

THE GLOBAL ECONOMIC IMPACT OF THE YEAR 2000 SOFTWARE PROBLEM

Version 4 – September 23, 1996

Abstract

From the start of the computing era until the early 1990's magnetic storage of information was expensive and storage capacities were limited. As a result, software applications routinely conserved space by using only two digits for recording calendar years; i.e. the year 1990 would be stored as 90. When the 20th century ends, many software applications will stop or produce erroneous results since their logic cannot deal with the transition from 1999 to 2000 when the dates are stored in two-digit form and their calendars change from 99 to 00. Because this problem is embedded in millions of aging software applications, the costs of fixing the "year 2000 problem" appear to constitute the most expensive single problem in human history.

If the problem is not fixed, then the errors in software associated with finance, taxation, insurance, and even operation of aircraft can lead to the most expensive litigation in human history. However, once the problem is fixed, enterprises will have much better knowledge of their software portfolios and application structures than ever before. A strong caution is indicated: failure to repair Year 2000 problems carefully with full testing and performance optimization can degrade software performance and data center throughput by more than 20%. Approximately June of 1997 is the last point at which year 2000 repairs can start with a reasonable probability of finishing before 2000.

Capers Jones, Chairman
Software Productivity Research, Inc.
1 New England Executive Park
Burlington, MA 01803-5005

Phone 617 273-0140 X-102
FAX 617 273-5176
Email Capers@spr.com
CompuServe 75430,231
Web http://www.spr.com

Copyright© 1996 by Capers Jones, Chairman, Software Productivity Research, Inc.
All Rights Reserved.

Table of Contents

Executive Overview	4
Introduction	5
Root Causes of the Year 2000 Problem	6
Hazardous Implications of the Year 2000 Problem	7
Beneficial Implications of the Year 2000 Problem	9
Fallacies Associated with the Year 2000 Problem	10
Terms and Concepts Associated with the Year 2000 Problem	12
Application	12
Software Portfolios	12
Data Base	13
Software Staff	14
Enterprise	15
Software Sites	15
Function Points Versus Lines of Code Metrics for the Year 2000 Problem	16
Function Points and Programming Languages	17
The Volume of U.S. Software Expressed in Function Points	18
Function Points and Mixed Language Projects	20
The Size of the Year 2000 Problem for the United States	21
The Volume of Software Installed in the United States	21
Performance Implications of the Year 2000 Problem	24
Year 2000 Repairs by Industry	25
Year 2000 Repairs by Company Size	26
Year 2000 Repairs by Programming Language	27
Year 2000 Repairs by State	28
Repairing Data Bases, Repositories, and Data Warehouses	33
Litigation Potential for the Year 2000 Problem	35
Business Failure Potential for the Year 2000 Problem	37
Business Failure Potentials Based on Company Size	37
Business Failure Potentials Based on Industry	38
Business Failure Potentials Based on Financial Health	39
Business Failure Potentials Based on Litigation	39

Table of Contents - Continued

Aggregation of All Year 2000 Software Related Costs for the United States	40
The Emergence of the Year 2000 Repair Industry	42
International Year 2000 Repair Effort	46
Year 2000 Repairs by Country	46
Year 2000 Repairs by Urban Area	51
Aggregation of All Year 2000 Software Related Costs Globally	53
Benefits of Solving the Year 2000 Problem	55
Summary and Conclusions	56
Additional Information on the Year 2000 Problem	57

THE GLOBAL ECONOMIC IMPACT OF THE YEAR 2000 PROBLEM

EXECUTIVE OVERVIEW

The year 2000 software problem concerns the widespread practice of storing calendar year dates in two digit form, so that the year 1997 would be stored as 97. At the end of the 20th century, many software applications will stop working or create erroneous results when the year switches from 1999 to 2000 at midnight on December 31. This is because many applications use dates for time-sensitive calculations, and the sequence of 99 followed by 00 can cause software crashes or incorrect results for many important calculations such as interest rates and mortgage payments.

The year 2000 problem will be one of the most expensive problems in human history. For the United States, more than four months of effort may be needed on the part of every software professional in the country to repair the year 2000 problem. The year 2000 repair costs may exceed \$2000 for every working person in the United States, or more than \$900 for every citizen of the United States.

The year 2000 problem is not trivial and will not go away if ignored. Failure to correct the year 2000 problem can lead to serious consequences, including several kinds of litigation, possible bankruptcy, and possible business failure.

Software and corporate executives may be held personally liable for some of the consequences of the year 2000 problem unless prompt and energetic actions are taken to correct the problem. The year 2000 problem may lead to lawsuits against corporate executives for violation of fiduciary duty, and against software executives for professional malpractice.

A variety of tools and services are now available to assist in finding and repairing the year 2000 problem. However, June of the year 1997 is the last month and year in which there is a reasonable possibility of finding and repairing all year 2000 instances before the end of 1999. Executives and managers are urged to explore the year 2000 situation in their own enterprises as rapidly as possible.

Although the year 2000 problem is very serious, the companies and organizations that are able to resolve the problem will find some significant benefits too. Software applications that are year 2000 compliant should be much easier to extend and maintain than their predecessors. Also, the work of solving the year 2000 problem will give enterprises a much better understanding of how much software they own and its business value than has yet been possible.

THE GLOBAL ECONOMIC IMPACT OF THE YEAR 2000 PROBLEM

INTRODUCTION

In popular detective-story fiction, it is well known that poisons such as arsenic accumulate slowly in the body. Tiny doses, each harmless in itself, can slowly accumulate until the victim perishes.

In some ways, the “year 2000 software problem” resembles the slow accumulation of arsenic. For many years, software applications have been built with two-digit fields for year dates; i.e. the year 1990 would be stored as 90.

Year by year these two-digit fields have been accumulating in software packages all over the world. The time is rapidly approaching when the slow and steady accumulation of these two-digit date fields will cause the applications containing them to perish. It may be that some companies will perish with them! Unfortunately this is not harmless fiction. This is the real world.

Since fiscal years are often decoupled from calendar years, even before the end of calendar year 1999 many applications will stop or will begin to produce incorrect results. Already in 1996 some credit card companies whose cards are valid for five years have encountered the Year 2000 problem.

The Year 2000 problem will not stop abruptly in the Year 2000 either. Continuing costs can be anticipated at least through 2005 AD and some will probably run even further. The post Year 2000 expenses will consist of: 1) Fixing obscure Year 2000 instances that were missed prior to the end of the century; 2) Hardware upgrades and retuning of applications whose performance was degraded by hasty repairs; 3) Litigation expenses for the host of anticipated Year 2000 law suits.

Computers and hardware devices will be affected by the Year 2000 problem too, since their internal clocks contain built-in calendar functions. Computers and software now drive the main operating components of every major company, government, and military organization in the world. Therefore the year 2000 problem is very likely to be one of the most expensive single problems in human history. There are four major aspects of the costs of the year 2000 problem that need to be considered:

- The costs of finding and fixing the year 2000 problem in millions of aging software applications. These costs can be apportioned into the costs for finding the myriad locations of the date fields, the costs of fixing the date fields, and the costs of testing the fixes to ensure that no other damage has occurred during the repairs.
- The costs of the litigation that will result in cases where the problem is not fixed. This second “year 2000” cost component will run well into the 21st century and will probably far exceed the direct costs of repairing the problem itself.

- The continuing costs for running applications that halt frequently, no longer execute efficiently, and degrade data center operations. These are likely to be long-term continuing problems that can degrade data center throughput and software execution speed by 20% or more. This problem may exist well into the 21st century since many companies are careless about testing applications after Year 2000 repairs are made, and even more lax about monitoring performance or reoptimizing the applications.
- The probability of business failures, bankruptcies, and damages to national economies as a direct result of the Year 2000 impact on the computers and software which contain vital business information.

One surprising aspect of the year 2000 problem is that some benefits can be envisioned as well as costs. Once the problem is fixed, enterprises will have far better data dictionaries than ever before. They will also have a much better understanding of the size, age, and complexity of their software portfolios and data bases.

Recall that major disasters such as the San Francisco fire and the Tokyo earthquake led to significant improvements in architecture and construction so that buildings are now much safer than they were prior to these disasters. It is likely that software applications constructed after the year 2000 crisis will be much safer and more stable than those built in the past.

In addition, enterprises will learn much more about their future software needs and the costs of both building and maintaining software than they know today. An unexpected byproduct of the year 2000 problem is that software may finally become a commodity with tangible economic aspects.

Root Causes of the Year 2000 Problem

Many articles have been written on why the year 2000 problem will occur, so it is only necessary to include a short background discussion here. The root cause can be traced back to the early days of computers, when information was stored on punched cards. Data storage was so limited and so expensive that any method that could save storage was readily adopted. Since no one in the 1950's or 1960's had any idea how long software would last, it seemed natural to store dates in two-digit form; i.e. 1965 would be stored simply as 65. This method was convenient and seemingly effective.

When magnetic storage was first introduced the cost of data storage declined slightly, but the early tape and disk based systems still were limited in capacity. Also, many card-based systems were transferred to tape or disk. But the original versions of the source code and the two-digit date logic continued to be used since there was no immediate reason to change it.

By the late 1970's and early 1980's it started to be noted that software applications were sometimes having remarkably long lives. For example, IBM's MVS operating system was approaching 20 years of age, as were a number of other widely used applications. Some tremors of alarm about date limits began to show up, but there was still no immediate serious alarm since the end of the century was 20 years away.

It was not until the early 1990's and the advent of optical storage that data storage costs declined to such a level as to be almost irrelevant. The early 1990's would have been the best time for addressing the Year 2000 problem, but for sociological reasons the human species is not very effective in disaster prevention.

Also, by the 1990's quite a significant amount of the damage had long been done. Millions of applications with two-digit date fields had already been written and many of them were in daily use throughout the world.

Hazardous Implications of the Year 2000 Problem

Here too, quite a bit has already been written about the kinds of applications that are going to be affected by the year 2000 problem. Suffice it to say the problem is not restricted to aging legacy applications. The year 2000 problem is also common in modern personal computer applications, and even embedded in the hardware of both mainframes and personal computers themselves!

Any software package that uses dates or has calendar routines as embedded functions is likely to be affected by the year 2000 problem. Some common applications that will have to be modified include:

- Software with long-range calculations such as mortgages, life insurance premiums, interest compounding, pension payments, and social security benefit calculations.
- Software calendar utility functions in commercial spreadsheets such as Lotus, Excel, Quattro, and many others. (Some spreadsheets can move past the year 2000 problem, but their calendars cannot handle dates past 2075.)
- Calendar applications embedded in personal information managers (PIM) and hand-held devices as well as those in personal computers.
- Software on board commercial and military aircraft, weapons systems, and also software on board satellites.
- Computer operating systems utilize calendar and clock routines, and many are likely to fail at midnight on 1999.

- Software that controls telephone switching systems is also subject to Year 2000 problems and this is likely to cause some errors in billing and possible disruptions of telecommunications services.

The most serious implications are the litigation consequences of the failure of calendar routines. Many important financial applications will be affected. Worse, some of the consequences may even threaten human lives and safety.

At least six kinds of potential litigation can be envisioned as a byproduct of the year 2000 problem:

- 1) Litigation filed by clients whose finances or investments have been damaged.
- 2) Litigation filed by shareholders of companies whose software does not safely make the year 2000 transition.
- 3) Litigation associated with any deaths or injuries derived from the year 2000 problem.
- 4) Class-action litigation filed by various affected customers of computers or software packages.
- 5) Litigation filed by companies who utilized outsource vendors, contractors, consultants, or commercial Year 2000 tools but where Year 2000 problems still slipped through and caused damage.
- 6) Litigation against hardware manufacturers such as computer companies and defense contractors if the Year 2000 problem resides in hardware or embedded microcode as well as software.

The end of the 20th century is likely to be a very hazardous time for many executives, and for almost all software executives. Any executive who is in a position of fiduciary responsibility should by now be taking energetic actions to solve the year 2000 problem. These same executives should also be in discussion with their legal counsels regarding the probable liabilities that they and their companies will be facing over the next 48 months and on into the 21st century.

Governments are not immune from the year 2000 problem. All government agencies associated with revenues such as state and federal tax agencies are probably going to have major year 2000 problems. This is also true of agencies such as social security that deal with benefits.

While national governments will be the most heavily impacted by the Year 2000 problem in terms of the volume of changes needed, the host of state, provincial, and local government bodies that use computers are likely to be damaged also.

It can be anticipated that major errors are likely to occur in a variety of governmental applications dealing with welfare, unemployment, taxation, even mundane accounts payable and receivable. A very likely byproduct of the Year 2000 problem will be sharp increases in local taxes to defray some of the Year 2000 expenses.

Although the military implications of the year 2000 problem are not widely discussed in the software press, the on-board computers in many weapons systems, ships, tanks, and military aircraft are going to be affected by the year 2000 problem. Logistics systems and various command and control systems will also be affected.

Since the U.S. military services are far and away the most automated and computerized armed forces in the world, the United States Department of Defense will be facing one of the largest military expenses in human history.

Beneficial Implications of the Year 2000 Problem

Since the magnitude of the year 2000 hazards are so enormous, most of the software literature has focused on the negative side of the problem. What is somewhat surprising until considered, is that there will be some significant benefits associated with solving the year 2000 problem.

Most companies do not really know how much software they currently own, and do not even have a clue as to how much software they truly need. This statement is true of data bases and information owned by enterprises. Also, both data bases and legacy applications tend to be added in a patchwork fashion in response to random but urgent needs.

Very few enterprises have really effective data dictionaries and some do not even know how many data bases they own. As a result, two of the most vital and expensive business commodities of world history, software and data, have no solid economic understanding.

In order to solve the year 2000 problem and minimize the hazards, enterprises will be forced to develop a very detailed inventory of the software applications and data bases which they own. In addition, it will be necessary to develop a very detailed enterprise wide data dictionary.

Therefore as companies begin to solve the year 2000 problem, they will find that they have much better knowledge and control over their software than they have ever had before. Within a few years after the end of the century, the companies that have solved their year 2000 problems will be in a very strong position to plan new applications and align their software and data assets with their business strategies.

Not only that, but the year 2000 problem itself is really only the tip of the iceberg. Date fields are not the only kinds of fields that were arbitrarily truncated to save space. Many applications also have trouble with financial calculations, name fields, and other kinds of information because not enough space was allocated when the applications were built.

The same tools and methods being applied to the year 2000 problem can also be applied to similar topics. Thus software created after the end of the 20th century should be far better structured and more robust than software typically created before the end of the 20th century.

Fallacies Associated With the Year 2000 Problem

Problems of the magnitude and seriousness of the Year 2000 problem are difficult to encompass mentally, and hence many executives are in what psychologists call the “denial phase” or asserting that the problem is not really going to occur.

The software trade press has alternated between denial and exaggeration of the problem, with some articles asserting that the Year 2000 problem will blow over, while others assert that the problem is severe enough to trigger a business depression and cause bankruptcy of a significant percentage of U.S. corporations.

One major fallacy is that there is no particular hurry associated with seeking out and fixing the year 2000 problem. Unfortunately, the year 1996 is the last year in which “average” programmers working without sophisticated automation could have had a reasonable chance of finding and fixing the year 2000 problem in a typical mid-sized corporate portfolio of 500,000 function points in size.

June of 1997 is the last time at which even specialists with automated search engines have a high probability of finding, fixing, and testing the year 2000 problem in a typical 500,000 function point corporate portfolio and finishing prior to midnight on December 31, 1999.

The Year 2000 problem is very serious, and will certainly have an affect on many businesses that depend upon computers and software. Some business failures will probably occur, especially among mid-size financial and insurance companies which have lagged in beginning their year 2000 repairs.

The U.S. software industry is also likely to suffer, and the year 2000 problem may be advantageous for countries such as India and the Ukraine which have a large surplus of software talent that could be applied to either fixing the year 2000 problem or to outsourcing other kinds of work while the U.S. software personnel are mired down in year 2000 repairs.

The most dangerous fallacy has been assertions by the press that the year 2000 problem will be solved primarily by scrapping aging legacy applications that embody the year 2000 problem, and replacing them with modern applications that are year 2000 compliant.

A quick look at average software development schedules will illustrate why that assertion is sheer folly. It is now late 1996. The only applications that could be built and deployed between now and the end of 1999 are those that are less than 2000 function points in size:

Table 1 Average Software Development Schedules in Terms of Calendar Months

Size	End-user	MIS	Outsource	Commer.	System	Military	AVERAGE
1FP	0.05	0.10	0.10	0.20	0.20	0.30	0.16
10FP	0.50	0.75	0.90	1.00	1.25	2.00	1.07
100FP	3.50	9.00	9.50	11.00	12.00	15.00	10.00
1000FP	0.00	24.00	22.00	24.00	28.00	38.00	27.20
10000FP	0.00	48.00	44.00	46.00	47.00	64.00	49.80
100000FP	0.00	72.00	68.00	66.00	78.00	85.00	73.80
Average	0.68	25.64	24.08	24.70	27.74	34.05	27.00

Table 1 is taken from the second edition of my recent book Applied Software Measurement (McGraw Hill, 1996). It shows the U.S. averages for six software subindustries and for six size plateaus, each an order of magnitude apart. The size is expressed in terms of function points and ranges from 1 function point to 100,000 function points. The subindustries are those of end-user development, management information systems (MIS), outsource or contract development, systems software, commercial software, and military software.

Note that any software development project larger than about 2000 function points that starts in 1997 will not be completed until sometime after the year 2000 problem has already occurred.

Numerically software applications less than 2000 function points in size comprise about 65% of the applications with year 2000 problems, while the other 45% are larger than 2000 function points. However, in terms of the work effort needed, the situation is reversed: about 65% of the effort for year 2000 repairs will be expended on larger systems in excess of 2000 function points. Indeed, some applications with year 2000 problems are in the size range of 100,000 function points and would take more than 10 years to redevelop.

(A simple rule of thumb for determining approximate software development schedules is to raise the size of the application in function points to the 0.45 power. That rule of thumb gives the number of calendar months from the start of deployment until delivery.)

In other words, none of the major software applications in either the United States or the rest of the world can be replaced between now and the end of the century. You have to fix the year 2000 problem in your current applications, like it or not.

TERMS AND CONCEPTS ASSOCIATED WITH THE YEAR 2000 PROBLEM

In order to evaluate the magnitude of the year 2000 problem, it is necessary to start with some basic terms and definitions of which six terms are critical to understanding of what follows:

- 1) application
- 2) software portfolio
- 3) data base
- 4) software staff
- 5) enterprise
- 6) software sites.

These terms will occur often in the future as the year 2000 problem is discussed, so it is well to begin with an understanding of what they mean.

Application

In the context of the year 2000 problem the term “application” is the general term for both individual computer programs and also the term for linked sets of programs that comprise larger entities termed systems. Software applications range in size from less than 10 function points at the low end to more than 100,000 function points for large systems such as defense applications.

Software applications that deal with calendar data can have a from few up to many hundreds of specific calendar calculation routines that will require modification and repair. Examples of applications that might be affected by the year 2000 problem include payroll applications, banking and financial applications dealing with interest calculations, and government applications dealing with taxes or social security.

However, since modern software applications share data with other applications, the year 2000 problem must also be considered at the level of all software owned by an enterprise.

Software Portfolios

In general use, the term “portfolio” applied to software is defined as the total volume of software owned by an enterprise regardless of whether the software was purchased, leased, or built by the enterprise itself. In the context of the year 2000 problem, this basic concept has some implications that need to be considered.

First, the software which enterprises built for their own use are assumed to be where the bulk of their work will be performed for fixing the year 2000 problem. Presumably the vendors of software packages will be responsible for fixing the year 2000 problem in their commercial offerings. Therefore the portfolio sizes discussed here are the estimated sizes of the applications built and owned by the enterprises themselves.

Thus a software portfolio for a bank or financial institution would consist of the specialized front-office, back-office, and automatic teller software which the bank has constructed or contracted for its own use. Other software used by the bank, such commercial products like Microsoft Excel or Lotus Notes, are assumed to be the responsibility of the vendors for making year 2000 repairs.

This report also assumes that the year 2000 problem will be fixed in a single master copy of an application, regardless of how many additional copies exist. For example, this report is being written using Microsoft Word for Windows Version 6.0. The Microsoft word product is approximately 5000 function points in size.

A large corporation might own 1000 copies of such a product. However, for studying the year 2000 problem it can be assumed that if the problem is fixed once, the fix will be valid for all copies of the product. Therefore in this report the phrase "portfolio size" is defined as the volume of a single copy of each application owned by the enterprise.

Under the single-copy definition most large corporations will own between 250,000 and 1,000,000 function points per site as the sum of one copy of each of their software applications. If each duplicate or all copies were considered, then the volume of software might be 500 to 2500 times larger. However, for dealing with aspects of the year 2000 problem such as finding the affected areas and repairing them, the author assumes that the work will take place on only one master version of each application.

An invisible component of software portfolios are applications built by end-users such as accountants or engineers. These applications are usually kept on personal computers, and except for the developers, they may not be visible to the professional software organizations of the enterprise. However, some of these end-user applications are used for business purposes and are affected by the year 2000 problem.

The end-user aspect of the year 2000 problem is going to be difficult to deal with. Fortunately, end-user applications are almost always less than 100 function points in size and even large corporations seldom have more than a few hundred of them. This means that as a percentage of corporate portfolio's, end user applications will probably not amount to much more than 1% of the total volume of software affected.

Data Base

In addition to software applications, modern enterprises own enormous quantities of information with about half of it stored in computerized data base systems. In order to conserve expensive magnetic storage, many data bases utilize two-digit date recording fields and hence are subject to the year 2000 problem.

Not only must repairs be made to software, but it will also be necessary to make repairs to many computerized data bases. (Information stored on paper is more or less immune to the year 2000 problem.)

Software Staff

Large modern software development organizations can have more than 100 specialized occupations in their total software employment: programmers, analysts, quality assurance, data base administration, testing, technical writers, etc. (For additional information on software specialization, refer to Patterns of Software System Failure and Success, International Thomson Computer Press, 1995).

Since the year 2000 problem is more than just a programming problem, the data shown in this report is based on the total complement of technical workers found employed by software producers.

Of the professional software personnel employed by major corporations, the occupations that will be most heavily impacted by the year 2000 problem are those of:

- Attorneys
- Data center managers
- Chief information officers (CIO)
- Computer operations personnel
- Configuration control specialists
- Cost estimating specialists
- Customer support specialists
- End users who utilize software
- End users who program their own software
- Data base administrators
- Maintenance specialists
- Network managers
- Programmers
- Programmer/analysts
- Quality assurance personnel
- Systems Programmers
- Systems Administrators
- Systems analysts
- Testing specialists

Because end-user applications are going to be troublesome for the year 2000 problem, the author and his colleagues at SPR estimate that the total of professional software personnel in the United States is about 1,920,000 but the number of non-professional personnel who can write end-user applications is about 8,000,000. These "invisible" end-user applications are going to be very troublesome for the year 2000 problem even though they are small in both number and size.

Enterprise

The author prefers the general term “enterprise” to the more restricted terms “company” or “corporation” because many of the groups that are affected by the year 2000 problem are government agencies, non-profit associations, and even religious groups.

In the context of the year 2000 problem, an enterprise is an organization that is assumed to own software. Examples of enterprises can include corporations such as IBM, government agencies such as the Internal Revenue Service at the Federal Level or the State of Florida’s Bureau of Motor Vehicles. An enterprise can also be a military unit such as the Navy’s Surface Weapons centers, or even a religious organization such as the Genealogical Division of the Church of Latter Day Saints.

Since software often resides on computers at each site, there is an assumption that some repair work will be necessary at each site. Of course there are also networked applications on a single host that operate at multiple sites. This is one of the reasons why a careful inventory of portfolios is necessary.

One major technical challenge of the year 2000 problem is the fact that enterprises are now linked together electronically. For an example an automobile manufacturer may have direct computer communications with more than 500 subsidiaries and suppliers. Thus the year 2000 problem will extend horizontally across more than individual enterprises. Entire industries are going to be affected.

Software Sites

Most large enterprises such as corporations and government agencies are geographically dispersed. For example, during the time which the author worked for IBM there were 26 major IBM programming laboratories scattered around the world. During the time the author worked for ITT, that multi-national conglomerate had software development groups in no less than 50 cities.

In the context of the year 2000 problem, it can be assumed that repairs will take place at every site. Therefore it is necessary to make assumptions as to how many sites exist for typical enterprises.

It can be assumed that large corporations in the Fortune 500 class will typically average about 15 sites or locations where software is developed an maintained.

FUNCTION POINTS VERSUS LINES OF CODE METRICS FOR THE YEAR 2000 PROBLEM

The first step in exploring the economic consequences of the year 2000 problem is to construct an approximate inventory of the total volume of software installed and operational in the United States, and then utilize the U.S. data as a jumping off place for evaluating the hazards of other countries.

This is not an easy task and probably cannot be done with high precision. But by making some reasonable assumptions, it is possible to put together a picture of U.S. software that can serve as a jumping off place for exploring the magnitude of the year 2000 problem on a global or at least multi-national basis.

Although some research organizations such as the Gartner Group have attempted to enumerate the costs of the year 2000 problem using the "lines of code" or LOC metric, that metric is not accurate enough for serious economic analysis of software problems.

Function Points and Programming Languages

As of 1996, the Software Productivity Research catalog of programming languages (Table of Programming Languages and Levels, Version 8, SPR 1996) identifies almost 500 programming languages in current usage.

For some of the languages that are impacted by the year 2000 problem, there is not an accurate definition of what a "line of code" is in that language. For example, it is very difficult to enumerate lines of code for languages such as query-by-example (QBE), the control functions of Visual Basic, spreadsheets such as Lotus and Excel, data base languages, and a host of others.

Even for procedural languages such as COBOL and FORTRAN, there are wide variations in how lines of code are counted. A survey of software journals carried out by the author found that about one third of the software literature used physical lines as the basis for determining lines of code, one third used logical statements, and the remaining third did not identify which method was used.

Since the difference between the number of physical lines and logical statements in a COBOL program can amount to more than 300%, it can be seen that the ambiguity associated with using lines of code is far too great for serious economic study.

In this study, the well-known function point metric will be utilized. Function point metrics originated in the 1970's within IBM, and have become the most widely used metric in the software world. The International Function Point Users Group (IFPUG) is the largest software measurement association in the United States, and there are affiliated associations in some 20 countries.

The function point count of a software application is based on enumerating five external attributes of the application:

- Inputs
- Outputs
- Inquiries
- Logical files
- Interfaces

These five attributes are assigned various weighting factors, and there are also adjustments for complexity. The actual counting rules assumed in this study are based on the Version 4.0 counting practices manual published by International Function Point Users Group (IFPUG Counting Practices Manual, 1995).

Because some readers may be unfamiliar with the function point metric, it is useful to show the relationship between function points and lines of code for various languages, using the rules for logical statement counts defined in Applied Software Measurement (McGraw Hill, 1996).

Table 2 Ratios of Logical Source Code Statements to Function Points for Selected Programming Languages

Language	Nominal Level	Source Statements Per Function Point		
		Low	Mean	High
1st Generation	1.00	220	320	500
Basic assembly	1.00	200	320	450
Macro assembly	1.50	130	213	300
C	2.50	60	128	170
BASIC (interpreted)	2.50	70	128	165
2nd Generation	3.00	55	107	165
FORTRAN	3.00	75	107	160
ALGOL	3.00	68	107	165
COBOL	3.00	65	107	150
CMS2	3.00	70	107	135
JOVIAL	3.00	70	107	165
PASCAL	3.50	50	91	125
3rd Generation	4.00	45	80	125
PL/I	4.00	65	80	95
MODULA 2	4.00	70	80	90
ADA 83	4.50	60	71	80
LISP	5.00	25	64	80
FORTH	5.00	27	64	85

QUICK BASIC	5.50	38	58	90
C++	6.00	30	53	125
Ada 95	6.50	28	49	110
Data base	8.00	25	40	75
Visual Basic (Windows)	10.00	20	32	37
APL (default value)	10.00	10	32	45
SMALLTALK	15.00	15	21	40
Generators	20.00	10	16	20
Screen painters	20.00	8	16	30
SQL	27.00	7	12	15
Spreadsheets	50.00	3	6	9

Table 2 merely illustrates why “lines of code” metrics are difficult to apply to general software economic issues that span multiple programming languages. There are hundreds of programming languages and they vary in power over an enormous range. Moreover, for a significant number of modern programming languages such as the “visual” languages the concept of a line of code is not truly relevant.

The Volume of United States Software Expressed in Terms of Function Points

By interesting coincidence each reference to a calendar date in a software application seems to require approximately one function point to encode in quite a large variety of programming languages. This coincidence makes expressing the effort and costs in terms of work hours per function point, function points per staff month, and cost per function point comparatively straight-forward.

In terms of languages that are affected by the year 2000 problem, these languages are probably the top of the heap in year 2000 impact, listed in order of the numbers of applications in U.S. software portfolios as of 1996:

Table 3 Impact of the Year 2000 Problem for Selected Languages

Language	Programmers	Applications	Function Points
COBOL	550,000	12,100,000	605,000,000
Spreadsheets	600,000	3,600,000	54,000,000
C	200,000	2,600,000	156,000,000
Basic	250,000	2,250,000	45,000,000
Query	150,000	1,950,000	29,250,000
Data Base	200,000	1,600,000	120,000,000
C++	175,000	1,400,000	105,000,000
PASCAL	90,000	1,080,000	54,000,000
Assembly	50,000	750,000	93,750,000
Ada83	90,000	720,000	54,000,000
FORTRAN	50,000	575,000	28,750,000

PL/I	30,000	270,000	13,500,000
Jovial	15,000	105,000	7,875,000
Other	1,000,000	7,000,000	336,000,000
TOTAL	3,450,000	36,000,000	1,702,125,000

Note that table 3 reveals a hidden aspect of the year 2000 problem. There are only about 1,920,000 professional software personnel in the United States, but table 3 shows a total of 3,450,000 programmers. The reason that the total for programmers is so large is that it includes applications developed by non-professional programmers such as accountants, managers, and engineers.

Computer literacy is very common in the United States and end-user applications constitute a large but invisible component of the year 2000 problem. For software that was done by professional software personnel the software usually resides in a formal corporate portfolio, often under formal configuration control.

End-user applications, on the other hand, typically reside on the C drives of personal computers in someone's office and are under no configuration control at all. Other than the originator of the application, it may be that no one in the company even knows of the existence of the software. Yet some of these privately-developed end-user applications are used for important business purposes within the enterprise.

Support for finding and fixing the year 2000 problem will be much tougher for some languages than for others. Assembly language applications will probably be the toughest, because many date calculations are hard to find since they are performed using register manipulation.

Another language that may be difficult in year 2000 terms is PL/I. IBM was pushing the PL/I language very hard as a business tool in the early 1970's, and several industries such as oil and energy began to adopt PL/I. As of 1996, however, there is a shortage of both tools for analyzing PL/I programs and also a shortage of trained PL/I programmers.

However, as this report was written several vendors have announced expanded Year 2000 support that includes PL/I among other languages. This support is welcome, but of course still does not compensate for the lack of experienced PL/I programmers on a global basis.

Other programming languages that manifest the year 2000 problem but have a current shortage of available tools or trained programmers available include Algol, APL, Basic, CHILL, CMS2, CORAL, Forth, Lisp, Modula, MUMPS, PASCAL, Prolog, Ratfor, RPG, and the host of proprietary languages which companies have built for their own use such as ITT's ESPL/I and IBM's PLS.

The language that probably has the highest incidence of year 2000 calculations is COBOL since that is a very old language dating back to the 1960's and also widely used for business applications. However, COBOL has the most plentiful supply of year 2000 tools and services of any language. Indeed, from a scan of the advertisements in software journals, it appears that COBOL may have more year 2000 support than all other languages put together.

Mixed Language Applications and the Year 2000 Problem

One other aspect of the Year 2000 problem that has not yet received adequate attention is that of dealing with applications that contain multiple programming languages. Among SPR's data base of information covering about 7,000 software projects, roughly 2,000 of them contain more than one programming language. Mixed language applications are very common among all classes of software: systems, military, information systems, and commercial. Some of the more common programming language combinations include:

- Ada and CMS2
- Ada and Jovial
- Basic and assembly
- C and assembly
- C and C++
- C++ and assembly
- COBOL and data base languages
- COBOL and PL/I
- COBOL and RPG
- COBOL and SQL
- COBOL and SQL and data base languages

About 30% of U.S. software applications contain at least two languages. The maximum number of programming languages which we have observed in a single system is 12. It is hard enough to find and fix year 2000 problems for applications containing only one language, and those containing multiple languages will be even harder.

Although the data is preliminary, rough rules of thumb can be hypothesized for the additional effort of finding and fixing year 2000 problems in mixed language applications:

Table 4 Impact of Multiple Programming Languages on Year 2000 Repair Effort

Number of languages in Application	Percent Increase in Year 2000 Repair Efforts
1	0%
2	15%
3	20%

4	25%
5	30%
6	35%

Hopefully the vendors and service bureaus that are now gearing up to perform year 2000 repairs will be able to handle mixed-language applications for at least the more common combinations such as COBOL and SQL.

THE SIZE OF THE YEAR 2000 PROBLEM FOR THE UNITED STATES

So far as can be determined by the author, there has never been an accurate or even approximate inventory of the total volume of software deployed within any country. The data presented here is known to be imperfect and have a high margin of error. However, for studying problems such as the year 2000 issue, it is better to have partly speculative data with a high margin of error than no data at all. Hopefully future research can correct any errors and improve the accuracy of the results. But if the information were not published at all, there might be no incentive to carry out research that will eliminate any errors shown here.

The Volume of Software Installed in the United States

Using the assumptions for portfolios, sites, and software staffs discussed earlier, table 3 shows the approximate size of the installed software portfolios in the United States. The data in table 3 is derived from multiple sources, and can be assumed to have a large but unknown margin of error.

**Table 5 U.S. Software Applications, Sites, Staff, and Software Portfolios
(Portfolio Volumes Expressed in Function Points)**

Industries	Applications	Software Sites	Software Staff	Portfolio Size in FP
Military	6,000,000	1,000	200,000	300,000,000
Manufacturing	1,800,000	8,500	250,000	200,000,000
Finance	2,454,545	2,500	150,000	135,000,000
Services	2,222,222	1,500	125,000	100,000,000
Communications	1,800,000	1,000	100,000	90,000,000
Insurance	1,800,000	1,500	90,000	81,000,000
Defense	1,600,000	2,000	100,000	80,000,000
Wholesale	1,777,778	1,500	100,000	80,000,000
Federal	1,333,333	500	75,000	60,000,000
Retail	1,200,000	3,500	75,000	60,000,000
Software	1,050,000	1,000	75,000	52,500,000
Health care	510,000	1,500	30,000	25,500,000

Municipal	533,333	1,500	30,000	24,000,000
Energy	500,000	1,000	25,000	20,000,000
Transportation	416,667	1,000	25,000	18,750,000
States	355,556	150	20,000	16,000,000
Other	8,002,668	15,000	450,000	360,000,000
TOTAL	36,000,546	44,650	1,920,000	1,702,750,000

Regardless of the margin of error in table 5, one fact is painfully obvious. The United States is the world's largest producer and largest consumer of software so the costs of the year 2000 problem will be greater in the United States than in any other country. The year 2000 problem is one of those comparatively rare problems that affects industrialized and computerized nations much more severely than those that are not yet fully automated.

This same statement is true of industries and governments as well as countries. Industries such as banking and insurance that are highly automated will have much higher costs associated with the year 2000 problem than less automated groups such as publishing.

Also, the potential liabilities associated with litigation also seem to correlate with the volumes of software used by an industry. However, the litigation potential is also affected by the probability that not fixing the year 2000 problem will cause economic damage.

In considering the impact of the year 2000 problem, it is obvious that the expenses will be large primarily because the problem is found throughout the entire portfolio of legacy applications. Changing any single application might not be too difficult, but when *every* application, or at least a high percentage, of applications owned by a large company is affected, the cumulative effort will be enormous.

Table 6 provides rough approximations of the percentage of portfolios that are likely to be modified, and the anticipated productivity rates for finding and repairing the year 2000 problem:

**Table 6 Productivity Assumptions for Year 2000 Repairs
(Productivity Rates Expressed in Function Points per Person Month)**

Industries	Portfolio Size in FP	Year 2000 Impact	Portfolio Changes	Productivity FP/PM	Effort PM
Military	300,000,000	7.00%	21,000,000	11.00	1,909,091
Finance	135,000,000	6.00%	8,100,000	18.00	450,000
Manufacturing	200,000,000	5.00%	10,000,000	18.00	555,556
Communications	90,000,000	8.00%	7,200,000	17.00	423,529
Services	100,000,000	10.00%	10,000,000	18.00	555,556
Insurance	81,000,000	10.00%	8,100,000	18.00	450,000
Wholesale	80,000,000	11.00%	8,800,000	17.00	517,647
Federal	60,000,000	10.00%	6,000,000	15.00	400,000
Defense	80,000,000	4.00%	3,200,000	12.00	266,667
Retail	60,000,000	11.00%	6,600,000	16.00	412,500

Software	52,500,000	7.00%	3,675,000	19.00	193,421
Municipal	24,000,000	10.00%	2,400,000	16.00	150,000
Health care	25,500,000	7.00%	1,785,000	16.00	111,563
States	16,000,000	10.00%	1,600,000	16.00	100,000
Energy	20,000,000	7.00%	1,400,000	16.00	87,500
Transportation	18,750,000	7.00%	1,312,500	16.00	82,031
Other	360,000,000	8.00%	28,800,000	16.00	1,800,000
TOTAL	1,702,750,000	8.12%	138,223,235	16.18	8,465,060

The technical work associated with finding and fixing the year 2000 problem can be broken down into four discrete activities:

- 1) Finding and isolating the year 2000 sections of applications
- 2) Modifying the applications to repair the problem
- 3) Testing the repairs to ensure that they work
- 4) Regression testing the application to ensure no secondary damage has occurred

Table 6 assumes that all four of these activities will be performed. However, from preliminary observations of companies that have already begun their year 2000 work, steps 3 and 4 (testing and regression testing) are sometimes performed in a very careless fashion. Carelessness in regression testing and validating Year 2000 repairs will have three damaging impacts later that can run well into the 21st century:

- Missed Year 2000 instances will be plentiful and troublesome.
- Bad fixes or fresh bugs accidentally injected will be common and troublesome.
- The performance or execution speeds of applications will be seriously degraded.

The approximate distribution of effort over the four aspects of the year 2000 problem will vary significantly by language, due to the presence or absence of available tools:

Finding and isolating the year 2000 problem should be easiest for object-oriented languages (i.e. Objective C, Smalltalk, etc.) where dates are handled in well-formed class libraries. Next would be COBOL, since there are several specialized tools that can seek out date references in COBOL applications. Such tools also exist for other common languages such as C and FORTRAN. The toughest language will probably be assembly, followed by languages that have tool shortages such as PL/I, LISP, FORTH and the like.

Year 2000 Activities

Percent Range of Total Costs

- Finding the year 2000 instances = 10% to 50%
- Fixing the year 2000 instances = 15% to 30%
- Testing the year 2000 repairs = 10% to 30%
- Regression testing the portfolio = 20% to 50%

The next aspect of the study of the year 2000 problem is to assign approximate costs for the overall repairs that must be performed. Here there is quite a bit of uncertainty, since

there are large variances in cost structure by industry and by geography. For example, the costs of updating financial applications in New York City or San Francisco will be much greater than the costs of updating retail applications in the rural South because urban areas have higher compensation rates than rural and because financial institutions have higher compensation rates than retail.

Table 7 shows the approximate United States costs for finding and fixing the year 2000 problem:

Table 7 United States Repair Costs for the Year 2000 Problem

Industries	Effort PM	Burdened \$/PM	Costs	Costs per Changed FP	Costs per Total FP
Military	1,909,091	\$7,500	\$14,318,181,818	\$682	\$48
Finance	450,000	\$11,000	\$4,950,000,000	\$611	\$37
Manufacturing	555,556	\$8,400	\$4,666,666,667	\$467	\$23
Communications	423,529	\$10,000	\$4,235,294,118	\$588	\$47
Services	555,556	\$8,000	\$4,444,444,444	\$444	\$44
Insurance	450,000	\$9,200	\$4,140,000,000	\$511	\$51
Wholesale	517,647	\$7,500	\$3,882,352,941	\$441	\$49
Federal	400,000	\$7,900	\$3,160,000,000	\$527	\$53
Defense	266,667	\$11,000	\$2,933,333,333	\$917	\$37
Retail	412,500	\$7,500	\$3,093,750,000	\$469	\$52
Software	193,421	\$9,000	\$1,740,789,474	\$474	\$33
Municipal	150,000	\$7,000	\$1,050,000,000	\$438	\$44
Health care	111,563	\$8,000	\$892,500,000	\$500	\$35
States	100,000	\$7,700	\$770,000,000	\$481	\$48
Energy	87,500	\$8,000	\$700,000,000	\$500	\$35
Transportation	82,031	\$8,000	\$656,250,000	\$500	\$35
Other	1,800,000	\$8,400	\$15,120,000,000	\$525	\$42
TOTAL	8,465,060	\$8,476	\$70,753,562,795	\$512	\$42

The platform on which the application resides will also affect the costs. Common platforms such as IBM mainframes and AS400 machines will be comparatively straightforward, as will standard IBM compatible personal computers. UNIX, OS/2, DOS, Macintosh, and various Windows versions will need repairs, but all are marketed by vendors that are actively moving toward year 2000 compliance. Tougher will be software that runs on "orphan" platforms or on specialized computers such as military on-board ANYUK computers.

Hardware and Performance Implications of the Year 2000 Problem

Fixing the year 2000 problem in software will have some possible implications for the performance of the applications, which may necessitate hardware upgrades to more powerful computers.

The performance and hardware implications of the Year 2000 problem are under reported in the literature and at conferences. Many mainframe software applications have been optimized to reduce machine utilization and maximize throughput. Any Year 2000 repairs that are made in a hasty or perfunctory manner without adequate testing and re-optimization can result in several major problems:

- Introduction of bad fixes or new bugs as a byproduct of Year 2000 repairs.
- Major degradation of application throughput and data center efficiency levels.

The author estimates that sloppy Year 2000 repairs are likely to affect mainframe data centers the most, since the bulk of aging U.S. legacy applications were originally built for IBM mainframe computers.

It is difficult to estimate precisely, but I suspect that the minimum degradation from the Year 2000 problem will be a de facto 10% loss in data center throughput. This is the most conservative estimate. The most probable estimate is about 20% or higher. The worst-case scenario might cause a 30% slowdown in data center efficiency and throughput, which could trigger a set of catastrophic secondary problems in things like delayed billings, slow processing of tax returns, and a host of other speed-related issues.

An independent model of performance degradation in response to sloppy Year 2000 repairs has also been reported by Dr. Howard Rubin. His model reaches a similar conclusion, and places the performance loss at something over 20%.

The normal response to a major degradation of throughput is either expensive optimization of software applications, major increases in hardware capacity, or both. I would predict that performance tuning of applications degraded by careless Year 2000 testing might add \$5,000,000 to the U.S. Year 2000 costs, while hardware upgrades to service the demand could add \$10,000,000 to \$20,000,000 in the years 1999 through about 2005 to U.S. Year 2000 costs.

Note that hardware and data center costs are not measured using function points, so these costs are over and above the software costs discussed later in this report.

In round numbers, preliminary analysis suggests that about \$40 will be expended for every function point in existing portfolios. Since most corporations own more than 100,000 function points and major corporations own more than 1,000,000 function points, the cumulative costs will be major indeed.

Year 2000 Repairs by Industry

Another interesting aspect of the year 2000 problem is to consider how much effort must be expended between now and the year 2000 by the software community? It appears that the problem is large enough so that essentially every technical worker in the software domain will become engaged for more than four months between now and 1999! If this

hypothesis is true, then the year 2000 problem may well be one of the largest and most expensive technical problems in all of human history.

Table 8 shows the approximate effort in terms of expended person months and costs between now and the end of the century on a per capita basis. That is, the effort and costs are expressed in terms of the impact on every software technical worker in the United States, ranked in descending order:

Table 8 Per Capita Effort and Costs for the Year 2000 Problem

Industries	Months per Staff Member	Costs per Capita
Military	9.55	\$71,591
Insurance	5.00	\$46,000
Communications	4.24	\$42,353
Federal	5.33	\$42,133
Retail	5.50	\$41,250
Wholesale	5.18	\$38,824
States	5.00	\$38,500
Services	4.44	\$35,556
Municipal	5.00	\$35,000
Other	4.00	\$33,600
Finance	3.00	\$33,000
Health care	3.72	\$29,750
Defense	2.67	\$29,333
Energy	3.50	\$28,000
Transportation	3.28	\$26,250
Software	2.58	\$23,211
Manufacturing	2.22	\$18,667
TOTAL	4.41	\$36,851

In general, the human race is not very good in preventing disasters and tends to wait until the last minute (or beyond) before taking action. That tendency is already visible in the context of the year 2000 problem.

The time to start fixing this problem was 1995. It is now 1996 as this report is written, and time is already beginning to run short. The year 1997 is the last year that mid-sized corporation can commence their year 2000 repairs with any hope of finishing before the end of the century even with an automated year 2000 search engine, as can be seen in table 9:

Table 9 Percentage of Year 2000 Repairs Completed Based on Start Year and Use of Manual or Automated Search Procedures

Year When 2000 Repairs Start	Percent of Applications Corrected by 1999 (Manual Search)	Percent of Applications Corrected by 1999 (Automated Search)
1994	100%	100%
1995	100%	100%
1996	99%	100%
1997	80%	99%
1998	70%	85%
1999	30%	60%

It should be noted that the year 2000 problem is one where automated search engines are going to be highly beneficial. However, the accuracy of these search engines is still something of an unknown value.

Right now no one is sure if automated year 2000 search engines will find 100% of the year 2000 instances. On the other hand, no one is sure if manual searches will find 100% of the year 2000 instances either. (There is also an opposite problem, or reporting false year 2000 instances but this is less serious than missing real year 2000 hits.)

An interesting question as this report is written in late 1996 is, "What percent of Year 2000 repairs have already been accomplished, and what percent remain to be done?" This is a hard question to answer.

Unfortunately, the great majority of companies and government groups are still in preliminary fact-finding mode and have not yet begun the tougher work of actually seeking out the year 2000 hits and making corrections. The result of this lag time in getting started means that a significant number of companies will still be working on year 2000 repairs when the clock runs out at midnight on December 31, 1999.

Further, repairs made during calendar year 1999 and especially during the last half of 1999 are probably going to be rushed and careless, so there will be a high probability of missed year 2000 hits, performance degradation, bad fixes, and the other problems associated with excessive haste and poor quality control.

From queries and surveys among our clients, the following expenditure pattern seems to be a rough approximation:

Table 10 Trail of Year 2000 Expenses for Software, Hardware, and Litigation From 1994 Through 2005 AD

Year When 2000 Repairs Start	Software Expense Percent	Hardware Expense Percent	Litigation Expense Percent
1994	1%	0%	0%
1995	3%	0%	0%
1996	10%	1%	1%
1997	15%	1%	2%
1998	20%	5%	5%
1999	<u>30%</u>	20%	10%
2000	15%	<u>25%</u>	20%
2001	4%	20%	<u>30%</u>
2002	2%	15%	15%
2003	0%	10%	10%
2004	0%	2%	5%
2005	0%	1%	2%

If the Year 2000 problem follows the pattern just illustrated, software repairs will peak during calendar year 1999. However, hardware upgrades and legal expenses will not peak until 2000 and 2001, respectively. In other words, the Year 2000 problem will leave a trail of major expenses until well into the 2¹century.

Year 2000 Repairs by Company Size

To chief information officers (CIO's), chief executives (CEO's) and other managers concerned with enterprise software, the most important topic is not what the problem is going to cost the country or their industry, but rather what the problem is going to cost their particular enterprise.

Unfortunately, it would require an on-site study of each enterprise to quantify the exact costs. Indeed, as this report is written many enterprises have already commissioned site studies to begin to quantify the expenses of the year 2000 problem.

However, by using some of the general assumptions shown earlier, it is possible to state the approximate overall costs of fixing the year 2000 problem for enterprises of various sizes. Table 11 is based on the total number of technical software staff employed, and shows approximate costs for enterprises with as few as five technical personnel, up to enterprises with as many as 10,000 technical personnel.

Some of the background assumptions going into table 8 include:

- A generic burdened cost per staff month of \$8400 is assumed.
- Large enterprises are geographically dispersed, and this raises repair costs.
- Large enterprises have more large systems, and they are harder to fix.

Table 11 has a significant margin of error and is not a substitute for a thorough on-site analysis of an enterprise's actual portfolio.

Table 11 Effort and Costs of the Year 2000 Problem by Size of Software Staff

Software Staff	Number of Sites	Portfolio Size in FP	Effort in Months	Total Costs	Cost per FP
5	1	6,000	23	\$197,784	\$33
10	1	11,500	45	\$379,087	\$33
25	1	27,500	107	\$906,511	\$33
50	1	50,000	194	\$1,648,203	\$33
100	1	95,000	416	\$3,523,033	\$37
500	2	450,000	1,969	\$16,688,051	\$37
1000	3	900,000	4,200	\$35,601,176	\$40
5000	5	4,500,000	21,000	\$178,005,882	\$40
10000	10	9,000,000	42,000	\$356,011,765	\$40
20000	15	18,000,000	84,000	\$712,023,529	\$40

In order to refine the information shown in table 11, readers are urged to carry out the following six-step analysis either on their own or aided by consultants who are specializing in the year 2000 problem:

- 1) Quantify the size of your portfolio of legacy applications.
- 2) Quantify the size of your software data bases and repositories.
- 3) Explore the incidence of year 2000 references in your legacy applications.
- 4) Explore the incidence of year 2000 references in your current data bases.
- 5) Estimate the effort to repair each year 2000 reference.
- 6) Estimate the effort to test and validate each year 2000 reference.

It should be obvious but unfortunately it is not, that a seventh step must also be included:

- 7) Stop using two-digit date fields in your new applications!

As stated earlier, by interesting coincidence each reference to a calendar date in a software application seems to require approximately one function point to encode in quite a large variety of programming languages. This coincidence makes expressing the effort and costs in terms of work hours per function point, function points per staff month, and cost per function point comparatively straight-forward.

Year 2000 Repairs by Programming Language

It is also interesting to consider the approximate costs of the year 2000 problem for selected programming languages. Table 12 gives the cost per language, although with a high margin of error.

Although COBOL is the language with the greatest number of year 2000 “hits” it will probably be among the least expensive to modify due to the large numbers of specialized tools and consulting groups in the COBOL domain.

Dr. Tom Love of the Worldstreet Journal software company and a well-known expert on OO topics reports that object-oriented languages such as Objective C, Smalltalk, Eiffel, etc. should be among the *least* expensive if the applications are well formed and the date calculations are handled by formal class libraries. Indeed, since object-oriented business applications are comparatively recent, many may already use adequate space for all date digits and hence the year 2000 problem may not be present in some OO applications.

On the other hand, the OO paradigm has a steep learning curve and there may also be OO applications with incorrect date calculations “hard coded” into the application just as they would be in procedural languages.

The most expensive languages will probably be assembly language and PL/I, both of which have shortages of tools and trained personnel. Table 12 shows the approximate overall expenses by language:

Table 12 Approximate Costs for the Year 2000 Problem By Language

Language	Function Points	\$ per FP	Total Cost
COBOL	605,000,000	\$28	\$16,940,000,000
Spreadsheets	54,000,000	\$35	\$1,890,000,000
C	156,000,000	\$35	\$5,460,000,000
V-Basic	45,000,000	\$30	\$1,350,000,000
Query	29,250,000	\$40	\$1,170,000,000
Data Base	120,000,000	\$45	\$5,400,000,000
C++	105,000,000	\$35	\$3,675,000,000
PASCAL	54,000,000	\$40	\$2,160,000,000
Assembly	93,750,000	\$80	\$7,500,000,000
Ada83	54,000,000	\$35	\$1,890,000,000
FORTRAN	28,750,000	\$35	\$1,006,250,000
PL/I	13,500,000	\$65	\$877,500,000
Jovial	7,875,000	\$60	\$472,500,000
Other	336,000,000	\$60	\$20,160,000,000
TOTAL	1,702,125,000	\$45	\$69,951,250,000

Although this data has a high margin of error, it appears that the repair costs for the year 2000 problem may be one of the largest single technology expenses in human history.

It is of interest to consider the relative proportion of expense among the four major activities associated with the year 2000 problem for selected programming languages:

**Table 13: Distribution of Year 2000 Expense by Programming Language
(Results shown in terms of Cost per Function Point)**

Language	Finding Year 2000 Instances	Repairing Year 2000 Instances	Testing Year 2000 Instances	Portfolio Regression Test	TOTAL
Assembly	\$25	\$20	\$20	\$15	\$80
PL/I	\$25	\$10	\$15	\$15	\$65
Jovial	\$20	\$12	\$13	\$15	\$60
CMS2	\$20	\$12	\$13	\$15	\$60
CHILL	\$20	\$15	\$15	\$10	\$60
Algol	\$15	\$12	\$13	\$10	\$50
Data Base	\$7	\$13	\$15	\$10	\$45
Query	\$10	\$12	\$12	\$6	\$40
PASCAL	\$5	\$10	\$15	\$10	\$40
C++	\$8	\$6	\$13	\$8	\$35
Spreadsheets	\$5	\$7	\$20	\$3	\$35
C	\$15	\$7	\$5	\$8	\$35
Ada83	\$5	\$7	\$8	\$15	\$35
FORTRAN	\$5	\$7	\$13	\$10	\$35
4-GLs	\$5	\$10	\$10	\$10	\$35
V-Basic	\$5	\$5	\$15	\$5	\$30
COBOL	\$5	\$6	\$7	\$10	\$28
SMALLTALK	\$3	\$3	\$5	\$7	\$18
Average	\$11	\$10	\$13	\$10	\$44

Note that table 13 includes CHILL, CMS2, Jovial, and Algol which are not in Table 12. Table 13 is only approximate and has a high margin of error. However, it is obvious that there will be significant differences from language to language based on the presence or absence of available year 2000 tools, and on the structure of the language itself.

Year 2000 Software Repairs By State

For problems as large and complex as the year 2000 problem, showing the approximate effort and cost at the national level results in numbers that are so large that the human mind has trouble grasping their significance.

Table 14 shows the approximate effort and costs for year 2000 repairs for all 50 states of the United States plus the District of Columbia, in descending order:

Table 14 Year 2000 Software Repairs by State for the United States

States	Software Personnel	Year 2000 Effort in Months	Burdened Monthly Salary	Year 2000 Costs Per State
California	214,000	1,061,440	\$9,000	\$9,552,960,000
New York	138,000	670,680	\$9,000	\$6,036,120,000
Texas	130,000	598,000	\$8,600	\$5,142,800,000
Illinois	98,000	450,800	\$8,300	\$3,741,640,000
Florida	92,000	447,120	\$8,200	\$3,666,384,000
Pennsylvania	94,000	432,400	\$8,300	\$3,588,920,000
Ohio	86,000	395,600	\$8,300	\$3,283,480,000
Michigan	70,000	322,000	\$8,300	\$2,672,600,000
New Jersey	60,000	291,600	\$9,000	\$2,624,400,000
Massachusetts	49,000	238,140	\$9,000	\$2,143,260,000
North Carolina	52,000	239,200	\$8,400	\$2,009,280,000
Virginia	46,000	218,960	\$8,400	\$1,839,264,000
Georgia	48,500	218,250	\$8,300	\$1,811,475,000
District of Columbia	41,500	205,840	\$8,200	\$1,687,888,000
Maryland	36,000	174,960	\$8,400	\$1,469,664,000
Indiana	40,000	176,000	\$8,300	\$1,460,800,000
Washington	35,000	170,100	\$8,500	\$1,445,850,000
Wisconsin	36,000	165,600	\$8,300	\$1,374,480,000
Tennessee	37,000	165,020	\$8,300	\$1,369,666,000
Minnesota	33,300	153,180	\$8,300	\$1,271,394,000
Missouri	34,500	150,420	\$8,300	\$1,248,486,000
Alabama	32,000	147,200	\$8,200	\$1,207,040,000
Louisiana	33,000	143,880	\$8,200	\$1,179,816,000
Connecticut	26,500	128,790	\$8,500	\$1,094,715,000
Kentucky	28,400	126,664	\$8,200	\$1,038,644,800
Oregon	24,000	116,640	\$8,700	\$1,014,768,000
Colorado	25,900	119,140	\$8,500	\$1,012,690,000
Arizona	26,500	121,900	\$8,300	\$1,011,770,000
South Carolina	24,000	110,400	\$8,300	\$916,320,000
Oklahoma	23,000	100,280	\$8,300	\$832,324,000
Iowa	20,500	89,380	\$8,300	\$741,854,000
Mississippi	19,500	85,020	\$8,200	\$697,164,000
Kansas	18,300	79,788	\$8,300	\$662,240,400
Arkansas	18,200	79,352	\$8,200	\$650,686,400
New Hampshire	12,500	57,500	\$8,400	\$483,000,000
West Virginia	12,750	55,590	\$8,200	\$455,838,000
Utah	11,500	52,900	\$8,300	\$439,070,000
Nebraska	11,500	50,140	\$8,300	\$416,162,000
New Mexico	11,000	47,960	\$8,300	\$398,068,000
Hawaii	8,600	37,496	\$9,000	\$337,464,000
Maine	8,700	37,932	\$8,400	\$318,628,800

Rhode Island	7,500	33,450	\$8,500	\$284,325,000
Idaho	7,500	32,700	\$8,300	\$271,410,000
Montana	5,750	25,070	\$8,300	\$208,081,000
Nevada	5,750	25,070	\$8,300	\$208,081,000
Delaware	5,150	23,690	\$8,300	\$196,627,000
South Dakota	5,000	21,800	\$8,200	\$178,760,000
Alaska	4,500	19,620	\$8,700	\$170,694,000
North Dakota	4,000	19,440	\$8,200	\$159,408,000
Vermont	4,000	18,400	\$8,400	\$154,560,000
Wyoming	3,700	16,280	\$8,400	\$136,752,000
United States Total	1,920,000	8,968,782	\$8,400	\$75,337,768,800

Note that table 14 uses simple ratios for dealing with compensation, so that the totals are slightly different from other tabular data in this report. Although table 14 has a high margin of error, it can be hypothesized that the software-intensive industries in the top 10 states are likely to be severely impacted by the work associated with the year 2000 problem. At the very least, there will be significant delays in starting new software projects because the year 2000 problem is likely to absorb almost all of the available software personnel for the next several years.

REPAIRING DATA BASES, REPOSITORIES, AND DATA WAREHOUSES

Software costs and economic studies can be expressed by means of function point metrics. Unfortunately, there is no equivalent metric for dealing with the volume of information stored in data bases, repositories, and data warehouses. In other words, the industry lacks a "data point" metric. As a result, there are no published statistics on the volumes of data and information owned by corporations and government agencies. Also as a result, there is no easy way to perform an economic analysis of the data impact of the year 2000 problem.

Thus as this report is written, the impact of the year 2000 problem on data bases, repositories, and data warehouses is still uncertain. However, preliminary indications of relative costs lead to the following hypothesis:

For every dollar spent on changing software applications, it will probably be necessary to spend another dollar on changing data bases. However for data-intensive industries such as insurance and finance, and for data-intensive government agencies such as Social Security, the data-repair costs will perhaps be twice those of fixing the software itself!

Due to the lack of data metrics, the volume of on-line information can only be approximated. Table 11 shows the approximate amount of data maintained by large corporations such as IBM and AT&T with perhaps 250,000 total employees:

Table 15 Relative Volumes of Stored Information in Major Corporations

Kind of Information	Pages Stored	Percent Stored On-Line	Year 2000 Impact?
Customer information	90,000,000	50%	Yes
Product information	50,000,000	50%	Yes
Software applications	40,000,000	75%	Yes
Email messages	30,000,000	95%	No
Reference information	15,000,000	20%	No
Personnel information	12,500,000	50%	Yes
Graphics/images	10,000,000	50%	No
Correspondence	5,000,000	10%	No
Defect information	2,500,000	50%	Yes
Supplier information	1,500,000	25%	Yes
Tutorial/training material	1,000,000	50%	No
Litigation/legal information	1,000,000	25%	Yes
<i>Total Volume</i>	<u>272,000,000</u>		

Since the example shown here is a corporation stated to have 250,000 employees, it is interesting to note that the total volume of corporate information stored by the case study corporation amounts to more than 1,000 pages per employee. How much of this information will require year 2000 changes is an important but currently unanswered question.

Table 16 shows the hypothetical costs of data repairs with a very large but unknown margin of error:

Table 16 Hypothetical Data Base Repair Costs for Year 2000

Industries	Software Cost	Data Base Cost	Total Cost
Military	\$13,363,636,364	\$12,027,272,727	\$25,390,909,091
Finance	\$4,950,000,000	\$5,445,000,000	\$10,395,000,000
Manufacturing	\$4,444,444,444	\$3,111,111,111	\$7,555,555,556
Communications	\$4,235,294,118	\$2,964,705,882	\$7,200,000,000
Services	\$4,166,666,667	\$3,333,333,333	\$7,500,000,000
Insurance	\$4,050,000,000	\$5,062,500,000	\$9,112,500,000
Wholesale	\$3,882,352,941	\$2,911,764,706	\$6,794,117,647
Federal	\$3,400,000,000	\$3,060,000,000	\$6,460,000,000
Defense	\$2,933,333,333	\$2,640,000,000	\$5,573,333,333
Retail	\$3,093,750,000	\$2,010,937,500	\$5,104,687,500

Software	\$1,740,789,474	\$1,392,631,579	\$3,133,421,053
Municipal	\$1,020,000,000	\$765,000,000	\$1,785,000,000
Health care	\$892,500,000	\$624,750,000	\$1,517,250,000
States	\$770,000,000	\$616,000,000	\$1,386,000,000
Energy	\$700,000,000	\$525,000,000	\$1,225,000,000
Transport.	\$656,250,000	\$459,375,000	\$1,115,625,000
Other	\$14,400,000,000	\$10,800,000,000	\$25,200,000,000
TOTAL	\$68,699,017,341	\$57,749,381,839	\$126,448,399,179

Unfortunately, because data base repair productivity rates are essentially a topic with no citations in the literature, the information presented here for the costs of data repairs is largely speculative.

LITIGATION POTENTIAL FOR THE YEAR 2000 PROBLEM

The costs of repairing the year 2000 problem can be quantified with acceptable precision for software itself, and guessed at for data base repairs. The last and most alarming component of the year 2000 problem are the potential expenses for litigation and possible damages for *not* fixing the year 2000 problem.

As stated earlier, six kinds of litigation can be envisioned in the context of the year 2000 problems:

- 1) Litigation filed by clients whose finances or investments have been damaged.
- 2) Litigation filed by shareholders of companies whose software does not safely make the year 2000 transition.
- 3) Litigation associated with any deaths or injuries derived from the year 2000 problem.
- 4) Class-action litigation filed by various affected customers of computers or software packages.
- 5) Litigation filed by companies who utilized outsource vendors, contractors, consultants, or commercial Year 2000 tools but where Year 2000 problems still slipped through and caused damage.
- 6) Litigation against hardware manufacturers such as computer companies and defense contractors if the Year 2000 problem resides in hardware or embedded microcode as well as software.

Potential litigation is not easy to predict. Further, when litigation does occur the outcome is not easy to predict. However the United States is a very litigious country, and damage awards are often set at astronomical values. It is possible that the litigation expenses (and

any damages if suits are lost) for the year 2000 problem can exceed the direct costs of repairs by as much as 20 to 1 in cases where negligence and violation of fiduciary duty are proven or at least confirmed by jury decisions.

Table 17 shows the litigation potential for the industries discussed thus far for four of the six kinds of litigation. Suits against vendors are subsumed under “client suits” rather than being shown separately. Suits against hardware vendors would be distributed across the other categories.

Table 17: Litigation Potential for Failure to Repair the Year 2000 Problem

Industry	Client Suits	Shareholder Suits	Injury Suits	Class Action Suits
Military	Low	Low	Very High	Low
Finance	High	High	Low	High
Manufacturing	High	High	High	High
Communications	High	High	Low	High
Services	High	High	Low	High
Insurance	High	High	Low	High
Wholesale	High	High	Low	Low
Federal	High	None	High	High
Defense	High	High	High	High
Retail	High	High	Low	High
Software	High	High	Low	High
Municipal	High	None	Low	High
Health Care	High	High	High	High
States	High	None	High	High
Energy	High	High	Low	High
Transportation	High	High	Very High	High

The senior executives of corporations have what is called a fiduciary responsibility to act in the best interests of the shareholders of the companies they serve. Failure to take action to repair the year 2000 problem has at least the potential to damage or even end the careers of about half of the senior executives in the United States.

Attorneys fees and damage awards for the Year 2000 problem are difficult to estimate using any kind of historical data, since a problem of this magnitude has not occurred before. For the U.S. as a whole, I suspect that legal fees associated with Year 2000 lawsuits will come to close to \$2,000,000,000 between about 1997 and 2005, which is the window of major Year 2000 litigation.

Damages and punitive damages are even harder to assess, but possibly \$100,000,000,000 is a likely number for the United States as a whole.

Expressed another way, a major bank, insurance company, or Fortune 500 company in general might expect to pay about \$750,000 a year in Year 2000 legal fees between 1997 and 2005.

Since major companies are very likely to end up suing each other, the potential of damages paid out might be offset by damages payments that come in. However, it would be surprising if less than \$100,000,000 per company ends up being paid out for Year 2000 damages among the Fortune 500 class of enterprises.

RISK OF BUSINESS FAILURE DUE TO THE YEAR 2000 PROBLEM

One of the most serious potential problems associated with the year 2000 crisis is that some companies may go bankrupt or fail either as a direct result of the year 2000 problem, or as a possible defensive measure to stave off massive damages due to year 2000 litigation.

There is insufficient data to predict the probable number of business failures that might be attributed to the year 2000 problem with any degree of accuracy, but from observing the recent history of business failures for other causes it is possible to form some preliminary hypotheses.

There are four ways of examining the business failure potential associated with the year 2000 crisis:

- 1) Failure potentials based on the size of the company.
- 2) Failure potentials based on the industry in which the company resides.
- 3) Failure potentials based on the financial health of the company.
- 4) Failure potentials based on probable year 2000 litigation against the company.

Following are some preliminary observations on failure potentials based on these four criteria.

Business Failure Potentials Based on Company Size

In evaluating the potential for business failure based on company size, the preliminary conclusion is that mid-sized corporations with from 1000 to 10,000 total employees are probably at greater risk than either larger or smaller enterprises.

Very large companies in the Fortune 500 class will be heavily impacted by year 2000 repairs, but many are already engaged in making those repairs and most have adequate financial resources to complete the task either on their own or with the assistance of specialized year 2000 tool and service vendors.

My estimate is that the chance of a business failure among the Fortune 500 class is only about 1%, unless some of them declare bankruptcy as an emergency measure to avoid damages due to litigation.

Very small companies with less than 100 total employees will be impacted by the year 2000 problem, and sometimes severely, but these companies usually do not own very much software so they can probably deal with the situation. My estimate is that the chance of failure for small companies as a direct impact of the year 2000 problem is about 3%, unless they are a direct target of year 2000 litigation for some reason. (Of course small companies fail all the time for a variety of reasons, so my 3% estimate is a delta on top of the already notable failure rate of small enterprises that exceeds 50% in the first two years after incorporation.)

Mid-sized corporations with from about 1,000 to 10,000 total employees have historically shown a distressing tendency to utilize quite a lot of software, but to be only marginally competent in how they build and maintain this software.

In a year 2000 context, mid-sized corporations will probably be late in getting started on their year 2000 repairs, will under estimate and under budget for their year 2000 work, will not bring in the appropriate tools and specialists, and will probably not have any contingency plans in place on what to do with applications that don't make the changes in time. I place the failure probability of mid-sized U.S. corporations at about 5% to 7%.

There are about 30,000 companies in the "mid sized" range in the United States, and a 5% to 7% business failure rate would mean that from 1500 to perhaps 2100 companies might close or file for bankruptcy as a result of the year 2000 problem. This is a significant number and it is an open question as to whether the impact of the year 2000 problem is severe enough to trigger a recession.

Business Failure Potentials Based on Industry

In considering the probability of year 2000 problems causing failures by industry, the industries that are most likely to be affected are those that utilize software for key business operations: banks, brokers, credit unions, health care, insurance, manufacturing, retail, wholesale. All of these probably have at least a 2% chance of going out of business or declaring bankruptcy to stave off damages and litigation.

Somewhat more ominous are the possibilities that industries not always recognized as software intensive will fail due to the year 2000 problem: city governments, county and provincial governments, public utilities, and telephone companies may also fail.

Perhaps the most hazardous of any of these is not an industry at all, but the affect of the year 2000 problem on state, provincial, country, and city government operations may well cause a rash of bankrupt government organizations or at least drastic reductions in the services they provide.

In the United States many local government agencies use computers and software for a variety of revenue and disbursement purposes. Unfortunately governments as a class are often not very sophisticated about building or maintaining their software. They are typically under funded, and many can't even afford to bring in year 2000 consultants.

The probable result will be a rash of lawsuits against a wide variety of government organizations. In any case, the impact of the year 2000 problem will no doubt reduce many government services because the money to fix the year 2000 problem has got to come from somewhere.

Public utilities for water and electricity, and all telephone companies, utilize software for both technical and administrative purposes. Some of these organizations are large and sophisticated in how they build software, but others have been somewhat careless. The year 2000 problem may cause some of these utility companies to fail, and in any case will probably raise the costs of their services to consumers.

Business Failure Potentials Based on Financial Health

The year 2000 problem is going to be very expensive no matter how the repairs are accomplished. This obvious fact means that companies whose cash flow and finances are already marginal will have a significant probability of failing under the added expenses of the year 2000 problem. Since perhaps 10% of small to mid-sized companies in the United States are already in some kind of financial distress, the added burden of the year 2000 problem may put many of them out of business.

What is not so obvious is that the venture capital community will be seriously impacted by the year 2000 problem. Right now venture capital is pouring into new year 2000 start-ups, and into many other kinds of software start ups as well.

The venture capital community should have included year 2000 compliance as part of the due diligence process starting in about 1994, but hardly any venture group even thought about the problem. When the costs of year 2000 repairs to venture-backed companies is factored in to business plans, the anticipated 10 to 1 yield which VC's expect will shrink to nothing. This means that 2nd, 3rd, or additional rounds of financing may evaporate.

Business Failure Potentials Based on Litigation

Readers of John Gresham's book The Rainmaker are aware of how a lawsuit against an insurance company triggered a deliberate bankruptcy filing in order to stave off having to pay damages.

The year 2000 problem will obviously cause a lot of litigation, and hence many organizations will consider filing for bankruptcy rather than face having to pay year 2000 damage claims.

The year 2000 crisis is likely to focus legal attention on three topics that have not been significant to software professionals and software companies in the past, but may well become major topics as the year 2000 problem manifests itself:

- Professional malpractice
- Violation of fiduciary duty
- Consequential damages

The topic of professional malpractice has long been a major source of litigation for medical practitioners, and a significant source of litigation against attorneys and some forms of engineers such as civil engineers.

The claim of professional malpractice has not been levied against software personnel very often, but the year 2000 problem may cause this to change. In particular, year 2000 vendors themselves who contract to make repairs but fail may find this charge brought against them.

The concept of professional malpractice is that a knowledge worker failed to perform duties in a way that matched the standard level of acceptable behavior for the topic in question.

One likely target of professional malpractice claims may be some of the year 2000 tool and service providers. It often happens when problems of a significant nature occur that a great many marginal organizations and even outright frauds move into the arena in order to make a quick profit. Companies seeking year 2000 assistance are cautioned to be alert to this fact, and to use due diligence when seeking year 2000 service providers.

It is somewhat ironic that the very insurance companies that offer malpractice insurance for physicians may find the same charges brought against them if they fail to repair the year 2000 problem in time.

Executives and boards of directors of corporations have what is called a fiduciary duty to act in the best interests of their corporations. Since the onrushing year 2000 problem is a very obvious one, the boards and top executives of companies that do not take effective and rapid action to solve the problem may will find themselves sued by shareholders.

The software industry has not yet encountered the risk of consequential damages for the bugs and errors that are common in software. The idea of consequential damages is that in addition to paying for repairs or replacement of a defective product, the vendor may have to pay for any lost business or secondary damages that result.

For example, suppose you use a \$200 spreadsheet to calculate a bid for a \$1,000,000 contract. If there is an error in the spreadsheet that causes you to not get the contract, then under the concept of consequential damages, the vendor might be ordered to pay you

not only for your out of pocket \$200 for the defective spreadsheet, but an additional \$1,000,000 for the business you lost due to using the spreadsheet.

Since the year 2000 problem permeates almost every piece of financial and long-range planning software ever written, the year 2000 problem is likely to elevate the topic of consequential damages to a very significant place in software litigation.

AGGREGATION OF ALL YEAR 2000 SOFTWARE RELATED COSTS FOR THE UNITED STATES

Because hardware upgrades, data base repairs, and litigation expenses are not normalized using function point metrics, the only kind of convenient overall aggregation of Year 2000 expenses is a simple summation of the various components. Table 17 gives a rough approximation of all major Year 2000 cost elements:

Table 17 Overall Total of United States Year 2000 Expense Elements

Year 2000 Topic	U.S. Year 2000 Costs
Software Repairs	\$70,000,000,000
Data base Repairs	\$60,000,000,000
Hardware Chip Replacements	\$10,000,000,000
Hardware Performance Upgrades	\$20,000,000,000
Litigation and Damages	\$100,000,000,000
U.S. TOTAL	\$260,000,000,000

It is an interesting “sanity check” to see how the Year 2000 costs relate to the overall cost per capita for United States citizens. Assuming a 2000 AD population for the United States of about 280,000,000 citizens then the per capita Year 2000 costs for the United States amounts to about \$928 for every citizen.

Assuming a U.S. working population at the end of the century of roughly 120,000,000 workers, then about \$2,167 would be the cost for every working person in the United States.

Assuming only software workers, with a century-end total of about 1,920,000 technical software staff plus managers, the overall costs would amount to about \$135,417 for everyone in the U.S. software business at the end of the century. This is a very significant expense that is likely to cause a number of software-intensive companies to file for bankruptcy prior to the end of the 20th century.

If we back out litigation and hardware costs, of course, the per capita expenses drop by more than 50%. They are still a rather significant amount and may well have a disrupting

affect on the U.S. economy and will certainly disrupt the cash flow and profitability of a large number of enterprises.

Although function points are not normally used for dealing with hardware costs or litigation, it is a simple calculation to divide estimated total U.S. expenses of \$260,000,000,000 by the anticipated U.S. software portfolio of 1,702,750,000 function points. This results in a per function point cost of about \$152.69 for every function point deployed in the United States.

Here too, backing out litigation and hardware costs would lower the amount by more than 50%. Even with this reduction, the overall total is quite an alarming figure for the United States.

THE EMERGENCE OF THE YEAR 2000 REPAIR INDUSTRY

Since this report was first begun, a new subindustry of Year 2000 repair companies has come into being. Indeed, new companies are entering the year 2000 market faster than almost any industry in human history: approximately 25 companies a month have begun to announce year 2000 repair tools and services starting roughly in January of 1995.

This new subindustry is beginning to coalesce into three broad categories:

- Companies that sell Year 2000 analysis and repair tools for various languages
- Companies that sell Year 2000 consulting and programming assistance
- Companies that sell both Year 2000 tools and programming assistance

It is an interesting but unknown question as to how many of these companies will be effective, how many will be marginal, and how many will fail in their stated business of finding and repairing year 2000 damages.

Prior to the year 2000 issue, we have often been asked to compare the overall results of software contractors and outsource vendors to the results achieved by their clients. The general conclusion was that outsource vendors usually had higher productivity and shorter schedules than their clients, as well as somewhat better quality control.

However, on the down side of outsourcing, my colleagues and I are sometimes commissioned to serve as expert witnesses in U.S. law suits between clients and outsourcers or contractors where breach of contract, poor quality, or some other form of contractual violation is one of the charges levied by the client. (We have not yet been asked to serve as expert witnesses in international outsource litigation, although we have worked with several companies where such litigation may occur due to dissatisfaction with the delivered projects.)

Although litigation potentials vary from client to client and contractor to contractor, it is reasonable to assume that the results of hiring outsource vendors to perform year 2000 repairs will approximate the overall results of other kinds of outsourcing within the United States.

The following distribution of outsource results is based on 24 months of contractual operations, which is likely to be the same time period as needed to accomplish year 2000 repairs for a large enterprise that owns in excess of 250,000 function points of software in its operational portfolio:

Table 18 Approximate Distribution of U.S. Outsource Results After 24 Months

Results	Percent of Outsource Arrangements
Both parties generally satisfied	70%
Some dissatisfaction by client or vendor	15%
Dissolution of agreement planned	10%
Litigation between client and contractor probable	4%
Litigation between client and contractor in progress	1%

From process assessments performed within several large outsource companies, and analysis of projects produced by outsource vendors, our data indicates slightly better than average quality control approaches when compared to the companies and industries who engaged the outsource vendors. This statement is true for management information systems. Outsourcers in the systems and military domain are approximately equal to their clients, but the systems and military domains have higher quality than the MIS domain.

However, our data is still preliminary and needs refinement for year 2000 repairs, which is still so new a task that we have not yet been commissioned to explore the success or failure of year 2000 vendors.

Our main commissioned research in the outsource community has been among the clients of the largest outsource vendors in the United States such as Andersen, EDS, IBM's ISSC subsidiary, Keane, and others in this class.

There are a host of smaller outsource vendors and contractors where we have encountered only a few projects, or sometimes none at all, since our clients have not utilized their services. Further, we have not yet been commissioned to explore the specific performance of outsource vendors in the year 2000 repairs, so my observations are incidental byproducts of other kinds of studies.

Outsourcing is possible for development projects, maintenance projects (defect repairs), enhancement projects (adding new features to existing software), conversion projects (moving software to a new platform), or all of the above. The special kind of outsource

project, fixing year 2000 problems, is now exploding through the contract and outsource domains.

Although the results vary from situation to situation, the overall results from the outsource data we've collected indicates the following in terms of how the outsource world compares to similar projects carried out by client companies that produce management information systems (MIS):

Table 19 Comparison of Outsource Results With Projects Produced by Clients

Project Type	Outsource Results Versus Clients		
	Schedule Reductions	Productivity Levels	Defect Levels
New software projects	- 10%	+ 15%	- 12%
Maintenance projects	- 50%	+ 50%	- 45%
Enhancement projects	- 10%	+ 20%	- 35%
Conversion projects	- 20%	+ 30%	- 30%
Year 2000 repair projects	- 50%	+ 75%	- 65%
Average	- 28%	+ 38%	- 37%

Although this data is very preliminary and needs additional validation, the early indications are that at least the larger of the specialized year 2000 outsource vendors are capable of finding and fixing year 2000 problems somewhat faster and more thoroughly than their clients might if they attempted the work themselves.

The most visible advantage in the domain of year 2000 projects are the specialized staff and tools available within the outsource community that are more plentiful than in-house tools and personnel.

The year 1996 is the last year in which a mid-sized corporation could have hoped to tackle the year 2000 problem in a 500,000 function point portfolio with a reasonable chance of finishing before the end of the century using their own staffs and manual search methods.

Utilization of automated search engines and year 2000 specialists can move the deadline until sometime in 1997 for common languages such as COBOL. If your portfolio has languages with limited year 2000 search engines such as MUMPS or Algol or APL, there is little or no time remaining to commence year 2000 repairs.

Note that the performance edge which year 2000 outsource contractors have over their clients in terms of work effort does not translate into a direct dollars and cents advantage, because the outsource fees and charges include burden rates and enough extra to make a normal profit.

When evaluating the tools and services of a proposed year 2000 vendor, the following checklist of key capabilities may be of use:

- Does the vendor have tools that support your primary programming languages?
- Does the vendor have tools that support mixed-language projects?
- Does the vendor have tools for searching data bases for year 2000 references?
- Will the vendor's search tools find more than 99% of year 2000 instances?
- Will the vendor's repair tools safely repair more than 95% of year 2000 instances?
- How many undetected year 2000 instances are likely to be left in your software?
- Will the vendor's year 2000 repairs damage throughput or performance?
- Does the vendor have adequate personnel for repairing your entire portfolio?
- Will the vendor offer a warranty or guarantee of year 2000 compliance?
- Will the vendor provide references from prior year 2000 clients?

The year 2000 consulting, repair, and tool business is so new that a full evaluation of its effectiveness is still largely unknown. Hopefully the major vendors in this domain are fully qualified and capable for the work at hand.

However, a strong caution is indicated. Problems of the seriousness of the year 2000 problem will attract many vendors with marginal capabilities. You should exercise due diligence in selecting an outsource vendor for performing year 2000 repairs. If the vendor fails to safely repair the year 2000 problem in your portfolio, it will be too late to resolve the problem via arbitration or litigation.

The legal topics of professional malpractice, violation of fiduciary duty, and consequential damages are all likely to become common topics as the year 2000 problem sweeps over the United States and the world.

INTERNATIONAL YEAR 2000 REPAIR EFFORT FOR THIRTY COUNTRIES

The Year 2000 problem is an interesting one because it affects industrialized nations more severely than those which are less dependent upon computers and software for business and government operations.

Although the year 2000 problem is of global concern, it is starting to appear that the most heavily industrialized and computer-intensive countries are going to bear the brunt of the expenses: The United States, Japan, Germany, France, the United Kingdom, and Brazil will probably be the most heavily impacted.

Conversely, countries such as India and the Ukraine may find themselves in an advantageous position as a result of the year 2000 problem, since they will probably have a surplus of skilled programming personnel available during a period that the heavy software countries are mired down in year 2000 repair work.

Year 2000 Repairs By Country

The first topic of interest is the year 2000 repair costs in the countries that have the most software personnel employed and the largest volumes of software in production. Table 20 gives a rough approximation of Year 2000 software repair effort in some 30 countries. There is a high margin of error with Table 20. It should be noted that some of the data in Table 20 is simply derived from the previous U.S. data and extrapolated for other countries. The demographic data for software personnel is limited to professionals and ignores end-user programming.

The portfolio sizes are based on ratios of U.S. portfolios, and the volume of Year 2000 software changes is artificially held constant at 9.5% which means that just under 10% of the code in global applications are assumed to require Year 2000 updates. This is a questionable assumption, but a reasonable starting place.

The effort to make the Year 2000 changes is also held constant at 16 function points per staff month. Here too, the assumption is questionable, but is at least a starting place for more detailed analysis.

Table 20 Estimated Year 2000 Software Repair Effort for Thirty Countries

Country	Software Staff (Professional)	Portfolio in Funct. Pts.	Year 2000 hits in Funct. Pts.	Year 2000 Repairs (Months)	% of US Effort
United States	1,920,000	1,570,560,000	149,203,200	9,325,200	100.00%
Japan	900,000	738,000,000	70,110,000	4,381,875	46.99%
Russia	770,000	539,000,000	51,205,000	3,200,313	34.32%
Germany	550,000	440,000,000	41,800,000	2,612,500	28.02%
United Kingdom	390,000	312,000,000	29,640,000	1,852,500	19.87%
Brazil	475,000	308,750,000	29,331,250	1,833,203	19.66%

France	385,000	308,000,000	29,260,000	1,828,750	19.61%
China	990,000	297,000,000	28,215,000	1,763,438	18.91%
Italy	375,000	290,625,000	27,609,375	1,725,586	18.50%
India	750,000	225,000,000	21,375,000	1,335,938	14.33%
South Korea	300,000	210,000,000	19,950,000	1,246,875	13.37%
Ukraine	260,000	195,000,000	18,525,000	1,157,813	12.42%
Mexico	275,000	178,750,000	16,981,250	1,061,328	11.38%
Spain	235,000	170,375,000	16,185,625	1,011,602	10.85%
Canada	185,000	144,300,000	13,708,500	856,781	9.19%
Turkey	210,000	141,750,000	13,466,250	841,641	9.03%
Thailand	175,000	105,000,000	9,975,000	623,438	6.69%
Poland	190,000	104,500,000	9,927,500	620,469	6.65%
Taiwan	125,000	93,750,000	8,906,250	556,641	5.97%
Australia	110,000	85,250,000	8,098,750	506,172	5.43%
Netherlands	100,000	77,500,000	7,362,500	460,156	4.93%
Argentina	110,000	77,000,000	7,315,000	457,188	4.90%
Indonesia	175,000	74,375,000	7,065,625	441,602	4.74%
Egypt	145,000	68,875,000	6,543,125	408,945	4.39%
Philippines	145,000	66,700,000	6,336,500	396,031	4.25%
Pakistan	135,000	57,375,000	5,450,625	340,664	3.65%
South Africa	75,000	56,250,000	5,343,750	333,984	3.58%
Belgium	65,000	50,375,000	4,785,625	299,102	3.21%
Portugal	65,000	45,500,000	4,322,500	270,156	2.90%
Sweden	60,000	45,000,000	4,275,000	267,188	2.87%
SUM	7,087,205,000	7,076,560,000	672,273,200	42,017,075	

It should be noted once again the Table 20 is built upon a series of assumptions and hypotheses which may be incorrect. There are possible errors in every column, and the data is only suitable for discussions and for preliminary economic analysis. However the importance of the Year 2000 problem is such that publishing preliminary data with a high margin of error may be better than waiting for corrected data, since the end of the 20th century is approaching very rapidly.

The data in tables 16, 17, 18, 19, and 20 reveals a potential problem for the U.S. software industry. Since the U.S. is the country with the largest software portfolio it will be the most heavily impacted by Year 2000 repairs. This fact may give other countries a chance to make significant headway in global software markets in several fashions:

- By offshore outsourcing Year 2000 repairs.
- By offshore outsourcing of other development and maintenance projects while U.S. software personnel are entangled in the Year 2000 morass.

A possible business outcome of the Year 2000 problem is that the United States will lose its dominant market position in the software industry, while countries such as India, China, Russia, and the Ukraine gain market shares since they are not as heavily impacted by Year 2000 work and will have a substantial surplus of software technical personnel while the U.S. enters a period of shortage.

Table 21 shows approximate costs for repairing software in 30 countries. This table uses a number of simplifying assumptions, such as basing all salary and burden rates on percentages of U.S. norms. This means that the costs are rough and only approximate, but should be within the “ball park” for the countries in question:

Table 21 Estimated Expenses for Year 2000 Software Repairs in 30 Countries

Country	Monthly Salary and Burden	Effort in Staff Months	Year 2000 Repair Costs
United States	\$8,000	9,325,200	\$74,601,600,000
Japan	\$9,600	4,381,875	\$42,066,000,000
Russia	\$4,000	3,200,313	\$12,801,250,000
Germany	\$9,200	2,612,500	\$24,035,000,000
United Kingdom	\$9,200	1,852,500	\$17,043,000,000
Brazil	\$7,760	1,833,203	\$14,225,656,250
France	\$9,200	1,828,750	\$16,824,500,000
China	\$1,000	1,763,438	\$1,763,437,500
Italy	\$7,760	1,725,586	\$13,390,546,875
India	\$1,200	1,335,938	\$1,603,125,000
South Korea	\$7,200	1,246,875	\$8,977,500,000
Ukraine	\$3,600	1,157,813	\$4,168,125,000
Mexico	\$7,200	1,061,328	\$7,641,562,500
Spain	\$6,800	1,011,602	\$6,878,890,625
Canada	\$8,400	856,781	\$7,196,962,500
Turkey	\$7,400	841,641	\$6,228,140,625
Thailand	\$5,200	623,438	\$3,241,875,000
Poland	\$6,000	620,469	\$3,722,812,500
Taiwan	\$7,520	556,641	\$4,185,937,500
Australia	\$7,760	506,172	\$3,927,893,750
Netherlands	\$8,800	460,156	\$4,049,375,000
Argentina	\$7,200	457,188	\$3,291,750,000
Indonesia	\$3,600	441,602	\$1,589,765,625
Egypt	\$6,000	408,945	\$2,453,671,875
Philippines	\$3,200	396,031	\$1,267,300,000
Pakistan	\$1,000	340,664	\$340,664,063
South Africa	\$7,600	333,984	\$2,538,281,250
Belgium	\$9,600	299,102	\$2,871,375,000
Portugal	\$7,200	270,156	\$1,945,125,000
Sweden	\$9,200	267,188	\$2,458,125,000
SUM/AVG.	\$6,580	42,017,075	\$297,329,248,438

Because table 20 uses generic data and rounded values, the overall results are slightly different from the more detailed data shown earlier by language and industry. Given the overall uncertainty of Year 2000 costs, this difference, while noticeable, is not significant. The costs should be alarmingly high no matter how precisely they are stated.

Note that table 20 shows only software repair costs. Over and above software repairs will be found several other expense elements:

- Data base repairs
- Hardware upgrades and retuning of applications
- Litigation costs
- Damages awarded from litigation

Another important factor not dealt with in this study, nor in the similar Gartner Group study, is the long-range impact of inflation on Year 2000 repair costs. Note that the financial data shown in this report assumes current 1996 dollars. By 2000 inflation will no doubt raise the dollar and other currency amounts significantly, and perhaps alarmingly.

Table 22 shows the Year 2000 repair costs in descending order of magnitude, starting with the United States. Although Year 2000 repairs will be troublesome everywhere, it can be hypothesized that the top half of table 22 will have a much greater set of problems than the lower half of table 22.

Indeed, it is possible for the countries with the lowest Year 2000 repairs to expand their software markets significantly by outsourcing surplus software engineering capacity to the countries whose software personnel are likely to be preempted by emergency Year 2000 upgrades.

The future is hard to predict, but the economic consequences of the Year 2000 problem are likely to be severe for the industrialized nations and potentially advantageous for countries that lagged in early automation and computerization.

Table 22 ranks the overall software costs for the Year 2000 problem, and shows the percentage of U.S. costs on a country by country basis. This table ranks the countries in descending order, and also shows the percentage of United States costs that are likely to be expended.

Note that table 22 deals only with software repair costs and does not include hardware upgrades, data base repairs, or litigation expenses.

Table 22 Ranking of Relative Year 2000 Software Repairs for 30 Countries

Country	Year 2000 Repair Costs	Percent of U.S. Year 2000 Cost
United States	\$74,601,600,000	100.00%
Japan	\$42,066,000,000	56.39%
Germany	\$24,035,000,000	32.22%
United Kingdom	\$17,043,000,000	22.85%
France	\$16,824,500,000	22.55%
Brazil	\$14,225,656,250	19.07%

Italy	\$13,390,546,875	17.95%
Russia	\$12,801,250,000	17.16%
South Korea	\$8,977,500,000	12.03%
Mexico	\$7,641,562,500	10.24%
Canada	\$7,196,962,500	9.65%
Spain	\$6,878,890,625	9.22%
Turkey	\$6,228,140,625	8.35%
Taiwan	\$4,185,937,500	5.61%
Ukraine	\$4,168,125,000	5.59%
Netherlands	\$4,049,375,000	5.43%
Australia	\$3,927,893,750	5.27%
Poland	\$3,722,812,500	4.99%
Argentina	\$3,291,750,000	4.41%
Thailand	\$3,241,875,000	4.35%
Belgium	\$2,871,375,000	3.85%
South Africa	\$2,538,281,250	3.40%
Sweden	\$2,458,125,000	3.30%
Egypt	\$2,453,671,875	3.29%
Portugal	\$1,945,125,000	2.61%
China	\$1,763,437,500	2.36%
India	\$1,603,125,000	2.15%
Indonesia	\$1,589,765,625	2.13%
Philippines	\$1,267,300,000	1.70%
Pakistan	\$340,664,063	0.46%
SUM/AVG.	\$297,329,248,438	

Table 23 gives a rough approximation of the total number of staff months per capita for Year 2000 repairs in the 30 countries selected for inclusion.

There is of course a high margin of error, but if indeed the Year 2000 problem is as serious and pervasive as believed, then the software development capacities of the industrial nations may be seriously degraded by the huge amount of effort absorbed by Year 2000 repairs.

If indeed up to 6 months of effort per capita on the part of most industrialized software personnel is really needed for Year 2000 repairs, then many other kinds of software development and maintenance projects will be slipped or canceled.

**Table 23 Staff Months of Effort per
Capita for Year 2000 Software
Repairs in 30 Countries**

Country	Staff Months	Percent of U.S.
Japan	4.87	100.24%
United States	4.86	100.00%
Germany	4.75	97.80%
United Kingdom	4.75	97.80%
France	4.75	97.80%

Canada	4.63	95.35%
Italy	4.60	94.74%
Australia	4.60	94.74%
Netherlands	4.60	94.74%
Belgium	4.60	94.74%
Ukraine	4.45	91.69%
Taiwan	4.45	91.69%
South Africa	4.45	91.69%
Sweden	4.45	91.69%
Spain	4.30	88.63%
Russia	4.16	85.57%
South Korea	4.16	85.57%
Argentina	4.16	85.57%
Portugal	4.16	85.57%
Turkey	4.01	82.52%
Brazil	3.86	79.46%
Mexico	3.86	79.46%
Thailand	3.56	73.35%
Poland	3.27	67.24%
Egypt	2.82	58.07%
Philippines	2.73	56.23%
Indonesia	2.52	51.96%
Pakistan	2.52	51.96%
China	1.78	36.67%
India	1.78	36.67%
Average	3.95	81.31%

As can be seen, the range of effort to make year 2000 repairs varies substantially. It is likely that the companies where the effort exceeds about 4.5 staff months per software professional will probably seek outsource assistance from the countries farther down the list.

The Year 2000 Problem and 35 Urban Areas

Software development is normally an urbanized occupation that takes place within a 30 miles radius of a major metropolitan area. Table 24 shows the approximate number of software workers and the year 2000 repair effort for a total of 35 international cities. Some American cities are also included to give an overall picture of the impact of the year 2000 problem on a number of urban economies where software is a major occupation.

Table 24 Year 2000 Software Repairs by City for 35 International Cities

International Cities	Software Personnel in Urban Area	Year 2000 Effort in Months per Staff Member	Burdened Monthly Salary	Year 2000 Costs Per City
Tokyo	165,000	4.87	\$9,600	\$7,714,080,000
London	136,000	4.75	\$9,200	\$5,943,200,000
Los Angeles	127,000	4.86	\$9,000	\$5,554,980,000
New York	116,000	4.86	\$9,000	\$5,073,840,000
Paris	85,000	4.75	\$9,200	\$3,714,500,000
Yokohama	75,000	4.87	\$9,600	\$3,506,400,000
Seoul	112,000	4.16	\$7,200	\$3,354,624,000
Sao Paulo	110,000	3.86	\$7,760	\$3,294,896,000
Osaka	67,000	4.87	\$9,600	\$3,132,384,000
Toronto	80,000	4.63	\$8,400	\$3,111,360,000
Mexico City	110,000	3.86	\$7,200	\$3,057,120,000
Chicago	75,000	4.86	\$8,300	\$3,025,350,000
Berlin	65,000	4.75	\$9,200	\$2,840,500,000
Montreal	68,000	4.63	\$8,400	\$2,644,656,000
Sydney	66,000	4.65	\$8,400	\$2,577,960,000
Washington DC	50,000	4.86	\$8,200	\$1,992,600,000
Rio de Janeiro	64,000	3.86	\$7,760	\$1,917,030,400
Moscow	112,000	4.16	\$4,000	\$1,863,680,000
San Jose, CA	36,000	4.86	\$9,000	\$1,574,640,000
Stockholm	35,000	4.75	\$9,200	\$1,529,500,000
San Francisco	33,000	4.86	\$9,000	\$1,443,420,000
Rome	40,000	4.60	\$7,760	\$1,427,840,000
Hong Kong	38,000	4.60	\$7,700	\$1,345,960,000
Madrid	46,000	4.30	\$6,800	\$1,345,040,000
Bangkok	70,000	3.56	\$5,200	\$1,295,840,000
Cairo	75,000	2.82	\$6,000	\$1,269,000,000
Buenos Aires	40,000	4.16	\$7,200	\$1,198,080,000
Lima	40,000	4.16	\$6,000	\$998,400,000
Leningrad	56,000	4.16	\$4,000	\$931,840,000
Jakarta	62,000	2.52	\$3,600	\$562,464,000
Beijing	90,000	1.78	\$1,000	\$160,200,000
Bombay	60,000	1.78	\$1,200	\$128,160,000
Shanghai	70,000	1.78	\$1,000	\$124,600,000
Delhi	58,000	1.78	\$1,200	\$123,888,000
Karachi	41,000	2.52	\$1,000	\$103,320,000
Total	2,573,000	4.03	\$6,768	\$79,881,352,400

Showing year 2000 expenses by urban area is in some ways more shocking than considering national levels. Most of us live in or near a city and know the numbers and kinds of companies in the area that employ software personnel. Dividing the urban costs by the number of local software employers leads to a very alarming economic picture.

Aggregation of Global Year 2000 Repair Expenses

Tables 16 through 23 show only the 30 largest countries in terms of their software populations and hence anticipated software expenses. Since there are roughly 200 countries in the United Nations, an interesting question is that of the magnitude of effort and costs for the 170 or so countries that are not shown separately.

My overall global demographic data indicates that the world population of software professionals circa 1996 totals to about 12,500,000 (with a high margin of error). The 30 countries enumerated here had a total professional population of just over 7,000,000. Assuming both sets of assumptions are true, the absent 170 countries would have a combined software population of about 5,500,000.

It is interesting that the average software population for the 30 countries that are enumerated here is about 233,000 per country. The average software population for the 170 countries that are not shown separately averages about 32,000 per country.

Assuming that the ratio of Year 2000 expenses is similar to the ratio of software personnel (a questionable assumption but at least a starting point) then the Year 2000 expenses for the missing 170 countries would total to about \$233,000,000,000. This amounts to about \$1,371,000,000 per country.

By contrast, the average cost for the 30 countries that are shown separately is about \$9,900,000,000 per country. Of course "averages" are not very meaningful when whole countries are considered, but the results do give a sanity check to the approximate costs on a global basis.

Adding the invisible costs for the missing countries to the visible costs for the 30 countries shown here yields an approximate global cost of \$530,000,000,000 for Year 2000 software repairs.

For Year 2000 software repairs, the U.S. total of roughly \$70,000,000,000 amounts to about 13.2% of the global total of \$530,000,000,000. This raises an interesting question as to the entire total of all Year 2000 repairs on a global basis when software, hardware, data bases, and litigation are all considered.

Table 25 is an attempt to assemble a global picture of all major Year 2000 cost elements simply to judge the rough magnitude of costs. For every cost element except litigation, the U.S. ratio of 13.2% of global totals was used. Note that unlike tables 15 through 17 which only showed 30 countries, table 19 is based on the assumption that all of the approximately 200 countries constituting the United Nations will have Year 2000 expenses.

For litigation, the United States is assumed to have a much higher total of global costs than for the other Year 2000 expense elements. For the litigation amount I estimated that U.S. litigation would amount to about 33% of world totals.

The rationale for this assumption is because the United States is a much more litigious country than the other major industrial powers such as Japan, Germany, France, the United Kingdom, and many others. My assumption on litigation percentages may well be wrong, but I'm not aware of any other data on this topic.

Table 25: Overall Total of Global Year 2000 Expense Elements

Year 2000 Topic	Global Year 2000 Costs
Software Repairs	\$530,000,000,000
Data base Repairs	\$454,000,000,000
Hardware Chip Replacements	\$76,000,000,000
Hardware Performance Upgrades	\$150,000,000,000
Litigation and Damages	\$300,000,000,000
WORLD TOTAL	\$1,510,000,000,000

No matter how things turn out, it is certain that the software industry in every country will be undergoing a major transformation between the years 1996 and 2005 in reaction to the Year 2000 problem.

One of the problems faced by the software industry is sociological. We have not been regarded by the older professions such as electrical and mechanical engineers as being true engineers or even true professionals. The fact that the software industry, collectively, has brought about one of the most expensive and hazardous problems in human history when it could easily have been avoided is going to lower our status even more.

Not only is the Year 2000 problem one of the most expensive problems in human history, it is also one of the most embarrassing. This problem has been theoretically discussed for more than 25 years, and its significance has been hypothesized with increasing alarm for more than 10 years. It is not a credit to the human race nor to the software industry that such an obvious problem with such a straight-forward technical solution should have reached the magnitude that is likely to occur.

On the other hand, the Year 2000 problem is symptomatic of a general human tendency to avoid trying to solve problems until the evidence is overwhelming. The historical difficulties which medical researchers such as Lister and Semmelweis had in introducing sterile surgical procedures, and the earlier resistance to Jenner's concept of vaccination illustrates that software is not the only learned profession that does not move swiftly to minimize potential risks.

BENEFITS OF SOLVING THE YEAR 2000 PROBLEM

It is no secret that software is the most troubling technology of the 20th century. A surprising aspect of the year 2000 problem is that the companies that solve this problem are going to end up knowing much more about their software than ever before. Even better, their software will be in much more stable and reliable condition than it is right now.

Recall that the year 2000 problem is only one instance of a large class of similar problems where storage limitations caused insufficient space to be reserved. The same tools that attack or solve the year 2000 problem can also be applied to the other instances so the software that is repaired should be more robust than now.

Further, the detailed inventory of software portfolios and corporate data bases and repositories needed for year 2000 repairs will give corporations and government agencies a much improved ability to match their software projects to the needs of their operating units.

It would be premature and highly optimistic to say that solving the year 2000 problem has a positive return on investment. However, there will certainly be significant positive value to solving the year 2000 problem.

If aspects of game theory are applied to the year 2000 problem, the results indicate that a rapid and thorough attack on the year 2000 problem is the minimax solution; i.e. the solution that will give the best return for the lowest expense.

Consider the following hypothesis: For every dollar spent on repairing the year 2000 problem, the value in terms of greater reliability and stability of software will probably return fifty cents in reduced maintenance costs for the applications in question.

For every dollar *not* spent on repairing the year 2000 problem, the anticipated costs of litigation and potential damages will probably amount to more than ten dollars.

The expenses of repairing the year 2000 problem are going to occur, like it or not. Since delays in attacking the year 2000 problem lower the probability of successful repairs and raise the probability of litigation, the overall conclusion is that a rapid attack on the year 2000 problem is the best economic solution.

SUMMARY AND CONCLUSIONS

The year 2000 problem is rapidly approaching, and is one of the most critical issues ever faced by the software industry or by any industry for that matter. Because the year 2000 problem has been developing for 25 years and affects many aging legacy applications, the one-time costs for year 2000 repairs are going to be alarmingly high.

However, the year 2000 problem cannot be ignored and will not go away by itself. The best response to the year 2000 problem is a rapid and energetic attack as early as possible. The worst response is to ignore the problem or to understate its importance.

The information presented in this report is preliminary and is believed by the author to have a large but unknown margin of error. However, even preliminary data is better than none when it comes to dealing with a problem as severe as the year 2000 problem.

ADDITIONAL SOURCES OF INFORMATION REGARDING THE YEAR 2000 SOFTWARE PROBLEM

Because of the magnitude and seriousness of the year 2000 problem, the opinions and data of a single organization such as Software Productivity Research should be confirmed or challenged by means of using multiple sources.

This section includes contact information for other independent research organizations which have been addressing aspects of the year 2000 problem. Readers are urged to contact any or all of these data sources for alternate evaluations of the year 2000 problem.

This set of contacts deals primarily with organizations that are not direct vendors of year 2000 services and tools, since these organizations may have a vested interest in their own products. The emphasis in this section is on organizations that can provide information about the costs and schedules of making year 2000 repairs. Both non-profit and profit making organizations are cited.

Digital Consulting, Inc.

DCI is a well known software conference and seminar company founded by George Schussel. DCI is a for-profit organization. DCI is primarily a conference group, with a smattering of management consulting as well. DCI sponsors a number of large conferences on the year 2000 issue in the United States and abroad.

Digital Consulting, Inc.
204 Andover Street
Andover, MA 01810

Phone 508 470 3880
FAX 508 470 0526
Web <http://www.DCI.Expo.COM>

Gartner Group

The Gartner Group is a for-profit research corporation that collects data on a wide variety of topics. The Gartner Group has produced a report similar to this one on the global impact of the year 2000 problem: Hall, B. and Schick, K. "Year 2000 Crisis – Estimating the Cost;" Gartner Group Application Development and Management Strategies (ADM) Research Note, Key Issue Analysis, KA-210-1262. However the Gartner approach uses lines of code rather than function point metrics.

Gartner Group
Gartner Park

Top Gallant Road
Stamford, CT 06904

Phone 203 964 0096
FAX 203 324 7901
Web <http://gartner.com>
Email info@gartner.com

The Information Technology Association of America (ITAA)

The ITAA is a non-profit association of software vendors. The ITAA has formed a year 2000 task force under the chairmanship of Peter Sheridan of Viasoft. The ITAA has published a useful although incomplete catalog of year 2000 vendors. Heidi Hooper at 703 284 5312 or hhooper@itaa.org are the contacts cited in the ITAA year 2000 vendor catalog.

ITAA
1616 N. Fort Myer Drive, Suite 1300
Arlington, VA 22209

Phone 703 522 5055
FAX 703 525 2279
Web <http://www.itaa.org>

Peter de Jager

Peter de Jager is an independent Canadian consultant, speaker, and collector of data on the year 2000 problem. Peter runs a very popular year 2000 web site, and is a frequent contributor of articles and lectures on the year 2000 problem. Since Peter has specialized in year 2000 economic issues, he is a good source of practical information.

Peter de Jager
22 Marchbank Crescent
Brampton, Ontario L6S 3B1
Canada

Phone 905 792 8706
FAX 905 792 9818
Web <http://www.year2000.com>
Email pdjager@hookup.net

Ken Orr Institute

Ken Orr, the founder and chairman of the Ken Orr institute, is a well-known author and lecturer on a variety of software topics. He has been a frequent key note speaker at year 2000 conferences, and has also performed a number of consulting studies in the year 2000 domain. Ken also explores issues in the data warehouse domain. His conclusion is that U.S. companies may not be moving fast enough to solve the year 2000 problem before the end of the century.

Ken Orr Institute
534 South Kansas
Topeka, KS 66603

Phone 913 357 0003
FAX 913 357 8446
Web <http://www.kenorrinst.com>

MITRE Corporation

The Mitre Corporation is a non-profit government-sponsored research institute that deals with a large number of military technology questions. The U.S. Department of Defense sponsored a study of the military and defense implications of the year 2000 problem. The MITRE study of the year 2000 issue is specialized, but very thorough. The research was headed up by Thomas Backman of MITRE's Burlington laboratory. The MITRE year 2000 research is somewhat alarming because it deals with the impact of the problem on satellites, military aircraft, military logistics, ships, weapons systems, and a number of other areas where failures can range from serious to catastrophic.

MITRE
202 Burlington Road
Bedford, MA 01730-1420

Phone 617 271 2725
FAX 617 271 6239
Web <http://mitre.com>
Email tkb@mitre.org

National Software Council (NSC)

The National Software Council is a fairly new non-profit organization created to assist the United States software industry in maintaining a favorable balance of trade. The NSC is still feeling its way in terms of missions, but the year 2000 problem is obviously one that

falls within the NSC purview. However, the NSC is not overburdened with funds and resources. The current NSC president is Larry Bernstein, formerly of AT&T.

National Software Council
PO Box 4500
Alexandria, VA 22303

Phone 703 742 7111
FAX 703 742 7200
Email lbernstein@worldnet.att.net

Rubin Systems, Inc.

Howard Rubin is a well known software lecturer, author, and researcher on a wide variety of topics. His also the designer of the ESTIMACS software cost estimating tool, and a tenured professor of software engineering at Hunter College. Howard is a frequent keynote speaker at year 2000 conferences and has accumulated a solid body of data on the year 2000 issue. Howard has performed an independent study on the performance degradation associated with careless year 2000 repairs and has reached a conclusion similar to the one in this report.

Dr. Howard Rubin
5 Winterbottom Lane
Pound Ridge, NY 10576

Phone 914 764 4931
FAX 914 764 0536
Email 71031.377@Compuserve.com

Society of Information Management (SIM)

The Society of Information Management is a non-profit association aimed at the software management community. SIM has a year 2000 working group that is accumulating data and information on the year 2000 problem. The co-chairman is Dr. Leon Kappelman of the University of North Texas.

Leon A. Kappelman
Business Computer Information Systems
Associate Director, College of Business Administration
University of North Texas
PO Box 13677
Denton, TX 76203

Phone 817 565 3110
FAX 817 565 4935
Email kapp@unt.edu

Software Productivity Group (SPG)

SPG is a for-profit conference, seminar, research, and publication company. Among their various offerings is the well-known software journal Application Development Trends edited by John Desmond. SPG also sponsors a variety of conferences and seminars on year 2000 issues. The Software Productivity Group (SPG) and the next company in this listing, Software Productivity Research (SPR), have similar names, but have no direct business relationship. This same statement is also true for the Software Productivity Consortium (SPC), Software Engineering Institute (SEI), and Software Research Associates (SRA). We all have the word “software” as part of our corporate names, but we are not related in any business sense.

Software Productivity Group, Inc.
386 West Main Street, Suite 2
Northboro, MA 01532

Phone 508 393 7100
FAX 508 393 3388

Software Productivity Research, Inc. (SPR)

Software Productivity Research, Inc. (SPR) is a for-profit research, development, and management consulting company located in Burlington, Massachusetts. SPR was formed in 1984. The author of this report, Capers Jones, is the SPR Chairman. SPR has been actively collecting data on the economic impact of the year 2000 problem, as can be seen from the report itself.

Software Productivity Research, Inc.
1 New England Executive Park
Burlington, MA 01803-5005

Phone 617 273 0140
FAX 617 273 5176
Web <http://www.spr.com>
Email Capers@spr.com

SPR

This company and the previous company, Software Productivity Research, share the SPR initials but do not have any direct business connection. This second SPR was formed in 1973 and is a mid-western corporation specializing in maintenance, renovation, and year 2000 software updates. The president is Rob Figliulo. Both of the SPR's collect data on the year 2000 issue, and both are occasionally mistaken for the other. Fortunately there is no direct competition between the two SPRs.

SPR

2105 Spring Road, 7th floor
Oak Brook, IL 60521

Phone 708 990 2040
FAX 708 990 2062
Web www.sprinc.com

Software Technology Support Center (STSC)

The Software Technology Support Center is a non-profit organization funded by the U.S. Air Force and located at Hill Air Force Base in Ogden, Utah. STSC performs research in a variety of software-related topics including the year 2000 problem. STSC is also the publisher of the military software journal, Crosstalk, which is surprisingly lively and interesting for a government journal. One of the STSC year 2000 researchers, Bryce Ragland, has written a useful book that is being published by McGraw Hill as this report is written: The Year 2000 Problem Solver: A Five-Step Disaster Prevention Plan

Software Technology Support Center
ALC/TISE
7278 Fourth Street
Hill AFB, UT 84056-5205

Phone 801 777 8068
FAX 801 777 8069
Web <http://www.stsc.hill.af.mil>
Email stscols@software.hill.af.mil

WSR Consulting Group

The WSR consulting group occupies an interesting niche: “management, technology, and litigation consulting. The founder and president, Warren Reid, is not an attorney but rather a management consultant who specializes in intellectual property and software litigation. Warren frequently works as an expert witness in arbitration and litigation

involving software issues. Obviously the year 2000 problem will be a fruitful domain for litigation. Warren is a frequent writer and speaker at year 2000 events, and tends to stress the legal issues, which other speakers and authors may not be aware of.

WSR Consulting Group
4273 Noeline Avenue, Suite 200
Encino, CA 91436

Phone 818 986 8842
FAX 818 986 7955
Email consult@primenet.com

End of Document