1
part

CEO/CIO: The Challenge

Y2K is a worldwide problem, as the Top Ten list by Swiss Bank's Doc Farmer of London points out in a humorous manner. While amusing, all but numbers 5 and 10 still make sense to some in IS or their bosses. Farmer also includes a brief list of non-IS Y2K consequences, potentially resulting in loss of lives.

Whatever the excuses for procrastination, anyone with corporate responsibility should realize that the risk is now personal. Delay any more, writes Dick Lefkon, and your only hope is that Y2K comes as a complete surprise to your competition and the general population.

The Y2K body of knowledge is old and stable—enough so for NYU to award CEU's in it. This book is the first reference to provide everything under one roof; but learning here what you need to fulfill your Y2K commitment doesn't require a Ph. D.

Ernst & Young warn about acquisitions and due diligence—especially SEC Regulation S-K item 303 and Comptroller of the Currency's advisory letter 96-4 demanding Y2K bank compliance. [Ed: For a full legal discussion, see Chapter 7, Y2K Contracts.]

Milt Habeck's six-page winning submission to our paper solicitation gives the whole exposure story in a nutshell, plus a three-page chart listing the Y2K blowup dates for 32 families of software applications!

The dozen "Actionable Caveats" by Ted Fisher and Chris Casey is one of three excerpts from "The Year 2000: Turning Oh-Oh into OK!" The Information Management Forum deserves special thanks for permitting AITP's SIG-Mainframe to show you nearly the entire text of this definitive enterprise-level management approach. See also Parts 3 and 6.

Dick Lefkon's "Seven Methods" tells the CEO / CIO concisely about date expansion, century windowing, intelligent digit, time shifting, and the rest. For an expanded look at each of these, managers and experts are referred to Part 7.

Ascent Logic's brief Compliance piece, along with the 8-digit-date FIPS and ANSI standards, clarifies what form Y2K certification should take; and Lefkon cautions about previous attempts at it.

NIST's CSL Bulletin relates a broad action plan, and Carruthers reminds the CEO of a dozen major exposures the CIO has never considered. Although not explicitly application software-related, some of these exposures can endanger an enterprise just as effectively as those newly 99-year past due accounts.

Lefkon returns to urge your establishment of a Project Management Office (PMO), described further in Part 12.

Finally, Gerhard Adam summarizes the software situation and offers some useful caveats. "First," he says, "review all products for Year 2000 support, especially those which provide automated functions based on data and time (i.e., scheduling systems, storage management, etc.)."

1 Top Ten List of Excuses Not to Address Year 2000 Issues

Dale F. (Doc) Farmer
SBC Warburg Corporate Audit

Y2K is a worldwide problem, as shown in the following. While amusing, all but numbers 5 and 10 still make sense to some in IS or their bosses. Also included is a brief list of non-IS Y2K consequences, potentially resulting in loss of lives.

Top Ten List of Excuses Not to Address Year 2000 Issues

1. You've got lots of time, it's only 1998!
2. You bought a magic bullet from a software salesperson.
3. You can afford to be without Accounts Receivables for a year or two.
4. When the time comes, you'll pay someone else to solve it for you.
5. You're getting into real estate anyway.
6. You like midnight phone calls from irate CEOs.
7. You believe maintenance is for wimps. Real managers create new systems.

Top Ten List of Excuses Not to Address Year 2000 Issues

8. You're not the head of IS, you only work here.
9. You believe that if you ignore the problem it'll go away.
10. You want to surprise your stockholders.

Safety Critical Software Issues

Air traffic control, rail systems, life support control, navigation systems
Will date structure or calculations cause any risk to life & limb?
Could wrong date = wrong location, wrong medication, wrong route?
Could your company be liable if date failure costs lives & property instead of mere cash?

IBM Retention Issues

IBM uses 99/365 to represent permanent retention of files
All permanent files will expire as of that date (31 December 1999)
How this will be addressed by IBM?
What impact to your operating systems and applications?
What impact to your permanent data?

2 *Director’s Guide to the Year 2000*

Dick Lefkon
Year 2000 Committee of AITP SIG-Mainframe

As a Director of a major corporation, you are no doubt familiar with the Year 2000 challenge:

> *Left untreated, IS programs and embedded-chip devices will perform as though the new 00-year is 1900, not 2000.*

Naturally, informed customers and business partners expect your company to use the coming years wisely to prevent

- ➢ time-activated vaults unable to open.
- ➢ customer credit accounts suddenly 100 years overdue.
- ➢ substantial fines due to electronically misplaced securities.

What Regulators Say

In 1996 the U.S. Controller of the Currency told all National Bank CEOs to complete Year 2000 Compliance by December 31, 1998.

In 1997 the U.S. Senate Banking Committee imposed the December, 1998, requirement on the Federal Reserve Bank and subsidiaries. Quarterly and annual corporate reports to the Securities and Exchange Commission must inform shareholders about the Year 2000 exposure, what you’re doing about it, and what it will cost.

What Will Fixing “Y2K” Cost?

The Gartner Group and others have estimated worldwide cost of “Y2K” at approximately two-thirds of a trillion dollars. This figure is about the same as the cost of the U.S. Savings & Loan mishap.

Only a third of that amount will be spent in North America: One and one half 1996 dollars per line of customized code.

- ➢ A small brokerage firm with one half million lines of in-house code will need to spend about $750,000 to ready that code for Year 2000.
- ➢ A medium-sized bank with 10 million custom code lines can probably hold its in-house migration costs to approximately $15,000,000.
- ➢ At least one large Eastern bank and one major securities firm are known to have budgeted more than two hundred million dollars to insulate their information systems and client accounts from Year 2000 vulnerability.

Are Directors and Officers Personally Liable?

Usually not, this time perhaps yes. All Directors are required to exercise *reasonable diligence and care*, making informed decisions.

If “Year 2000” were a surprise to your competitors and the general population, you would not be liable.

Fortunately for our economy, companies like those just cited are well on their way to curing the Year 2000 problem. And this encouraging news is disseminated daily via newspapers and television.

How Can We Stay Within D&O Insurance Coverage?

D&O insurance protects Directors and Officers in instances of normal human fallibility.

Make sure your paper trail shows you took reasonable preventative measures:

- ➢ Establish Y2K project leadership and adequate funding.
- ➢ Inventory your source code and hardware/software packages.
- ➢ Perform an analysis and write a plan.
- ➢ Use automated packages and outside help appropriately.
- ➢ Track your progress and report it honestly to employees, customers, and regulators.

Your costs per code line will not vary widely from others in your industry. Try to avoid writing budgets with patently unrealistic cost figures.

If you lack the necessary incremental funds, consider following the example of the U.S. Army:

- ➢ Immediately shut down information systems that are not absolutely mission-critical, and re-deploy their resources to Y2K.
- ➢ Promptly halt all new development on remaining applications (except for production emergencies) until after they are all Y2K-ready. [Ed: See Chapter 20, *You Might Receive a “C” Grade*.]

Why December 31, 1998?

Ask your IS Director about “Y2K” and you will hear these facts among others:

- ➢ Every in-house program must be inspected and potential date impacts modified.
- ➢ The necessary code and/or file changes are fairly simple; but they’re pervasive.
- ➢ Using industry benchmarks, every million lines of code you change will introduce 100 to 1000 undetected errors.
- ➢ Because of interrelationships, every single program you own *must be re-tested even if it has not been changed*.

December of 1998 is your last chance to “shake down” your modified end-of-year automated processes in a production setting—except for Year 2000 itself!

3 The Millennium Rollover Is Not a Surprise

Dick Lefkon
New York University

The Y2K body of knowledge is old and stable—enough so for NYU to award CEUs in it.

- ➢ The Millennium Rollover is not a surprise.
- ➢ The necessary body of knowledge is well defined.

New York University
School of Continuing Education
Fall Semester, 1996
X53.9807 (SEC 1): Year 2000 Computing Best Practices
1.0 CEU
Instructor: Dick Lefkon

Description

The course surveys four main components of Year 2000 methods: Strategic issues, needs, solutions, and legal aspects; Technical view of 3GLs, COTS, and custom software; Large scale project management techniques for hardware, software, and testing; Product classes: code converters, version control with capture / replay, test scripters, and clock simulation.

Text

Lefkon (ed.), *Year 2000: Best Practices for Y2K Millennium Computing:* Panic in Year 2000, AITP-SIG Mainframe, 1996

Lessons

1. Needs and Resources: New FIPS and ANSI Standards; project costing; Congressional Final Report on resources; per-line versus per-point cost. Identifying exposures in software and automated processes.
2. Legal and Accounting Issues: FASB and GASB on expensing, Contract wordings from viewpoints of vendor and owner. When one can modify, when demand modification.
3. Awareness and Needs Analysis: Involving the enterprise; needs surveys; HW / SW checklists; formulating a template based on others; success.
4. Five Technical Approaches: Sliding century window, date expansion versus compaction; date code regression beyond the IDD; bridging; interfaces.

5. Four Technical Challenges: Embedded dates; extended-interval aging; isolating production from Millennium work; lost source code.
6. Project Management: The usual seven-step description; models for managing testing; techniques for project coordination / acceleration.
7. Staffing Management: Non-monetary compensation; on-going training for replenishment; staff allocation; determining and managing outsourcing.
8. Present State of Leading, Most-Used Products on PC, Mainframe, UNIX.
9. More on Products: Quiz.

4 *I'm Not Worried…*

Ernst & Young, LLP

I'm Not Worried...

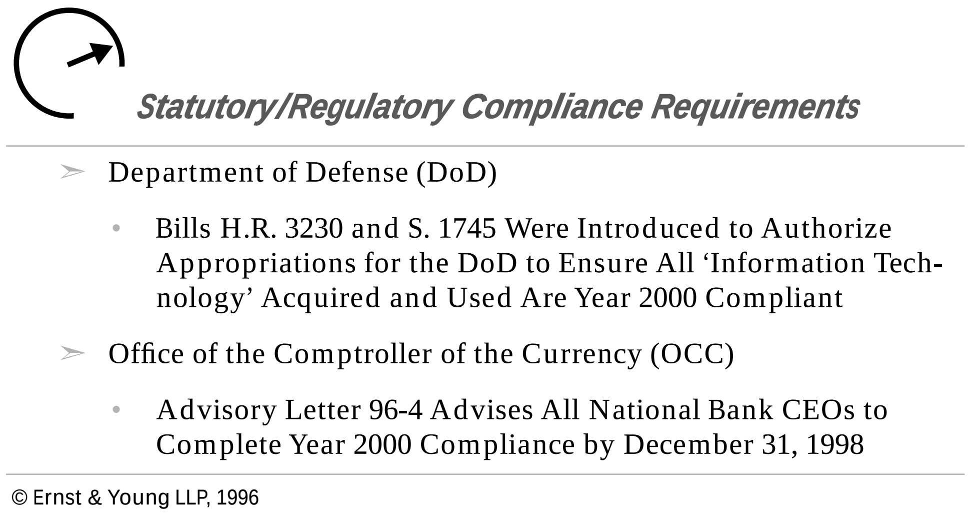
- Vendor / Outsourced Software
  - Inventory and Review All Formal Agreements
  - You May Have to Pay for the Year 2000 Fix if It Is Not Part of Your Maintenance / Outsourcing Agreement
  - When Will the Vendor / Outsourcer Be Year 2000 Ready?
- It's 1997, Do You Know Where Your Systems and Data Are?
  - System and Data Integration Over the Past Decade May Hide Date Problems
  - Imbedded Date Fields in Nomenclature May Need to Be Addressed
  - Third Party Data Transfers May Contaminate Your 'Fixed' Applications and Databases

I’m Still Not Worried…

- ➢ Software License Problems
  - Can You Load Your Package Software on Another CPU Without Incurring Additional Costs?
  - Will It Even Load to Another CPU?
- ➢ Due Diligence
  - Be Careful When You Acquire or Merge
  - Reserve the Right to Terminate Any Agreements Due to Year 2000 Reasons

Now I’m Worried…

- ➢ U.S. Securities and Exchange Commission
  - Reg. S-K, Item 303, “Management’s Discussion and Analysis of Financial Condition and Results of Operations” Must Contain
    - Matters That Would Have Impact on Future Operations and Have Not Had an Impact in the Past
    - Matters That Have Had an Impact on Reported Operations and Are Not Expected to Have Impact on Future Operations
- ➢ Shareholder Litigation
  - Likely to Occur If Year 2000 Causes Business Disruption
  - Civil and Criminal Actions May Take Place



Statutory/Regulatory Compliance Requirements

- ➢ Department of Defense (DoD)
  - Bills H.R. 3230 and S. 1745 Were Introduced to Authorize Appropriations for the DoD to Ensure All ‘Information Technology’ Acquired and Used Are Year 2000 Compliant
- ➢ Office of the Comptroller of the Currency (OCC)
  - Advisory Letter 96-4 Advises All National Bank CEOs to Complete Year 2000 Compliance by December 31, 1998

5 The Century Date Problem: How Bad Can It Be?

Milt Habeck
Unbeaten Path International

The short answer to the question posed by this headline to the executive management of today’s modern business enterprises is “worse than you can ever imagine.” In fact, it is virtually certain that the viability of your business will be severely threatened if your company does not make Year-2000 date compliance of its software systems the top priority of information processing staff right now.

But prudent managers make decisions based on facts, not emotions or fad-driven hysteria. Therefore, as an information processing consultant whose firm has examined first hand the effect the century-date change will have on the major systems applications which drive today’s modern businesses, I would like to present a synopsis of our findings.

The Root of the Problem

Most application software operating today’s businesses, whether purchased from an outside vendor or created in-house, was designed to accommodate dates only up through calendar year 1999. Though this seems foolish today, the systems in use have evolved over the last 10 to 25 years, from times when the Year 2000 seemed too far away to matter.

Because the software developed in past years stores only the last two digits of the year, if today's system attempts to add one more year to 99 years, there won't be enough room to store the three-digit result and the computer will do what the odometer on your car would do at 99,999 miles. It will turn to zero.

Problems will begin long before the 1999 Christmas season. Many planning lead times based on calendars in various system modules extend for months and sometimes years. As soon as one of those planning horizons or forward-postings reaches January '00, the system will become confused and begin to cause unpredictable problems.

When your company built or acquired its complete, comprehensive, fully integrated software application, the strength of the technology was its ability to use centralized information in many functional ways. Unfortunately, that strength greatly exaggerates the century-date problem. With complex interrelationships, date sequence errors introduced into centralized data, the problem spreads like a deadly plague to many other parts of the system, and erroneous data ends up in places where reliable data is required for daily decisions. At that point, the whole system becomes functionally useless.

In the following pages, we will review how major systems use dates in the processing of vital information and then relate the practical effects on business for a company that fails to make those systems century-date compliant.

Table 5.1 Typical Date Proliferations is a report based on an actual analysis run against the integrated manufacturing system of a major worldwide processed-products manufacturing company. We wanted to know, for a product scheduled for manufacture in January, 2000, what would be the earliest possible occurrence of a date requirement for each module of the integrated system.

Though many modules will be unaffected prior to 2000, the Quality Module, which dictates the ingredients to be used in the bill-of-materials, will be affected starting in October, 1996. The next most serious problem will occur in the Manufacturing Requirements Planning module in October, 1997.

Dates on Screens and Reports

In most large, integrated systems, dates are stored in a standard format which is then translated into whatever format is required by the users of that data. For example, some nations express December 25, 1999 as 12/25/99, and others write 25/12/99. Both of these numbers are derived from the stored format 991225.

Table 5.1 Century Date Problem for Typical ERP System (Page 1 of 3)
Resuscitator 2000 Binoculars™ Report

Appl. ID	*Application Description*	*First Date Problems*	*Practical Illustration of How Century Change Will Create Issues*
AAA	Accounts Payable		System will not allow your company to earn cash discounts if discount period spans 1/2000
BBB	Accounts Receivable	Nov-99	Payments not recognized for cash discount; faulty aging report calculations
CCC	Process Manufacturing	Sep-99	Planned release date for processing orders would be after order completion dates
DDD	Billing	Dec-99	Potential post-shipment invoicing problem if activity spans century; order history sorts inaccurate
EEE	Bill of Materials	Jun-98	Engineering change management functionality diminished by out-of-sequence effectivity dates
FFF	Capacity Planning	Dec-97	Backward scheduling calculations could yield negative values/ unpredictable errors
GGG	Data Collection	Jan-00	Transaction records would only have six-digit date information
HHH	Cash Management	Jan-00	Report selections which span the century would be disallowed
III	Cost Accounting	Dec-98	Limited problems except for rolling up costs with out-of-sequence BOM effective dates
JJJ	Foreign Exchange Translation	Jan-00	Exchange rate maintenance will cause calculation faults at change of century
KKK	Help Text	Jan-00	Very little effect...just report heading faults
LLL	Distribution Resources Planning	Jun-99	Resupply orders will have due dates pre-dating order release dates, unpredictable problems

Table 5.1 Century Date Problem for Typical ERP System (Page 2 of 3)
Resuscitator 2000 Binoculars™ Report

Appl. ID	*Application Description*	*First Date Problems*	*Practical Illustration of How Century Change Will Create Issues*
MMM	Forecasting	Dec-97	Problems occur when year values are truncated to send data to Master Production Scheduling
NNN	General Ledger	Oct-98	Faulty accounting period definitions; accounting calendar won't allow "00" close to "99" start
OOO	Inventory Management	Jan-00	Inventory transaction history would be mis-sorted; reports spanning century disallowed
PPP	Repetitive Manufacturing	Oct-99	Planned orders from MPS/MRP won't be accurately converted to scheduled releases
QQQ	Quality Scheduling	Sep-99	Quality staff scheduling disrupted when MRP & CRP encounter date discontinuity
RRR	Multiple Factory Management	Jan-00	Should be ok except for selection parameters that include date ranges
SSS	Foreign Exchange Management	Jan-99	Exchange gain/(loss) calculations faulty if recognition date and settlement date span century
TTT	Material Requirements Planning	Dec-97	Horizon date logic disabled across century change; order release date calculation faults
UUU	Customer Order Processing	Nov-99	Orders shipped Dec '99 will immediately look overdue, causing credit hold conditions
VVV	Performance Management	Jan-00	Inaccurate standard readings for date-related measurements spanning century date
WWW	Marketing Planning	Jul-99	Discontinuity in promotion calendar will disrupt definition of start, end dates for deals

Table 5.1 Century Date Problem for Typical ERP System (Page 3 of 3)
Resuscitator 2000 Binoculars™ Report

Appl. ID	*Application Description*	*First Date Problems*	*Practical Illustration of How Century Change Will Create Issues*
XXX	Purchasing	Jun-98	PO inquiry spanning century disallowed; faults in measurement of vendor due date performance
YYY	Quality Management	Oct-96	Retest dates for production samples will be sorted to past-due position
ZZZ	Sales Analysis	Jan-00	Sales analysis reports that span century cannot be initiated
ABC	Shop Floor Control	Sep-99	Confusion will result when job completion dates pre-date job release dates
DEF	Menus	Jan-00	Just the date on the menu display would be affected
GHI	System Controls and Parameters		Listing routines, date retrieval, retrieval/validation routines, and job submission routines affected
JKL	Pop-up Information	Jan-00	Any pop-up window that uses dates affected; for example, additional order information
MNO	Reference Routines	Jan-00	Report headings affected, but nothing critical
PQR	Standard Source Library		Some date routines in standard source library would not handle century logic
	Totals RPG & CLP Programs: 2,500 Programs with Dates: 58% Number of Dates: 39,000		

If century information is added to the internal storage number of December 25, 1999, that number would become eight digits long, or 19991225. But if the end-use numbers are not adjusted, they, being six digits long, would be at a loss for how to handle eight digits of information.

The net result would be that the reports and displays used to run your business would not work at all or they would be unintelligible.

Dates as Information Drivers

Your software stores and processes millions of transactions for your company, including the payment of invoices, receiving a shipment, booking a collection, scheduling an order, etc.

In all applications, transaction data must be displayed and processed in sequence, using either an ascending or descending order. Examples of this function include order inquiry, inventory transaction history, sales data, manufacturing planning and pegging, capacity planning, general ledger transaction detail, accounts receivable invoices, accounts payable invoices, purchase order inquiry, shop order inquiry and inventory lot allocations, to name a few.

Since dates are stored internally in YYMMDD (YearYear, MonthMonth, DayDay) format, the numerical value of the date field determines the sequence. For example, a transaction with a date of 960925 (September 25, 1996) will be processed before a transaction dated 970901 (September 1, 1997) because the former is a lower number than the latter. As you can see, dates in the next century will not be correctly processed relative to dates in the current century because the first two digits of the date field will be zeros. Therefore, even though January 10, 2000 is later on the calendar than December 25, 1999, the numerical processing will put Christmas before New Year's because 991225 will be processed long after 000110. Thus, for all those transaction-based functions listed above, the workload will be turned upside down, resulting in unusable output.

Sorting by dates will be equally futile. Imagine this scenario:

> *It is October '99. Your supply of boric acid is getting low, and your software application calculates that an order should be placed with the supplier in three months, January '00. The computer dutifully stores that information in numerical date order back before World War I.*
>
> *The system then notices that no boric acid is scheduled for future delivery, so it schedules another purchase order for January '00 and proceeds to store that one before World War I as well. Then it does it again, and again, until someone pulls the plug before the system fills the computer with automatically generated boric acid purchase orders that are 99 years overdue.*

Dates in Calculations

In all application areas, dates are used in calculation routines in order to determine other related dates. Here are a few examples:

- If the century date is not fixed, calculations that cross the century mark will require customer payments 99 years before shipment to qualify for a cash discount because in billing and accounts receivable, the due date and discount date are calculated by adding the “customer terms days” to the “invoice date.”
- Similarly, orders shipped mid-December on 30-day terms will immediately show up as being overdue by 99 years, which will cause future orders from that customer to be put on credit hold and overdue carrying charges to be vast. This will cause interesting customer relations. One more glitch: the program that prints aging reports won’t work.
- If century dates are not fixed in the accounts payable software, calculations that cross the century mark will require your company to make payment 99 years ago to earn a cash discount.
- In manufacturing requirements planning, the century date problem may calculate negative release dates, meaning orders will be released before Joseph and Mary took off for Bethlehem to celebrate the first Christmas.
- In shop floor control, the system will generate complicated circular logic because some jobs would be scheduled for completion before World War I, decades before those same jobs were scheduled to start. That’s because “operation start dates” and “end dates” are determined by either backward or forward scheduling.
- In inventory control, many lot dates are calculated by the system based on the transaction date and the “lot expiration days” combined with the “lot reset dates.” This means the system will automatically change your inventory policy from FIFO (first in, first out) to OINO (once in, never out).
- In manufacturing planning, where ingredients are changed by “effective dates,” the integrity of finished product could be seriously affected by ingredients not being substituted at the specified time or simply not being available at the specified date.
- Sales forecasting dates will cause turmoil when passed to master production scheduling, and forecasted demand for 99 years ago will not be acted on by the system.

- Engineering change management will become challenging because effectivity dates in the new century will be nonsense numbers.
- Sites using multiple currency functionality will learn that applications will be confused as they try to figure out foreign exchange gains and losses for forward currency obligations that run across the century mark. Multi-national profitability will be reduced to guesswork.
- Integrated promotion / order processing / billing systems will become very confused. Companies that have not become century date compliant will probably need to resort to manual customer service systems by late 1999.
- Comparing Dates for Information Retrieval
- Many applications maintain interrelated calendars for cut-off dates or trigger-dates with associated information. Based on a transaction date, a lookup is performed and corresponding data is retrieved for further processing. Therefore, it is critical for the system to properly determine date sequences. If they do not:
- in the General ledger, transactions will end up posted in the wrong accounting periods.
- master production scheduling data and material requirements data will be reported in the wrong time periods, completely confusing material planners.
- capacity plans relying on the shop calendar to determine work days and holidays as well as the number of hours available for scheduling will plan to have the world end after Christmas 1999, as there will be zero work days available.
- Inquiries, reports and transaction processing programs which require “date range” selections as “from” and “to” ranges depend on the ability of the system to determine the relative sequence of the transaction dates. Thus, the system must be able to correctly process transactions selected from 12 / 07 / 99 through 010 / 31 / 00. If they cannot, general ledger inquiries won’t work, sales analyses will generate nonsense data and user-generated inquiries will produce useless reports.

Solving the Problem…The Urgency, the Costs and the Process

I hope, by sharing these findings, that we leave the impression that the century-date conversion challenge is not a simple maintenance task. I also hope we have enlightened various business managers that this is not just a systems problem, it is a business problem, for not only must each individual enterprise prepare itself for the Year 2000; it must ensure its suppliers and customers prepared as well. In case after case, we have found that people underestimate the date conversion task by a factor of five to ten. . . eight man-months typically turn into eighty.

Fixing the problem will cost a considerable amount of money. To some, it will seem like a painful, unnecessary expenditure. Some will consider it "business survival insurance." Others will take funds from the marketing budget, assuming that the competition will not take care of his own problem in time and this business, being well prepared, will gain a significant competitive edge. Whatever the cost today, it will double every six months until January 1999, at which time it will be virtually impossible to perform an adequate century-date conversion project.

In June of 1996, Gartner Group reported that 75 percent of U.K. businesses and 90 percent of U.S. businesses were in a state of total denial on the century-date problem. By the end of 1996, those numbers had shifted in some industries. After witnessing the depth of the problem, I cannot understand why businesses are not actively working to solve it today.

We have summarized the effect of non-century date compliance for each of the modules studied.

Editor's Note: The manufacturing sample showing a dozen dates per program is much too low for the financial industries. The "sub-assemblies" of a typical bond can consist of anywhere from several dates to literally hundreds of dates.

Four Steps to Success

Taking action is a simple four-step process.

1. Make the Year-2000 issue a top priority for your I/S staff.
2. Tell them you want to know the real size of the job at hand.
3. Authorize them to buy whatever Year-2000-specific tools they need to use the computer's own processing power to analyze date fields and automatically rebuild them into century-compliant formats.
4. Ensure that the conversion plan leaves at least a year for testing of converted systems.

I submit that your company's life is at stake; your systems professionals are eager to get this project behind them. The deadline will not move. The detonator is ticking. It's time to take action now.

6 Actionable Caveats

Chris Casey
Bytewise Consulting, Inc.

Ted Fisher
Sperduto & Associates, Inc.

This chapter summarizes the *Actionable Caveats*, or important conclusions, which we feel are critical to the success of any compliance project. These statements are at times warnings to help managers be aware of certain pitfalls which could derail or stall their process. Other thoughts are preventive measures which can be implemented to allow the organization to handle the process in a pro-active fashion in order to capture some of the positive opportunities presented by the problem.

1. *The Year 2000 compliance effort is a survival issue.*
   For those who are prone to argue about cost, be prepared to prove that survival is the issue and ask what survival as a firm is worth.
2. *The compliance of others' systems is also a survival issue.*
   The compliance of customers, key suppliers, city, county, state and federal governments, utilities, banks, etc., is also a survival issue for us all. If those with whom you do business fail as a result of their non-compliance, your efforts toward compliance will have been in vain. Enterprise management must realize that they have a real and critical stake in the compliance of their commercial partners.
3. *In most cases, a significant amount of time must pass from the moment when the CEO is informed of the Year 2000 problem to the time when the CEO makes the necessary commitment to address the problem.*
   The amount of time seems to depend upon the CEO's temperament, the general state of the firm and the economic outlook. The range of time required is about three to nine months. With so little time left before January 1, 2000, it is obviously important to complete the awareness-to-commitment process as quickly as possible.
4. *In order to gain the commitment of enterprise management, business and technical managers should assemble a fact-based case for presentation to enterprise management.*
   The first step toward compliance is to convince the CEO and other enterprise management of the scope and potential impact of the Year 2000 problem. An objective case must be prepared and presented to the CEO. This case must convey an appropriate sense of urgency to convince the CEO in a reasonable period of time to make the commitments

necessary to achieve compliance (including testing) within the time available. The presentation must be persuasive and objective, and to the extent possible, should incorporate real data which can be obtained using third-party analysis tools and methods.

5. *Current resources are nowhere near sufficient to deal with the world-wide need.*
   Those organizations which begin the compliance efforts early will have the benefit of best-of-breed technicians. The cream of the crop will be fully committed in a short time. Companies which wait to start their compliance process will face a market of scarce resources.
6. *Competing priorities can derail the compliance process by diverting resources or distracting enterprise management.*
   Competing survival issues can arise (a fire, serious accident, merger. etc.), and attention and commitment to the Year 2000 compliance process can diminish or, at least temporarily, cease. Some form of contingency planning is necessary. So is a means to maintain the involvement and commitment of enterprise management.
7. *There are positive outcomes to compliance that need to be studied for purposes of exploitation.*
   Enhanced customer relations, new customers and prospects, and opportunities for favorable publicity are just a few of the achievable positive outcomes of the Year 2000 compliance process. Cooperative activities with customers, prospects and key suppliers will draw you into closer relationships, will strengthen the trust of the parties involved, and will result in added business and profits.
8. *Exposure to the possibility of lawsuits abounds.*
   Firms must pro-actively manage their legal exposures before, during and after Year 2000 compliance. Intelligent strategies must be developed to minimize such exposures. Some of the exposures include claims against software vendors for failing systems, shareholder actions against board members of non-compliant firms, and customer actions against service organizations, such as banks, brokerage houses and insurance companies.
9. *All stakeholders in the enterprise need to be involved in the compliance process.*
   All parties, including partners, customers, suppliers, media, governmental regulators and shareholders, need to be consulted, kept informed and, to the appropriate extent, involved in the compliance process. Stakeholder involvement is one of the most effective ways of avoiding the erosion of enterprise management’s commitment.

10. *A significant percentage of medium and large sized businesses will not meet the compliance deadline ,and the likelihood of their failure is great.* If it appears that a particular business is not going to be able to achieve compliance, it may be advisable to pare down the business being done with that enterprise. Eventually, it may be necessary to cease electronic contact with them to avoid the risk of contaminating the data in your system. If their contribution to your business is unique and necessary, identify such businesses early and proactively assist them in their compliance efforts through education, technology transfer or direct involvement.
11. *Compliance-related activities will continue well beyond January 1, 2000.* Many companies will not succeed in achieving full compliance before January 1, 2000, for a variety of reasons. Faced with Year 2000-related system failures, these companies will be forced to revert to manual methods to replace their failed computer information systems. Some companies will successfully tread water with manual systems, while others will drown, discovering too late their dependence on date-sensitive automation. Those that do survive will have to bring their information systems into compliance at some point. As such, January 1, 2000, does not mean an end to compliance efforts. Not all systems and applications will be run on January 1, 2000, or for that matter, in January at all. Special quarterly, semi-quarterly and annual processing may reveal isolated pockets of non-compliance.
12. *Don't permit your enterprise to become over-reliant on vendors and solution providers during the compliance process.*
    Vendors and solution providers have their limitations, just as you have yours. While you may be able to outsource a significant share of the work, you cannot outsource the responsibility and management of the compliance effort.

7 Have You Selected These Seven Methods?

Dick Lefkon
Year 2000 Committee of AITP SIG-Mainframe

There are seven main approaches to the Year 2000 software upgrade task. "Outsource" is not one of them, and if you outsource you'll still need to choose the approach. If you can only list two, you'll have made real decisions about the other five. Deciding not to decide is a decision.

1. **Prune the business.**
2. **Wait.**
3. **Replace the application by a purchase or new build.**
4. **Expand YY year fields to YYYY.**
5. **“Intelligent” digits, other encoding.**
6. **Date “Window(s).”**
7. **Date-shift (“encapsulate”) code or data.**

Once your orange systems are made Y2K-compliant by one of these methods, if your green ones aren’t yet ok, you’ll need to “bridge” from one application to another by using intermediate files or other techniques. When you near completion, bridges between conformant applications are removed. Keep them available, though, for times you want to access your archives.

All seven main approaches are easy to understand, especially number 2. You barely have time to convert 40 percent to 60 percent of your programs fully. Part of your triage approach should be to determine which functions or programs can safely be ignored until next decade because their shortcomings will be cosmetic at worst. For instance, your salespeople won’t lynch you for having to scroll past the 12/31 and earlier sales to the 00/04 ones on a “show recent sales first” CICS screen, since this ugliness will only last for days and they’ll know their competitors are probably suffering the same inconvenience. Save some cosmetic changes for when there’s time.

At the opposite end of “do nothing” are systems which are absolutely critical to the business but have a truly prohibitive Y2K compliance cost. If the company (or line) will become unprofitable due to that outlay, consider shutting it down or selling it. Fast food chains do this frequently with unprofitable stores, and IBM itself has thus far avoided announcing Y2K conformance for the 43xx and 30x1 lines of installed mainframes.

Another “pruning” alternative is to sell that business part to a Y2K-ok competitor—but the SEC won’t let you foist it on an unsuspecting party.

Third on the list is a full software replacement. This could modernize your business, but you won’t be able to regression-test incrementally. A new build is both costly and deadline-risky, and most business-specific software replacements can’t help requiring massive customization efforts.

One smart triage using number 3 might be to toss your low-data store Executive Info system, installing a conformant one to be fed by your revamped apps.

Number 4, full YYYY field expansion, is the most appealing and elegant Y2K date fix approach. You regularly make similar changes to keep up with industry-wide formats. But there aren't time and resources to expand everything to YYYY.

Number 5, "Intelligent digit," is one of three unusual methods. It keeps dates the same length by using the leftmost nibble of the leftmost year byte to hold a century-millennium flag. To use this approach, you have to add various date-encoding and -decoding lines that are anything but obvious. "Seven digit date" is a cross between this method and the one preceding.

```
WWWWWWWWWWWWWWWWWWWW
1                  1
9                  9
1                  1
9                  9
1                  1
9                  9
WWWWWWW 1930 WWWWWWW
2                  2
0                  0
2                  2
0                  0
WWWWWWWWWWWWWWWWWWWW
```

Sixth of the seven methods is the so-called century window. Pictorially, this looks just like the kitchen window on the TV series *Honeymooners*. The top glass pane represents part of a century of which the YY numbers are above a century "pivot" point. In the diagram, the middle bar of the window is labelled "1930." All years ending with numbers 30 or above are to be interpreted as "19" plus YY. "98" is interpreted as "1998," etc.: IF YY > 30, CC = 19 ELSE CC = 20. Numbers below the bar are future: "20" plus YY.

Coding for a century window probably looks familiar, since much of your legacy code already contains date logic for left-affixing "19" or "20."

A so-called "sliding century window" defines the pivot year as being a certain number of years into the past, starting at the current year. Thus, your 100-year window keeps on advancing, preventing shrinkage of the number of years in your future horizon: IF YY > (now - 50), CC=19.

Last on the list is the quickest—and dirtiest—approach, one which makes no pretense of using the "true" date, as long as the calculations come out right. This date-shift approach downshifts the year by 28. There are seven days to a week and four years to a leap-year cycle; multiplying together and getting 28, you'll find a year shifted this much will have 4th of July and all Sundays in the right place. Taking proper calendar care for new holidays and religious ones, date-related calculations should in theory come out just fine.

Shifting the date requires changing programs (-28 / +28 at the start / exit) or changing the data-store to hold (phony) shifted dates, not true ones. But known-incorrect file data raises a possible need to store duplicate data. And so, a code fix is probably the better means to shift dates.

A single canned program segment can repetitively be used to downshift all relevant dates in a given named copybook. The same holds for upshifts. On the whole, "encapsulation" of programs will probably entail the lowest amount of coding, errors, and testing among the seven approaches.

Nobody wants to use such an artificial approach based on false numbers. However, since it is pretty safe and pretty quick, you might want to put a tab labeled “summer 1998” on this page and read it again when time is short and deadlines are close and “Q&D” is less a dirty word.

(A goofy-sounding but common variation of number 7 is to use the Lilian date, storing untrue dates that are the interval since a certain time in history, not untrue dates measured 28 years backward from the present.)

To summarize: There are seven main approaches to software which isn’t yet ready for the Year 2000. Choosing the best mix for you—even if it’s executed by a mediocre workforce—will make you look better than selecting the wrong approach and implementing it through a truly outstanding, top-quality outsourcing firm or in-house team.

This decision is yours to make and cannot safely be outsourced.

8 Four Primary Compliance Criteria

Yngvar Tronstad, Jan Peterson, Joe Ramirez,
Jack Ashburner, Grant Robinson
Ascent Logic Corporation

Although the four criteria as summarized in Table 8.1 fully define century compliance, the essence of compliance implies a balance between cost and risk based on a business perspective, rather than on a technical yardstick. However, such a balance will vary with each enterprise according to its business needs and its technology base. Thus these generic century-compliance criteria need to be decomposed and refined within the context of a particular business case.

The first compliance criterion, General Integrity, is one of several high-level criteria that should guide the effort of making a business century compliant.

General Integrity Criterion

“Desired operations” will continue uninterrupted regardless of value for current date.

Elaboration: As a system date advances normally on a host processor, each date rollover must not lead either the host process or any software

Table 8.1 The Four Primary Compliance Criteria

Criterion	*Definition*
General Integrity	"Desired operations" will continue uninterrupted regardless of value for current date.
Date Integrity	All manipulation of calendar-related data, such as dates, duration, days of week, etc., will produce desired results for all valid date values within the application domain.
Explicit Century	Date elements in the interfaces and data storage permit specifying the century to eliminate date ambiguity.
Implicit Century	For any date element represented without a century, the correct century is unambiguous for all manipulations involving that element.

executing on the host to erroneous processing. The term "desired operations" is intentionally broad and needs to be interpreted for specific businesses and technologies.

The technical component of the Year 2000 challenge involves the accurate acceptance, creation, manipulation, and output of calendar-related data. Although the largest concern is whether the system can handle the rollover date to the Year 2000, century compliance is much broader and more diverse. This particular rollover event is a high-risk event that needs to be mitigated; there are several other risk areas. These events can occur not only at rollover to the Year 2000 but well before and well after. It is therefore important to identify and categorize different events and their associated event horizons, which pinpoint the probable timing for the occurrence of a Year 2000 failure.

Date Integrity Criterion

All manipulation of calendar-related data, such as dates, duration, days of week, etc., will produce desired results for all valid date values within the application domain. This century compliance criterion covers correctness of manipulations of date data within manipulation categories such as arithmetic, branching, formatting, storage, and extended semantics. Several examples of these issues are given in Table 8.2. These manipulations need to be made reliable over the range of dates that an application is expected to handle. For example, sales order processing many handle dates from five years in the past to one year into the future. In contrast, an employee database may store dates of birth from early in the 20th century to planned retirement dates well into the 21st century.

Table 8.2 Date Integrity Examples

Category	*Examples of manipulation*
Arithmetic	Calculate the duration between two dates
	Calculate date based on starting date and duration
	Calculate day of week, day of year, week within year
Branching	Compare two dates
Data Storage	Storing and retrieving
	Sorting and merging
	Searching
	Indexing on disk file or database table
	Moving data within primary memory
Extended Semantics	“99” as a special value for year
	“99.365” as a special value for Julian date
	“00” as a special value for year

Explicit Century Criterion

Date elements in the interfaces and data storage permit specifying the century to eliminate date ambiguity. This criterion essentially requires the capability to store explicit values for century. For example, third-party products that can use a 4-digit year in all date data elements stored and passed across each interface, including the user interface, would satisfy this criteria. A base and offset representation of dates covering all centuries of interest would also satisfy this criterion. Whether this capability should be used to eliminate century ambiguity is part of the last criterion.

Implicit Century Criterion

For any date element represented without century, the correct century is unambiguous for all manipulations involving that element. This last criterion requires that if the century is not explicitly provided, then its value must be correctly inferred from the value of date provided with 100 percent accuracy. For example, the range of values for an "invoice date" would rarely span more than 10 years. Because the century can always be interpreted correctly for an invoice with a 2-digit year, this date element would satisfy this criterion. This criterion permits cost-risk trade-offs that may minimize changes to existing date formats, given that you evaluate the cases for interpretation within the context of your business functions.

9 FIPS Publication 4–1 Change Notice

National Institute of Standards and Technology

Following is the text of change 1 to FIPS Publication 4–1, FIPS PUB 4–1, Representation for Calendar Date and Ordinal Date for Information Interchange, dated 25 March, 1996.

Specific Change

Page 2: In reference to paragraph 10, Specifications:

> *For purposes of electronic data interchange in any recorded form among U.S. Government agencies, NIST highly recommends that four-digit year elements be used. The year should encompass a two-digit century that precedes, and is contiguous with, a two-digit year-of-century (e.g., 1999, 2000, etc.). In addition, optional two-digit year time elements specified in ANSI X3.30-1985(R1991) should not be used for the purposes of any data interchange among U.S. Government agencies.*

10 Proposed Draft Revised American National Standard X3.30, Representation of Date for Information Interchange

American National Standards Institute

1. Scope, purpose, and application

1.1 The purpose of the American National Standard is to provide a single standard means of representing calendar date for interchange among data systems.

1.2 The scope of this American National Standard is limited to the representation of calendar date for interchange among data systems; it does not describe how the date is determined. This standard was not designed for (nor does it preclude) usage by humans as input to, or output from data systems. This standard does not address how data is converted by data systems to be internally processed and / or stored.

1.3 The representation of calendar date specified in this standard is compatible with other national and international standards. In ISO 8601:1988, the representation specified by this American National Standard is referred to as calendar date-complete representation-basic format.

2. Normative references

The following standard contains provisions that, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below.

ISO 8601:1988, Data elements and interchange of formats—Information interchange—representation of dates and times.

3. Definitions

For this American National Standard, the following definitions apply:

3.1 Calendar date: A particular year, month, and day of the Gregorian Calendar.

3.2 Calendar day: A particular day within a Gregorian Calendar month.

3.3 Calendar month: A particular month within a Gregorian Calendar year.

3.4 Calendar year: A particular year according to the Gregorian Calendar.

3.5 Gregorian Calendar: A calendar introduced in the year 1582 A.D. and now in general use.

4. *Specifications*

4.1 The calendar date shall be represented by eight numeric characters. Calendar date shall be represented in the order of calendar year (YYYY), calendar month (MM), and calendar day (DD). No characters may be added or omitted.

4.2 The allowed values for calendar year are “0001” through “9999”. Calendar year is presumed to be anno Domini (A.D.) unless otherwise specified.

4.3 The allowed values for calendar month are “01” through “12”, with a leading zero where the number representing the month has only one digit. January is represented by the ordinal number “01”, and subsequent months are numbered in ascending sequence from “02” to “12”.

4.4 The allowed values for calendar day are “01” through “31”, depending on the number of days in a month, with a leading zero where the number representing the day has only one digit. The first day of the month is represented by the ordinal number “01”, and subsequent days are numbered in ascending sequence from “02” to the end of the month.

5. *Example*

The fourth day of July in the year 1776 is represented as “17760704.”

[Editor’s Note: The proposed X3.30 revision is related to the combining and revision of time standards X3.43 and X3.51. There will be formal recognition of a leap second. This probably won’t affect your applications, but ask your communications providers about it.]

11 Product Certification and Year 2000 Infrastructure

Dick Lefkon
Year 2000 Committee of AITP SIG-Mainframe

As of mid-1997, the two Y2K issues of largest financial significance remained unresolved: certification and capitalization.

True Year 200 Conformance certification of software and / or hardware is critical to all user organizations—if perhaps not to the Y2K-pooh-poohing PC houses whose income will slow dramatically as budgets adjust to Y2K.

True certification would collapse many Y2K conversion projects down a purchasing agent sitting at a desk surrounded by vendor catalogues.

GSA, among others, posts an internet list of Y2K product status vendor statements.

But as of late 1997, we had no announcement of real Y2K testing centers paralleling Underwriters Labs "UL" mark for electrical safety, or Corporation for Open Systems' "COS" mark for network interoperability.

Among others, ITAA offered a "registry" or "process survey," depending on the essential honesty of the applicant vendor. Certification is not awarded to individual, tested products. No UL or COS mark yet.

Mitre Corporation does test individual software product Y2K conformance for the U.S. military: specific products but not a service to the public. Comdisco offers a custom offsite testing program for companies' Y2K-modified code: Apparently restricted to programs' main paths, this others' offsite Y2K testing helps consulting houses do Y2K conversions.

Capitalization issues affect neither conformance nor performance, but they may sink many companies and are meaningful, too, for government agencies.

Most of Europe makes no tax distinction between a business expense and a capital outlay. We do: Most office PCs are expensed this year, although originally most were capitalized over several years.

The accounting standards boards (FASB and GASB) treat new software builds or purchases as capitalizable, but today mere modifications (like Y2K) must be expensed in the year incurred.

Taken on its face, expensing Y2K means noticeable drops in corporate (and supplier) income, a $100–$200 million Federal tax shortfall, and a stock market crash as shareholders sell upon learning their favorite companies will be reporting a nickel less profit per share.

One solution proposed by the author (and surely others) consists of Federal legislation offering companies both declaration alternatives for Y2K costs, sufficiently well drawn to exclude IS carpet-cleaning and other irrelevant budgets from that Y2K category sunsetting in 2001.

In a network posting, H Husman offered a silk purse for the 90 percent of Y2K which isn't code change, explaining it "should be rolled out to the rest of the organization to provide ongoing 'infrastructure' advantages." For example:

- ➢ General
  - Standards definition and implementation
  - Mainframe platform and client server platform impact analysis
  - Test environment assessment, operating environment inventory
  - Vendor contact; external interface contact
  - Tool assessment; tool implementation / validation / training
  - Inventory control, impact analysis control
  - Project estimating
  - Facilities management and planning (workspace and access utilization)
  - Capacity planning (system utilization)
- ➢ Configuration management process
  - Change management, issue tracking / prioritization / classification / assignment, change requirement definition (work packet), change request assignment
- ➢ Checkout
  - Work packet assignment, cross reference, impact analysis
- ➢ Quality assurance
  - Migration to test, rebuild, make, test, review of development standards compliance, functional standards compliance, delta versus production version, concurrent maintenance reconciliation, linked component reconciliation
- ➢ Production migration
- ➢ Customer verification
- ➢ Development / maintenance cycle:
  - Track work packet, accept work packet, review metrics, review cross-references, review other impacted work, review change impact analysis, analyze re to change request, change, recompile, review, unit test

CEOs and CIOs may want to make sure the CFO involves the firm's top accounting/audit specialists, since optimal classification of Y2K proportions can materially affect the firm's published financial results.

12 Millennium Rollover: The Year 2000 Problem

NIST Computer Systems Laboratory

At one second after January 1, 2000, millions of people will celebrate the beginning of a new year. Many people will also rue the day because of computer hardware and software problems that will create havoc for those who are not prepared. Simply put, many hardware and software systems will cease to work or will produce wrong answers when the Year 2000 arrives. This bulletin provides information that CSL has collected from a variety of sources on the extent of the Year 2000 problem, what organizations are doing about the problem, and where help can be found to deal with the problem.

The Year 2000 Problem

For 30 or 40 years, programmers have stored date information in "MM/DD/YY" format to conserve space in disk storage and computer memory. They adjusted computations to take the two-digit year into consideration when computing time periods, ending dates, and the like. Programmers represented years in the twentieth century as two digits without considering what might happen once the Year 2000 rolled around. At that time, most programmers and project leaders figured that their programs would not last into the twenty-first century. In hindsight, it seems these people should have known better, but they were trying to perform a service to their management by conserving expensive disk and computer memory. Adding two century digits to a date field for a 100-million record file would have added at least 100 megabytes of storage requirement to a disk that cost upwards of $20,000 for 15 to20 megabytes. It made economic sense to lop off the two century digits.

Now, the industry faces the problem of adding those two century digits back into the date field in order to keep software running and producing correct output. The problem, however, is not isolated to software. Hardware will also cause difficulties for system administrators and chief information officers. System clocks on virtually every personal computer will wind up with corrupted dates on January 1, 2000.

In some cases, the date will appear to roll over to the correct date, but when the machine is turned off and then back on for the next session, an odd date will have taken its place. It may appear as January 1, 1980; January 4, 1980; January 1, %000; or some other combination of characters, all of which will produce erroneous results. The dilemma is not limited to personal computers. Some workstations, minicomputers, mainframes, elevators, and automobile central computers will fall victim to the insidious problem. In most cases, software patches can alleviate the problem to a more-or-less livable extent, but in some cases, the date issue can be resolved only by replacing the hardware.

In software, the problem will be most visible in sorting routines that sort on two-digit year fields. Storing 1999 as 99 and 2000 as 00 will cause the 00 date fields to sort out before the 99 date fields. The consequences of this action can be determined only after the context of its use is understood. Additional difficulties will crop up, and already have.

In one case, a bank's irreplaceable backup tapes were almost used as scratch tapes when a mainframe operator discovered the discrepancy and pulled them from the scratch tape bin. The problem came from the tape management software's use of the date "00/00/00" as the scratch tape indicator in the tape label retention date field. In 1995, tape backups were made with a retention date of December 31, 2000, which was stored in the tape header as "12/31/00." The tape management software looked at only the year portion of the retention date and decided that they had been around long enough. Thank goodness for the observant operator!

Year 2000 horror stories abound, all with the same lesson to be learned. Hopefully, senior executives and chief information officers will realize the severity of the problem and take preventive action. Unfortunately, the solution is expensive and labor-intensive, but there is hope and experience from those who have already taken corrective measures.

What Organizations Can Do About the Problem

William M. Ulrich, in *Application Development Trends'* February 1996 issue, describes the essential elements of a strategy for assisting organizations in solving the Year 2000 problem. These elements include:

- performing an enterprise-wide assessment of the extent of the problem;
- assessing the infrastructure in place and additional requirements to support any new functions associated with the solution;
- deploying strategies for solutions;
- defining validation strategies for testing modifications and assessing the compliance of new software to standards;

- detailing budgeting strategies.

Foremost in deciding what to do is estimating the extent of the problem. For software, the Gartner Group estimated that it will cost between $0.50 and $1 or more per line of executable code to analyze, modify, and test the software. [Editor: Gartner has now approximately doubled this cost-per-line estimate.] Organizations in general have found that 1–2 percent of code will be affected and will have to be modified, but all of the code must be analyzed to make this determination. Estimates translate into one staff-year per 100,000 lines of code! Some organizations, such as banks, may have as many as 10 million lines of code with a higher affected rate than organizations that use information technology to keep accounts, mailing lists, personnel records, etc. This translates into 100 staff-years of effort.

Time Is of the Essence

Once the extent of the problem has been defined, organizations need to formulate a time frame for corrective action and start the process as soon as possible. All of the work should be done before the start of the year 1999 in order to have a sufficient shakedown period for testing changes. With only 220 effective workdays per year (after two weeks of vacation, holidays, and sick leave are factored out), approximately 600 workdays remain until the end of the year 1998. One hundred staff-years over 600 days requires at least 35 persons working on the problem full-time. A large organization may spend between $5 million and $10 million on corrective action. The Gartner Group estimates that Fortune 500 companies will spend between $10 million and $40 million each. Worldwide, the figure is $300–600 billion.

Table 11.1 presents statistics collected over six months from various messages and notices on the World Wide Web (WWW). While not rigorously measured, these figures give an indication of what others have found in trying to deal with the enormity of the problem. The average from this information is that 167,000 lines of code per staff-year can be analyzed, modified, and tested. The scope of the problem for individual organizations can be bounded using a ballpark estimating factor between 100,000 and 167,000 lines of code per staff-year.

A Plan of Action

The most reasonable solution is to attack the problem one step at a time. A suggested means of planning for the work may include the following steps:

Table 12.1 Size and Effort Estimates

Comments	*Lines of Code*	*Estimated Staff-hours*
Manufacturing system	1,200,000	2,000
Commercial off-the-shelf (source code available)	2,000,000	2,500
2,000 programs	7,000,000	38,000
Retail system	7,500,000	75,000
401K system	1,300,000	9,000
7,000 COBOL programs (83% were affected)	12,000,000	200,000
Total	***31,000,000***	***326,500***

- Select a product to assist in managing the inventory of software and databases involved. Select one or more products to assist in analyzing the software and estimating the extent of the problem. Some of these products will also modify the software and data automatically, but cannot do so for every case. (Some computations are date-related, but cannot be determined from the source code. In such cases, an individual must analyze the source code line by line.)
- Inventory applications, libraries, databases, extraneous files, documentation, and other items that have importance within specific systems. Identify who is responsible for each item.
- Analyze the applications and data. Estimate modifying the source code alone to change those locations that perform date computations and logic operations based on dates. Perform a second estimation that includes modifying databases and all source code that references data fields and all source code affected in the first estimate. If there is an insignificant difference between the two estimates, the recommended course of action is to modify both the databases and the source code. It may be less expensive in the short run to modify only the source code, but more expensive in the long run if maintenance problems crop up over time due to the date processing fixes.
- Assemble a team of programmers, application experts, database designers, and project management based on the overall system requirements. Once estimates are known, the number of personnel

required can be determined, particularly in view of the automated tools selected for use.

- Modify the system. Three major options are: a) modify the source code to manipulate and perform computations on dates with century digits included; b) use a sliding window time frame to determine date context for computations; and c) incorporate packed date fields and use specialized subroutines for performing the computations. All three of these are expensive and may lead to further maintenance problems in the long run.
- Test the modifications. Allow 40–50 percent of the overall project resources for testing, even more if the database is modified. This includes testing documentation to ensure that directions are correct and correspond to the changes made.

Sources of Help for Dealing with the Problem

The major obstacles in succeeding with a Year 2000 problem are:

- getting executive management to acknowledge the problem and take serious action;
- finding the right suite of tools to assist in the conversion process; and
- enlisting the help of knowledgeable professionals.

13 *So You Can’t Program Your VCR?*

Harold Carruthers
Edward Jones & Co.

[Editor’s Note: As you peruse “Year 2000” magazine articles, books, and even Congressional testimony, you may think you have found a look-alike predecessor to this landmark article by Harold Carruthers. They’re look-alikes, all right, but it is Carruthers who spoke first. Congress has had a 1996 version of this book (and Carruthers’ chapter) since it first appeared. An expert witness received his permission to use it in widely-reported testimony. Others have expanded his alert. e.g., on “Fire Suppression Systems” into a long list of fire suppression devices—interesting to software specialists but quite superfluous to your Fire Control unit. And so, read this definitive chapter with relish—then staple it to the chairs of your health manager and major premises officers![

Some processes are so complex that the most intelligent, highly trained person could not control them due to the reaction speed required, the amount of information to analyze, and the accuracy required in responding to "events" as they occur. If you doubt these facts, then I invite you to control every aspect of a space shuttle launch by using people alone. No automation, no devices, no chance. It isn't possible to maintain safety and still launch a shuttle without automation or automated devices.

Automated devices don't malfunction unless they have a physical defect or the software, microcode, firmware, ROM, PROM, and EPROM program code driving the device malfunctions. Typically, that device encounters a situation that its software was never designed to recognize or act upon. Notice, I didn't say that the situation would never occur. Instead, I said the software was never designed to recognize or act upon the situation. Automated devices must be considered *limited function* computers and subject to the same vulnerabilities as their *full-function* counterparts, such as mainframes, personal computers, or client / server-based systems.

Because automated devices are *limited-function* computers, we can use the same approach for Year 2000 correction processes that we use for *full-function* computers. That process would be:

- Get an inventory of all automated devices. Anything that uses electricity is a candidate.
- Initiate specific Y2K compliance requirements in any RFP devices, include proper language in contracts, and improve device testing procedures to include Y2K compliance.
- Determine every device where date, time, or duration related information is processed, computed, sent, or received.
- Prioritize each device's function to the business's success and customers' well being.
- Determine if any device failure is episodic (one-time occurrence at ultimate midnight) or ongoing (all other times). Many failures can be missed by turning the device *off* and *on* as needed.
- Determine the vendors for each automated device.
- Contact all vendors and request Y2K compliance information for the device and each sub-component. This is much more complex than getting Y2K compliance statements for the more typical business software packages. You'll probably need to send letters to each vendor of each sub-component to get compliance statements. Expect to spend a lot of time here with frustratingly little results.
- Replace, retrofit, or retire the automated device based on failure point, priority, vendor responsiveness, and the time required to accomplish a replacement / retrofit. Retrofit would be an ordeal

requiring disassembly, replacement of the affected components and reassembly.

- Test the Y2K compliant devices for proper function and repeat all steps as required.

Some of the following automated devices have either shown Y2K problems or are date, time, or duration dependent. This is just a partial list, and you need to more fully investigate your company's devices.

Critical systems

- *Fire suppression systems:* A Y2K failure here will shut down most of your facility.
- *Security systems functions included in badge readers, elevators, surveillance systems, parking lot gates for off-hours, vaults of many kinds:* The controlling system may have authorization based on from-to dates and / or from-to times. You still need to get to work.
- *Elevator control:* Some elevator systems will go to the bottom floor and stay there if the automation believes maintenance hasn't been performed as required. That decision is based on a comparison of the current date and the date of last maintenance as entered into the systems.
- *Time-dependent controls such as parking lot lighting, programmable thermostats controlling HVAC, elevator functions:* Some devices work only during certain times of the day and / or only certain days.
- *Power-management functions for HVAC usage and control, UPS backups and related components, off-hour power availability for lighting the building:* These are very complex issues because there are many levels on levels of monitoring. You want to be able to use and monitor devices once you get to the building and do so in relative comfort.
- *Environmental-safety systems for detecting changes in humidity, temperature, CO2 levels:* "Extreme" changes are monitored. Some of those changes are based on duration and / or *spike* measurements. You want your employees to be comfortable and safe while you are at work.
- *Phone systems including PBX, voicemail, switching, and fax services:* You may need to call your vendors for help.
- *Robotics systems:* These include automated assembly processes driven by functions that happen in a certain order for a certain amount of time. Failures in robotics systems have caused products to be wasted for a reason as simple as a container not being at the end of a particular assembly line. The next robotic system continued to visit the area faithfully and on schedule but seeing

no product (it was all on the floor) decided there was nothing to do and went back where it came from. Failures of this type are definitely episodic.

- *Any automated device sensitive to the change to and from daylight savings time.*

Non-critical systems

- *Electronic timeclocks:* Do you really want to fight a labor suit (real or imagined) because a timeclock fouled up and didn't record employee time properly?
- *Landscaping systems:* Nothing like lawn sprinklers or water fountains going off in the middle of winter to cause problems. Your corporate sign might not light up at night.
- *Vending machines:* Some machines have direct interfaces with the vendor to indicate low-on-stock and stale-dated items. These systems order more items that will immediately go stale and then order more items that go stale then…Similar failures have already occurred.
- *Miscellaneous times:* Coffee pots and other equipment or timers can be programmed to operate on specific days and at specific times.

Obviously, getting ready for the Year 2000 will require lots of planning and preparation, but the pay-off will be to ensure that your company doesn't experience a millennium meltdown.

14 The Successful Year 2000 Project Office

Dick Lefkon
Year 2000 Committee of AITP SIG-Mainframe

During 1997 and 1998, organizations will staff up on legacy application programmers, testers and managers.

Whether stationed in-house or at a consulting workplace, the coders and testers will probably come to do an acceptable job in spite of apparent or discovered experience shortfalls.

Not so the managers. Soon-to-be-anointed Y2K project leaders typically have held the title less than 2.5 years and never themselves ran a huge, tight-timeframe project. They graduated to project leader from being lead technician, and may or may not have received intensive project management training. Also—except for those in brokerage during the 1995 “T+3” implementation[1]—most may never have seen an enterprise-wide upgrade.

A visible structure for Project Management should be in place at each enterprise before doubling its IS budget to accommodate the Year 2000 rollover. Not only will this provide guidelines for newly installed project managers, it also will enable in-house and outsourced resources to be controlled, coordinated and optimized.

Having run a number of general PMOs in years past, the author had set up Y2K-specific PMOs for three multi-billion dollar organizations in the year preceding publication.

Overall, a Project Management Office will usually:

- Report to the information executive: CIO, treasurer, etc.
- Serve as conduit for outsourcing and purchasing.
- Set up and administer the “funnel” to prioritize projects.
- Oversee and provide templates for others’ project management.
- Delineate (sometimes coordinate) system assurance testing.
- Coordinate use of Y2K “time machine” / firewall vs. other CPUs.
- Provide periodic project progress reports and evaluations.
- Manage external contracts—including Y2K conversion progress.
- Provide central functions such as
  - Disaster recovery planning & management
  - Central data dictionary and repository
  - Coordination / planning of interplatform / interproject matters.

The Project Management Office may be devoid of specific projects, or it may be assigned management of a sensitive effort such as Y2K. Because there simply aren’t enough seasoned managers to go around, it is critical to implement a published structure within which all work.

Having an established successful work intake “funnel” gives the PMO an ongoing ability to triage nonessential conversions away from critical ones. And, when the inevitable bulge arises, organizations can divert resources from non-Y2K / non-production efforts—not be forced to start inventing the wheel near the scheduled end of the Y2K effort.

1. Because of a hard-deadline regulatory requirement, stock and bond trading standard “settlement” was shortened from five to three business days. Money payment and securities delivery were required to occur two business days sooner, and legacy code had to be changed enterprise-wide for T+3 instead of T+5. To modify such consequences throughout all systems, parallels much of the logic component of Y2K, albeit not the data component.

15 Year 2000: The End of IS?

Gerhard Adam
Syspro, Inc.

The end of IS? How can the Year 2000 possibly be the end of IS? The industry is moving in dozens of directions with newer, better technologies being constantly introduced and exploited. Surely, the Year 2000 problem isn't that serious, or that difficult to fix.

In truth the problem is technically simple to fix. The most serious problem faced by the I/S industry regarding the Year 2000 is the almost uniform lack of concern for fixing it. Make no mistake, this is a significant logistical problem and is extremely labor intensive. For many organizations, it is already too late to easily fix it. So what happened?

What happened is what always happens in I/S; the ability to put off fixing a problem which will occur tomorrow in favor of the one occurring today. This may seem to be sensible, but in reality everyone has known that the Year 2000 would be an issue for years; yet instead of addressing it when it would be easy, most organizations simply ignored the problem. The time is past for which ignoring this problem can be continued without peril. Does this sound too ominous? Too much "doomsday" talk? Well, let's examine the problem and see.

The Problem

The problem of the Year 2000 is actually two separate problems. The first and most difficult involves the current use of two-digit years instead of the full four digits. Since only two digits are used, any operation in which the date is used for comparisons, calculations, sorts, etc., will be in error since the "00" in the Year 2000 will be less than the "99" of 1999. This simple condition causes most instances of date use to be compromised.

The second problem is whether the Year 2000 is a leap year or not. Briefly, it is. The rules governing leap year are normally that the year must be divisible by four. However, at the century mark the requirement is that the year be divisible by 400 as well.

The Consequences

What happens if we don't make these changes? Will systems fail or stop working? Probably not. What will happen is infinitely worse. Systems will continue to operate and simply calculate, sort, and compare every date based only on its numeric value without regard for the results. The ability to corrupt untold volumes of data is truly frightening.

Many articles discuss the Year 2000 with absurd outcomes for date calculations and imply that this is what will be encountered. For example, a baby born in 1999 will be calculated as being –99 years old. In most cases this is unlikely and systems will behave much worse.

For example, an individual born in 1953 will become 53 years old instead of the correct 47 years old. This error is not quite so obvious and could easily become part of a permanent data base without anyone noticing anything wrong. This error would occur because of numerous assumptions made by programmers. First, the subtraction of 53 from 00 would result in an answer of -53. However, most programs do not allow a signed result and so the number is arbitrarily made positive, hence 53 years old.

Imagine how many inventory systems could erroneously calculate shipping times or modify schedules because of the transition between 1999 and 2000. For example, if a business wanted to determine which ordered items were scheduled to arrive within the next sixty days, the calculation would typically take the shipping date and subtract the ordering date from it to determine if it is sixty days or less. If, however, this calculation occurs during the transition between 1999 and 2000, the results will be off probably by several months (see attached Example #1 pg.46). How is such an error going to be detected?

What about loans which involve "aging" an account? How many collection systems are going to be kicked off because of improper aging? How many accounts may be written off as unrecoverable debts because of improper calculations?

Imagine a personnel department attempting to obtain a list of all employees hired after June 1999. If, for example, this query were to occur during June 2000, no employees hired during that year would be included in the query since 00 obviously occurs before 99, and so the data base would not return those results. We would be missing six months worth of data, but the system wouldn't even indicate the loss.

Many people imagine that this problem exists only on large mainframes because of some implicit belief that it is all the fault of "legacy" systems. This is completely wrong. Test your PC[2] using DOS/Windows and see if it can handle the Year 2000 transition. In 70 percent of the sys-

2. To test your PC simply set the date to December 31, 1999 11:50 PM. Power off the system and wait about fifteen minutes. Power up the system and check the date. In a large number of cases, the system date reverts to the BIOS date of the machine (Jan 4, 1980).

tems tested by my company, the system reset the date to Jan 4, 1980. How many PC's might be connected to other systems that would take action based on that date? Is it possible that a centralized backup / archival mechanism could take this "1980" data and archive, or simply delete it? The most recent work of an individual would be treated as if it were 20 years old. Is your organization even aware of this exposure? If not, you can begin to understand why this problem is so serious.

The Scope

How can this be fixed? The most straightforward approach is simply to expand the date fields of every program and file which uses them. The requirements of this expansion are not difficult, but they are quite labor intensive.

First, review all vendor products for Year 2000 support, especially those which provide automated functions based on date and time (i.e.: scheduling systems, storage management, tape management, etc.). It is also necessary to ensure that the proper language support is available so that programs can use the four digit year format.

Second, every program will have to be reviewed for date references. Don't be reassured if the program uses it only in a report. The file which the program uses may have to be changed to support a program which maintains it, and consequently the report program will have to be changed as well. The easiest and most effective approach is to assume that all programs will have to be changed.

Third, each program must be changed, and the files it uses will have to be converted as well.

Fourth, programs which display or print the date will have to reviewed to see if the expansion of the date field results in changes to terminal displays, workstations, or reports.

Fifth, a concerted effort to trace all secondary uses of data (i.e., user constructed queries, spreadsheets, personal data bases) to ensure that users understand the consequences of this format change.

The final step is to test these systems and replace faulty existing ones.

There are other technical approaches which may be used, but despite their initial appeal their use may actually make the conversion more difficult and error-prone. The technical alternatives, windowing and date encoding, can be summarized as follows.

Windowing is based on the recognition that the two digits representing a year will only occur once in any 100 year "window" and therefore the application can be coded to exploit this. For example, if a "window" is chosen between 1950 and 2049, the application would simply check whether the date was less than fifty (50) to indicate that it has a century indicator of "20". Conversely, if the date is greater than or equal to fifty (50), then the century indicator is a "19". The major drawback to this technique is that the determination of the century is firmly embedded in the logic of the program and unavailable to anyone else. [Editor's Note: See Part 7 for a possible solution.] If the data were downloaded to a PC spreadsheet, there would be no way to assign a century indicator without knowing explicitly which window the application used.

Date encoding attempts to use the existing date field to indicate the proper century by setting bits or assigning binary codes. This method has numerous caveats which should be carefully reviewed. First, file transfers may become impossible since EBCDIC to ASCII conversions cannot allow binary data to be imbedded in the file. In addition, high level languages can only handle binary data with difficulty so the likelihood of errors in programming logic increases. If the method chosen involves assigning a code, then a file conversion may still be necessary to allow "downward" compatibility with existing data. This system will also fail if the data is already represented in a binary or packed decimal type format.

Summary

It is important to understand that this problem is not technically difficult, nor complicated to correct. What is difficult is getting people interested in fixing it. Everyone has an excuse as to why they have more important things to do, or how there is plenty of time to fix this problem. But let's have a reality check.

Assuming that only programs and files need to be modified and taking about a day to do this for each program, the important question is how many programs need to be changed? Consider that there are fewer than 1000 days until the Year 2000. At one day each,[3] it doesn't leave a lot of time for very many programs. An organization with 10,000 programs would have some serious thinking to do.

Don't think to console yourself with the idea that this is a vendor problem, or that the vendors will develop some magical tool to fix this. While vendors certainly have a role to play, the problem is ultimately yours to deal with. Even if a vendor supports the four digit year, that is

3. The assumption of one day per program includes the time necessary to convert the files as well. In most cases, the initial research and coordination effort will take more time than the actual programming changes themselves so the estimate of one day (average) per program is probably conservative.

no guarantee that your organization is actually using it. There is even the possibility that programs written to support a four digit year could be modified within your organization and subsequently lose that support.

Another surprising suggestion which is sometimes made is to convert all existing applications to client / server and thereby derive a technical benefit as well as solving the Year 2000 problem. This recommendation fails on several points:

1. If you don't have time to convert existing applications, you most certainly don't have time to completely rewrite them.
2. Even under the best of circumstances, the attempt to completely replace all systems with client / server would be quite ambitious given the time available.
3. Many applications do not have client / server solutions (i.e., batch jobs), so major redesign and / or business re-engineering efforts would be involved.
4. It is an invalid assumption to suppose that PCs don't have a date problem.

Example 1

An organization wants to plan for ordered items which are scheduled to arrive within the next 60 days so that storage space can be planned for and provided (i.e., warehouse, shelf space, etc.). A COBOL program might consist of the logic shown in Figure 15.1 to perform this function.

Let's assume that on January 3, 2000, a batch application is run to produce a report for the warehouse staff with the following data:

Performing the logic as coded in the program, various intermediate calculation errors will cause negative values to be treated as positive (no sign fields) as well as truncation for large values. This will result in a final calculated answer of 222 days, rather than the correct answer of 8 days.

With this result the program logic indicates that this item should be ignored and not included in the report. The error is not intuitively obvious, and would probably go undetected. What would happen to this company once it is discovered that all the reports and planning are completely wrong?

Note: This example does not account for leap year and is intended only to illustrate the potential integrity exposures rather than be a model for programming techniques.

This article is intended as a "wake-up" call to the I / S people who understand the problem but haven't done anything about it. It is too easy to find excuses for not handling this problem, or why it isn't as severe as

Figure 15.1 *COBOL Program*

```
Working Storage Section
77      YEAR-NO                          PIC 9(2).
77      YEAR-DAYS                        PIC 9(3).
77      NEW-DDD                          PIC 9(3).
01 ...
        05        CURR-DATE.
                  10          CURR-YY    PIC 9(2).
                  10          CURR-DDD   PIC 9(3).
        05        ORDER-DATE.
                  10          ORDER-YY   PIC 9(2).
                  10          ORDER-DDD  PIC 9(3).
        05        SHIP-DAYS              PIC 9(3).
PROCEDURE DIVISION.
Subtract ORDER-YY from CURR-YY giving YEAR-NO.
Multiply YEAR-NO by 365 giving YEAR-DAYS.
Add CURR_DDD to YEAR-DAYS giving NEW-DDD.
Subtract ORDER-DDD from NEW-DDD giving SHIP-DAYS.
If SHIP-DAYS > 60 go to IGNORE-IT.
```

represented, or how there is plenty of time. In fact, the only way to know how bad the problem is, is to look at it. Once it is examined, a course of action can be determined. It has been suggested that the Year 2000 problem is being over-hyped and reminds one of the "Boy Who Cried Wolf." This may be, but it is also useful to remember how that story ends.