized organization, it is not feasible to begin a project of system re-development now and finish it by the year 2000, because the time available is too short for such an effort. Such replacement is possible only when the size of a system is less than 1000 function points, whereas the size of systems in a medium organization is estimated to be around 500,000 function points (Jones, 1997).
For managing the work flow between the software owner and the conversion vendor, the system under consideration is divided into several modules, called upgrade units. An upgrade unit is converted by the conversion vendor and delivered to the owner as one unit. Before beginning work on an upgrade unit, certain set up activities such as creation of directories, creation of databases for testing, creation of directories for backup are to be performed. An upgrade unit should neither be very small nor very big. Large upgrade units reduce the possibility for parallel execution of activities by the contractor and on-site teams. As an upgrade unit is delivered as a whole, large upgrade units would lead to delay in integrating the converted code into the production system. On the other hand, very small upgrade units lead to frequent set ups and as a result, high set up and communication costs. Upgrade unit formation lies at the heart of making an initial project network, which is used extensively in project planning and monitoring. The factors that need to be considered while forming upgrade units are as follows:

- If a sub-module or program is still under regular maintenance, it is desirable to reduce the interference between the regular maintenance and Y2K conversion.
- Some modules may be required to become Y2K compliant urgently, because of reasons such as pressure from customers.
- It is desirable to combine programs into upgrade units such that the interaction between different upgrade units is very low.
- As far as possible, the functional grouping of the modules should be preserved; for example, software belonging to finance function should be formed as one upgrade unit and software belonging to purchase function should be formed as another.
- All the different upgrade units should be approximately equal in their level of difficulty of conversion.

The cost and effort of conversion are also affected by the method of conversion adopted to fix the Y2K bug. There are several choices available even for the method of conversion and a careful evaluation of these forms an important step of the assessment phase.

**Choice of a Conversion Method**

We outline below the major alternatives available for handling the new date requirements. It should be noted that (i) not all alternatives are applicable for all systems, and (ii) different alternatives require different efforts and costs (Glass, 1997; Lefkon, 1997; Brigham, 1997; Wagner, 1997; Martin, 1997; Eldridge and Louton).

- **Database Expansion:** Ideally, in making a system Y2K compliant, each data field in each database should be expanded to accommodate explicitly two digits to hold the century number; that is, if a date field is currently represented in the format DDMMYY using six character positions, the field should be expanded by two more positions, assigning it eight positions in all, to convert it to the format DDMMYYYY. After this expansion, the databases should be reloaded, and the logic of the programs operating on these databases should be suitably modified. In reality, however, expansion of each data field may not be a practical idea. Only those fields which form a basis for sorting are expanded first. The drawback of the date expansion strategy is that it makes the systems sluggish to process, especially in operations such as taking backups. The conversion itself will turn out to be quite time consuming.

- **Windowing:** If the dates being processed by the data processing system of an organization span only a few decades, not exceeding a century, then the century to which a date belongs can be inferred from just its two digit year field. For this, what is known as a pivot year has to be specified. Suppose, for example, that the pivot year is 80. This means that the years 80 through 99 are to be interpreted as belonging to the 20th century and the years 00 through 79 to the 21st century. Surprisingly, a large number of computer applications are found to be capable of being solved by the windowing technique. For example, many invoicing applications deal with data that belong to a time range not longer than 10 years. In this method of conversion, only the logic of the programs gets changed, while the databases do not undergo any change. However, one drawback of this method is that it is not valid forever. Sooner or later, the converted systems need to be converted again. Especially after the end of the time window, the logic becomes invalid. In some systems, the pivot year is fixed whereas in some others, the pivot is incremented by one every year. The former method is called static window technique, and the latter sliding window technique. Windowing is simpler to implement than date expansion.

- **Encoding:** Without introducing two more digits explicitly in an existing date field, it is possible
to represent century data, provided the coding system for representing a date is changed from the current decimal system to hexadecimal or binary. It is easy to see that, if we use binary system, six character positions (i.e., six bytes) are enough to represent day numbers spanning a few billion years. Hence, this encoding approach provides a solution which lasts almost as long as the humanity lasts. But, the method has a catch associated with it. First, it requires database conversion, even though no database expansion. Second, the logic of the programs needs to be changed to encode and decode the dates. Thus, it is no less expensive than the other methods of conversion. Further, this method is likely to slow down the processing speed of the affected programs.

- **Encapsulation:** One little known property of our calendar is that it follows a 28 year cycle; that is, this year’s (1998’s) calendar is the same as the calendar 28 years back (1970), in terms of the correspondence between dates and week days. Similarly, the calendar for year 2000 will be the same as that for year 1972. Therefore, the necessity for a data processing system of facing the dates of the 20th century can be postponed for a few more years by setting back each date in the data processing system by 28 years. However, the subtraction of 28 years from each date that enters a program requires the setting up of software bridges. Reformatting of the dates in the outputs also requires similar efforts.

- **Methods Used in Actual Practice:** A shoe manufacturing company using a client server platform adopted the ordinal number approach in which a date is represented as the number of elapsed dates since January 1, 1900, in a six digit space, that is, as DDDDDDD. Previously, the date representation was according to the format YYMMDD. The reason for choosing the ordinal number approach was that windowing was not applicable to their systems. Date expansion was ruled out because the positions of fields adjacent to the date fields in databases would be affected by date expansion.

An oil and chemicals company for converting a large number of COBOL programs running on IBM MVS platform adopted the date expansion approach. As VS COBOL did not handle four digit years, they had to first convert VS COBOL programs to MVS COBOL which handled four digit years. An MRI imaging center for converting a large number of COBOL programs running on IBM MVS platform adopted the date expansion approach. As VS COBOL did not handle four digit years, they had to first convert VS COBOL programs to MVS COBOL which handled four digit years.

A security dealers’ association preferred to adopt different conversion approaches for different systems. For a database with a short shelf life, windowing approach was found suitable. Databases with longer lives were subjected to date expansion method.

A life insurance company adopted a 100 year window from 1950 to 2049 in converting their systems. Years less than 50 were interpreted as belonging to 21st century whereas those equal to or greater than 50 were interpreted as belonging to 20th century. Date field expansion was necessary only for accounting applications which formed a small percentage of the total code.

A railroad company used a combination of a hexadecimal date standard and a 15 year sliding window in their Y2K conversion.

A health insurance company decided to use windowing approach for most of its Y2K conversion. As it is possible for an individual to live for more than 100 years, the date of birth field wherever it existed was expanded. Other data fields were not expanded because it was felt that the risk and cost of such expansion would be excessive.

**Code Conversion**

Code conversion is a technical task performed by software professionals. During this phase, the defective code is located and modified. Location of defective code is a challenging task, especially when the concerned software professionals are not completely conversant with the functionality of the program. Often, a program, before conversion, has to be ported to the platform on which the conversion is taking place; and, after conversion, the reverse has to take place. The conversion vendor should be careful not to introduce any new bugs while removing the old ones.

In performing this phase, the project managers should set up an effective and easy to use monitoring system to measure the productivity of the software professionals and to give feedback to them on their performance. Further, the progress of the different activities should be kept track of and corrective actions to meet the deadlines should be taken whenever needed.

**Testing**

The testing phase consumes about 40 per cent of the total time of a Y2K project. It is rather unusual for a software project to be so intensive in testing. The reason for this high requirement of testing is that the total number of lines modified in a Y2K conversion
would be only about one per cent of the number of lines forming the code, whereas almost the entire system is to be dealt with during the testing. There are several peculiar problems associated with Y2K testing\(^3\) (Chabbouni, 1997; Jager, 1997).

- The same test case fed at different times to the same program may yield different results, if current time is one of the variables affecting the calculations.
- Regression testing, that is, testing the new code with old data is necessary but not sufficient.
- Testing the converted code with future dates is a must; however, one may not know the expected results during this testing.
- As the functional specifications of the systems under conversion are usually not fully available, detailed testing of the logic using the principles of white-box testing is not possible. Instead, peer review, which is review of the converted code by programmers other than the one who edited the program, is more extensively relied upon than usual.
- When the system involves on-line data to be fed through interactive screens, the testing by the offshore team becomes rather cumbersome. A certain part of testing is done by the off-shore team whereas the remaining by the owners and the on-site team.

The managerial concerns related to this phase are: specification of test cases, delineation of responsibilities between the different teams during testing, when to declare a converted upgrade unit as acceptable, and specification of a proper testing environment.

There are several hazards associated with testing when the system date is rolled forward to simulate future dates. The future date might trigger disastrous events: some important data files which are due to be retired on that date might get wiped out; passwords and some software licenses might expire. The best way of averting such disasters is to set up an isolated system and devote it completely to Y2K testing so that the testing does not interfere with other running systems.

Extensive checklists for Y2K testing have been published. These checklists are based upon the dates which can trip Y2K non-compliant systems. These are: dates just before and just after the end of the current century, the dates December 31, 1999 and January 1, 2000, and surprisingly, the last date of February 2000. It is to be noted that the year 2000 is a leap year whereas the year 1900 was not. Hence, if year 00 in the computer system is interpreted as 1900 then the last date of February of that year would be 28th and if year 00 is interpreted as 2000 then the last date of February of that year would be 29th.

II. Tools and Productivity

If each of the above phases were to be executed manually without computer support, it would be absurd to attempt Y2K conversion, except in the case of tiny systems. Fortunately, several computer utilities, called Y2K tools, are available to support each phase of a Y2K conversion project. Several vendors market these tools and train their customers in using them. While there is at least one tool related to each phase, no single tool seems capable of supporting all the phases (Newcomb and Scott, 1997; Zvegintzov; Freeman, 1997). Further, different tools work on different platforms. Therefore, while selecting tools for a Y2K project, the customers must keep in mind the fact that a mix of tools is unavoidable and that there should be a good match between the functionality of a tool and the conversion phase in which it is intended to be used, and between the platform on which the tool works and the platform the customer uses.

The various functions performed by Y2K tools are:
1. inventory analysis, 2. generation of source code from object code, 3. identification of date variables and faulty lines in a program, 4. correcting a code, 5. test case generation, and 6. project management.

Some conversion vendors develop their own tools. In such cases, the tools are developed by assuming certain requirements on the part of the prospective users. However, once the tool is in use, its performance and the way it is used should be constantly monitored so that the tool can be improved in its features and efficiency.

One of the tasks that is not supported properly by Y2K tools is the walk through method of testing adopted during peer reviews. Automated support to this phase is important, because, as mentioned already, most of the off-shore testing belongs to this category; moreover, more than 90 per cent of the conversion mistakes are usually caught through peer review. The functions that can be relegated to a tool during the peer review of a code are: 1. storing of checklist items, 2. navigating the code, 3. highlighting the piece of code which needs to be checked next and displaying the specific check to be applied, 4. keeping a log of the mistakes caught, their nature, details of the author, reviewer and the time at which the bug was noticed.

5. support in updating the checklist, and 6. giving feedback to programmers on their performance in code conversion and in code review.

In fact, if some of the checks contained in a checklist can be applied automatically by the tool itself, the productivity of testing would improve even further.

As the Y2K deadline is fast approaching, there are only two alternatives for completing the massive amount of pending work: 1. by increasing the number of professionals devoted to this task, 2. by improving the productivity of existing professionals. Between the two approaches, the first is clearly not possible because of a shortage of suitable professionals. The second alternative becomes possible if the Y2K tools are enhanced as mentioned above.

To emphasize the importance of productivity, let us analyse the data of a typical conversion project. Suppose this project consists of 3 million lines of code. Suppose a conversion vendor has accepted the project at the charge of 60 cents per line of code, and has estimated that the project would consume 25 man-years of effort. Let us assume that 20 per cent of the work involved is related to code correction and 40 per cent to testing. This means 5 man-years would be consumed in code correction and 10 man-years in testing. Suppose, the tools being used are improved such that they bring about 10 per cent increase in productivity. This means that, in renovation, 5 man-years would be saved, and, in testing, 1 man-year would be saved. Thus, in the whole project, the saving would be 1.5 man-years, which means $100,000 more earnings for the conversion vendor because the man-years saved can be employed in another project.

III. Legal Issues of Y2K
Towards the beginning of the next century, Y2K problem is certain to spark off a barrage of legal battles involving vendors of non-compliant hardware and software, directors and officers of organizations owning non-compliant software, Y2K solution providers, certification agencies, and software and hardware selection consultants. The legalities related to the possible liabilities of the above parties are still uncertain. However, it is certain that the Y2K litigation will cost as much as the Y2K conversion itself. We examine below the legal issues pertinent to different parties in the Y2K affected software (Reid and Brower; Cirillo et al.; Appleton, 1997).

• Software Vendors of Non-compliant Systems: The major point of controversy related to this category of vendors is whether a vendor is liable for Y2K non-compliance when such non-compliance is observed and disputed in a court of law, long after the warranty period for the software sold by the vendor was over. While a warranty period extends over only a few months, the statute of limitations for fraud extends for several years. Therefore, how exactly the Y2K bug is classified in legal terms determines the kind of liability that a software vendor has. In future, software vendors would do well to incorporate unambiguous disclaimers in products whose Y2K compliance is in doubt.

• Directors and Officers of Corporations Owning and Using Y2K Non-compliant Software: When the Y2K bug is expected to have serious adverse impact on the business of a corporation, especially on its financial performance, it is imperative on the part of the directors and officers to inform the employees and other stakeholders about the implications of the bug, and to take action in time to rectify the errors and to report the progress in conversion regularly in their financial statements. Failure to do so might render them personally liable.

While awarding software contracts and purchasing software in future, it is advisable for an organization to explicitly incorporate Y2K compliance requirement in their agreements.

• Y2K Solution Providers: A Y2K conversion contract has to clearly specify the responsibilities of the Y2K solution provider. For example, the contract should clarify the responsibility of the solution provider for bugs and problems likely to be encountered after the converted product is accepted by the customer. Further, it should be specified whether the solution provider is responsible for the non-date related bugs that may surface later or during the course of conversion. Y2K conversion vendors end up scanning and tampering with a whole system, and this makes them vulnerable to unexpected litigation even though it may be related to non-date related bugs.

Closely related to the legal aspect is the insurance aspect of the Y2K problem. The solution providers are many times required by their customers to cover themselves with appropriate insurance, so that in case they are required to pay penalties for omissions and errors later, they would be able to meet those penalties through insurance.

As the Y2K conversion represents a major upgrade of existing software, it is natural on the part of the management to appropriately classify the
conversion costs, so that the overall cost to the company due to this crisis is minimum. It should be noted, however, that only the cost of new systems and costs of system replacement can be capitalized. Y2K conversion is only an effort to fix a bug, and hence does not come under system replacement nor new system development. Therefore, the cost of a Y2K conversion project is to be shown only under expenses incurred.

IV. Y2K and Human Resources

As remarked earlier, there is an acute shortage of professionals available for Y2K work all over the world. Organizations in the developed countries subcontract this work to Indian software houses mainly to overcome the problem of personnel shortage. Even for the Y2K solution providers situated in countries like India, the manpower problem is no less severe. India boasts of possessing a large pool of trained manpower, but in reality, the fraction of this pool that can be actually employed for the work under consideration is not large enough. The problem is further worsened with the liberalization of immigration quotas in many of the developed countries for software professionals, causing the number of young professionals migrating to the developed world to increase day by day. Because the conversion work is done mostly by fresh graduates and because the turnover rate among such professionals is very high, the Y2K vendors do not have any option but to assign the supervisory and project management work to professionals too young to bear such a burden. Thus, the major hassle faced by Y2K conversion vendors is to recruit, retain, and retrain ('3 Rs') software professionals. Many companies adopt the policy of recruiting young inexperienced graduates independent of whether they possess any computer background, provided they are trainable in computer work. The standard features of pay and compensation packets for these professionals are: high compensation, performance bonus, retention bonus, stock options, and special working conditions (Delancy). Most young graduates shy away from accepting Y2K jobs on the ground that Y2K conversion is unfashionable and unchallenging. The recruiters should dispel such fears by emphasizing the role played by automatic tools in Y2K conversion and by highlighting the similarities between Y2K conversion and other maintenance work.

V. After Y2K What?

Y2K solution providers, especially in India, are wary of undertaking Y2K work on a large scale for fear that once the year 2000 comes, the need for this work will be over and hence the expertise gained during the course of Y2K projects will go waste. The hundreds of software professionals devoted to this task would also find themselves unemployed. Because of such apprehensions, a few large vendors are not prepared to accept all the Y2K work that comes their way and are in fact imposing on themselves a ceiling on the amount of this work that they would undertake. We consider such apprehensions as misplaced in light of the following:

- Accepting a Y2K conversion contract from a major multinational is of strategic importance for a Y2K vendor, because, if the conversion is successful, further software maintenance of the organization is likely to be awarded to the same vendor.
- It is not necessarily true that with the turning of the new century, the need for Y2K work would abruptly stop. In fact, as observed before, several organizations have not yet realized the need for Y2K conversion, and, sooner or later, such organizations also would begin the conversion.
- The (Organizations which have already undertaken conversion would face secondary bugs and further work on fixing these bugs would start immediately after the beginning of the new century. Thus, for a few more years after beginning of the next century, the demand for Y2K conversion would continue.
- The expertise gained during the Y2K conversion can be effectively used in maintaining systems with problems similar to the Y2K problem. Thus, if Y2K is viewed as a special case of a general class of problems, most apprehensions related to working on it on a large scale seem unjustified. We outline below a few similar problems afflicting the software systems in many parts of the world.

- **The GPS Date Problem**: The global positioning satellites, meant mostly to aid marine and aerial navigation, relay information on position, velocity and time to suitably equipped users on or near the surface of the earth. For this purpose, they measure time with great accuracy through astronomical means. Their data, particularly the clock, is used not only in navigation, but also by many ground stations in operating software for applications such as funds transfer, which require to know time to a high degree of accuracy. The problem with the clock driven by these satellites

is that the maximum week number that can be stored in the clock is 1023. This means that the clock is reset to 0 once in every 1024 weeks, or roughly once in every 20 years. The next such rollover is due at midnight on August 21, 1999. Therefore, at that time, the associated programs are likely to misbehave. Worse, apart from their own clock problem, these systems are affected by the Y2K problem.

- **The Euro Problem:** The European Union has decided that all its member countries should switch over to a unified currency, called the Euro, between 1999 and 2002. The timing of the Euro conversion is not chosen well, because, in order for the switchover to take place, several data processing systems have to undergo conversion. This task seems impossible to achieve in time, because, during the scheduled switchover period, most of the data processing community of the world will be busy with Y2K conversion. Further, several systems besides those related to stock markets, banking, and financial institutions will have to be modified because of the switchover to Euro currency. For example, many on-line analytical processing, data mining, and data warehousing systems which are engaged in analysis of long-term trends will be affected, because they may have to show all their projections entirely in the new currency or entirely in the old currency or in both. This conversion requires enormous resources. An additional complicating factor is that different commodities and different regions will be assigned different dates for switching over to the new currency. Worse, most computer systems do not include the Euro currency symbol in their character sets.

- **Social Security Number Problem:** Currently, the US government uses nine digits for representing a social security number. The social security number used for an individual would never be repeated for another. By about the year 2050, it is estimated that the nine digit numbers would get used up and the number of digits in a social security number has to be increased.

- **Telephone Number Problem:** A telephone number in the US currently possesses three digits for area code and nine digits for the telephone number. Unless the area code is increased to five digits and the telephone number to nine digits, the existing length will not be sufficient beyond the year 2025. This expansion necessitates upgrading of corresponding software, and the software upgrades, in turn, are estimated to cost billions of dollars.

**Concluding Remarks**

From the above discussion, it is clear that most companies are using makeshift methods of conversion to achieve Y2K compliance. The reason for this is that by the time an organization decides to act on this problem, the time available for replacing a system with a completely new one is too short. While doing the conversion mentioned above, an organization should simultaneously plan for replacement of the systems. Otherwise, the organization will be in the same position as it is now when the converted systems become invalid again in a few years.

It should not be forgotten that myopia in computer system design was one of the factors that caused the present crisis. With the experience of the Y2K crisis, it is advisable for all the software designers to assume that some of the systems currently under design will live longer than expected. Otherwise, even if we successfully weather the present year 2000 crisis, another date crisis will emerge in the year 10000 and the next one in the year 100000, and so on. It is a good idea in reviewing software designs to choose a date far beyond the time horizon for which the design was originally intended to work, and examine which parts of the system would fail at the chosen date.

The current practice in the design of data processing systems is to mostly use fixed length codes for a given data field. It is unusual to come across a system which uses several different types of code for the same data field in different parts of the same system. In light of the Y2K crisis, it is good to encourage development of new methods that facilitate flexible coding systems, so that the format of a data field can be changed as and when needed without changing the old data or programs. This would be possible, if along with an instance of a code, information related to its interpretation can be made available. Object oriented methods would be of great use in developing such new schemes.

Several new legal disputes arising out of the Y2K crisis will turn out to be so challenging for the legal experts that the disputes are likely to trigger new legislation in this field.

Software maintenance with sketchy and improper knowledge of the full functional specifications should become a subject of research and new methodologies for such maintenance should emerge to ensure better quality maintenance.
A few professionals tend to argue that Y2K crisis has done more good than harm by becoming a mainstay for software export houses, tool vendors, and other consultants, and by generating a whole lot of new employment. One might as well decry programs to eradicate epidemics lest the medical community and pharmacy sector should face a decline in demand. Let us not forget that the Y2K crisis has tremendously slowed down new software development in many organizations. Indeed, it will be a pity if we learn from this crisis merely how to cope with another crisis should it ever arise, and not how to prevent another from occurring. As has often been said, people who ignore history will find themselves reliving it soon.

References


