The Century Date Change Software Problem

Robin Guenier – *September 2011*

Executive Summary

Many commentators, comparing its seemingly trouble free outcome with predictions of disaster, have claimed that the Century Date Change Problem (better known as Y2K or the Millennium Bug) was little more than an unjustified scare and an irresponsible waste of money. The objective of this paper is to demonstrate, by explanation, example and reference, that they are wrong.

The paper is divided into eight sections. The first is a short introduction and the second a note on the problem's historical background, noting that it wasn't exclusively a computer problem and explaining how the use of two digits, not four, to designate the year – e.g. "70" instead of "1970" – was a standard practice in early computer programming. Section 3 ("**The Challenge**") shows how this became a serious problem: whereas two digit references worked perfectly well when only twentieth century dates were being processed, they didn't when twenty-first century dates were involved. Thus, in the twentieth century, it was possible to calculate someone's age by subtracting their birth year (say 50 – i.e. 19**50**) from the current year (say 90 – i.e. 19**90**) – getting 40. But, when, for example, 50 (i.e. 19**50**) is deducted from 08 (i.e. 20**08**), the answer is *minus* 42. The section explains, with examples of actual failures, how – had this not been fixed – it would have had major and widespread impact. This is supported by references to statements by the Bank of England and the Bank for International Settlements. Finally, section 3 reviews the special problems that arose in relation to personal computers and so-called "embedded chips" (devices that monitor and control a wide range of electronic systems, e.g. safety monitors in power generation plants). It reviews how PCs and embedded chips turned out to be rather less of a problem than had been initially feared.

Section 4 ("**The Solution**") reviews what had to be done to solve the two-digit problem. It shows how the seemingly simple task of turning two-digit references into four-digit references was far from simple in practice – especially when, as was commonly the case, a system had to work smoothly in relation to third party systems. It shows how "fixing" a system wasn't the end of it – fixes had to be rigorously tested and often adjusted and retested. It was this fixing and testing that cost so much money: almost every large organisation in the developed world had systems containing potentially critical two-digit year references.

Section 5 ("**Criticisms**") deals with the "unjustified scare" and "irresponsible waste of money" claims. Referring back to the earlier sections, it demonstrates how the problem and its potential impact were real and justifiably worrying. So why has it become so widely regarded as a scare? A suggested reason is that it seemed so unlikely that the IT industry would have allowed such a dangerous state of affairs to exist: surely, if it were real, the industry would have dealt with it

long before the 1990s? Unfortunately it didn't and the problem was permitted to reach a critical point. Another possible reason was the media's characterisation of warnings ("*if action isn't taken*, the consequences **could be** dire") as predictions ("the outcome **will be** dire"). The classic example was the commonly repeated "experts tell us that planes will fall from the sky". No "expert" said that. Most avoided prediction altogether.

Section 5 deals also with the claim that businesses and, especially, countries that did little or nothing had few or no problems. The reality is that, whereas many small and medium sized businesses had few problems (essentially it was a problem for big organisations with significant amounts of "legacy" software), most countries throughout the world were involved in remedial action. Banking and telecommunications are, for example, intrinsically international activities: a system was not fixed until all the systems to which it related were fixed. Nonetheless, the problem was considerably more serious in some countries than in others: there was, for example, little computerisation in Chad and Haiti. Also, unlike the developed Western economies, economies that came late to computing didn't rely on software originally developed in the 1960s and 1970s. Moreover, those businesses and countries that started Y2K remediation late were able to take advantage of the experience gained and lessons learned by the pioneers. Such considerations explain the – essentially false – claims that, for example, Italy and Spain did little about Y2K.

Then there's cost: was it really necessary to spend around $300 - $500 billion? The answer is that it was a huge task and an organisation's cost – when spread over 2 to 3 years – was usually not a big percentage of its IT budget. Thus the average UK clearing bank spent about £300 million: not much when compared with the cost (£270 million) incurred last year by a German bank in fixing just one small date related problem.

The section ends by noting that, although the outcome was relatively benign, the examples of actual failures referred to show that, without the action taken, it might well not have been. Those who got on and fixed the problem took the only rational and responsible course.

Section 6 reviews some date-related problems that occurred in 2010 (the Year 2010 (or Y2.01K) Problem) and Section 7 ("**Some Observations"**) some useful matters that arose from Y2K remediation: unexpected benefits (e.g. the development of business continuity planning as a standard procedure), important lessons about project management (lessons seemingly not learned), and a better understanding of the importance of computing to business and society.

Section 8 ("**Conclusion"**) notes that anyone who believes Y2K to have been a scare, a hoax or a waste of money hasn't tried to understand it: it was, in fact, a bizarre, unnecessary but real and seriously worrying problem.

The paper ends with references, notes and links.

1. Introduction

This paper was prompted by an article[i] by Donna Laframboise dated 9 August 2011 and entitled *The Y2K Scare, the Media & Climate Change.* In it, she quoted various catastrophic predictions about the Y2K computer problem and, referring to the fact that in the event little went wrong, she drew parallels with current climate change scare stories – her assumption being that these stories are equally poorly based. She observed "*potentially hundreds of billions of dollars that could have been spent finding a cure for cancer were flushed down the toilet by governments and corporations in a mad rush to avert an imaginary Y2K catastrophe".* In other words, she assumed that, because scary scenarios didn't occur, the effort to avoid them was unnecessary – referring to the whole thing as "*an embarrassing … episode*".

I believe she, and other commentators who have made similar points, are wrong about this and my objective here is to explain why. I draw no conclusion about whether or not she's also wrong about climate change – that's an issue well beyond the scope of this paper.

A note on terminology

I've called this paper "The Century Date Change Software Problem" as I consider that the most accurate description of what came to be widely and conveniently known as "Y2K". Other names – such as "The Century Date Change Problem", "The Year 2000 Date Change Problem" and "The C2000 Problem" were also used, as was the misleading but catchy "Millennium Bug" – misleading because a computer "bug" refers typically to an error, mistake or flaw (for example in logic or syntax) in a computer program. Y2K was not a bug – nor was it a "virus". As explained below, it was a deliberate and, at the time, sensible and successful solution to a serious problem: the huge cost of computer memory in the early days of computing. The mistake was allowing software incorporating that solution to continue for so long.

A personal note

I first heard about the problem in early 1996 when I was Chief Executive of the UK government's Central Computing and Telecommunications Agency (the CCTA), reporting to the Cabinet Office. It was an interim appointment – I was a general manager recruited from the private sector to assess the future of and to reorganise the CCTA. I had no specialist IT experience. From later in 1996 to 2000, I was Executive Director of Taskforce 2000, tasked by the government with raising awareness of the Y2K problem.

2. Background

The Y2K problem derives from the common and useful practice of using two digits rather than four to designate a year – common because it's an easy shortcut and useful because, over most of a century, it's usually obvious what is meant. Thus, in the old song *My Darling Clementine*, the description of Clementine's father as a "*miner forty-niner*" would have originally – and for many years – been understood as referring to someone who joined the 1849 Californian gold rush. Even today it's obvious what a reference to the "Twenties" or the "Sixties" means and most credit cards still have the year embossed on them in two digits. Useful, yes – but it can go wrong. Thus there's a story that, when her husband died, his wife had a tombstone prepared for them both: his dates were inscribed as "1900-1991" and hers as "1920-19 ". But she died after 1999[ii]. This was not an isolated case: it's reported that in the United States there were around 500,000 cases where an expected death in a year beginning "19" had already been set in stone for someone still alive in 2000[iii].

So the problem wasn't exclusively a computer problem. The use of two digits for the year for computers probably goes back to the practice of using two digits in punch cards in the early years of the twentieth century: punch cards were space limited (under 100 character columns), so it made obvious sense. Likewise, when computers were first introduced for business purposes in the 1960s, electronic storage was extremely expensive – about a million times more than it is today. So it was a practical and seemingly harmless practice to store the year element of a date in a two-digit rather than a four-digit field: e.g. "31-12-70" instead of "31-12-1970". Here's a comment by Alan Greenspan[iv]:

> *"I'm one of the culprits who created this problem. I used to write those programs back in the 1960s and 1970s, and was proud of the fact that I was able to squeeze a few elements of space out of my program by not having to put a 19 before the year. Back then, it was very important. We used to spend a lot of time running through various mathematical exercises before we started to write our programs so that they could be very clearly delimited with respect to space and the use of capacity. It never entered our minds that those programs would have lasted for more than a few years. As a consequence, they are very poorly documented. If I were to go back and look at some of the programs I wrote 30 years ago, I would have one terribly difficult time working my way through step-by-step."*

It usually worked well: for example, in 1965 the age of someone born in 1910 would have been calculated by deducting 10 from 65, giving 55. However, it wouldn't work for someone born before 1900, so workarounds had to be devised for such special cases. But that didn't, for example, apply to Mary Bandar of Winona, USA who, in 1992, was invited to join an infant class as she was born in "88" – but Mary was 104 years old[v]. However, such examples were not seen as a serious problem: computers were then relatively rare and, as the years passed, there were increasingly fewer people born in the nineteenth century.

The use of two digits became a programming standard that continued into the 1970s, and even the 1980s. It worked, it saved expensive space and, in any case, as Alan Greenspan said, programmers didn't think their work would last for more than a few years: the fact that it might cause problems at the end of the century wasn't really considered. But it did. So "31-12-00" would be read as "31-12-1900" (or rarely as "31-12-19100"), not as "31-12-2000". (It's already hard – in 2011 – to appreciate just how unreal a year starting with the digits "20" seemed to someone in 1970, even in 1990 – hence the widespread practice of referring to the first year of the new Millennium as "The Year 2000", not simply "2000".)

Unfortunately many programs utilising these two digit date references were not changed or replaced. Instead, they were incorporated into new programs and gradually buried within millions of lines of computer code – commonly losing the original programmer's notes and/or source code (the "key" that enables programmers to understand or modify a program). This was especially so for very large custom designed programs typically used by large organisations such as banks and government departments. And it should be noted that these old date references existed as part of established programs that had stood the test of time: and they had worked reliably – unlike many more recently introduced systems. And thus they were taken for granted.

3. The Challenge

So the essential problem was the widespread use of two-digit date references throughout the software commonly employed by large organisations. It was extraordinary (and frankly irresponsible) that as late as 1996, only four years before the date change, few people in the IT industry seem to have said much about it. Yet software that was unable to process date information correctly when, for example, a date in a pre 2000 database was matched or compared with a post 1999 date, was plainly going to cause problems, many potentially serious. For example, in the finance and insurance industries, interest calculations would be in error, credit card expiry dates would fail, fund transfers would be disrupted, interest payments would be incorrectly calculated, lease records would cease to make sense, policy due dates would be incorrect, annuities, pensions and other entitlements would be miscalculated and billing records would malfunction.

Probably the best way to demonstrate how a failure might happen is to consider two simple examples of the relatively few failures that actually did occur, one before the date change and one afterwards.

> The first[vi] concerned credit card swipe machines in the UK that rejected cards with post 1999 expiry dates. It happened because it read them as having already expired (e.g. in 1900) and affected about 20,000 machines – 2% of UK retail outlets using such machines. The manufacturer, Racal

Electronics, said it "*was unable to explain how the malfunction was missed when the company went over its plans for the year 2000 date change*". Clearly, were it not for such plans, Racal would have been in serious trouble.

The second[vii] came to light in 2002. A Health Visitor in Yorkshire noticed a higher than usual number of babies with Down's Syndrome in her area. What had happened was that pregnant women who were referred to the National Health Service's Northern General Hospital in Sheffield as possibly being at risk of having babies with birth defects were initially screened by a routine designed to identify those at highest risk. A major factor was age (women over 35 were at higher risk) so that was a main focus of the screening process. Unfortunately, the PathLAN computer used for the task used a two-digit system. Therefore, if a woman born in 1962 presented in 1999, the computer deducted 62 from 99, getting 37 – over 35, so she was at risk. But, if the same woman presented in 2000, it deducted 62 from 00, getting *minus* 62 – under 35, so (it concluded) she was not at risk. This affected over 150 women.

In the 1990s, Racal Electronics was the UK's leading manufacturer of electronic devices for *inter alia* the finance, telecommunications and defence sectors, both in the UK and overseas. Had it decided in 1997 that money spent on fixing the Y2K problem was – to use Ms Laframboise's phrase – "*flushed down the toilet*" and had not instituted its date-change plan, there would have been very many more malfunctions than those relatively few card readers that escaped their net. Likewise, the NHS utilises vast numbers of dates for a huge range of purposes, from scheduling operations to "use-by" dates for drugs. Had it decided it would be a waste of money and effort to institute its massive and costly Year 2000 compliance programme (better perhaps to spend the money on cancer research), vastly more people would have been affected than those unfortunate women in Yorkshire. There would have been nothing imaginary about the resulting catastrophe for UK healthcare. And, were all these failures happening at around the same time, it needs little imagination to see that the impact on society would have been most significant – more significant than the combined impact on the businesses and individuals affected individually. And that assumes the problems affected only electronic equipments and healthcare. But of course the correct processing of date information is critical throughout nearly all aspects of modern life.

A prime example is banking – both nationally and internationally. Here's an extract from a News Release[viii] (the first of a series) issued by the Bank of England (the UK's central bank) in February 1998:

> *The origin of the problem lies in the programming methods used in earlier years, when computer memory was expensive and the use of two digits rather than four to represent the year was an efficient short-cut. Many of those programs, or elements of them, are still in use, and while the problem is simple to describe* ***it is so widespread and pervasive that the***

> ***cost and complexity of correcting it represents a massive burden on business***. [My emphasis.]

The then Governor of the Bank (Eddie George) is quoted as saying:

> "*The financial system - especially in a centre as large and diverse as London - is highly interdependent and the failure of one quite small part can easily have substantial knock-on effects. And* ***the failure of parts of the infrastructure could be catastrophic***." [My emphasis.]

The Bank, and especially its Governor, is noted for caution and understatement. So Mr. George's use of the word "catastrophic" is significant. He plainly didn't see the potential for catastrophe as "imaginary" (Ms Laframboise's word). But neither was he making a prediction: there's a vast difference between prediction and his "*could be*" warning.

The Bank of England was not alone in issuing a serious warning. Its statement followed a press release[ix] and a technical paper[x] published in September 1997 by the Bank for International Settlements in Basel (*The year 2000 - A challenge for financial institutions and bank supervisors*) and addressed to central banks worldwide. Here's an extract from the Press Release:

> *At their meeting in Basle on 8th September 1997, the central-bank Governors of the Group of Ten reviewed the need for financial institutions to check all their computer applications in advance of the new millennium. Not only will a large number of applications have to be converted or replaced, but extensive testing will be necessary to ensure that all operations run smoothly after conversion.*

And from the Introduction to the technical paper:

> *Failure to address this issue in a timely manner would cause banking institutions to experience operational problems or even bankruptcy and could cause the disruption of financial markets.*

Following that statement, similar warnings were made by other central banks in the developed economies – for example by Mr. Ernest T Patrikis, First Vice-President of the US Federal Reserve Bank, in a detailed statement made in June 1998 before a Committee of the US House of Representatives.[xi]

The reason why the potential impact of the date-change problem on banking was regarded as especially important was, as Eddie George pointed out, the interdependence or interoperability of international financial systems whereby a failure in one bank could cause knock-on failures elsewhere. Finance in the developed world had become, in effect, a single, highly complex and therefore very vulnerable system. For example, in his evidence to the US House of Representatives (see above), Mr. Patrikis noted that banking activities are dependent on "*a large, geographically diverse and highly computer intensive*

global infrastructure for each of the key phases of … activity – from trade execution through to payment and settlement". He illustrated this with an example:

> *"…consider the daily financial market activities of a hypothetical US-based mutual fund holding stocks and bonds in a number of foreign jurisdictions. Such a mutual fund would likely execute trades via relationships with a set of securities dealers, who themselves might make use of other securities brokers and dealers, including some outside the United States. The operational integrity of the major securities dealers in each national securities market is critical to the smooth functioning of those markets. In addition, securities trading in most countries is reliant on the proper functioning of the respective exchanges, brokerage networks, or electronic trading systems and the national telecommunications infrastructure on which these all depend. Financial markets today are also highly dependent on the availability of real-time price and trade quotations provided by financial information services. For record-keeping, administration, and trade settlement purposes, our hypothetical mutual fund would also likely maintain a relationship with one or more global custodians (banks or brokerage firms), who themselves would typically maintain relationships with a network of sub-custodians located in various domestic markets around the world."*

International finance was particularly vulnerable: References **viii**, **ix**, **x** and **xi** provide a comprehensive view of that vulnerability. Yet its full nature was poorly understood at the time – a benefit of Y2K is that it is better understood today. But, although international finance raised especially serious concerns, the problem had, as noted already, far wider implications. For example, re healthcare, the UK's National Health Service was mentioned above (page 6). And, in the USA, a committee of the House of Representatives reviewed progress in resolving the healthcare issues in May 1998[xii]. Wider implications were well illustrated by what was probably the UK's first government note on the matter (an overview briefing for Members of Parliament) issued in December 1996[xiii]. It included a short but comprehensive explanation of the problem and referred to the dangers of knock-on failures:

> *Because of **the ubiquitous** nature of the effects and the **very short time interval** in which they may act at the turn of the century, some see a danger that a 'domino effect' might cause IT systems to fail throughout whole sectors of the economy.* [Original emphases.]

Nevertheless, the briefing avoided any hint of alarm (note that "*might cause*") and stressed, in particular, uncertainty about the consequences of failure – noting however that an IT failure can be grave. It urged organisations to treat Y2K (then referred to as "Y2000") "*as a business, not just a technical, problem*". This briefing, the bankers' warnings and the US healthcare report referred to above were all serious warnings but were a long way from illustrating Ms

Laframboise's assertion about a "*mad rush to avert an imaginary Y2K catastrophe*".

This paper has, so far, focussed on how computer software, especially in large systems, had inherited a programmers' solution to the extreme cost of memory in the early days of computing. It was this that caused the most serious problems. But there were other date-change concerns. For example, one that caused some difficulty[xiv] was uncertainty about whether or not 2000 was a Leap Year. The rule is that a year divisible by 4 is a leap year, unless it's divisible by 100. However, there is an exception to this: years divisible by 400 are leap years. Therefore, although 1800 and 1900 were not leap years, 2000 was a leap year. Problems were likely to emerge at the end of 2000 – and at least one did[xv]. (An overview of this, *Blame the madness on Dennis the Short*[xvi], is interesting).

Two other areas of concern were personal computers ("PCs") and so-called "embedded chips":

Personal computers

> Some PC operating systems (e.g. older versions of "DOS" and Windows) included elements that couldn't cope with the 1999/2000 transition. And some might contain software that could produce unexpected results after the date-change: e.g. older versions of the Compuserve email system used a two-digit year that caused incoming messages in January 2000 to be treated as if they were older than messages received in 1999 or before: just as "98" is smaller than "99", so "00" is smaller than either. The result was that incoming emails, which the recipient would expect to find at the top of their Inbox, were at the bottom and thus appeared to be lost. Another example might be where someone had (unwisely) created a spreadsheet using two digits for the year. Working well when only dates in the twentieth century were being processed, this could produce unpredictable problems as soon as it had to deal with dates in the twenty-first century. Further, earlier versions (typically produced before 1997) of some standard business applications, e.g. accounting packages, might not handle the date-change correctly.
>
> In the event, although there were some problems (largely affecting personal users rather than businesses), few were serious and most were easily remedied, e.g. by users ensuring systems were upgraded to the latest version of software. Advice to PC users was widely available[xvii] (note: there were many fewer users than there are today) and commonly suppliers – e.g. suppliers of standard business accounting packages – provided their customers with compliant versions at no charge well before the end of 1999.
>
> Despite some genuine initial fears (and media scare stories and commentaries), most who understood the issue did not expect the transition to 2000 to cause important difficulties for PCs and other similar

equipments such as so-called microcomputers and workstations. And that proved to be the case. This was especially true of small or medium sized PC (or microcomputer) reliant businesses where, in most cases, software suppliers had ensured their customers were using compliant software. In any case, problems that occurred after the date change were usually easily remedied.

Embedded chips

In contrast, so-called embedded chips were regarded, from the beginning of the campaign to resolve the Y2K problem, as a serious potential problem, largely because it was hard to quantify what the effect of their failure might be. Such chips are monitoring devices commonly installed at the heart of a wide range of electronic systems such as cash registers, security doors, process controllers, some medical equipment and, in particular, safety critical systems such as devices ensuring that equipments such as lifts (elevators) would not operate unless they had been recently serviced. What, for example, would be the effect of a chip "thinking" that an action – such as routine maintenance – undertaken after the transition from 1999 to 2000 had been taken 100 years ago?

There was particular, and understandable, concern about the possibility of major breakdowns in the telecommunications and utilities industries (power generation, water supply etc.) and, for example, in chemical plants and oil refineries where embedded chips control a vast range of functions and where failure could potentially have far-reaching consequences[xviii].

And some problems did occur. For example, in May 1998, the *Ottawa Citizen* reported that Chrysler closed an assembly plant and turned the clocks to 31 December 1999 to satisfy itself that all would be well at the transition. But it was disappointed: as Robert Eaton, Chrysler's Chairman, is reported as saying, "*We got lots of surprises. Nobody could get out of the plant. The security system absolutely shut down and wouldn't let anybody in or out. And you obviously couldn't have paid people, because the time-clock systems didn't work*." And the UK briefing for Members of Parliament (see page 8 and Reference **xiii**) mentioned another example: at Marks & Spencer (a British retailer) a computer required new canned goods to be discarded because their bar codes indicated sell-by dates of "02" – which it read as meaning 1902, not 2002.

Although initial fears were justified, in the event few problems turned out to be serious. That was largely because key industries – most especially the telecommunications and utilities industries – had taken the threat seriously and put a lot of effort into checking and, where necessary, into remedial[xix] and contingency action (see, for example, Reference **xviii**). Moreover, chip manufacturers advised customers of potential problems and, where necessary, replaced non-compliant with compliant equipments. The Racal Electronics action referred to above (p.6) is an example of this although, as

noted, some credit card swipe machines were nonetheless overlooked – a case of the exception proving the rule. The Otis Elevator Company provides another example of a manufacturer taking responsibility. It was confident that its equipments would not cause any problems but nonetheless worked with customers to allay any fears[xx].

In summary, although PCs and embedded chips were a cause of justified concern, on the whole it proved to be relatively easy for users and equipment manufacturers and suppliers to ensure effective remedial action was taken well before the date-change – this applied in particular, to most small and many medium size businesses that had to do little and experienced few problems.

But that was not true of the essential challenge: overcoming the widespread use of two-digit date references throughout the custom software utilised in large, often interconnected, computer systems: the UK government briefing referred to in page 8 (and see Reference **xiii**) considered that "*about 80% of all mainframe computer systems contain two digit year references in their programs*"). It was that inheritance that was the most serious problem and the one most difficult to eradicate.

4. The Solution

Resolving that challenge turned out to be a massive task[xxi]. And the fact that it was largely successful is a tribute to the outstanding work done by those involved. It wasn't a high tech operation: it didn't require esoteric computer skills. What had to be done is simply stated: find two-digit year references and ensure (e.g. by turning them into four-digit year references) that they would not cause problems in relation to the date change. But achieving that was not simple. And it turned out to hugely expensive.

Although it may sound easy enough in theory, in reality the task was quite exceptionally complex. For example, two-digit date fields were "hidden" within untold millions of lines of computer code, some of it about thirty years old. And as already mentioned, in many cases, the original programmer's notes and source code were long since lost. So those engaged in this thankless task had to drag their way through line after line of dense computer code (commonly written in a variety of programming languages), often without any guide as to what they were looking for or to where they might find it. And often while continuing to run their daily work. They had to examine not only programs and files in current use, but also those that had been archived or stored. And, having found what appeared to be a two-digit year reference, they had to be sure it really was – after all, "55" might mean 1955 or it might refer to a myriad of other possibilities, such as someone's age. Then they had to determine if it really had to be fixed (it might never, for example, relate to the new century) and, if so, how best to fix it. The obvious solution was to simply expand the date field from two to four digits so that, in due course, "1999" would be succeeded by "2000". But it

was not unusual for that to be impossible because of the nature of the code where the date was found. For example, a change might have damaging knock-on effects elsewhere. In such cases, the only solution might be to write or purchase a completely new suite of software. But even that solution could then cause difficulties with other software with which it interfaced within a wider system where, for example, that other software had been fixed by using a different date solution. Moreover "fixing" a system was commonly not enough: it, in turn, had to be able to work seamlessly with third party systems. Thus a bank's trading system would have to be compatible with that of all the other trading systems with which it corresponded – many of them overseas. See, for example, Reference **xi**: a US Federal Reserve statement made to a House Committee in 1998. And, of course, those corresponding banks were probably fixing their own date-change problems. (Also it should not be forgotten that the teams dealing with all this usually had to ensure also that there were no problems with PCs and embedded chips.)

A solution that avoided the common problem of expanding a two-digit field to a four digit-field was a technique called "windowing". This might involve software being reconfigured so that years entered as, say, 00-19 were assumed to represent 2000 to 2019, and years entered as 20-99 to represent 1920 to 1999. But this has serious drawbacks: for example, a system reconfigured in this way wouldn't for example recognise 1919, would have problems interfacing with systems that were differently configured and, above all, would start to fail as 2020 approached: see Section 6 (page 17) and Reference **xxxix** for examples of failures arising around 2010 ("The Y2.01K" problem). Windowing was one of various unsatisfactory Y2K fixes[xxii].

But, although difficult enough, finding and fixing the software wasn't the end of it: once "fixed", the software had to be rigorously tested. And there would inevitably be some failures (there nearly always are with complex software) so, quite often, new fixes had to be devised and retested. Moreover, the testing couldn't of course be confined to that specific software – all those third-party interfaces had to be tested as well. And the testing could itself could cause problems: for example, a serious incident occurred in testing of a US spy satellite system[xxiii].

Overall, it was a massive task: almost every large corporation or government department in the developed world had systems that at least potentially contained critical two-digit date references. The exceptions were usually those that, for one reason or another, had replaced their principal software systems in the 1990s – some in anticipation of the date-change. And even they couldn't always be completely sure they were exempt. Probably the world's biggest ever computer-related undertaking, it was – unsurprisingly – very expensive. Yet, despite the justified fears expressed in 1996, 1997 and 1998, it was largely successful. The world owes a debt of gratitude to those, often relatively junior, IT staff who carried out this quite exceptionally boring and unglamorous undertaking. Yet, not only is that debt largely unrecognised, but the fact that the job was done at all is commonly criticised – even mocked.

5. Criticisms

The efforts of those who carried out the huge, important and largely successful task outlined above have largely gone unrecognised because it's become almost universally accepted that Y2K was, at best, grossly exaggerated and, at worst, was a hoax – a scam, an illusory scare created so that a few computer "experts" could make a lot of money.

Yet a review of the facts, references, evidence, examples, etc. set out above demonstrates plainly enough that the problem was real – and sufficiently serious to warrant urgent remedial action. After all, anyone claiming it was a hoax or was imaginary (and therefore that action was unnecessary) must logically believe one of the following: (1) that two digit date references were not used in early computer software; or (2) that, if they were, they would, without human intervention, somehow be able to cope with the interaction between dates in the twentieth and twenty-first centuries; or (3) that, even if they were unable to cope, the consequences of that inability would be trivial; or (4) that, even if there were some serious consequences, they would be easily rectified after the event. But the evidence set out in this paper and referred to in the material to which I have provided links demonstrates that none of these propositions is valid. I should add that the sceptic must also believe, for example, that "embedded chips" would not have caused any problems – let alone a threat to the continuity of, for example, telecommunication systems and utility provision – and that their replacement was therefore unnecessary and another unwarranted expense. But here too I believe I have produced sufficient evidence to rebut any such belief.

So how is it that Y2K has come to be regarded as a prime example of an unnecessary scare?

Well, when I first heard about it in early 1996 (see *A Personal Note* on page 3), I didn't believe it: it seemed to me it was exceptionally unlikely that, for all its faults, the mighty computer industry could have been so negligent, even foolish, as to have allowed the perpetuation of such a simple, seemingly obvious and, as I was advised, dangerous state of affairs. Surely, if it were true, it would have been sorted out long ago – and well before the advent of the new century? But I was wrong: this absurdity had been allowed to happen; in my view, this was extraordinarily irresponsible. I believe that that was, and still is, an embarrassment to the IT industry. It was also a nuisance: contrary to common belief, there wasn't a lot of money to be made from its solution: the big money was substantially spent internally by organisations utilising their own IT staff. And, in any case, the industry was far more interested in blowing up what became the dotcom bubble. Moreover, the skills acquired in Y2K remediation had only limited future value. Therefore the IT industry was, on the whole, not unhappy that, after 2000, the matter, widely thought to be little more than another false scare, came to be largely forgotten.

Another factor – one that strongly influenced the characterisation of Y2K as an unnecessary scare – was the attitude of the media. Those trying to draw attention to the need for urgent action found initially that it was quite difficult to get the media interested: computer-related stories are widely seen as boring. But, when the media did get involved (largely when the embedded chip concerns surfaced), they began to take the line that it was little more than a scary millenarian fantasy. There were remarkably few serious attempts to understand the issue – problems with complex "legacy" software in large, typically financial, organisations were seen as especially boring. Instead, there was a focus on the possible consequences of failure: advocates of the need for action were constantly pressed to say what might happen if things went wrong and, all too often, only that part of the interview was reported. So legitimate warnings were ignored and replaced by wild prediction: "experts tell us that, at the stroke of midnight on December 31st, planes will fall from the sky" … … "ICBMs will be launched from their silos" … "nuclear plant will go critical" … "there will be riots in the streets", etc. Yet no one who was involved with and understood the problem made any such prediction[xxiv] – indeed most tried to avoid predication altogether. This is evidenced, for example, by the bankers' warnings set out in References **viii**, **ix**, **x** and **xi**. Nonetheless, subsequent commentators continue to use the fact that such "predictions" came to nothing as evidence that the whole exercise was little more than a scare and a colossal waste of money.

Another common assertion is that some businesses and, especially, some countries did little or nothing about Y2K, yet experienced few or no problems. What's usually missing from such claims is any hard evidence. The reality is that, whereas it's true (as explained above at page 11) that most small and many medium size companies had little to be concerned about (and, in any case, most could easily fix, say, a faulty invoicing system after the date-change), remedial action was taken in most countries throughout the world. It was essentially a global, not a national problem. For example, in an interview prior to a meeting of the United Nations International Y2K Cooperation Center in June 1999[xxv] (said at the time to be the largest meeting in the history of the UN), its Director, Bruce McConnell, said,

> *"Its clear that there's a global effort underway. This is demonstrated by the fact that we have over 160 countries coming to the meeting. Virtually all the countries of the world have Y2K programs that are moving forward."* [He added that he was] *"very pleased at the level of activism, energy, and initiative that people are showing and willingness to work outside their own area and pitch in and help out on a group basis."*

Banking, in particular, is intrinsically an international activity – see the statements and advice from the Bank for International Settlements referred to in page 7 and detailed in endnotes viii and ix. Note also that the World Bank produced a "tool kit" re Y2K for developing economies[xxvi]. Moreover, no bank or other financial institution could regard the matter as resolved unless it had been resolved also in overseas banks and by other, not necessarily financial,

organisations with which it had dealings. So banking action had a substantial knock-on effect throughout the world. Similar considerations applied to telecommunications[xxvii] and other essentially international concerns such as meteorology[xxviii] and air traffic control[xxix].

So, contrary to the impression given by much of the media, international Y2K remedial activity was substantial and widespread. Nonetheless, it's true that the problem was considerably more serious in some countries than in others. For example, unsurprisingly, little action was taken – or was necessary – in underdeveloped countries such as Chad, Haiti and Afghanistan where few businesses or services were computerised. But also some more developed economies, such as countries in Eastern Europe, needed to do relatively little. And that, paradoxically perhaps, was because they had introduced digital computing on a substantial scale relatively late; therefore, unlike developed Western economies, they did not rely on systems incorporating software originally developed in the 1960s and 1970s. And, in any case, countries that were less technologically developed and had simpler infrastructure had less to go wrong and any problems were much easier to resolve. For example, many developing countries' telecommunications were controlled by analogue systems (with gauges) not digital systems (with readouts) and thus were not at risk[xxx]. Another factor affecting the amount of effort that was necessary is, again paradoxically, that businesses and countries that started late were able to take advantage of the experience gained and lessons learned by those that preceded them: tools, shortcuts and efficient methods of working had been developed, fears that turned out to be unnecessary had been thoroughly investigated and eliminated (embedded chips in lifts (elevators) are a good example – see Reference **xx**) and uncertainties had been tested and resolved. Such research and information was, on the whole, made freely available. It's considerations such as these that lie behind the – essentially false[xxxi] – claims that little (or even nothing) was done in some developed Western economies such as Italy and Spain.

Finally the cost – was it unnecessarily large? Well, certainly it was extremely expensive: it's been estimated that between $300 and $500 billion was spent worldwide. Was it really necessary to spend so much? Well, in one sense, I suppose the answer is no: as noted above (page 13), the IT industry had neglected or simply forgotten about something that should have been dealt with far sooner. Had that happened, the task would undoubtedly have been easier and less expensive.

Another factor affecting cost was that nothing like this had ever been done before. Therefore, those involved in remediation (particularly at the beginning of the exercise) had no established experience or expertise on which to build and, unsurprisingly, wrong decisions were made, attempts at remediation proved inadequate and blind alleys were investigated. That this was unnecessary can only be contended with the benefit of hindsight: it's obviously true that, given the experience and skills we now have, a similar job today would cost rather less.

In any case, the cost per organisation was not so great. For example, in the UK, British Telecom spent about £300 million – as did an average clearing bank. The top UK businesses (the FTSE 100) averaged around £60 million. These sums, especially when spread over 2 to 3 years, were small percentages of those businesses' IT budgets. And, as well as fixing the Y2K problem, the exercise brought some unexpected benefits: see Section 7 (p.18).

But an example of failure referred to in Section 6 (p.17) puts the issue of cost into sharp focus: see the section's final paragraph on page 18. Note the extraordinary cost of fixing just one relatively small failure (see Reference **xlii**). If it cost £270 million to put that right, how can the £300m million spent by the average UK bank to avoid the catastrophe of multiple failures possibly be described as having been wasted?

Fears that things might go very badly wrong were understandable even if they turned out to have been over cautious or exaggerated. After all, it was hard to believe that most organisations would get it right, that nearly every programme would be finished on time, that nothing of significance would be missed – anywhere throughout the developed and developing world. That didn't fit with normal experience of big projects. So maybe it was (just about) understandable that some journalists, media commentators, prominent politicians and others (usually who had made little attempt to understand the issue) made predictions of disaster that went well beyond legitimate warning about the probable consequence of failure to act.

And it's easy to be wise in 2011. When the pioneers started their Y2K programmes in 1995 and 1996, very little had been done: so their fears were then fully justified. Arguably greatest was the unknown: no one really understood our dependence on computers. Or the interdependencies between computer systems, both within and external to organisations. Or the impact on wider society if things went wrong. The risk that they might go badly wrong was clear and valid: it would have been irresponsible to ignore it. Even some overkill was arguably prudent: for example, senior Russian and American military leaders established the Center for Year 2000 Strategic Stability[xxxii] to ensure that no accidental missile launchings occurred the transition. Was this really necessary? Perhaps not – yet those involved presumably considered it a sensible precaution. Who can say they were wrong? It's significant that no insurance company, whose essential business is the assessment of risk, would provide Y2K cover except under near impossible conditions. And none of today's "it was a scam" experts were then advising large organisations that action was unnecessary. Senior people who examined the risk decided that the job had to be done. I know of none that made a deliberate decision to ignore it. Uncertainties about the outcome continued right to the end: I know of no organisation with a major programme that scaled it back as it learned more about the nature of the problem or that felt it had wasted its time or money; or subsequently felt it would have done just as well with less expenditure.

Nonetheless, in retrospect, I think it's probable that more had been achieved by early 1999 than many appreciated. Some did. For example, Peter de Jager – who had taken a lead the early 1990s in warning about the dangers of Y2K[xxxiii] and who had issued some of the most serious warnings about the issue – certainly thought the task was essentially done by early 1999. In January, he claimed, "*We've finally broken the back of the Y2K problem*"[xxxiv]. (And see Reference **xxiv**.) I thought then that was being rather optimistic[xxxv]. However, I considered an EU assessment published in December 1999[xxxvi] to be broadly accurate. It stated:

> ***Countries and sectors throughout the EU now report that their preparations are essentially complete****, and that the rigorous contingency plans which they have established to cope with exceptional events have been tested and reinforced to cope with possible Y2K problems. They consider themselves to be ready and expect no material disruption to their operations. Their confidence is supported by the unprecedented degree of industry and private-public collaboration which has taken place during 1999 to address this issue. Indeed, many believe that their ability to detect and respond to problems which may occur at year end is now greater than at any other time.* [Original emphasis.]

In the event, of course, the worst fears were not realised and although, as noted elsewhere in this paper, there were various failures throughout the world (others included, for example, failures at nuclear plants and telecommunications centres in Japan[xxxvii] and (one that may interest Ms Laframboise) an error that, when corrected, established that 1934, not 1998, was the hottest year on record in the United States[xxxviii]), most were fixed without serious trouble and without fanfare. The latter is hardly surprising: in view of all the prior publicity, those who experienced problems were reluctant to report it.

So the outcome was relatively benign. But it might well not have been – and those who pioneered solutions, got on with it and did the boring job of fixing the problem took the only rational and responsible course. None, I believe, regrets it today.

6. The Year 2010 (or Y2.01K) problem

As noted in relation to Leap Years (page 9), the century date change was not the only date-related problem for computers. A number of problems occurred in 2010[xxxix] (and more may occur in 2020), the most serious affecting banking systems.

Why? Well, there were various possibilities. One is confusion between hexadecimal number encoding and binary encoding – this affected some mobile phones and so-called "EFTPOS" terminals (processing debit cards at point of sale). Another arose as a result of "windowing" (see page 12) whereby two digits

were retained but the code adjusted so that numbers up to and including, for example, 10 were regarded as being in the twenty-first century and numbers higher than 10 in the twentieth century[xl]; thus, for example, "09" would be read as "2009" and "12" as "1912". That would be fine so long as a more permanent fix was devised well before 2010. But, in some cases, that seems not to have happened. As IBM said in referring to such a solution[xli]:

> … *keep in mind that when the new value comes around in the future, it will need to be changed again. Also consider that the higher the value, the less it can handle older dates, such as for date-of-birth or other back dates where 19 is desired when the century control year is exceeded. … Note: The only permanent solution is to provide the actual century (CC) within the input …*

The most important failures were in Germany, where about 30 million bankcards had problems, and in Belgium, where Citibank's "digipass" customer identification chips stopped working.

The German failure was reported[xlii] to have cost €300m ($420m). That's hugely significant: it puts into perspective claims that money spent on Y2K remediation was wasted. That one relatively minor failure could cost so much to fix after it had failed, emphasises the massive advantage of spending comparatively modest sums (see pages 15 and 16) to avoid the possibility of countless such failures.

7. Some Observations

Y2K remediation brought unexpected benefits

Many organisations, when faced with Y2K, decided that, as they were going to replace their old legacy systems within the next few years, they might as well replace them now, thereby avoiding many of the date-change difficulties. This accounts for much of the overall expenditure. One of the consequences of this was that there appear to have been marked productivity gains in the US and elsewhere as better, more productive IT systems were brought on stream, existing systems were much improved and improved methods of working were established – all a direct result of Y2K remediation[xliii].

One important example was that Y2K obliged organisations to sharpen up their disaster recovery and contingency planning[xliv], ensuring that, if there were failures, the business would be able to cope. (This is another reason why fewer problems were reported than many feared.) Today such "business continuity planning" is standard business practice – particularly where new IT systems are being introduced. That was not so prior to 2000.

It's been claimed, for example, that New York City's efficient computer systems recovery after 9/11 was a result of system redundancies developed to combat Y2K[xlv]. And here's a comment from an IMF report[xlvi] on the post 9/11 resilience of the US dollar Payment System:

> *Contingency planning advanced substantially in the run-up to the century date change (Y2K). Y2K planning incorporated the involvement of senior management and boards of directors and asked business managers to consider how they would operate their businesses (and not just their back offices) in the event of a Y2K-related disruption. In a letter to financial institutions in March 2000, federal bank regulators noted that detailed contingency plans developed for Y2K specified the minimum level of output and services necessary for each major business process. They also noted that simulated operational failures and scenario building helped reduce the time needed to respond to operational problems and improved decision making and internal communication.*
>
> *External communication was a special focus of Y2K planning and included maintaining lists of contact numbers for financial institutions, regulators, and key infrastructure providers.*

Another important Y2K lesson was that the resolution of computer problems required a high level of cooperation – between businesses and between nations. For example the Bank for International Settlements (see References **ix** and **x**) established the Joint Year 2000 Council[xlvii] – an ad hoc group that comprised market regulators and insurance regulators as well as bank regulators – covering over 200 major financial institutions from around the world. This level of cooperation had never happened before.

Project management

One important lesson, however, seems not to have been learned. The remarkable success of Y2K remediation demonstrates that a very large computer project (and they don't get much larger than this) can be completed successfully and on time – yet the reasons for that success have not been widely understood and put into practice. It was, of course, a network of interrelated projects – but pretty well each of them was characterised by a few simple features:

(1) It had senior management support from the outset.

(2) It had a clearly defined and widely understood objective.

(3) It had a fixed, unalterable end-date.

(4) What had to be done was clearly established at an early stage.

(5) The need for the project was widely understood and accepted.

(6) Communication, internal and external, was prompt and comprehensive.

If more projects were based on these features – and there's no reason why they should not be – there would, I believe, be far fewer disastrous IT project failures.

Y2K caused the importance of computers to be better understood

The computer date-change problem demonstrated how poorly we understood the potential impact of a major computer upset and, in particular, the importance of computers to business and society. The advent of the so-called "digital age" brings huge new threats as well as opportunities: we are becoming increasingly dependent on the smooth running of IT systems. Partly as a result of Y2K remediation efforts, that dependency – and especially information security and the reality that it is far more than an IT matter – is much better (although probably still inadequately) understood.

8. Conclusion

Y2K related problems occurred widely over several years. Their effect was local and - with some unfortunate exceptions - relatively unimportant. In particular, because the vast majority of potential problems were fixed, there was no example of the catastrophic knock-on effect that some had feared and of which the Governor of the Bank of England had warned (see p.7). That there was not is largely because his and other such warnings were heeded and acted upon. Anyone who confuses such warnings with predictions, and failed predictions at that, or with scaremongering or who regards Y2K as a hoax or believes efforts to resolve it were a waste of money, hasn't tried to understand what was, in fact, a bizarre, unnecessary but real and seriously worrying problem, resolved only because of the massive effort deployed throughout the world.

Notes and references

[i] http://nofrakkingconsensus.com/2011/08/09/the-y2k-scare-the-media-climate-change/

[ii] http://searches2.rootsweb.com/th/read/GENCMP/1999-02/0917875049

[iii] http://reason.com/archives/1999/07/01/grave-problem

[iv] From the testimony by Alan Greenspan, ex-Chairman of the Federal Reserve before the Senate Banking Committee, February 25, 1998, ISBN 9780160579974

[v] See this and similar examples here: ftp://csl.sri.com/pub/users/neumann/cal.ps (p 2). (This article provides a useful review of date-related computer problems.)

[vi] http://airwolf.lmtonline.com/news/archive/1230/pagea15.pdf

[vii] http://news.bbc.co.uk/1/hi/health/1541557.stm

[viii] http://www.bankofengland.co.uk/publications/news/1998/024.htm

[ix] http://www.bis.org/press/p970908a.htm

[x] http://www.bis.org/publ/bcbs31.htm

[xi] http://www.bis.org/review/r980626d.pdf

[xii] http://www.hhs.gov/asl/testify/t980507a.html

[xiii] http://www.parliament.uk/documents/post/pn089.pdf

[xiv] http://www.cplusplus.com/forum/beginner/23306/

[xv] For example: http://www.greenspun.com/bboard/q-and-a-fetch-msg.tcl?msg_id=004KIW

[xvi] http://www.ianchadwick.com/essays/madness.html

[xvii] See, for example, http://y2k.berkeley.edu/computers/fixpcs/ and http://www.fastcompany.com/magazine/24/powertools.html

[xviii] A good example was railway control systems: http://www.indianexpress.com/Storyold/117946/ (This story also demonstrates the detailed work that was undertaken in India.)

[xix] See for example this article re the need to replace older systems: http://www.prosoft-technology.com/content/view/full/3780

[xx] http://articles.courant.com/1999-10-17/business/9910180929_1_elevator-companies-y2k-problem-building-systems

[xxi] A review of the size and technical complexity of this task is provided by *The Year 2000 Software Problem* by Capers Jones – published by ACM Press in 1998.

[xxii] http://www.itbusinessedge.com/cm/blogs/lawson/revenge-of-the-y2k-bug--some-fixes-strike-back/?cs=40434

[xxiii] http://web.caller.com/2000/january/14/today/national/5810.html (Note incidentally the vast cost ($3.6 billion) reported here to have been incurred in fixing the Pentagon's computer systems.)

[xxiv] Peter de Jager, who as noted below (**xxxiii** and **xxxiv**) was a prominent advocate for action at an early stage and who became convinced that the problem was largely fixed by early 1999, demonstrated his confidence by booking a flight from Chicago to London some months before the end of 1999 and was in the air at the date change: http://www.theglobeandmail.com/archives/former-prophet-of-doom-flies-to-london-to-prove-y2k-defeated/article577224/singlepage/
The reality of course was that, were any aircraft at risk of falling from the sky, it wouldn't be in the sky. Manufacturers and airlines carried out exhaustive tests to ensure all was well: e.g. http://news.bbc.co.uk/1/hi/sci/tech/523239.stm (It's interesting to note from this that China had especial Y2K difficulties because of its use of pirated software.)

[xxv] http://www.greenspun.com/bboard/q-and-a-fetch-msg.tcl?msg_id=000y7g

[xxvi] http://allafrica.com/stories/199901120152.html

[xxvii] http://www.itu.int/itunews/issue/1999/03/y2k.html

[xxviii] http://www.wmo.int/pages/prog/www/reports/Y2K-Reading.html

[xxix] http://www.rense.com/politics5/y2kyeahsure.htm

[xxx] African airports, for example, illustrated both extremes: on the one hand were newly built airports such as Kenya's Eldoret International, equipped with new and therefore fully compliant systems, while on the other were older-generation airports where semi-automatic and manual systems remain in widespread use. In both cases, this made them largely immune to Y2K problems: http://www.flightglobal.com/articles/1999/11/24/58790/catching-african-bugs.html

[xxxi] See Reference **xxxvi** (below) re an EU report that notes (in September 1999): "*Countries and sectors throughout the EU now report that their preparations are essentially complete…"* and this report (*Reuters* and others) detailing Italian readiness: http://www.greenspun.com/bboard/q-and-a-fetch-msg.tcl?msg_id=0029SZ
These reports (re banking in South Africa, nuclear power in Korea and preparations in Russia) are also interesting in this context and demonstrate the absurdity of claims that nothing was done in such countries:
http://articles.chicagotribune.com/1999-11-30/news/9912010110_1_y2k-bug-tito-mboweni-readiness
https://www.oecd-nea.org/nsd/docs/1999/cnra-r99-3/1-09.pdf
http://news.bbc.co.uk/1/hi/sci/tech/294912.stm
(Likewise, note Reference **xviii** above re India)

[xxxii] http://www.defense.gov/advisories/advisory.aspx?advisoryid=1366

[xxxiii] http://www.nhne.org/news/NewsArticlesArchive/tabid/400/articleType/ArticleView/articleId/6450/language/en-US/Peter-De-Jager-Receives-Award-For-Helping-Avert-Y2K.aspx Note that de Jager's *Doomsday 2000* article (http://www.window95.com/y2k/article/dejager.html) was a wake up call written in September 1993 – over 2 years before the British Government (for example) has even heard of the problem. And it was 5 years before he was convinced the problem was essentially fixed: see **xxxiv** (below) and **xxiv** (above).

[xxxiv] http://greenspun.com/bboard/q-and-a-fetch-msg.tcl?msg_id=000Yi9

[xxxv] Although, by March, I was rather more optimistic: http://news.bbc.co.uk/1/hi/sci/tech/296713.stm

[xxxvi] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:1999:0651:FIN:EN:PDF

[xxxvii] http://archives.cnn.com/2000/TECH/computing/01/03/japan.nukes.y2k.idg/

[xxxviii] http://wattsupwiththat.com/2007/08/08/1998-no-longer-the-hottest-year-on-record-in-usa/

[xxxix] http://www.cairns.com.au/article/2010/01/02/85845_local-business-news.html
http://www.crn.com.au/News/163864,bank-of-queensland-hit-by-y201k-glitch.aspx
http://www.tuaw.com/2009/03/04/wwnc-09-official-announcements-and-the-2010-bug/
http://www.basissap.com/2010/01/sap-spool-issue-affects-all-releases/
http://www.thelocal.de/sci-tech/20100104-24353.html

[xl] See the last paragraph here: http://www.theregister.co.uk/2010/01/05/symantec_y2k10_bug/

[xli] https://www-304.ibm.com/support/docview.wss?uid=swg21419889

[xlii] http://www.guardian.co.uk/world/2010/jan/06/2010-bug-millions-germans

[xliii] http://archive.healthmgttech.com/archives/h0100managedcare.htm
http://research.cs.queensu.ca/~cordy/Papers/IWPC03_Keynote.pdf

[xliv] http://www.projectauditors.com/Papers/BCP/BCP.html

[xlv] http://web.mit.edu/newsoffice/2002/terror-1120.html

[xlvi] http://www.imf.org/external/pubs/ft/fandd/2002/03/cumming.htm

[xlvii] http://www.bis.org/press/p990108.htm