YEAR 2000 PROBLEM - THE LESSONS LEARNED

Bogdan Pilawski

Wielkopolski Bank Kredytowy SA, Kozia 10, 61-835 POZNAN, Poland
phone +48 61 8564064, fax +48 61 8564063, mobile 605 253124
e-mail : bogdan.pilawski@wbk.com.pl

Poznañ University of Economics, Department of Information Technology,
Aleja Niepodlegoœci 10, 60-967 POZNAN, Poland

„The millennium bug is a vivid and powerful reminder of the ways that we are growing ever more interdependent as we rise to the challenges of this new era"

President Clinton

Abstract

It is virtually impossible to cover all of multiple lessons learned from Year 2000 Problem. For that reason this paper concentrates on an arbitrarily selected subset of those, leaving more general key issues to another consideration. Among subjects covered one can find calendar date handling by IT systems, date standards, management role in the year 2000 crisis and others. For the obvious reason of limited space of this paper these subjects are only signalled and the reader is directed to the sources presenting more detailed consideration.

Introduction

With the most critical dates related to the Year 2000 Problem being over, now one can try to look for some lessons learned from this experience. No doubt these are going to cover a wide area, and it is no wonder since the Year 2000 Problem has equally threatened things from autopilots in aeroplanes to desktop PCs, and more advanced home appliances.

Generally the changeover from 1999 to 2000 was not that harmful as initially predicted, but that by all means doesn't say it has not done any harm at all. The number of real failures reported was very limited (nearly one thousand of cases of more serious nature known world-wide). This was mainly because anyone experiencing these failures was doing everything possible to hide them from the public. The rationale behind it was simple: it would be a big shame to admit Y2K related failure after all that media hype and pressure to be prepared. But ask any Clipper programmer and you will get the right picture: they are still struggling to get things right. The system they care for, usually behaves so-so in general, but there are plenty of secondary functions which simply fail to operate or do yield wrong results. The estimates for Poland say there are about 30 thousand Clipper (or dBASE, to be exact) related systems in use country wide. How many of these operate properly? How many do run with the calendar date set back to a year from

the past? How many users finish off computer generated reports manually? How many files interchanged between systems are manually editor-corrected before they can be processed?

Beside this now we know our IT systems much better than ever. We learned a lot about what we had no time to study before. We needed to slow down and to look carefully on what's going on. Maybe the picture we had seen wasn't gloomy, but it wasn't pretty either.

Lesson no. 1 - Calendar date handling by IT systems

The Year 2000 Problem revealed numerous weaknesses of calendar date handling capabilities of both computer hardware and software. The concept of atomic clock and the introduction of UTC has made the time measure one of the most precise measurement techniques none to mankind. While this can deviate by only one second over several million years, the more rough measure of calendar dates is still the source of problems, and especially in the area of computers and their software [Jone98a].

Whether it's hardware clock or computer software, they are quite unable to cope with most possible calendar dates, and with four digit years in particular. This is equally applicable to old and new products in that area.[1] The good example of this is the hardware clock of common PC. It is around since 1984 when AT model of PC was introduced. It has been designed to handle two digit year only and this has not changed till now. The truth is the Year 2000 compliance of a PC is provided by a BIOS code and not by the clock itself.

The calendar we're using is also to be blamed[2]. It is far from being precise and hopelessly tries to get together the sun cycle (farming) and 28 day moon month (religion), using a unit of the day, which is absolutely incompatible with either of them. If that's not enough for trouble, just add to that a concept of seven day week, for which no rationale exists.[3] [Seid97]

There are only two software packages claiming they are capable of properly handling any possible calendar date, whether current, future or from the past. These are SAP R/3 and ICL's GENTIA. This is extremely hard to prove that, especially bearing in consideration the period of almost 350 years after Gregorian Calendar was announced, during which various countries of Europe gradually adopted it. So far there is no satisfactory solution in sight, what is best illustrated by a leap year problem: some sources say year 3600 needs to be dropped from leap year list [Ulri97, p.6] to adjust for difference between calendar and the position of Earth, while others claim it is year 4000 instead [Seid97].[4]

---

[1] this was one of the most astonishing discoveries made while coping with Y2K problem

[2] this is the Gregorian Calendar, introduced in 1582 by pope Gregory XIII and gradually accepted in most countries of the world; there are hundreds of other calendar systems in existence, but these have only local or mainly religious meaning; the idea of New World Calendar, raised in 1931 and initially supported by 32 national committees and the League of Nations (United Nations after the World War II) was abandoned in 1955, after USA withdrew its support because - as it was stated - it would harm the religious feelings of US citizens [WCA45, p.147-150, USA55, p.629]

[3] the Moon takes 29,53059 days to go round the Earth, what - assuming twelve months - gives the so called moon year of 354,36706 days, while the Sun year (the time needed for the Earth to complete its run around Sun) lasts for 365,242198 days [Gou98, p.117,110]

[4] the leap year rule of Gregorian Calendar says, every year which number is divisible by 4 is a leap year, except for those divisible by 100, which are leap years only when also divisible by 400

In many organisations world-wide the date of 29th February 2000 resulted in more problems than the changeover from 1999 to 2000. This was especially true in banks, financial institutions and also in case of control electronics. There is also a commonly known error of Microsoft's Excel software, which claims year 1900 was a leap year, while in fact it was not. Microsoft says this error was initially introduced into Lotus 1-2-3 package, and deliberately repeated by Microsoft for the sake of full compatibility with Lotus.

Year 2000 problem preparations have revealed many more calendar dates critical for IT systems. Usually they relate to a particular hardware or software. Few most significant examples of these dates are given in Appendix 1.

Lesson no. 2 - Standards

There are a number of date standards in existence. The most prominent example of those is the European Standard EN 28601, which replaced former ISO 8601 standard.[5] There are also US, Canadian and Japanese date standards and British equivalent of EN 28601.[6] From an IT point of view all these standards lack one thing - they do not require four digit year field as compulsory. They allow various date format, provided the full date can be deducted using the so called inference rules. That allows for the year field, for that instance, being omitted altogether if current year is to be assumed.

The only exception is the FIPS PUB 4-1 standard of US state administration. It strongly insist on the yyyy-mm-dd date format to be used in all IT programs, data files and data exchanges. The only problem with that standard is, its use is not compulsory and acts as an advise only.

Since none of the existing standards were suitable to govern the year 2000 preparations, a provisional British Standard DISC PD2000-1 was widely adopted for the purpose. Even that needed to cope with existing practices and for that reason could not insist on one, universal format of the date.

If however some proper and exact solution will not be found and accepted in the near future, we will be doomed to continuous problems with date formats in various aspects of IT.

Another matter is the internal representation of the date adopted by particular software manufacturers. There is an enormous variety of these, and they do not conform to any standards at all.

Lesson no. 3 - windowing techniques (handling of two digit year fields)

In 1996 Palace Produce International supermarket chain experienced serious problem with „00" year field in expiry dates of some credit cards. Every attempt to pay with such card resulted in total failure of their new IT system and temporary shop closure. [Tate97, p.10] Since the software company was unable to get things right, the whole affair ended up in court.[7] This case

[5] the Polish standard PN-90/N-01204 is an exact repetition of ISO 8601; see [PN01204]

[6] these are: ANSI X3.30 [ANSI85] in the USA, FIPS PUB 4-1 [FIPS88, FIPS98] in US state administration (more restrictive superset of ANSI X3.30), JIS X 0301 in Japan, and CAN/CSA-Z234.5-89 in Canada

[7] Palace Produce won the court case and a $250 thousand compensation; the related legal material (including detail protocols from court proceedings) was made available at http://www.2000law.com/html/lawsuits.html

has turned attention to much wider problem of various payment cards. Despite all the differences in their characteristics, they had one thing in common: the year field in the date encoded on magnetic strip was of two digits only. To change that rule it would mean replacing several hundreds of millions of cards along with amendments to card handling software. The card organisations decided to keep the existing cards and to adopt a software solution called „windowing technique". This technique is based on an simple assumption: the two digit year numbers starting from 00 to a number called a „pivot year" are software extended with front digits of „20", giving together a four digit year number, e.g. 20 + 04 = 2004. All numbers from pivot + 1 up to 99 are extended with „19", e.g. 19 + 95 = 1995. This works fine, however there are at least two reservations about that method:

- various pivot years were assumed for different applications[8]
- they only postpone re-appearance of the problem by pivot years.

The windowing techniques resulted in many applications failing to exchange date data just after putting their year 2000 compliant versions under tests. This was simply because various organisations adopted different pivot years.

In Poland the so called Citizen Register Number is commonly used by state administration and businesses, among others in most payroll and tax IT systems. The birth date (two digit year) is a vital part of this number. The State Office running that system adopted the so called Roman System of deciding whether a particular birth date comes from 19th, 20th or 21st century. For birth dates with year starting with „18" they add 20 to the month number (3 January 1895 becomes 032195), while 40 is added to dates with year starting with „20" (3 January 2000 becomes 034100). The dates in which the year starts with „19" got the direct month number (3 January 1995 becomes 030195). This rule is simple and easy to adopt in computer programs, provided one knows about the very existence of such a solution. The State Citizen Register Office says the leaflet explaining that rule is freely available on demand. The problem remains, to request it one needs to know it does exist in the first place.

Lesson no. 4 - software development

The advent of PC in the early 80s beside other things has brought and unprecedented and illusive easiness in creating pieces of software. The whole new generation of IT people has grown up which does not see the need of stage wise approach to development of IT systems, not to mention things like maintaining versions, documentation and change control. This was true world-wide but especially in the countries like Poland where pre-PC IT culture was limited to a very narrow group of professionals.

Out of date documentation (or no documentation at all) and lack of latest source codes were the most commonly faced problems during year 2000 related preparations. This has put an enormous burden upon those responsible for the appropriate projects, and imposed huge expenditures on businesses. The advent of the Year 2000 necessitated the upgrade and standardisation of various systems, resulting - among others - in a more cost-effective asset base.

[8] in its Excel95 Microsoft assumed year [20]19 as pivot, while in Excel97 this is [20]29

It remains to be seen, whether they will build upon the level thus achieved, or will again plunge into usual state of disorder.

On the positive side - the Y2K problem has turned more attention to maintaining the software development life cycle, to IT metrics and quality measures like CMM or ISO9000 family of standards. In many organisations year 2000 problem has lead to implementation of a method of an independent verification and validation of software systems used. All these subjects were widely and in detail discussed in various papers dealing with year 2000 problem and its aftermath, and also in many earlier publications. [Yourd97, Peng93, PGF96, Dods95, Offen99, Estes97, Kaul97 and others]

Lesson 5 - The Role of Management

Management played a significant role in year 2000 preparations. It was far more to that than only managing the things to develop and go smoothly. At the very beginning of year 2000 problem there was the so called „early denial stage", with management denying the very existence of the this problem and refusing to spend any money on it. [Carm98, Eddy98, BCS97, p. 4, Lewi97, p. 17]

Printed in 1993 „Doomsday 2000" article by Peter de Jager is commonly recognised as the first paper on the subject. [Jage93] In fact one can find a number of earlier writings, each of these enough to raise the attention to the problem and its implications. [e.g. Beme79, Babe82]

These many early warnings were widely ignored and substituted with early denial. The result in many organisations had a number of serious consequences: higher costs, delays in normal development of software applications, lower reliability of systems hurriedly made Y2K compliant and others.

On the other hand - many IT managers have demonstrated their ability to get things done on time and (sometimes) within budget. This aim however would probably be never accomplished without strong support from top management of the businesses. Now they know their IT environment much better and in more detail. It was for the first time they learned how many redundant hardware and software exists around, and how costly it is just to keep it going. A special part of it is the „dormant software". This is a software being part of bigger applications, which is no longer used for some historical reasons. It is however still maintained, compiled, backed-up and it still occupies disc space and memory of computers on which it is run. R. Kendall, former IBM executive, has noted that in IBM data centres somewhere between 40 to 70% of the total number of applications present in software program libraries were found to be no longer in use. [Jone97a, p.32] This example clearly shows how big are reserves of IT, and how much it is orientated on moving forward, leaving a huge disorder and waste behind.

Conclusion

Every single lesson learned from Year 2000 Problem practice presented here could be a subject of a separate paper. Hopefully these will be written and we will all gain even more from this unique experience. There are also many more lessons to be learned. One of those is the IT systems operational security. Almost every single day we are alarmed with the news about security breaches, financial loses and successful hacking attacks. Year 2000 Problem

preparations revealed how low the average IT security level is, whether it's government, bank or manufacturing company. In many instances the makeshift updates made to the systems have brought this level even lower. Many concerns were raised about possible time bombs being implemented into software while getting it Y2K compliant. [see: Coop97, p.3] In the USA the FBI found even a family with many previous links to organised crime, which established a Y2K software company just to get hold of details of software of businesses served, and most probably to exploit this knowledge later to its own advantage. [Gary98a]

All that means the Y2K crisis is far from over and its aftermath could appear out of a sudden in some least expected places. After implementation change freeze period introduced in many organisations and businesses is gradually lifted[9] IT returns to normal course of events. However one needs to stay vigil and watch carefully, remembering that - according to Gartner Group research - each 50th line of source code of average computer program used in business relates to calendar date[10].

[9] e.g. in the banks of AIB Group, WBK is member of, this ban on implementing changes to IT systems was in force from 1 October 1999 till 7 April 2000, to cover end of 1999, 1999/2000 change over and critical dates like: first working day of year 2000, first month closure in 2000, 28,29 February, 1 March and first quarter closure in 2000

[10] each 10th line in banking software

Appendix 1.

Example Future Critical Dates
Hardware and Operating System Related Dates

No.	Date (Time)	Details	Remarks
1	1980-0-0	The smallest possible file creation date in MS-DOS operating system	Non-real date
2	1980-1-1	The smallest possible date in MS-DOS operating system	
3	2001-9-8	The *time_t* variable in UNIX operating system will change from 9 to 10 digits	Date dependent screen and printout formats will become disorganised
4	2036-1-1	Final overflow of date register in Unisys Series A computers	Initial date related problem was experienced by Unisys on 1987-1-1. Another one is expected on 2003-1-1. The problem results from especially complicated method the date is registered there (it is a decimal number containing number of years since 1970, followed by number of the day within a year, converted into two byte hex integer with sign indicated by the bit before last)
5	2038-1-19 3:14:07	UNIX operating system time register overflow	Some systems will change onto 1970-1-1, while others onto 1901-12-13
6	2042-9-18	IBM 360 time register overflow	
7	2041-11-16	Unisys's BTOS and CTOS operating systems will change into 1952-3-1 at midnight	
8	2042-9-17 23:53:47	IBM 370 time register overflow	This register counts the so called long seconds since midnight 1900-1-1 (1 long second = 1,048576 sec.)
9	2071-5-10 11:56:53	AS/400 internal clock will change to 1928-8-23	
10	2080-1-1	Windows File Manager operating with ISO-8601 date format will subtract 100 years from each file date	

Software Related Dates

No.	Date (Time)	Details	Remarks
1	1601-1-1	First day available in ANSI COBOL85 calendar	
2	1899-12-30	Day 0 of Borland Delphi software	*TDateTime* variable
3	1900-2-29	Non existing date	It exist (an error!) in Lotus 1-2-3 and Microsoft Excel spreadsheets
4	2010-1-1	Time register overflow in ANSI C library modules	
5	2036-12-31	Visual C^{++} version 4.x maximum date	

Service Related Dates

No.	Date (Time)	Details	Remarks
1	2002-1-1	Euro to be introduced as a real money	
2	2002-6-30	The end of double currency accounts (Euro and domestic currency)	
3	2002-7-1	The end of circulation of money other than Euro	
4	2???-?-?	Poland to go Euro	

Source:. [Army98], [Cind98b], [Fred98], [Jone98a], [Kolb98a],[Mitr98], [Stock98b]

Literature

[ANSI85] *ANSI X3.30-1985(R1991), American National Standard for information systems - representation for calendar date and ordinal date for information interchange,* American National Standards Institute, New York, 1992

[Army98] *Critical Y2K Testing Dates*, http://www.army.mil/army-y2k/Testing_Dates.htm, December 1998

[Babe82] Baber, Robert Laurance, *O oprogramowaniu inaczej* (original title: The Socially Responsible Programming of Our Computers), WNT, Warszawa, 1989

[Barb98] Barber, David, Buffa, Joseph, *Year 2000 Test Procedures*, General Motors Corporation, 1998

[BCS97] British Computer Society, *The Year 2000 - A Practical Guide for Professionals & Business Managers*, Swindon 1997

[Beme79] Bemer, Robert, *Time and the Computer*, in: Interface Age Magazine p. 74-79, February 1979

[Carm98] Carmichael, Douglass, *Social psychology of Y2K: Trying to understand the denial*, http://www.tmn.com/~doug/dcnote1.htm, 1998

[Cind98b] *Year 2000 Test Criteria*, http://www.cinderella.co.za, 1998-7-20

[Coop97] *Organising the Year 2000 date change while taking advantage of synergies*, materials of COOP BANK, Basle, Switzerland, 1997

[Dods95] Dodson, William R., *Harnessing End-user Computing within the Enterprise - The Achilles Heel of the Enterprise*, Bill Dodson Associates, Boston, 1995

[Eddy98] Eddy, David, *Senior Management and Y2K Denial*, Westergaard Year 2000 Technology Corner, http://www.y2ktimebomb.com, 13/5/1998

[Estes97] Estes, Don, *Year 2000 Strategic Project Design: Risk Assessment, Cost Control And Automated Testing*, materials of: 2000 Technologies Corporation, Lexington, USA, 1997

[FIPS88] *FIPS PUB 4-1*, Federal Information Processing Standards Publication, *Representation of Calendar Date and Ordinal Date for Information Interchange*, version of 27/1/1988

[FIPS98] FIPS PUB 4-2, Federal Information Processing Standards Publication, *Representation of Calendar for Information Interchange*, version of 15/11/1998, US Department of Commerce, National Institute Of Standards And Technology

[Fred98] Fredrickson, Janet, *Comprehensive List of Potential Y2K Problem Dates*, Mitre Organisation, http://www.mitre.org, 1998-10-29

[Gary98a] *The Mafia Gets Into the Y2K Repair Business,* Gary North's Y2K Links and Forums, http://www.garynorth.com/y2k/Detail.CFM?Links_ID=1833, 17/6/1998

[Gou98] Gould, Stephen J., *Pytania o millennium*, Prószyñski i Ska, Warszawa 1998

[Jage93] Jager, Peter de, *Doomsday 2000,* Computerworld, 6/9/1993

[Jage97] Jager, Peter de, Bergeon, Richard, *Managing 00 - Surviving the Year 2000 Computing Crisis*, Wiley Computer Publishing, 1997

[Jone97a] Jones, Capers, *The Global Economic Impact of the Year 2000 Software Problem,* papers of: Software Productivity Research, Burlington,1997

[Jone98a] Jones, Capers, *Dangerous Dates for Software Applications*, version 2, March 1998, Software Productivity Research Inc. - Year 2000 Information Center, http://www.year2000.com/archive/dangers.html

[Kaul97] Kauler, Barry, ERUPT: pragmatic software development lifecycle, http://www.goofee.com/erupt.htm, 1997

[Kolb98a] Kolberg, Volker, *Y2K-related critical dates to test*, Y2K Navigation Center, http://privat.schlund.de/v/vk/vk_y2k03.htm, 1998-12-3

[Lewi97] Lewis, David, *The Ultimate Obstacle,* in: „Beyond The Year 2000", Information Strategy (An Economist Group Publication) Special Edition, 1997

[Mitr98] *Critical Date Transitions*, Mitre Organisation, http://www.mitre.org/.../CRITICAL_DATES.html, 1998-12-3

[Offen99] Offenbacher, Steven J., *Building Usable Software Through Early Testing,* John Hopkins University, http://www.apl.jhu.edu, 1999

[Peng93] Peng, Wendy W., Wallace, Dolores R., *Software Error Analysis,* US Department of Commerce, National Institute of Standards and Technology (special publication 500-209), Gaithersburg, March 1993

[PGF96] Park, Robert E., Goethert, Wolfhart B., Florac, William A., *Goal Driven Software Measurement - A Guidebook*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1996

[PN01204] *Polska Norma PN-90/N-01204, Numeryczne zapisywanie dat i czasu dnia,* Warszawa, 1991

[Seid97] Seidel, Roland, *The Calendar,* http://www.skeptics.com.au/journal/calendar.htm, 14/4/1997

[Stock98b] Stockton, J.R., *Critical and Significant Dates,* http://www.merlyn.demon.co.uk, 1998-11-29

[Tate97] Tate, Paul, *The Day After*, in : Beyond The Year 2000, Information Strategy (An Economist Group Publication) Special Edition, London, 1997

[Ulri97] Ulrich, William M., Hayes, Ian S., *The Year 2000 Software Crisis - Challenge of the Century*, New Jersey, 1997

[USA55] *U.S. Department of State Bulletin*, 11/4/1955

[WCA45] *History of the World Calendar Association*, Journal of Calendar Reform, tom XV, no. 4,1945

[Yourd97] Yourdon, Edward, *Why are Year 2000 Projects So Difficult and Risky?,* Year 2000 Journal, vol. 1, no. 6, November/December 1997