GRESHAM COLLEGE

4 APRIL 2017

# WHAT REALLY HAPPENED IN Y2K?

PROFESSOR MARTYN THOMAS

## Introduction

During the 1990s, the newspapers contained a growing number of stories and warnings about the Century Date Problem or "Millennium Bug" that would cause computers to fail at midnight on 1 January 2000. There were fears that there would be a power blackout and failure of the water supply, and that computer records of bank accounts would disappear, simultaneously wiping out both savings and debts. Governments worried about safety and security, auditors questioned whether they could approve company accounts if companies might collapse, airlines and passengers did not know whether planes would fail in flight or whether airports would be able to remain open and radars and air traffic control would continue to function.

Software errors were a familiar experience, of course. What made this one different is that it could potentially cause very many systems to fail at the same time.

Committees were formed and issued reports and advice. Auditors insisted that companies obtained assurances about their essential systems, or replaced them. New consultancy companies were created (and many existing firms created new service lines) to investigate systems and equipment, to advise on Y2K risks, and to manage Y2K projects. Old programmers were brought out of retirement to look again at the systems they and their colleagues had written years or decades before. For a while it seemed that the launch of the European common currency, the Euro, would have to be delayed from the planned date of January 1st 1999.

As the new millennium approached there was growing anxiety in boardrooms, among technical staff and across many parts of society in the UK and other advanced economies. Despite the huge expenditure and efforts, it was difficult to be sure that nothing vital had been overlooked.

Then nothing happened, or so it seemed, and the feeling grew that the whole thing had been a myth or a scam invented by rapacious consultants and supported by manufacturers who wanted to compel their customers to throw away perfectly good equipment and buy the latest version.

I led the Y2K services internationally for Deloitte & Touche Consulting Group for several years in the 1990s and later acted as the ultimate auditor of the Y2K programme for the UK National Air Traffic Services, NATS, so I was well placed to see what happened before, during and after the first of January 2000. This lecture explains the nature of the Millennium Bug, describes some of the unprofessional behaviour that made the problems so much worse and some of the heroics that made the consequences so much better than they would otherwise have been. In the final section, I shall show why the risks of catastrophic failure are actually much higher now than they were in 1999 and highlight some of the lessons that have not yet been learnt.

## What was the Millennium Bug?

The root of the problem was that it was common to leave out the century when referring to a year. Just as most people refer to "the Twenties" or "The Sixties", computer systems commonly just assumed that all dates were in the twentieth century. There were actually *five* problems related to Y2K that had to be considered when making sure that computer systems would handle dates in 2000 properly, and there were some additional issues that increased the required work.

### 1. Two-digit years

The biggest problem was that many systems and a huge amount of existing data used and stored dates in the format YYMMDD, with two digits each for the year, month and day but omitting the century (In the COBOL programming language, commonly used for commercial applications, the dates would usually be in Binary-Coded Decimal BCD format which allows convenient arithmetic). Dates are often sorted into date order, or compared to see which date is earlier (for example, to check whether an expiry date has passed) or used to calculate the number of days between two dates (for example, to work out when a medicine or a security pass should expire). It will be obvious that by omitting the century and only using two digits for the year, these operations give the wrong results when the earlier date is in the 20th century and the later date in the 21st century.

Using two digit years was a sensible decision in the 1960s, when data was often on 80-column punched cards and when computer storage was very expensive. The convention persisted, partly for compatibility but mainly because programmers did not consider that their programs might still be in use by the time that this could become a problem. Indeed, Alan Greenspan, former Governor of the US Federal Reserve Bank reportedly told congress in 1998[i].

> *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.*

Two digit years are still common, as you can see from the front of your bank or credit cards.

Systems that could fail included those that display the current date, calculate someone's age from their date of birth, check expiry dates, generate dated printouts, carry out specific functions on specific days, or store information for averaging or trending purposes—a wide range of systems, such as personal computers, surveillance equipment, lighting systems, entry systems, barcode systems, clock-in machines, vending machines, switchboards, safes and time locks, lifts [they maintain dates for maintenance and alarms], faxes, vehicles, process monitoring systems, and production line equipment.

The risks were slowly recognised to be serious and widespread. A UK Government report said that in 1996, the number of embedded systems distributed worldwide was around 7 billion and that according to research conducted in 1997, around 5% of embedded systems were found to fail millennium compliance tests. More sophisticated embedded systems had failure rates of between 50% and 80%. In manufacturing environments, an overall failure rate of around 15% was typical[ii].

These were real business risks. For example, when I was assisting Heathrow Airport with their Y2K programme they told me that they would have to close the airport if their lifts failed.

**2. Real-time clocks in PCs and PC Software**

The second biggest problem was that many real-time clocks, including the clocks in some PCs would not handle the roll-over from 1999 to 2000 correctly.

The IBM PC (model 5150) and its 1983 successor the extended Technology PC[iii] had a system timer based on a crystal oscillator that also drove the central processor chip[iv]. The oscillator ran at 14.31818 MHz, which was divided by 3 to get the 4.77 MHz clock that the 8088 CPU needed. Dividing 4.77 MHz by 4 provided a 1.19318 MHz source for the system timer[v]. These timers generate an interrupt every 65536 pulses, which equates to 18.206 interrupts per second and allowed the operating system (DOS at the time) to update the date and time (from the value that had to be set manually when the PC was booted up)[vi].

There was originally no way to maintain the date and time whilst the PC was switched off. This was inconvenient, so a second clock was added when the PC XT was replaced by the faster PC AT (Intel 80286 16-bit CPU). This Real Time Clock (RTC) was a battery-driven CMOS chip with a small amount of memory that could be read by the BIOS during the PC boot sequence and used to initialise DOS with the date and time. The RTC chip only updated the two digit year; there was a separate century location in the memory that was normally set to 19 but that could be set manually to 20.

IBM PCs and the many PC-compatible computers built by other companies used a variety of BIOS firmware which handled the RTC in different ways when the century moved from 19 to 20. On many of the later BIOSs the roll-over would be handled correctly; some introduced a test that if the year was 00 to 79 the century would be set to 20, otherwise it would be set to 19; some went wrong in a variety of ways.

DOS was written with the assumption that the date would lie between 1980 and 2099; if it received a date outside this range from the BIOS, it reset the date to 1 April 1980. Quite a lot of PCs showed this behaviour when tested; on the first boot after Y2K the system date was 1-4-1980 and if this was not corrected then new files would get the wrong creation date, new data would get wrong dates, and the new data would typically be sorted to the front of a date-ordered file and vanish from subsequent processing. On the next boot, the same date would be set again, causing multiple entries with identical dates and further problems, especially with backups.

Many of these PCs only needed the date to be set correctly once in the new century and they would work correctly after that (at least until the battery for the RTC had to be replaced or there was a hardware problem). The date could be set manually or by software.

Some versions of the BIOS created bigger problems. For example, the 1994/95 Award v4.50 BIOS[vii] had been designed so that it did not handle dates before 1994 or after 1999 (I have been unable to find out why). If it found the date outside this range during the boot sequence, it reset the clock to a date in 1994 and this could not be overridden by software. The result was that computers with this BIOS either had to replace the BIOS (which was often not straightforward) or to replace the PC.

Many PC software products were not Y2K compliant, including Microsoft Windows 95.

It should be remembered that rack-mounted PCs were widely used as controllers for industrial equipment, so this Y2K problem went far beyond home and office computing.

A further complication arose if a PC had interrupts disabled at the instant when the new century started (which could be the case if some time-critical process was running and the program was in DOS "real mode") because disabling interrupts stopped the DOS clock from updating and when interrupts were re-enabled the BIOS had to read the RTC and, if the century had been crossed meanwhile, the date would again be reset to 1 April 1980.

**3. Real time clocks in Programmable Logic Controllers (PLCs)**

Programmable Logic Controllers replaced the hard-wired logic that was used in control applications throughout all industrial sectors up to the early 1980s. They were typically programmed in a language called *ladder logic*. PLCs are often part of a larger system with many input and output devices that contain their own logic and may have their own timing devices to time-stamp data. PLCs are often connected to Supervisory Control and Data Acquisition (SCADA) systems that monitor a number of PLCs and build trend reports and track the variance of specific points over time. Time and date stamping of this data is essential. PLC's have been used in applications such as building management systems, process control, traffic control, security and safety shutdown systems. PLC firmware mainly used two-digit year fields and therefore had Y2K issues; PLCs raised extra Y2K difficulties because operating systems were usually proprietary and monitored using PC-based platforms, the intelligent I/O devices sometimes took their time and date from their own internal Real Time Clock and the ladder logic software was often undocumented.

### 4. The Leap Year

The year 2000 was a leap year, which some programmers had overlooked. Since 1582, a year has been a leap year if it divided by 4 *except* if it divided by 100 – *except* that if it divides by 400 it is again a leap year[viii]. 2000 was the first century leap year since 1600; there will not be another until the year 2400. So programmers who wrongly thought that there was always a leap year every 4 years got 2000 right, whereas some slightly-better-informed programmers who knew the 100 year rule but overlooked the 400 year rule got it wrong, though they may get their revenge in 2100.

Because the RTC was blind to the century and there was no reason for a BIOS to be programmed to worry about 2100, we never found a PC that had a problem with the leap year calculation, though some companies insisted that this should be checked.

### 5. Special uses of dates

Programmers use a lot of tricks to implement their programs and one popular one was the use of special dates to indicate the final record in a file. Quite often, the year **99** was used, because it ensured that the record sorted to the end of the file and it was so far in the future when the program was written that the programmer assumed that it would never be a problem – or that they would have moved to another job before it was. This was a Y2K problem that would start to cause problems in 1999 at the latest and that needed to be corrected as part of the wider Y2K projects. Some programmers used the year **00** for special purposes, such as marking records that should be ignored – perhaps because they were invalid or as a way of moving records to the front of a file once they had been processed.

### 6. Other issues

Quite a lot of systems had the century built into data or in a program as a constant **19**. Many organisations had **19** pre-printed on forms, contracts, purchase orders, invoices and cheques. All these forms and pro-forma data fields had to be changed for the new century – often just for reasons of clarity and professional appearance but sometimes because giving the wrong date could have legal consequences such as making a cheque or other legal document invalid. This added to the effort and costs of Y2K projects and increased the pressure on time and resources.

There were even many cases reported of 19—having been carved into family headstones and causing difficulties when an elderly relative unexpectedly survived into the new century.

## Recognition of the Y2K Problem

Some early failures helped to raise the alarm. It was said that sometime in the late 1980s the UK supermarket Marks and Spencer rejected a delivery of tinned meat because the stock control system detected that it was nearly 90 years old. The expiry date, in 2000, was read as 1900. In 1992, 104 year old Mary Bandar of Winona, USA was invited to join an infant class for four-year-old children because she was born in "88".

It took some time for the seriousness of the threat to be widely recognised but more failures in the first half of the 1990s and a large number of speeches and articles by academic and industrial IT experts gradually convinced leading companies, regulators and Governments that the problem had to be taken very seriously.

Government and United Nations committees were formed to generate awareness and to provide information and guidance. In the UK, a briefing for Parliamentarians[ix] in December 1996 said

> *The potential implications of the millennium date, the earlier failure of the main computer and software companies to compensate for it and the lack of awareness and preparation in industry, raise concerns in government circles worldwide. In the USA, the problem gained prominence in early 1995, and many organizations have Year 2000 compliance programmes in hand (e.g. New York Stock Exchange has completed its project, after 7 years of effort at a cost of US$30M). The House of Representatives Science Committee has an on-going inquiry into millennium compliance. The latest findings suggest that many US Government systems (including NASA!) may not meet the Year 2000 deadline.*
>
> *In the UK, a survey of 535 public and private sector organisations (May 1995) found that while 70% of IT managers were fully aware of the problem, only 15% of senior managers were, and only 8% of organizations had conducted a full audit. Awareness began to grow in 1996 and there was an Adjournment Debate in the House of Commons in June in which the Minister for Science and Technology urged all IT users to tackle the problem. In July 1996 the DTI, CBI and the Computing Services and Software Association (CSSA) co-sponsored Taskforce 2000 to raise awareness in the private sector. As far as Government IT systems are concerned, in June 1996, the Deputy Prime Minister wrote to all Departments to establish their current positions. The Central Information Technology Unit (CITU) of the Cabinet Office has contracted the Central Communications and Telecommunications Agency (CCTA) to coordinate activities and CCTA has formed a 'Year 2000 Public Sector Group' to support departmental activities and provide a forum for sharing solutions.*

By 1997 most audit firms had written to their audit clients advising that they would need to ensure that all their business critical systems and equipment had been certified to work throughout Y2K, so that the auditors would be able to approve the accounts on a *continuing business basis*. This drove a flurry of activity to get assurances from suppliers and to correct or replace systems and equipment. Inevitably many suppliers saw this as a business opportunity and forced their customers to upgrade unnecessarily, in some cases refusing to certify as Y2K compliant equipment that they knew had never contained any form of clock.

In late 1996 the UK Conservative Government prompted the setting up of the agency Taskforce 2000 (a public/private sector body) to raise awareness of the problem. After the 1997 election, the incoming Labour decided that a wholly government-backed and better funded body was necessary, setting up Action 2000 with a substantial budget. Thereafter the two agencies continued in parallel. *Action 2000* had an initial budget of £1m; this was later increased to £17m. Many other countries created similar Y2K programmes, some with financial support from the World Bank. In the 1998 budget the UK Government allocated £70 million to help small and medium sized businesses develop their IT skills to assess and fix systems which will be affected by the bug. European national Y2K programmes included *Millennium Platform* (Netherlands), *Year 2000 Action Plan* (Denmark), *Forum 2000* (Belgium), and *IT Commission* (Switzerland). Recognising the seriousness of the Y2K threat, the issue was on the formal agenda of the G8 summit in Birmingham in May 1998[x].

In June 1998, the building company Bovis wrote[xi] to the owners of 870 buildings it had built since 1984 warning them that integral building systems – ranging from ventilation and heating to intruder alarms and connections to the electricity supply network – might fail. This was because many building systems use microprocessors to identify dates for switching machinery on and off and to alert maintenance staff to the need for servicing.

These actions had a big impact. By October 1998 Governor Edward W. Kelley, Jr. of the Federal Reserve Bank described the huge efforts that had been undertaken and said that he was cautiously optimistic that enough had been done for the US to prevent major disruption[xii].

The *International Y2K Cooperation Center* (IY2KCC) was created in February, 1999 under the auspices of the United Nations, with funding from the World Bank. Their mission was to promote increased strategic cooperation and action among governments, peoples, and the private sector to minimize adverse Y2K effects on the global society and economy[xiii].

In 1999, the International Y2K Cooperation Center reported[xiv]:

> *Many PLC systems will be unaffected by the Year 2000, but if they are affected, they tend to fail in a catastrophic manner. This is due to the date reference being central to the operation of the PLC and memory limitations, there being very little in the way of failure control programming. Redundancy is no help here, as the redundant system is of a similar design and will be impacted just like the primary system. Failures of PLCs used as intelligent I/O devices could cause erratic behavior of the machinery being controlled and make trending/historical records suspect.*
>
> *If wrong data is fed back into a newly-renovated business system, the corporate database could become corrupted with unknown and far-reaching implications to the organization's viability. Such an impact could be immediately catastrophic or could grow in an organization's vital areas until it becomes catastrophic.*

The IY2KCC said that some incidents had already occurred by March 1999:

> *There have been a few significant failures reported, and a large number of devices that have failed in testing. The testing failures have indicated that if many of these devices are left unremediated, catastrophic shutdowns or uncontrolled operation of equipment will occur sometime in 2000. Some examples are as follows:*
>
> **An aluminum plant in Western Australia** *had certain critical PLC-based control systems which performed date-related tasks (running certain procedures after a certain number of days, etc.). It calculated the date using the following formula: Today = 366 – Number.of.Days.Already.Counted*
>
> *On December 31, 1996, the system subtracted day 366 (1996 being a leap year) from 366 and received a result of 0 when it was programmed to only get a positive number. This resulted in an immediate shutdown of the entire system and caused damage to operating plant equipment and an extended production delay. The same software was operating a sister plant in Tasmania. When they were informed of the New Zealand plant shutdown, they were able to operate a manual override and stop the same event from happening there. This saved millions of dollars in losses.*
>
> **Kraft** *accomplished tests on all of its business and manufacturing systems. In the manufacturing systems, they tested 832 PLCs and found that 10% had some type of date-related functionality. Of these 83, 4% had date-related problems functioning in 2000. The problems were in a subtraction routine and included some leap day problems. All of Kraft's PLCs are listed as mission critical because of quality control and safety. If they fail or produce errors Kraft must shut down the specific process because all of their processes cannot be allowed to go on in an unmonitored condition to insure health and safety. For a production line to continue to produce food that is either under or over cooked is unacceptable. If you don't know for sure, the food has to be destroyed.*
>
> **Chrysler Corp.** *shut down its Sterling Heights Assembly Plant in mid-1997 and turned the plant's clocks to December 31, 1999. "We got a lot of surprises. Nobody could get in or out of the plant. The security system absolutely shut down and wouldn't let anyone in or out. And you couldn't have paid people because the time clock systems didn't work." noted Chrysler Chairman Robert Eaton.*

Failures occurred throughout the 1990s as systems first encountered security passes, library cards and other documents that had expiry dates beyond 2000. Even as late as December 1999, a Racal credit card system failed and caused problems[xv]

> *LONDON — A Y2K-triggered failure in credit card swipe machines caused frustrating delays for thousands of retailers and customers trying to ring up purchases across Britain on Wednesday. The machines, manufactured by Racal Electronics and supplied by HSBC, one of Britain's largest four banks, improperly rejected credit cards because of a failure to recognize the year 2000, a bank*

*spokeswoman said. Retailers claimed they have so far lost $5 million in sales due to the problem, are reportedly threatening to bring a class action lawsuit against the bank.*

*The glitch occurred because some of the bank's new swipe card terminals are programmed to look ahead four working days in processing merchant transactions to ensure they are registered within that time period.*

## What needed to be done?

In January 1997, BSI published a standard PD2000-1[xvi] with four rules that specified how software should handle dates.

- **Rule 1: No value for current date will cause any interruption in operation.**
- **Rule 2: Date-based functionality must behave consistently for dates prior to, during and after year 2000.**
- **Rule 3: In all interfaces and data storage, the century in any date must be specified either explicitly or by unambiguous algorithms or inferencing rules.**
- **Rule 4: Year 2000 must be recognized as a leap year.**

These rules were increasingly referenced in contracts and audit standards. The challenge was ensuring that every system complied.

Year 2000 projects had a standard structure – inventory, evaluation, remediation, implementation and asset management. For many companies, this was the largest IT project they had ever undertaken (one oil company had more than 40,000 microprocessors on its offshore oil platforms). It also had an immovable deadline of 31 December 1999, though most companies set an internal goal of December 1998 for safety (though some systems that had a longer look-ahead in their data or algorithms could still fail earlier that that).

A complete inventory of all equipment and systems was vital and proved a problem for many organisations. Poor control of software meant that many companies did not have the latest source code for all their business critical systems. I addressed one meeting of 60 departmental IT directors of a FTSE 100 group, and discovered that none of them was confident that they could rebuild their key systems without introducing errors. A typical explanation was that many enhancements and error corrections had been made by programming staff using versions of the software that they held in their own directories, without documentation, and that these staff had now left. (There was very high staff turnover in the late 1990s as experienced staff were scarce and in high demand, so salaries for new positions kept increasing).

Remediation work involved finding where problems existed, by scanning the software source or by testing or both, and then designing and implementing suitable changes to the software and data. Several solutions were applied to legacy systems; the main ones were:

### Date expansion to four-digit years

Two-digit years were expanded to include the century and this had to be done consistently in programs, files and databases and coordinated across all systems that shared or exchanged data throughout whole supply-chains. This was the best solution as it resulted in unambiguous dates that would result in software that was easy to maintain. But this method was costly, needed extensive testing and conversion efforts, and often affected many connected systems.

### Windowing

In this solution, two-digit years were retained in the data and the programs were changed to calculate the century value when it was needed for date comparisons and other date calculations. So, for example, years 00 to 19 would be treated as 2000 to 2019 whereas years 20 to 99 would be treated as 1920 to 1999 .This solution involved many small local changes to programs; it was simpler to test and implement and cheaper than date expansion but it had three main disadvantages:

1. it was quite easy to make mistakes or to overlook places in the software that had to be changed;

2. it was important to ensure that the same range of years was used for all connected systems, which could require detailed co-ordination across a whole supply chain of different companies;

3. it was not a permanent solution. No-one knows how many systems still exist that are going to fail as we approach the end of their chosen "window".

A variant of this method was the *sliding window* that used a set of years before and after the current date.

**Replacement**

Many companies took the opportunity to replace their systems rather than to spend the money required to correct the errors. This was the option that suppliers and consultants often recommended because they would make more profit. Unfortunately most companies could not resist adding new features rather than seeking a direct replacement, which complicated the projects and, then as now, IT projects commonly overran. In many cases the replacement projects came up against the hard Y2K deadline and testing was cut short, leading to later failures that were rarely publicised as having been Y2K related.

Y2K remediation was a major business opportunity for software companies and consultants. Many large and small consultancy firms created and trained teams that could help their clients find and correct Y2K errors, and specialist companies developed software tools to scan software source code and data to find places where 2 digit years were used. COBOL programmers came out of retirement and commanded high salaries (Scott Adams, the creator of the Dilbert cartoons, always drew these COBOL programmers as dinosaurs). It was particularly irritating to see extremely high salaries being paid to staff who had created many of the problems themselves through oversight or incompetence.

In 1996, cost estimates were $1 to $2 per line of business mainframe software code to make Y2K repairs. (A typical accounting program has hundreds of thousands of lines of code.) The cost was expected to rise to over $4 per line by 1998 as a shortage of skilled programmers developed. In fact, the cost gradually fell to only a few pennies per line because automated Y2K tools became extremely accurate and efficient at fixing the code. This had the fortunate result that those countries and companies that had started their remediation programmes late were able to make up the lost time.

## What Failures were prevented?

Thousands of errors were found and corrected during the 1990s, avoiding failures that would otherwise have occurred. Here are just a few of them.

**The UK's Rapier anti-aircraft missile system**[xvii] The Ministry of Defense admitted that the millennium bug could have left Britain vulnerable to air attack. It discovered that the Rapier anti-aircraft missile would have failed to retaliate. The problem was identified inside the field equipment which activates the missiles and it would have made the system inoperable.

**Swedish nuclear plant -** In mid-1998, a nuclear utility in Sweden decided to see what would happen if it switched the clocks in its reactor's computers to read January 1, 2000. The response surprised utility officials, who had expected business as usual. The reactor's computers couldn't recognize the date (1/1/00) and shut down the reactor[xviii].

An error was found and corrected in **BP Exploration's oil platforms** that would have caused significant disruption. BP Exploration told the consultants that finding that one error justified their whole Y2K programme[xix].

**US Coast Guards** found[xx] errors that would have caused loss of steering control on ships, and failure of fire detection and control systems.

10% of **VISA swipe-card machines** were found not to handle cards that expired after 1999. Visa asked member banks to stop issuing cards with 00 expiry dates as a temporary measure[xxi]. Visa then prepared to fine banks up to £10,000 per month until their systems became Y2K compliant[xxii].

> *The company, itself a consortium of 20,000 banks, is launching the penal system a year after its first deadline for Year 2000 compliance. It estimated that 1.3 million outlets worldwide are still unable to deal with cards with expiry dates reading '00'. Britain is believed to account for only 40,000 of the faulty terminals. After April, banks that have problems processing Visa's cards will be charged between UKP600 and UKP 100,000 per month, depending on volume, until they correct the bug. Visa says that 90 per cent of terminals accept the new cards but an unacceptably high number still throw up an error when told a card was issued in '97 and expires in '00. Jim Dickie, vice president of Visa's operations and services in Europe, said the move was the next logical step to safeguard the credit card's brand name.*

Many thousands of date problems were found in **commercial data processing systems** and corrected. (The task was huge – to handle the work just for General Motors in Europe, Deloitte had to hire an aircraft hangar and local hotels to house the army of consultants, and buy hundreds of PCs).

Many Y2K failures occurred in the 1990s and were then corrected. A typical example was an **insurer** that sent out renewals offering insurance cover with dates that ran from 1996 to 1900 rather than to 2000.

When the systems for the Millennium Dome in London were tested by advancing the system clock to 23.55 on 31 December 1999 and seeing what happened at midnight, so many errors occurred that they scrolled off the bottom of the system console before they could be read[xxiii]. All these errors had to be corrected in time for the VIP reception on December 31st 1999.

## What Failures Occurred after 2000

Despite the massive international efforts to find and correct Y2K errors, some failures did occur on January 1 2000. Many of these were easily corrected by rebooting the system or resetting dates manually; one example was the Runway Visual Range systems that report the visibility on NATS airfields in the UK; at 4am on January 1, the systems made a routine health-check with the master computer, discovered a significant disagreement about the date and shut down. (The problem was quickly resolved and there was never any risk to aircraft or passengers, because the weather was fine and anyway, no-one was flying that morning!).
Other reported[xxiv] problems included[xxv]:

1. 15 nuclear reactor shut-downs (in Spain, Ukraine, Japan and the USA)[xxvi].

2. Many credit card systems rejected valid cards.

3. The oil pumping station in Yumurtalik shut down, cutting off supplies to Istanbul.

4. There were power cuts in Hawaii and cable television feeds failed.

5. Glitches hit government computers in Hong Kong and mainland China. Police testing the sobriety of drivers in Hong Kong had to enter birth dates on breath-testing machine because of an apparent Y2K malfunction. Courthouse computers in Italy mixed up prisoner dates by 100 years. A few ATM machines shut down.

6. Government computers in Hong Kong failed to display the correct date, but no records were lost and everything ran smoothly, officials said. Weather observations in part of mainland China had to be made by hand after the circuit board of a solar measuring device in the remote north-western region of Ningxia failed to roll over to 2000

7. In Tokyo, about a dozen small brokerages reported Y2K-related glitches in a record-keeping system. They were quickly fixed. Ten small Hong Kong companies reported minor hardware or software problems possibly caused by Y2K. The heat went out in apartments for about 900 families in Pyeongchon, South Korea.

8. A hospital in western Norway reported that an X-ray machine had failed. News reports said cash registers at a handful of 7-Eleven convenience stores also failed there, and some ATM machines weren't working.

9. Ticketing machines on some buses in Australia briefly jammed. Forecasting maps at the French weather service initially got the New Year Day date as ``01/01/19100." Eight hundred slot machines shut down at a Delaware horse track. Taxi meters broke down in a China province.

10. Three dialysis machines stopped functioning in Egypt hospitals, but the problem was quickly fixed.

11. A provincial court in South Korea issued automated summonses to 170 people to appear for trial on Jan. 4, 1900 instead of Jan. 4, 2000.

12. A customer at a New York State video rental store had a bill for $91,250, the cost of renting the movie 'The General's Daughter' for 100 years.

13. Glitches were reported in train-pass vending machines in Tokyo, in a program of France's Syracuse II military satellite system, and at the Islamabad Stock Exchange in Pakistan, where computers said the year was 1900. Exchange members were forced to manually record their transactions.
14. The Palatka-Putnam, Florida reporting station, conveyed that the county's 911 service was experiencing intermittent problems with the name and number link.

15. Low-level Windshear Alert (LLWAS) systems failed at Tampa, Denver, Atlanta, Orlando, Chicago O'Hare, and St. Louis during the rollover. The systems displayed an error message. Air Transportation system specialists at each site rebooted LLWAS computers to clear the error, and the last system was in normal operation in just over two hours. Impact on operations was minimal.

16. The Direct Access Radar Channel monitor at Albuquerque, N.M. failed immediately after the rollover.

17. Kavouras Graphic Weather Display Systems at flight service stations in 16 locations around the country failed about ten minutes after the rollover. Data supplied to automated flight service stations was not updating properly. Specialists discovered the system sent data bearing the date "2010," resulting in rejection of National Weather Service data and incorrect updates of weather data in the system.

18. An Automatic Backup to the central computer complex at the Cleveland Air Route Traffic Control Center failed to activate after the date change.

19. ARINC Oceanic Display and Planning System printers at Oakland, CA and Islip, NY, failed the transition from 1999 to 2000. These are not FAA printers but are a redundant system used in the relay of data from ARINC communications centers to air traffic controllers handling oceanic traffic.

20. The Federal Reserve Bank of Chicago experienced Y2K related problems in transferring $700,000 in tax payments. The bug was fixed and the payments were made the next day.

21. Weather Message Switching Center Replacement, Atlanta, GA, stopped recognizing and processing certain kinds of Notices to Airmen (NOTAMS) because of a software problem involving a failure to recognize years ending in "0" in the NOTAM time and date code.

22. Several problems remained in Microsoft Visual Basic and Access[xxvii].

23. Payroll software at Berlin's German Opera denied certain employees government mandated subsidies for families with children. When year 2000 arrived, the computers date was 1900. This caused a person born in 1995 to appear 95 years old, making the parents ineligible for the government subsidy.

24. The Pentagon had a self-inflicted Y2K mis-fix that resulted in complete loss of ability to process satellite intelligence data for 2.5 hours at midnight GMT on the year turnover, with the fix for that leaving only a trickle of data from 5 satellites for several days afterward.

25. The Pentagon DefenseLINK site was disabled by a preventive mistake.

26. The Kremlin press office could not send e-mail.

27. In New Zealand, an automated radio station kept playing the New Year's Eve 11pm news hour as most recent, because 99 is greater than 00.

28. Toronto abandoned their non-Y2K-compliant bus schedule information system altogether, rather than fix it.

29. Birth certificates for British new-borns were for 1900.

30. Some credit-card machines failed, and some banks repeatedly charged for the same transaction -- once a day until a previously available fix was finally installed.

31. Various people received bills for cumulative interest since 1900.

32. At least one person was temporarily rich, for the same reason.

33. In e-mail, Web sites, and other applications, strange years were observed beginning on New Year's Day (and continuing until patched), notably the years 100 (99+1), 19100 (19 concatenated with 99+1), 19000 (19 concatenated with 99+1 (mod 100)), 1900, 2100, 3900, and even 20100. Some Compaq sites said it was Jan 2 on Jan 1. U.K.'s NPL atomic clock read Dec 31 1999 27:00 at 2am GMT on New Year's Day[xxviii].

One serious UK problem[xxix] was only recognised when a Heath Visitor in Yorkshire noticed an unusual number of babies were being born with Down's Syndrome. It transpired that more than 150 pregnant women were given the wrong results from pathology laboratory tests because the *PathLAN* computer system that was used in nine hospitals calculated the women's date of birth incorrectly from January 2000; it had worked perfectly for the previous decade. The result was that women who should have been identified as being in a high-risk category were wrongly told that they did not need further testing.

Many other problems occurred because replacement systems had been implemented hastily or by incompetent or malicious staff. To give just two examples: (a) a utility company in Australia lost the ability to monitor its financial position, to collect direct debits and to issue accurate bills; it ran out of money and had to be rescued by the State Government; the Directors were replaced[xxx]; (b) A software failure resulting from an attempted last second fix caused delays in air transportation all over the east coast of the USA.

## Why were the problems far less significant than feared?

There had been fears that there would be far worse problems than actually occurred. With hindsight, there were several reasons for this:

- A huge number of problems had been found and corrected before 2000, as the result of the enormous international investment in Y2K remediation projects and fixing the individual failures that had occurred over the preceding decade.

- The companies and countries that seemed to have started their Y2K programmes too late were rescued by the progress that was made by others – in getting suppliers to correct software and equipment and in developing high-productivity tools for remediating legacy software.

- The feared domino effect of cascade failures in interconnected systems did not happen because the major supply chains contained the largest companies and these companies had given high priority to finding and correcting problems in their systems and in interfaces.

- Systems were actually not as tightly interdependent as had been feared, so there was redundancy in supply chains that provided resilience.

- The threat had been exaggerated to some extent, partly by suppliers and consultants who wanted to sell equipment, systems and services, partly as a necessary step to attract attention to the seriousness of the threat and to stimulate and accelerate the required remediation action and contingency plans, partly by headline seekers, and partly by religious and survivalist sects that saw the new millennium as having apocalyptic significance.

## What did it Cost?

The estimated global costs were between \$300B and \$500B, though this can only be an estimate and is not well evidenced. The International Y2K Co-ordination Centre and reported the international costs in February 2000[xxxi] as follows:

> *According to data for 50 countries developed and copyrighted by the International Data Corporation (IDC), national public and private Y2K expenditures (1999) include estimates of \$34 billion in the United States, \$2.5 billion in Italy, and \$100 million in Venezuela. Of course the United States, being a large country, has many more digital systems than Italy or Venezuela. Although there is little reliable data on the number of digital systems in a country, a rough index can be fashioned using the number of telephones as a surrogate. In 1997, according to the International Telecommunications Union (ITU), there were 172 million telephone subscriber lines in the U.S., 26 million in Italy, and 2.6 million in Venezuela. Thus the weighted Y2K Spending Index (the 1999 Y2K cost per telephone) was about \$195 in the U.S., \$97 in Italy, and \$38 in Venezuela.*
>
> *Industrialized countries are highly dependent on digital systems, and there are many interconnections between the systems. Interconnected networks increased the complexity of Y2K work in countries more dependent on digital systems. Again, no good data exists on the degree of countries' dependency on digital systems, but "teledensity," the number of telephones per 1000 persons, provides a rough surrogate. For the 50 countries whose Y2K expenditures were tracked by IDC, teledensity ranged from 19 phones per 1000 people in India to 679 in Sweden (ITU, 1997 data). When normalized for this complexity, it becomes clear that, in general, countries with greater dependencies on information technology spent proportionately more to fix their digital systems.*
>
> *One additional factor may account for spending differences across countries. As noted in Chapter 10, Lessons Learned, there was an economic advantage to starting late. The countries most dependent on technology started early, and they paid for the development of tools that made the job cheaper for those who started later.*

There were many benefits from this expenditure, besides avoiding Y2K failures. Most companies gained far better control over their IT inventory and far more detailed knowledge of their supply chains and their position in other's supply chains. The importance of IT to the business became far clearer to boards of directors and led to more companies having Chief Information Officers with a reporting line to a main board director. Many IT systems were upgraded and equipment was replaced.

## Were the Right Lessons Learned?

The Y2K problem is often referred to now as an example of an exaggerated threat or to support an argument that expert warnings – for example about the threat from climate change – should be ignored. In my opinion, this is completely the wrong conclusion to draw from the fact that the worst predictions did not occur.
I believe that the real lessons should be these:

1. **The problem was caused by poor software engineering**. Peter Neumann observed that systematic use of concepts such as abstraction, encapsulation, information hiding, and object-orientation could have allowed the construction of efficient programs in which the representation of dates could be changed easily when needed.
2. Y2K was a particularly serious threat because it could cause many unrelated systems to fail simultaneously. Such **single-points-of-failure should be strenuously avoided**.
3. Systems were not as tightly coupled as had been feared; looser coupling and some **redundancy provides useful resilience** by reducing or removing the risk of cascading failures.
4. **It is possible to achieve remarkable urgency, effort and co-ordination once it is seen to be essential and this can be very effective**.
5. **Changes to audit standards and regulation cause Board-level attention and action**.

Y2K should be regarded as a *signal event*: a near miss that should serve as a major stimulus to change the way that IT systems are designed and built, so that no similar problem can ever happen again. Y2K remediation should be seen as a major success, showing that it is possible to deliver major IT systems on time if they have clear objectives, clear timescales, senior management support, and commitment to provide the necessary resources, clear communications and acceptance throughout the affected organisations that the project has the right objectives and deserves the highest priority. Robin Guenier has observed that the disastrous National Programme for IT in the UK National Health Service (NPfIT or Connecting for Health) might have succeeded if the right conclusions had been drawn from the Y2K programmes and implemented in future large-scale IT projects.

Unfortunately, little seems to have changed, perhaps because Y2K remediation was so successful and because it is only after a catastrophe has occurred that major reorganisations are brought about in the way an industry operates.

1. Almost all software-based systems are still designed and developed with cost and time-to-market taking priority over modularity, robustness, security, ease of modification and other software engineering principles. Testing is still the primary way in which programmers assure that software is fit for purpose, despite decades of evidence that testing can never find most errors.
2. Single points of failure are still introduced without considering the possible consequences. One current and critical example is the extraordinarily widespread dependence on the GPS signal for positioning, navigation and timing. Other examples are the almost universal reliance on encryption systems that could be destroyed by advances in quantum computing, and the growing dependence of multiple systems on identical software components or on identical sources of open data.
3. In the interests of efficiency, supply chains have become far more tightly coupled and redundancy has been removed with little thought about the impact on resilience, making cascade failures more likely.
4. There is an increased unwillingness to use regulation to encourage private companies to write better software.

**The current cybersecurity crisis is one consequence of this failure to learn from Y2K but as there is no deadline to compel urgent action the risks to society from badly designed and insecure software are certain to continue to increase.**

## Acknowledgements

I am very grateful to Robin Guenier[xxxii], Michael Mainelli[xxxiii] and Rob Wirszycz for their recollections from the Y2K front line. All remaining errors are my fault.

[i] http://fm2x.com/The_Century_Date_Change_Problem.pdf by Robin Guenier, former Chief Executive of the UK Central Computing and Telecommunications Agency and Executive Director of Taskforce 2000.
[ii] *Embedded systems, A guide to evaluating how this problem could affect your business,* Millennium bug campaign brochure, February 1998, from Action 2000 [a UK Government awareness campaign]
[iii] model 5160 – it had a hard drive, unlike the 5150
[iv] the Intel 8088 – chosen over the faster Intel 8086 because the 8088's 8-bit bus made the rest of the motherboard cheaper
[v] Intel 8253 or 8254 Programmable Interval Timers
[vi] http://www.daqarta.com/y2kure.htm
[vii] http://www.daqarta.com/y2kure.htm#prob3
[viii] https://www.theguardian.com/science/2011/feb/28/leap-year-alex-bellos
[ix] http://researchbriefings.files.parliament.uk/documents/POST-PN-89/POST-PN-89.pdf
[x] The Birmingham Summit, Final Communiqué, Sunday 17 May 1998, http://birmingham.g8summit.gov/uk/docs/final.shtml.
[xi] *Bovis warns of computer threat to buildings,* Financial Times, pg 6, 06.06.98
[xii] https://www.federalreserve.gov/boarddocs/speeches/1998/19981029.htm
[xiii] http://www.iy2kcc.org/
[xiv] http://web.archive.org/web/20010227234942/http://www.iy2kcc.org/Advisory99012.htm
[xv] http://airwolf.lmtonline.com/news/archive/1230/pagea15.pdf
[xvi] http://parsifal.membrane.com/y2k/pd2000-4.htm
[xvii] The Times, 9 Dec 1998, p2
[xviii] Reported in http://xwalk.ca/Y2K.html
[xix] Reported to me by my Y2K consultants in Praxis/Deloitte Consulting, who had found the error.
[xx] Reported in http://xwalk.ca/Y2K.html
[xxi] Reported in http://www.msnbc.com/news/42003.asp but no longer available online.
[xxii] The Times (London, UK), Mon Jan 6 1997, Business News p41
[xxiii] Personal communication from the head of the Y2K project for the Dome.
[xxiv] http://xwalk.ca/Y2K.html
[xxv] https://www.cs.swarthmore.edu/~eroberts/cs91/projects/y2k/Y2K_Errors.html
[xxvi] Originally on http://www.informationweek.com/wire/story/TWB20000102S0002 but removed
[xxvii] https://www.theregister.co.uk/2000/01/07/y2k_bug_lurking_in_microsoft/
[xxviii] These are all from comp.risks, the Forum on Computer-Related Risks moderated by Peter Neumann.
[xxix] https://www.theguardian.com/society/2001/sep/14/NHS1
[xxx] I was an expert witness in the subsequent court case.
[xxxi] http://web.archive.org/web/20010108225500/http://www.iy2kcc.org/Appendices.htm#AppendixD
[xxxii] http://fm2x.com/The_Century_Date_Change_Problem.pdf
[xxxiii] http://www.zyen.com/index.php?option=com_content&view=article&id=163&Itemid=359