THE ORIGIN OF THE YEAR 2000 DATE PROBLEM: AN ALTERNATIVE HYPOTHESIS

Richard Kingsford[22]
Parliamentary Information Systems Office
PARLIAMENT OF AUSTRALIA

Leone Dunn[1]
School of Information Technology and Computer Science
University of Wollongong, Australia

ABSTRACT

This paper aims to identify the primary origin of the year 2000 date problem, in which many computer-based systems may fail because they cannot correctly interpret two-digit year dates which lie in and after the year 2000. The paper proposes a hypothesis within a cultural anthropological framework that the fundamental origin of the problem lay in a pre-existing, commonplace norm of Western culture - the pervasive use of two-digit year dates. This cultural practice set Western society on a collision course with the technology of the future. The cultural norm underpinned and influenced the actions taken within the IT industry from the 1960s onwards, which laid the physical foundation for the problem.

KEYWORDS

year 2000, Y2K, century, millennium, information technology, information systems, human-computer interaction, culture, society, anthropology

INTRODUCTION

The impact which technology can have on society has been widely recognised (Drucker 1970; Kranzberg and Davenport 1972). However it has also been demonstrated that social and cultural factors may have a formative or shaping influence on technology (Bijker et al 1987; Bijker 1995; MacKenzie and Wajcman 1987). This latter insight underpins the hypothesis proposed in this paper. The social factors may be ones which pervade a whole society, or more localised aspects. In this context "technology" is taken to include associated human actions and knowledge, and does not simply refer to "hardware".

The year 2000 date problem (or "Y2K bug", century problem or "millennium bug") refers to a serious issue which many nations of the world have faced in recent years. Many computer-based information systems, and devices containing embedded computing systems, may fail because they are unable to correctly interpret two-digit year dates which lie in and beyond the year 2000. The immediate cause of the problem is that the developers of such systems used two-digit dates in their systems, instead of full four-digit year dates. This means that if in a system "10/05/01" is intended to mean "10 May 2001", the system is likely to interpret it as something different, often "10 May 1901". This is likely to cause the system to fail, often with uncertain consequences (e.g. de Jager 1993, 1997, 1999; de Jager and Bergeon 1997; Gartner 1997; Jones 1998a; Schick 1995; Ulrich and Hayes 1997; Yourdon and Yourdon 1998). The problem was probably first encountered by the banking and insurance industries from the 1960s onwards, when some systems were required to handle dates 30 years or more into the future (de Jager 1998b; IBM 1996; Keogh 1997). Many such companies therefore ensured that such systems could handle four-digit dates. However it has been estimated that year 2000 compliant systems amounted to around only one in twenty of all systems (Jones 1998b).

Embedded systems refer to devices which contain microprocessors or microcontrollers which contain hard-wired instructions. These devices are installed in diverse equipment including vehicles, aircraft, alarm systems, factory machines and robots. Many such devices utilise two-digit dates, and there are estimated to be many millions world-wide which may experience problems (Bylinsky 1998; Gartner 1997; Greif 1997).

It has been estimated that the Australian government will need to spend $600 million to correct "mission-critical" systems alone, while the total bill may be $3 billion (Davis 1998). The full cost of the problem in Australia has been estimated at $5 to 10 billion (Australian 1998; Financial Review 1998; Binning 1998). The US government cost is likely to be $6.4 billion - however some estimates place it at $10 to 12 billion (Luening 1999). The US Defense department alone may spend up to $4 billion (Burnett 1998). Large US businesses may

[22] This paper is a revised version of a technical report written by the authors (Kingsford and Dunn 1998). It is not based in any way on the authors' current places of employment.

spend up to $50 billion in all. It has been variously estimated that the world-wide cost will be $US200 to 800 billion (DCI 1999; Luening 1999; Strassman 1998).

Various possible outcomes have been forecast, ranging from inconvenience to more serious aftermaths (Ulrich and Hayes 1997; Yourdon and Yourdon 1998). In one recent US survey, 93 per cent of responding organisations said that if they did nothing, they would be damaged (Luening 1998a). Businesses and governments are interdependent, and it has been suggested that year 2000 problems may have adverse repercussions. The legal ramifications have also been canvassed (Australian 1998; Dunn 1998).

Many of the systems developed since the 1960s use two-digit dates, but some very recent IT work does also. Some IT schools continued using two-digit dates in teaching into the 1990s (Morgan Stanley 1997). Recent client/server systems are generally better than earlier systems, but are not immune (Ghosh 1997). In late 1998 it was estimated that 64 per cent of some 5000 commercial software packages had year 2000 problems (Gifford 1998a). One practitioner found that much Java code on the Internet was not compliant (Amon 1998a). Of around 250 million personal computers (PCs) in use, many are not year 2000 compliant, and even many new ones have problems (Australian 1998; Feilder 1997; Ulrich and Hayes 1997). In late 1998 it was estimated by a consultant that 11 per cent of new PCs would fail on year 2000 hardware tests, while *all* would fail the British standard test for operating systems (Gifford 1998a).

In 1991 a large consulting firm installed a non-compliant system for a US retail chain. The design "explicitly set out a two-digit year dating format", but both parties were satisfied at the time (Luening 1998c; Overby 1998). The authors examined systems developed in 1997 which were not compliant. When asked, the developer of one explained the use of two-digit dates by saying, "They just looked alright, and I didn't really think about them." Some programmers have even re-coded two-digit dates back into systems which were already compliant (de Jager 1998a). A pertinent question is why the managers in such cases did not enforce the use of four digits. In summary, the issue has been seen as "a dust-like problem in which two-digit dates have been blown into every nook and cranny of technological societies" (Dunn 1998).

To achieve accuracy and quality, system development and IT work in general requires close attention to detail. How should we explain the widespread neglect of dates? As only around one in twenty developed systems were year 2000 compliant, were about 95 per cent of IT practitioners incompetent or negligent? This has been suggested by some, and some negligence may of course have occurred (Keogh 1997). Some have gone further and implied that the IT profession may, or should be held responsible in some way (Hayes 1997; Ratcliffe 1998). For example: "While the all-pervasive nature of the problem has become apparent, there is still a widespread belief that the blame for the whole mess should be sheeted back to the IT industry" (Connors 1998). It has been suggested that IT professionals may ultimately pay a price: "But once the crisis is over, all the anger now building up will be released" (M. Vitale, quoted in Connors 1998). The hypothesis presented in this paper suggests that such inferences are unwarranted.

CULTURAL HYPOTHESIS

This paper proposes the hypothesis that the primary origin of the year 2000 problem lay in a commonplace cultural norm which has existed in many Western nations since the late nineteenth century or earlier, namely the widespread use of two-digit dates, especially in written form. This norm has underlain all subsequent human-computer interaction, in its broadest sense. This hypothesis draws on anthropology in terms of identifying culture as a set of widely shared values, behaviours, beliefs and norms (e.g. Harris 1975; Spradley and McCurdy 1975). Cultural traits are deep-seated and usually long-established, and culture has a powerful influence on the actions of its members. The elements of a culture are often strongly influenced by its history and the shared experiences of its members. Conversely, the culture of a group is often a formative influence on events which contribute to its history.

This paper refers to the aggregated societies of Western nations as "Western society", and the set of common elements of the cultures of Western nations as "Western culture". Some may think of Western culture in terms of great art, architecture, classical music, opera and so on. However to anthropologists culture is just as much about unobtrusive, widely shared elements, which can strongly influence people's everyday behaviour because they *are* commonplace. Such aspects may be undocumented and even rarely articulated, but they nevertheless seem natural and familiar to members of the culture. Every time actions are influenced by such a cultural norm, this process also reinforces the legitimacy of the norm. This paper identifies the widespread use of two-digit dates as a common cultural practice in many countries - mainly Western - which use the Gregorian calendar. The origin of the practice is beyond the scope of this paper. The convention influenced all people concerned, including analysts, programmers, engineers and users. Common European and Australian practice is to abbreviate dates in

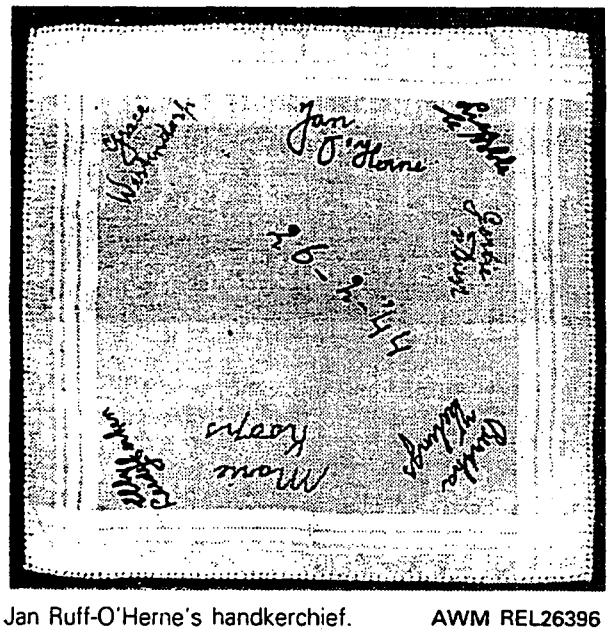
the form "dd/mm/yy" or similar. A common US practice is to abbreviate them as "mm/dd/yy", while Canada appears to use both forms. In each case members of the respective societies readily make the inferential leap to the full date with unconscious ease. Two-digit dates are as much a part of human behaviour as of technology.

There are few direct studies of behaviour in the IT workplaces of the 1960s and 1970s, and few if any studies in any period of past behaviour and attitudes regarding two-digit dates. This implies that any real understanding of such behaviour must now rely largely on the recollections of people who were there. The cultural hypothesis is based on published evidence, on the authors' own experience in the IT industry, and on discussions with veterans of IT from the 1960s onwards. Qualitative and anecdotal evidence are relevant and valuable in this case. However the year 2000 problem received extensive publicity in the late 1990s, often implying that past actions were at least short-sighted (Hill 1998; IBM 1996). This has led to some tendency to rationalise and justify what are now widely seen as past mistakes. IT work in general is analytical, logical and precise, and many find it hard to understand how something now seen as misguided could have occurred so widely - hence there is a strong emphasis on explanations within a "rational" genre such as storage limitations. This issue is examined further below.

Outside the realm of computing, two-digit dates have been used extensively since at least the late nineteenth century. Some suggest even earlier: "Since at least the early 1800's the culture had standardized on the two digit year" (Carmichael 1998). The prospectors in the California gold rush of 1849 were commonly referred to as "Forty-Niners" (Van Steenwyk 1991). An Australian magazine commented in 1898, "Let's see, this year's '98; and next year's '99; and the year after - '00?" (Bulletin 1898). The account records of a small Australian business from the late 1890s were viewed by the authors. The records were written by hand with two-digit dates, and they progressed smoothly from 98, 99 to 00 - obviously the owner had no problem in making the inferential leap to 1898, 1899 and 1900. Hollerith punched cards were also in common use from late last century until the 1970s, and often recorded two-digit dates (Austrian 1982; Morgan Stanley 1997). This helped minimise storage to some extent, but it also reflected common practice.

Later collections of business and personal letters, records of meetings, cheques, paper forms and manual date stamps were viewed, and again there was a definite tendency over many decades to use two-digit dates (see also Ulrich and Hayes 1997). More recent documents sometimes reflect the fact that word processors can generate four-digit dates - although most still offer two-digit dates as options even now.

As one example, a prisoner in World War Two in Java received a signed handkerchief from fellow prisoners, bearing the date they were taken away by the Japanese - this is now in the Australian War Memorial (Begbie 1998) (see Figure 1).



Jan Ruff-O'Herne's handkerchief. AWM REL26396

Figure 1: Handkerchief, World War Two (Begbie 1998)

Some significant but brief suggestions have been made which imply support for the cultural hypothesis. For example: "Computer software generally stores dates as two-digit numbers (for no better reason than social convention)" (P. Bailes, quoted in Mathieson 1998). There are similar comments, although these often go on to nominate other principal causes of the year 2000 problem. For example, it has been noted that two-digit dates were already a shorthand habit for our society in the 1950s (Ulrich and Hayes 1997). One suggestion is that "People think and work in two digit dates. Computers think the way people work" (Microsoft spokesperson, quoted in Luening 1998b). Another comment is that "In our society, people automatically represent years in two digits" (Microsoft 1998). One source suggests that "human nature seems to dictate that writing and keyboard entry of two digits is more practical than four digits" (IBM 1996). The reference to human nature serves to underline how deep-seated the practice is - but it is a component of Western culture, not human nature. It is human nature for people to communicate. It is also commonplace to use shared abbreviations, but just how this is done is an aspect of culture. For example, abbreviations such as "Dr" and "Mr" are common, and are often also embedded in computer systems. It is sometimes implied that the convention of two-digit dates actually stemmed from IT decisions to use them in earlier years (Keogh 1997; Microsoft 1998; N. Weiss-Dolev, in Radio National 1997). In reality it was surely the other way around.

MAIN EXPLANATIONS THAT HAVE BEEN SUGGESTED

Various suggestions have been put forward to explain the use and continued existence of two-digit dates in computer systems (e.g. Gerner 1997). The main ones focus on the demands of users, the need to conserve storage, the lack of an official date standard, the focus on core priorities, short-term planning, the demands of system maintenance, the high mobility of IT staff, the impact of end user computing, the force of precedent and denial by managers. These are discussed further below. Some suggestions acknowledge that over the last 40 years various inter-related technological and business decisions and actions were taken which combined to physically create the problem. It can be demonstrated that these actions were not inevitable, and that in many cases there were feasible alternatives entailing the use of four-digit dates. In most instances these options were not taken up. In all aspects of human-computer interaction the cultural norm of two-digit dates was a common underlying thread, which essentially implied, "If it seems alright, it *is* alright" - and it influenced behaviour.

User demand

User demand has been suggested as a key causative factor in the year 2000 problem (Gerner 1997; Swaine 1998). On this argument, two-digit dates were specified by customers to reflect their everyday practice. While there is some validity in this, the organisation of work in the early years of business computing is also relevant (see also Haselkorn 1998). This was the period of development of many large systems which survived for decades as "legacy" systems, and which used two-digit dates. In the 1960s and early 1970s IT was a different world in many ways. Mainframe computers dominated, and it was primarily larger organisations which developed software. In the experience of the authors and their informants, in many such organisations there was typically a marked role division between systems analysts and programmers. The analysts interacted with business users and, acting on behalf of the users, wrote specifications for programmers to follow. The analysts specified the two-digit dates, often in explicit detail in file and report layouts. Programming was then a specialised technical craft, and analysts quite often did not fully understand the internal working of the computer. The analysts' skills were typically oriented mainly to the business of the organisation, while technical IT knowledge was left to the programmers.

It has been suggested that the IT industry had "no adult supervision", and that this contributed to the year 2000 problem (L. Kappelman, quoted in Hayes and Bullers 1998). However systems were not developed by programmers alone. While it is true that programmers were often aged around 18 to 25 years, the analysts and user managers were more usually aged 25 to 50 years. Analysts generally had higher relative status than programmers, who often simply did as instructed (see also C. Jones, in Foremski 1998). The IT work environment of the early period was typically quite hierarchically structured - often more so than in later years.

There are indications that some IT practitioners did raise early concerns about the century risk, but were generally treated with disbelief (Bemer 1979; Morgan Stanley 1997). Being removed from users and management, programmers had little influence. One college IT student brought up the problem in class in 1974, but was greeted with "stunned silence then... incredulous laughter" by colleagues, even though they understood it (Guy 1998). Anthropologists recognise this as a common reaction of people confronted by a concept outside their normal frame of reference. An IT practitioner developed and attempted to market a windowing tool to address the problem in 1983, but was "repeatedly shown the door, or laughed at" (Foremski 1998). The year

2000 issue at the time was "acknowledged occasionally in trade conference jokes and DP shop banter but... attracted little serious attention" (Gillin 1984).
On this interpretation, the two-digit dates were specified mainly by those who did not fully understand their implications. Those who could have better understood the issue generally lacked decision-making power. Some members of the latter group did occasionally voice concerns, but this was relatively rare - implying that two-digit dates generally appeared acceptable or at least innocuous to most of these people also.
From the 1970s the separate roles of analysts and programmers began to merge in many IT shops into the profession of analyst/programmer, narrowing the gap between user and developer. However the use of two-digit dates continued. A significant factor was the underlying cultural norm which influenced both perceptions and behaviour.

Storage limitations

The need to save on storage has been widely publicised as an explanation and is an oft-repeated mantra, amounting almost to "received wisdom". This proposition assumes that from the 1960s business computing was driven by the storage limitations of punched cards, paper tape, memory and magnetic tape and disk (e.g. de Jager 1993, 1997; Gerner 1997; IBM 1996; Jones 1998a; Keogh 1997; Microsoft 1998; Ulrich and Hayes 1997). The need to conserve storage steadily declined with hardware advances from the 1970s onwards (E. Yourdon, in Hayes and Bullers 1998). However the use of two-digit dates did not abate.
There was no official date format standard in the 1960s and 1970s (Gerner 1997), and no overall coordinated strategy for date storage. Given the extensive spread of IT in Western countries, this implies that a very large number of independent decisions were taken to store two-digit dates. In fact a wide variety of date formats have been observed, but many share the common factor of two-digit years. It is now not possible to directly establish the reasons underlying most decisions to use such dates. It is probable that most were *not* taken with a specific aim of conserving storage (see also Haselkorn 1998). The IT veterans consulted invariably said they did not think much about the issue at the time, sometimes adding that the dates were specified by others anyway. None said that they had been aware of any conscious decision to use two-digit dates for the explicit purpose of conserving storage. But most IT practitioners also failed to clearly draw attention to potential problems. There are factors indicating that this situation was widespread, and these are discussed below.
Punched cards have been in use since the late nineteenth century (Heide 1997). It has been suggested that as cards are of fixed size, there was little option but to use two-digit dates, and that this practice was carried over into computer systems (Chorafas 1998; de Jager 1997). Was this the full story? A typical business application would require the recording of a reference number, one or two dates, a name and address of 60 characters or more, and further details. The non-date information typically overwhelmed the dates in space occupied. Economising on dates occasionally prevented a record overflowing a card, but rarely saved much overall - and in any case was not done when there was a business need for four-digits. However there was a cultural norm which favoured two-digit dates, and this was an underlying driver.
Even when cards carrying pre-set two-digit dates were used as input to a computer system, programmers still had the option to implement simple windowing logic to interpret them in context as four-digit dates, replicating human reasoning (Moore and Foley 1998). (The term "windowing" is used here to mean the process of interpreting a two-digit date as a four-digit date within a window of 100 years which may span two centuries - regardless of how the date is subsequently stored.) For example, in 1965 a birth date of "10/05/90" would be interpreted as 10 May 1890, while "10/05/40" would be 10 May 1940. Any ambiguities - such as "10/05/60" - would generate error reports to be clarified later (this was common practice for other purposes). But programmers only rarely coded such logic as they, like everyone else, were influenced by the prevailing cultural norm.
In the early period of IT, card records were sometimes copied intact to tape or disk as an initial step. The systems would often then process these "card images" instead of actual cards. This was done mainly to reduce the time taken for a deck to pass through the card reader. However it may also have tended to reinforce the apparent legitimacy of using two-digit dates in magnetic storage. From the 1960s direct key-to-tape and key-to-disk systems began to displace punched cards for data entry, and became the precursors of later on-line systems. This tends to diminish any argument that there was no alternative to two-digit dates because cards were "unstretchable". Even when two-digit dates were keyed in, programmers had the same option to use windowing, but rarely did. It has been suggested that two-digit dates were used to save keying effort by operators (de Jager 1999; Keogh 1997). However dates typically comprised relatively little of the keying workload, and this is also a tenuous argument when applied retrospectively to a period prior to widespread awareness of repetitive strain injury. In fact the operators usually entered two-digit dates directly from forms designed by analysts or users, originating in the business realm. The cultural norm underlay such decisions.

The use of two-digit dates in magnetic storage was not inevitable. Again, the non-date information typically overwhelmed the dates in terms of space. Memory was very limited, but generally only relatively few records were held in memory at any one time. Most systems afflicted by the year 2000 problem stored two-digit dates on tape, disk or drum as six text characters, such as "100568" for 10 May 1968. A preliminary IEEE statement commented "It is wrong to believe that the Year 2000 date problem [resulted] simply from a rational but short-sighted decision to save memory and storage space by using two digit years rather than four. The problem is far more complex... much more could have been saved by boiling these dates down to binary" (Haselkorn 1998). There are various ways of compressing four-digit dates in magnetic storage (IBM 1996; Moore and Foley 1998). In the 1960s and 1970s, most IT shops employed programmers skilled in assembly language, even when high-level languages were in use such as COBOL (COmmon Business-Oriented Language). Assembler and similar low-level languages were widely used to manipulate data at the binary digit (or bit) level. If saving on date storage had been a conscious aim, it would have been a trivial exercise to write an assembler routine to compress four-digit dates into bits occupying the same or less space than two-digit dates expressed in characters. This was confirmed by former assembler programmers consulted, who frequently faced far more difficult problems. A typical example (only) of a simple compressed date format is:

	Range	*Bits required*	*Stored in*
Day	1 to 31	5 bits	8 bits
Month	1 to 12	4 bits	8 bits
Year	1 to 9999	14 bits	16 bits
			32 bits total

That is, in a byte-addressable computer similar to an IBM machine, a four-digit date could be compressed into four bytes (32 bits) or less. (The example above would fail within 8000 years.) Similar binary compression schemes could be implemented in other computers, depending on their basic architecture. Various other techniques were also possible, such as the use of Julian dates, that is, a count of days forward from a fixed starting point, with the result stored as a binary integer (Moore and Foley 1998).

Such techniques could maintain human readability by expanding all output dates. It has been suggested that the computation time needed to compress and expand dates would have significantly extended batch processing times (Ted 1998). However it is unlikely this would have been excessive, and it would be about the same as the inbuilt date compression routines which some COBOL compilers offered, but which were also rarely used in practice. The simple answer is, if binary date compression had been recognised as a business need, the batch runs would have been adapted to it - just as they accommodated the much slower overall processing rate of the period. A date compression routine could be used both in programs written in assembler, and also in high-level languages. COBOL had the ability to invoke assembler routines from within a program - and this was commonly done for other purposes, such as calculating a "checksum" used to help confirm that records were written and read correctly. Some COBOL compilers had only limited date formats, but "date" formats as such were never essential for date processing - many programmers created their own (see also Haselkorn 1998). It can also be noted that whenever date calculations were needed, a date in characters had to be converted to a numeric form in any case, and this took time - again diminishing any argument that date conversion would have been too time-consuming.

Within the "rational" school, storage limitations are seen as a key causative factor: "A problem of rational origin that requires a rational solution" (Kappelman and Cappel 1996). It has been suggested that the problem stemmed from IT professionals applying "sound business judgement" in a period of limited and expensive storage: "It is estimated that conserving precious disk space with the two-digit year format saved the typical organization over $1,000,000 per gigabyte of total storage in the 30 year period from 1963-1992 (in 1992 dollars)... To mitigate the potential response of anger from top management, IS managers need to emphasize the rational origin of the problem" (Kappelman and Cappel 1996). However studies within this genre beg the question of motive and intent. If such savings were achieved deliberately, why were equal or greater economies not achieved through date compression - which would have simultaneously prevented the year 2000 problem? There is evidence that conscious decisions to use two-digit dates with the aim of conserving storage were relatively rare, and if so, calculations of imputed monetary savings have less relevance.

The occasional cases of the use of one-digit year dates underline the fact that storage was limited. Some US defence organisations and car manufacturers used such systems in the 1970s, mostly redeveloped from systems using two-digit dates first written in the 1960s (Gates 1998; Gerner 1997). Such decisions were therefore taken against a baseline of the use of two-digit dates as a "given", and this influenced thinking. These systems no doubt included forward planning applications which stored dates some years ahead. Problems inevitably soon emerged when dates in the 1980s were encountered. At that time, systems were not as widely interconnected as they were later, and correction was simpler. However these instances could be seen as demonstrating negligence,

in that conscious decisions were made to use one-digit dates while knowing that problems would shortly arise. If the use of one-digit dates had in fact been widespread, such negligence would also have been widespread - and the accusations of incompetence directed at the IT industry would be justified. However on the basis of published reports, actual instances of the use of one-digit dates were relatively limited.
The techniques and skills were available to use binary date compression from the earliest years of IT. However such techniques were rarely used, and mostly only when there was a known business need for four-digit dates, as in some banks. The UNIX operating system was first developed in 1969, and it was also designed to use compressed four-digit Julian dates (Bach 1986). However some UNIX system utilities and services have date problems. Application systems developed to run on UNIX, or migrated to UNIX, are also often not immune (Inman 1998).
The basic question is not so much why storage needed to be conserved - this is unarguable. The more pertinent question is why two-digit dates were so widely and thoroughly *accepted* without question as a valid abbreviation, which tended to ensure that binary date compression was rarely considered. This argument does not deny the possibility that in the early decades of IT some practitioners - perhaps many - noticed the use of two-digit dates and, in a prevailing climate of storage minimisation, assumed that the dates were an aspect of this practice. However this does not logically imply that deliberate decisions were made to use two-digit dates for the explicit purpose of economising on storage. Such decisions were rarely made. It was primarily the underlying cultural norm which informed the decisions to use two digits, and this also helped legitimise the actions taken by the parties involved.
It is acknowledged that the informants for this paper represent only a very small minority of the millions of IT practitioners whose work entailed date storage. However if it could be demonstrated that many of these people consciously used two-digit dates with the explicit aim of saving storage, the IT profession could then legitimately be asked to explain why it did not widely adopt binary date compression, which would have both saved on storage and "future proofed" systems against the year 2000 problem.

Lack of official date standard

The lack of any widespread official date standard has been suggested as a factor in the year 2000 problem. There has been little consistency among date formats used, apart from the common factor of two-digit years. Some localised standards have existed, such as a US government standard which in 1968 recommended the use of two-digit dates. In 1971 this standard advised, but did not mandate the use of four-digit dates (Landers 1998). By 1988 the standard still only "highly recommended" them (Peckinpaugh 1999). There was no international four-digit date standard until 1988 (ISO 8601) and this was rarely mandated or widely accepted anyway (Gerner 1997; Haselkorn 1998; Jones 1998a). This was partly related to the fact that there were differences in informal date formats among countries, for example common practice in Europe and Australia is day/month/year, whereas one common North American practice is month/day/year. Neither of these match the "official" standard, which is year/month/day. This reflects the difficulty of attempting to impose a standard which conflicts with pre-existing conventions.
The limited adoption of the standard was also compounded by the fact that the IT industry has historically had difficulty in agreeing on *any* standards (Landers 1998). This largely stemmed from a widespread adoption of minimum standardisation wherever possible by IT vendors, to gain commercial advantage. This entailed the deliberate use of proprietary standards, aimed at "locking in" customers to a vendor. For example, IBM, a major player in the period 1960 to 1980, wrote "basic principles" for itself in 1968 which included: "Implementation of industry standards is a business decision based upon market considerations. Standards are "minimums" - do not preclude added features nor product different from the standards" (DeLamarter 1986). This led to a general acceptance among IT practitioners of the prevailing lack of uniformity. In summary, while it is true that if there *had* been a mandatory four-digit standard the year 2000 problem might not have emerged, this in itself does not explain the widespread adoption of two-digit dates. In the absence of official standards, everyday conventions took over.

Focus on core priorities

A focus on "core priorities", to the relative exclusion of other issues, has been suggested as a factor (Gartner 1997; Gerner 1997). This is related to a focus on productivity and profit (or in the public sector, cost saving). In fact this general approach was applied in all professions over the history of IT. Thus analysts, designers and programmers concentrated on the issues they saw as most important to achieve their respective goals. These usually related to getting the accounts correct, or the refinery operating and so on - and not to how the dates were stored. For programmers, the main focus was often simply on getting the program to run. It is sometimes suggested that the use of two-digit dates saved effort for programmers. One IT consultant wrote, "However... I

have found, compliant code would have been just as easy, if not easier, to produce. This confirms to me that most programmers, like most of the population, haven't given this matter any thought at all" (Amon 1998a). Another consultant said, "People in the software industry have not paid attention to this problem. It's an issue that people have completely overlooked" (quoted in Overby 1998). The authors of IT textbooks also did not mention the issue until very recent years (Foremski 1998). The underlying cultural norm shifted the focus away from two-digit dates. In only relatively few cases a focus on core business *did* in fact entail attention to dates and the use of four-digit dates.

Short-term planning

Short-term planning is a related issue, and this has been common practice in both business and IT (Gartner 1997). It has been suggested that in the 1960s and 1970s there was a view that the use of two-digit dates would not be a problem for 20 years or more (de Jager 1997; Keogh 1997). However there have always been some systems which stored future dates - sometimes far distant. It has also been suggested that systems were then not expected to last for 20 years or more, and therefore their use of two-digit dates did not matter (de Jager 1999; Gerner 1997; Ulrich and Hayes 1997). This may have been so in some cases, but often no such assumptions were made. One programmer wrote, "The year 2000 was 20 years away, and we programmers were in unspoken agreement that our programs would be well and truly replaced..." (Armstrong 1999). But this suggests a post hoc rationalisation. Even if any "unspoken agreement" did in fact exist, the writer could not know how widely it was shared, if at all. What did "replacement" really mean? It was expected that the hardware would be replaced, but not necessarily the application software and especially the data (Ulrich and Hayes 1997). From the earliest years many data holdings - often containing two-digit dates - were retained relatively intact while both hardware and application code were updated. For example, in the 1960s many assembler programs were converted to COBOL, leaving the master data files with their dates intact - and these were clearly visible to analysts and programmers.
Such arguments are also countered by the fact that it became apparent quite early that many systems were surviving for long periods. In fact by the late 1970s there were clear signs that legacy systems were emerging (Keogh 1997). For example, IBM's MVS operating system was then approaching 20 years old - and is still running (Jones 1998b). By the 1980s, the Australian government was still using systems developed in the 1960s. One pertinent comment is, "There's a myth in our industry that we change software every five or seven years. But the real systems, the ones that really drive our business, do not get changed every year. They don't get changed every ten years. The air traffic control system [in the US] is more than 20 years old" (P. de Jager in Radio National 1997). Some accounting systems have lasted 30 to 40 years, and 10 to 20 years is common in manufacturing (McKenney et al 1995). In the early decades of IT there was usually no way of forecasting how long any particular system would last, and to suggest otherwise is simply rationalising after the event.
The influx of end user computing in the 1980s led to rising pressures to replace legacy systems, driven largely by the availability of improved user interfaces and better access to and control over information. In some cases legacy systems were "encapsulated" in new interfaces, leaving their two-digit dates intact. In other cases a system was redeveloped in relational database or client/server form, and the old data or data model migrated to the new system (Gartner 1997; Johnston 1998). In both cases either dates or date formats were visible to developers or testers, but were often ignored. Sometimes the survival of an older system was necessitated by the failure of a redevelopment project, but there was often also a reluctance to disturb systems which met their core business needs. The cultural norm minimised the perceived importance of two-digit dates, and prolonged their use.

System maintenance

System maintenance could have rectified the problem, once a system using two-digit dates survived. However maintenance usually only addressed the problem if there was a recognised business need for four-digit dates. But this was rare, and there was often a large backlog of other more urgent corrections needed - the IT industry has rarely got it right first time (de Jager 1999; Gibbs 1994). As a result the focus of maintenance was on immediate priorities, not longer-term issues. Maintenance has also typically had lower status and prestige than system development, and been allowed only limited funding (Haselkorn 1998). As the years advanced, two-digit dates became progressively more deeply embedded in systems. This made it increasingly difficult to address the problem through routine maintenance, as opposed to special projects - but such projects were rare. More recently, downsizing, budget cuts and outsourcing have often compounded the problem (Gerner 1997). A related suggestion is that some IT shops may have waited until a "silver bullet" emerged to solve the year 2000 problem, despite that fact that this was always highly unlikely (Keogh 1997; Ulrich and Hayes 1997). All of these factors exacerbated the year 2000 problem, but the underlying convention of two-digit dates was also a significant influence.

High mobility of IT professionals

The high mobility of IT professionals among jobs has been seen as implying that IT staff did not care about date problems that might emerge after they left an organisation (de Jager 1997). The same observed mobility could equally imply that year 2000 awareness should have spread more than it did. That is, IT professionals who had experienced the date issue in one organisation should have taken that awareness to the next. However a curious but genuine "amnesia" seemed to afflict many people after they moved to another job, if their new role did not specifically target four-digit dates. For example, in the mid-1990s an IT quality inspector worked at a site employing 200 IT staff. She held regular meetings of analysts and programmers to "work through all... specifications to detect any problems and oversights. In the six months I held this position I worked with 90% of all programming staff, reviewed 90% of all work to be done and regularly attended meetings with management on quality issues. In that time the year 2000 issue was never raised - not by myself, not by management and not by any of the programmers or analysts" (Amon 1998a). It is improbable that none of these IT professionals had ever encountered date issues before. The authors have experienced similar effects, and believe that the cultural norm shifted the focus away from two-digit dates.

End user computing

The end user computing wave which followed the microcomputer revolution of the 1980s introduced widespread systems development by non-IT professionals. Such developers were relatively shielded from the internal complexity of computers. Nevertheless they certainly saw and used two-digit dates, which were often offered as options in end user support products such as system generators - and this reinforced their legitimacy. Some such products used windowing in their system software. Not all applications so generated were fully year 2000 compliant, unless the business context was accurately reflected in the application. The client/server computing trend of the 1990s again emphasised the role of users in system development (Adhikari 1994), and it is known that many such systems also have year 2000 problems (Johnston 1998). The underlying cultural norm served to promote and legitimise the use of two-digit dates.

Force of precedent

The force of precedent exacerbated the year 2000 problem, once it had become entrenched. This was related to the development of new systems by integrating into them pre-existing components including code modules, programs, designs, whole systems and data holdings, and this occurred at a steadily increasing rate from the 1970s onwards (Freeman 1987; Gerner 1997). Even in the 1960s, established business procedures such as the use of punched cards had helped shape later systems. If a pre-existing component used two-digit dates, the new system often did so as well. Related to the integration process were the need for the compatibility of new versions of a product with previous versions, the encapsulation of legacy systems and legacy data within new user interfaces, the integration of entire systems into larger networks, and the development of embedded systems. In general these processes all entailed the integration of existing components into new systems. In fact the history of computing largely entails a progression of ever more complex integration of components. As just one example, remnants of a BASIC interpreter some 16 years old were found at the heart of a modern PC: "The computer was suddenly revealed as... a series of skins, layer on layer... We build our computers the way we build our cities - over time, without a plan, on top of ruins" (Ullman 1998). The PC itself is a legacy system - little wonder that manufacturers have found it difficult to achieve full year 2000 compliance.

Designers and developers often treated pre-existing components as "black boxes", and did not look inside them. However they necessarily had to attend to the interfaces among system components. If two-digit dates were used in a "black box" they usually crossed its interfaces in some way. Either dates or date formats were visible to developers or testers - but were often seen as insignificant and not relevant to the core priorities, and overlooked. The cultural norm influenced such perceptions. There was also pressure in many cases to simply integrate the "black box" without any changes to it. If the developer had actually perceived a serious problem in a component, it may have been either corrected or rejected. This did sometimes happen, but only rarely because of the use of two-digit dates.

The IT revolution has brought dramatic changes to the world economy over the last fifteen years, and this has exacerbated the year 2000 problem. Computing was originally mainly a Western phenomenon, but it later spread to many non-Western nations. There are now many large, diverse and interdependent systems connected together

widely on almost a global scale, often using and exchanging two-digit dates (de Jager 1999; Ulrich and Hayes 1997). The cultural norm has contributed to a neglect of two-digit dates in these systems and their interconnections. Many Asian economies are vulnerable to the year 2000 problem (Economist 1998). However historically many non-Western nations did not use the Western calendar (Eade 1995; O'Neil 1975). The fact that many such nations now have year 2000 problems underlines the extent of their adoption of the Western calendar, and also the extent to which they have derived their systems and business practices from the West.

Denial

Denial that a serious problem exists has been common in recent years, especially among business managers and some IT managers (Amon 1998b; Furman and Marotta 1994; Ulrich and Hayes 1997). Such denial can result in an organisation doing little or nothing. For example, in 1997 few Australian government agencies had fully gauged their "year 2000 exposure" and estimated the costs (ANAO 1997). In late 1997 it was estimated that two-thirds of US companies did not have a year 2000 plan (American Banker 1998). A 1998 survey of US small businesses found that 82 per cent were at risk, but only 23 per cent had taken action (Fysh 1998). The Australian Bureau of Statistics found that by late 1998 about 43 per cent of 6500 businesses surveyed had not started work (Business Today 1998). A New Zealand survey in late 1998 found that fewer than a third of responding organisations were "well placed" to manage their risks effectively. Responses were "along the lines of 'she'll be right'... they 'are under-informed and over-confident'" (Gifford 1998b). The underlying cultural norm acted to diminish the perceived importance of two-digit dates and promote denial - if the dates looked alright, how could there be a major problem?

Denial may in some cases be based on a view that the year 2000 issue is a "racket" or "ruse" driven by the IT industry (Zvegintzov 1996). Such arguments are countered by the large budgets allocated by many technologically astute organisations - evidently for sound business reasons, and with much of it already spent. For example, the Australian telecommunications company Telstra has allocated up to $500 million, while the National Australia Bank has budgeted about $250 million (Financial Review 1998; Gottliebsen 1998).

CONCLUSION

The hypothesis proposed in this paper is based on the fact that a seemingly insignificant, everyday component of culture can nevertheless strongly influence behaviour. One such aspect is the Western cultural norm relating to two-digit dates, which constitutes a common thread underlying the various actions taken within the IT industry over the past four decades. These actions in themselves reinforced the legitimacy of the cultural norm - the dates seemed right, and therefore *were* right. The year 2000 problem resulted from the collision of this long-standing cultural norm with twentieth century technology, a collision which drove two-digit dates deeply into our systems. The agents of this process were ordinary people working in various professions, whose actions were simply influenced by their culture. In all cases it is people who are the bearers and agents of culture. For these same reasons, the convention of two-digit dates is also likely to persist into the next century.

It has been demonstrated that social and cultural factors can shape technology (Bijker et al 1987). Technologies, tools and material artefacts reflect the societies and cultures which create them (Harris 1975; Schusky 1975). One pertinent comment is that "values pervade the invention and production of a technology, and these values are put into technology in such a way that they cannot be separated from it... the technology may therefore be tainted because of the practices, institutions, and people involved in its creation..." (Johnson 1997). It is evident that cultural norms may also pervade technologies and artefacts. A computer-based information system is a prime example of a modern artefact which reflects the culture of both the organisation and the people involved in its creation and use.

The year 2000 problem highlights the importance of bringing a wider perspective to bear on human-computer interaction - an approach which is often overlooked or seen as less "serious" in information system studies. Virtually all information systems are conceived, designed, built and used by people - all of whom invariably bring their own distinct understandings and expectations to their roles. The technology is clearly important, but the human factors also can have a substantial and all too often unforeseen impact on information systems, and on their success or failure. The year 2000 problem has been seen as "part of the flawed integration of technology with society and with... real human beings" (Carmichael 1998).

With regard to dates, the full human requirement has always been to: a) accept and display all dates in forms natural and familiar to users, and b) interpret and process dates accurately at all times and in all contexts. For many computer-based information systems, only part (a) of this requirement was met. The full requirement was rarely even made explicit. This in itself was influenced by the cultural norm, which evidently implied (falsely) that the requirement had been met. Ironically, the manual information systems of the late nineteenth century satisfied the requirement in full. These records often used familiar two-digit dates, which were correctly and

effortlessly interpreted by users. The challenge for IT is to better understand the full range of human needs, and to reflect them more accurately and carefully in systems which are more "human-centred" (Kling and Star 1998). The development of such systems would require significantly improved analysis of social and cultural factors, including values, behaviours and norms.

Will modern civilisation be destroyed by the year 2000 problem? Such an outcome is unlikely. Will society pay a high price for its failure to properly understand and fulfil a basic human requirement of numerous information systems? This is now beyond doubt.

REFERENCES

Adhikari, R. (1994) **Implementing client/server technology**, Charleston: Computer Technology Research Corporation

American Banker (1998) **Year 2000 bankers conference**, American Banker and Strategic Research Institute

Amon, J. (1998a) "The year 2000 problem: a programmer's perspective on the dilemma of the century", **Computerworld NZ**, 20 July

Amon, J. (1998b) "Programmer denial threatens Y2K projects", **Computerworld NZ**, 23 November

ANAO (1997) Managing the year 2000 problem - risk assessment and management in Commonwealth agencies, Australian National Audit Office, Audit Report No. 27

Armstrong, D. (1999) "Not guilty - confessions of a programmer", **The Press (NZ)**, 20 January

Australian (1998) "Y2K: the bug that ate the future", **The Australian**, April-December

Austrian, G. (1982) Herman Hollerith: forgotten giant of information processing, New York: Columbia University Press

Bach, M. (1986) **The design of the UNIX operating system**, Englewood Cliffs: Prentice-Hall

Begbie, R. (1998) "They endured the unimaginable", **Canberra Times**, Panorama, 27 June

Bemer, B. (1979) "Time and the computer", **Interface Age**, February

Bijker, W. (1995) "Sociohistorical technology studies", *In* **Handbook of science and technology studies**, Jasanoff S., Markle, G., Petersen, J. and Pinch, T. (eds.), Sage Publications

Bijker, W., Hughes, T. and Pinch T. (eds.) (1987) **The social construction of technological systems**, Cambridge Mass.: MIT Press

Binning, D. (1998) "Cost of Y2K blows out by more than 100 per cent", **Sydney Morning Herald**, 25 September

Bulletin (1898) Untitled item, **Bulletin** (Sydney), 24 December, p. 28

Burnett, R. (1998) "Defense funds likely to chase bug", **Orlando Sentinel**, 31 October

Business Today (1998) "Australian businesses slow to start work on millennium bug", **Business Today**, 8 December

Bylinsky, G. (1998) "Industry wakes up to the year 2000 menace", **Fortune**, 27 April

Carmichael, D. (1998) Year 2000: who will do what and when will they do it? Towards actions, 23 July, http://www.tmn.com/y2k/y2kwho.htm

Chorafas, D. (1998) **Agent technology handbook**, New York: McGraw-Hill

Connors, E. (1998) "It's a $1 trillion jigsaw puzzle without borders", **Australian Financial Review**, 20 April

Davis, I. (1998) "Canberra's Y2K outlay on the rise", **Australian Financial Review**, 28 April

DCI (1999) "Y2K's worldwide financial impact", **DCI**, 13 January

de Jager, P. (1993) "Doomsday 2000", **Computer World**, 6 September

de Jager, P. (1997) "Computers on strike! You've got to be kidding", **Year/2000 Journal**, January-February

de Jager, P. (1998a) "Keynote address", **Year 2000 bankers conference**, American Banker and Strategic Research Institute

de Jager, P. (1998b) "Financial community leads the Y2K race", **Wall Street Journal**, 24 September

de Jager, P. (1999) "Y2K: so many bugs, so little time", **Scientific American**, January

de Jager, P. and Bergeon, R. (1997) **Managing 00 - surviving the year 2000 computing crisis**, New York: Wiley

DeLamarter, R. (1986) **Big blue: IBM's use and abuse of power**, London: Macmillan

Drucker, P (1970) **Technology, management, and society**, London: Heinemann

Dunn, A. (1998) "Year 2000 bug likely to infest courtrooms", **Los Angeles Times**, 8 September

Eade J. (1995) The calendrical systems of mainland South-East Asia, London: Brill

Economist (1998) "Survey: the millennium bug", **Economist**,19 September

Feilder, K. (1997) "Unloved forgotten: year/2000 issues affecting personal computers", Year/2000 Journal, July-August

Financial Review (1998) "Special report: the millennium bug", **Australian Financial Review**, 20 April

Foremski, T. (1998) "Date change dilemma", **Computer Weekly**, 17 December

Freeman, P. (ed.) (1987) **Tutorial: software reusability**, IEEE Computer Society PressFurman, J. and Marotta, A. (1994) "Year 2000 denial", **Computerworld**, 28 (43), 24 October
Fysh, G. (1998) "Wising up to 'Y2K'", **News Tribune**, 27 September
Gartner (1997) The year 2000 crisis: an enormous challenge that must be addressed, Gartner Group, Strategic Analysis Report, March
Gates, D. (1998) "Before Y2K: the 1980 problem", **PreText Magazine**, May
Gerner, M. (1997) "Why has the year/2000 problem happened?" **Year/2000 Journal**, March-April
Ghosh, D. (1997) "Surprise! client/server has the year/2000 problem, too", **Year/2000 Journal**, November/December
Gibbs W. (1994) "Software's chronic crisis", **Scientific American**, 271(3)
Gifford, A. (1998a) "Bug carries a nasty bite", **NZ Herald**, 6 October
Gifford, A. (1998b) "Y2K - why we should start to worry", **Business Herald**, 18 September
Gillin, P. (1984) "The problem you may not know you have", **Computerworld**, 13 February
Gottliebsen, R. (1998) "Time runs out for Canberra on millennium bug", **The Age**, 18 April
Greif, S. (1997) "Embedded systems: surprising exposures to the year/2000 problem", **Year/2000 Journal**, May-June
Guy, J. (1998) **Y2K fun in 1974**, http://webhome.idirect.com/~joeguy/y2000.htm
Harris, M. (1975) Culture, people, nature: an introduction to general anthropology, New York, Crowell
Haselkorn, M. (1998) **The year 2000 problem**, Preliminary draft technical information statement, version 1.2, IEEE Technical Activities Board, New Technologies Development Committee (www.mindspring.com/~pci-inc/Year2000 in 1998)
Hayes, D. and Bullers, F. (1998) "Chances to fix year 2000 problem squandered for decades", **Kansas City Star**, 26 September
Hayes, F. (1997) "Year 2000: a regular laff riot", **Computerworld**, 31(31) p.101
Heide, L. (1997) "Shaping a technology: American punched card systems 1880-1914", **IEEE Annals of the History of Computing** 19(4)
Hill, D. (1998) "If business is late on Y2K, govt has some blame", **Computerworld NZ**, 28 September
IBM (1996) The year 2000 and 2 digit dates: a guide for planning and implementation, fifth edition, IBM Corporation
Inman, G. (1998) "The year/2000 problem and its importance to UNIX users", **Year/2000 Journal**, September/October
Johnson D. (1997) "Is the global information infrastructure a democratic technology?" **Computers and Society**, 27(3), September
Johnston, L. (1998) "Getting a grip on client/server applications - year/2000 isn't only a mainframe problem", **Year/2000 Journal**, September/October
Jones, C. (1998a) The year 2000 software problem - quantifying the costs and assessing the consequences, Reading, MA: Addison Wesley
Jones, C. (1998b) **Dangerous dates for software applications**, Version 2, 24 March, http://www.year2000.com/y2karchive.html
Kappelman, L. and Cappel, J. (1996) "Facing the year 2000 problem", http://www.coba.unt.edu/bcis/faculty/kappelma/critical.htm (earlier version in **Journal of Systems Management**, July/August 1996)
Keogh, J. (1997) **Solving the year 2000 problem**, Boston: AP Professional
Kingsford, R. and Dunn, L. (1998) **The origin of the year 2000 date problem: an alternative hypothesis**, Technical Report 1998/2, School of Information Technology and Computer Science, University of Wollongong
Kling, R. and Star, S. (1998) "Human centred systems in the perspective of organizational and social informatics", **Computers and Society**, 28(1), March
Kranzberg M. and Davenport, W. (eds.) (1972) **Technology and culture, an anthology**, New York: New American Library
Landers, J. (1998) "Lost chances litter year 2000 bug's path", **Dallas Morning News**, 4 October
Luening, E. (1998a) "Study: Y2K already a problem", **CNET News**, 25 March
Luening, E. (1998b) "Microsoft debuts Y2K strategy", **CNET News**, 15 April
Luening, E. (1998c) "Andersen faces Y2K suit", **CNET News**, 31 August
Luening, E. (1999) "Who will pay the bills?" **CNET News**, 20 January
MacKenzie, D. and Wajcman, J. (1987) **The social shaping of technology,** Milton Keynes: Open University
Mathieson, N. (1998) "Beware the bug!" **Graduate Contact,** University of Queensland, 17

McKenney, J., Copeland, D. and Mason, R. (1995) **Waves of change: business evolution through information technology**, Boston: Harvard Business School Press
Microsoft (1998) **Microsoft year 2000 resource centre**, April 15, http://www.microsoft.com
Moore, R and Foley, D. (1998) "Date compression and year 2000 challenges", **Dr. Dobb's Journal**, May
Morgan Stanley (1997) **Technology: enterprise software**, Morgan Stanley US Investment Research, 25 April, http://www.year2000.com/y2karchive.html
O'Neil, W. (1975) **Time and the calendars**, Sydney: Sydney University Press
Overby, S. (1998) "Andersen Consulting sues to block J. Baker's request it pay for Y2K flaw in systems", **Business Today**, 3 September
Peckinpaugh, C. (1999) "Who to blame for the Y2K problem?" **Federal Computer Week**, 11 January
Radio National (1997) "The millennium "bug": the year 2000 problem", **Background Briefing**, Radio National Transcripts, 23 March
Ratcliffe, M. (1998) "Responsibility: business and tech pros will bear the brunt of blame", **ZDNet News**, 7 Oct
Schick, K. (1995) **Three certainties: death, taxes and the year 2000**, Gartner Group, Application Development and Management Strategies, Research Note, Strategic Planning SPA-215-1116, 25 January
Schusky, E. (1975) **The study of cultural anthropology**, New York: Holt, Rinehart and Winston
Spradley J. and McCurdy, D. (1975) **Anthropology: the cultural perspective**, New York: Wiley
Strassman, P. (1998) "Hard data (at last) on year 2000 costs", **Computer World**, 15 May
Swaine, M. (1998) "A chat with Bob Bemer", **Dr. Dobb's Journal**, 23(5) p.115
Ted, B. (1998) "The mailbox is full of ideas about the causes and effects of the year-2000 problem", **InfoWorld**, 20(10) p.108
Ullman, E. (1998) "The dumbing-down of programming", **Salon**, 13 May
Ulrich, W. and Hayes, I. (1997) **The year 2000 software crisis - challenge of the century**, Upper Saddle River: Prentice Hall
Van Steenwyk, E. (1991) The California gold rush: west with the Forty-Niners, New York: Watts
Yourdon, E. and Yourdon, J. (1998) **Time bomb 2000: what the year 2000 computer crisis means to you**, Upper Saddle River, NJ: Prentice Hall
Zvegintzov, N. (1996) "The year 2000 as racket and ruse", **American Programmer**, February