International Journal of Information Technology and Knowledge Management
January-June 2008, Volume 1, No. 1, pp. 71-78

REENGINEERING COST FOR Y2K CHALLENGE

Rajesh Verma & R. K. Chauhan

Abstract

Y2K projects tend to be complex; time consuming and risky as well as they can cause considerable cost because they belong to the class of corrective maintenance projects. Examples of beneficial aspects range from speeding up initiatives such as reengineering because of Y2K projects, to finding out-of-date systems that can be phased out, with longer term saving on maintenance contract, hardware capacity etc. This article is intended to highlight the critical technical, legal and regulatory issues which need to be addressed and the potential role of the Year 2000 Commission in meeting that challenge. This article will discuss (a) the scope of the Year 2000 problem at the business industry (b) the status of reengineering efforts, (c) technical, legal and regulatory issues which may influence how the business industries have approached Year 2000 corrective efforts. (d) Benefits of Y2K project from an enterprise-wide perspective

1. Introduction

The technologies to solve the Y2K problem are re-engineering technologies. Many tools are provided by tool venders to support various areas of Y2K activities. These activities are categorized into the following phases: the preparation phase, the analysis and design phase, the modification phase, and the testing and verification phase. Among these, the key phase is the analysis and design phase. The key technology is often called impact analysis or ripple effect analysis, which helps customers to identify source code that needs to be modified. However, there are no automatic tools to change/fix all the source code which requires modification. Also, the required effort and times heavily depend on customers' current status of applications and their related standardization and documentation. This is why the Y2K project manager has such difficulties making their Y2K plan. We have a maximum of 2 years 5 months to complete the project.

At the turn of the century organizations in all sectors affected by the year 2000 computer problem also known as millennium bomb. The Y2K projects stimulate awareness of the need of for best practice in IT system procurement, development which can benefit the industry and other organizations in the longer term. In many respect, Y2K projects are different from ordinary maintenance tasks. They are most difficult projects faced by the IT community [14]. For such projects, it is very crucial to be completed in time. Any delay produces the considerable risks for an enterprise. Examples of Y2K projects are Y2K POLITICAL ACTION PROJECT, Y2K CULTURAL

CREATIVES OUTREACH PROJECT, Y2K BREAKTHROUGH FINANCING PROJECT, Y2K ACTIVIST OUTREACH PROJECT, Y2K-BREAKTHROUGH Q&A PROJECT [40].

Several benefits from Y2K projects can be gained if these projects are managed well [3], [23]. In this paper two groups of such possible benefits are distinguished and explained.

- Small-scale, maintenance related projects
- Large-scale, enterprise-wide benefits.

Since the early days of electronic computing, only 2 digits have been used in very many cases to represent the year in date fields (YYMMDD): therefore in many applications the Year 2000 will interpreted as the year 1900, causing failures in arithmetic computations, comparisons, sorting and access to data. This is further complicated by the frequent use of "99" and "00" as reserved values with a particular meaning (e.g. "never delete this") as well as by the cases where 2-digits representations are embedded in the program logic and, in spite of the correct four digit representation, only the last two digits are used and the first two are not initialized (thus containing meaningless data). Last but not least, many programs may not take into account that Year 2000 is a leap year1. Although the turn of the century is the date when most problems are likely to occur, many date-dependent algorithms and forward-referencing systems are already beginning to fail and several systems will not show failures until later in Year 2000. The original reason why programmers took this approach was to save on what used to be expensive computer resources (mostly magnetic storage). More recently, they kept using the same date representation partly in order to allow new applications to interoperate with older ones, and partly as a matter of habit. Although the consequences of this choice now appear in all their seriousness (see sections 3), it has to be noted that – as outlined in [2] – the cumulative savings on magnetic storage due to the 2-digit convention probably exceed the costs of the correction.

The structure of the paper is as follows: In section 2, an analysis of the history and background concerning the reasons for the occurrence of the Year 2000 problem will be explained. In section 3, the cost of reviewing and rewriting codes will be discussed. In section 4, Benefits of Y2K project from an enterprise-wide perspective. In section 5, Future work

2. An Analysis of the History and Background Concerning the Reasons for the Occurrence of the Year 2000 Problem

It is generally understood that software programmers beginning approximately thirty years ago used six digits to represent the date, month and year in business application

software in order to save processing memory, which was expensive at the time. In more recent years, six digits were used by programmers out of habit or in order to assure that new software programs would interface smoothly with older programs using six digit date To complicate the Year 2000 corrective effort, many business application software packages must also be modified to account for the fact that the year 2000 is a leap year.

An issue which the Year 2000 Commission could examine is the role that industrial standards played in accommodating the Year 2000 problem (also variously known as the "Millennium Bug", the "Century Date Change Problem" and the "Y2K Problem"). The Year 2000 Commission could also analyze what role the industry should play in the future, if any, in mandating computer standards to avoid problems in the future similar to the Year 2000 problem.

The Year 2000 Commission also might be able to discover useful information on how the industries have coped with similar computer software changes in the past

3. The Cost of Reviewing and Rewriting Codes for Software Industries Including a Legal Analysis of Responsibilities for Such Costs

Year 2000 Corrective Cost in the Billions

The Gartner Group has estimated that the cost of correcting the Year 2000 problem worldwide is approximately $300 billion to $600 billion (see the article entitled "'Year 2000 Problem' Gains National Attention", at the URL of "*http://www.gartner.com/aboutgg/pressrel/pry2000.html*"). In an *Industry Analysis*, J. P. Morgan Securities, Inc. has estimated, based on its own research, that the cost approximates $200 billion (see W. Rabin, "The Year 2000 Problem", at *http://www.jpmorgan.com/MarketDataInd/Research/Year2000/index.html*").

The $30 billion estimate for the federal government is not surprising, since (a) the federal government is reportedly the largest single purchaser of information technology, spending approximately $25 billion per year on IT services and products (see, Testimony of Stephen M. Smith, Managing Partner, Federal Government Practice of Andersen Consulting, before Congress, Federal News Service, July 17, 1996).

Essentially, the Year 2000 software corrective process is an example of "software reengineering". Following the Institute of Electrical and Electronic Engineers, Inc. (IEEE) definition of "software reengineering", a baseline inventory is the first step, followed by analysis and only then by the changing of the software code. Although federal and state agencies may be tempted to skip the inventory phase and start immediately with the corrective work, this short-cut should be discouraged. If software programs are missed, the computer system may not test out as being Year 2000 compliant after corrective work. Without any computer system inventory as a baseline, it may be difficult to pinpoint the reason for the testing failure.

Indeed, under the *Paperwork Reduction Act of 1995*, each federal agency is required to inventory its computer software assets. In many instances, the inventory can be automated by the use of scanning or parsing software, reducing the time to complete the inventory and the personnel time required. An estimate can then be made as to the total cost of making the hardware and software Year 2000 compliant. For an example of one formula for estimating this cost, see the formula at the URL of "*http://cfcse.ncr.disa.mil/jexhome/y2estm8r.html*".

A cost-benefit analysis must then be made by the agency as to whether it makes more sense to migrate to a newer computer system architecture which is already Year 2000 compliant, or to correct the older system currently being used. According to the Federal Aviation Administration (FAA) advisory document entitled "*Guidance Document For Year 2000 Date Conversion*", located at the Internet URL of "*http://www.faa.gov/ait/year2000/y2kguide.htm*", the inventory phase requires 25-40 per cent of total effort, the corrective work phase requires 10-20 per cent of the total effort and the testing phase requires 40-55 per cent of the total effort.

It is possible that agencies which have contracted for long-term maintenance or data processing outsourcing lasting past the year 2000 may be able to request the third party vendors to absorb part or all of the Year 2000 corrective work. For example, under some outsourcing agreements, the vendors agree to correct any "bugs" or "defects" in the computer system at the vendor's cost. An issue exists as to whether the Year 2000 problem qualifies as a system "bug" or "defect" which properly is the vendor's responsibility. Government counsel will need to review the relevant contracts to determine the parties' relative rights. However, if the federal agency simply proceeds to correct the Year 2000 problem without first making claim against the vendor, it likely will have waived its right to claim reimbursement from the vendor (see, e.g., Jeff Jinnett, "*Legal Issues Concerning the Year 2000 'Millennium Bug*'", located in the article archive at Peter de Jager's Year 2000 site at the URL of "*http://www.year2000.com/archive/NFlegalissues.html*").

There are in excess of two thousand software programming languages in existence, with perhaps 500 programming languages in current usage, and date fields are sometimes used for purposes other than signifying the date (such as the use of "99" to signify the end of a file). (See, e.g., the Internet URL of "*http://cuiwww.unige.ch/langlist*" for a database of existing programming languages; see also, the informative article by Capers Jones entitled "*The Global Economic Impact of the Year 2000 Software Problem*", located at the Internet URL of "*http://www.spr.com/library/y2k00.htm*").

Therefore, industries might have to locate and retain programmers skilled in many different languages to assist the agencies in correcting the non-compliant date fields. In this regard, the agencies will be competing with the private sector for an increasingly

scarce supply of programmers and the cost of programming help may increase as the year 2000 draws near, with the private sector outbidding the public sector for the better programmers. In testimony before Congress, Kevin Schick, Research Director of Gartner Group, estimated that when federal, state and local governments finally get funding at the start of fiscal 1998 for their Year 2000 corrective work, the cost of that corrective work will be over two times the cost of doing the same work in 1996 (see Testimony of Kevin Schick before the House Government Reform and Oversight Committee, Subcommittee on Government Management, Information and Technology, Federal News Service, April 16, 1996). It is quite likely that the careful investigations of all the applications yield reusable components. A Y2K project can be combined with the introduction of a reuse program [19].

4. BENEFITS FROM AN ENTERPRISE-WIDE PERSPECTIVE

From an enterprise-wide perspective, the use of information technology should yield a maximal competitive example at minimal cast. An Y2J project seems to gain competitive advantage. This goal can be achieved by the following four types of enterprise-wide benefits; a fast development should better associate the business needs (type 1), it should be possible to optimize processes based on the use of information technology (type 2), new information has to be made available (type 3) and an effective and efficient development should be enabled (type 4). A Y2K project can be seen as an enabling factor for different development activities covering exactly the above requirements.

- **Business Re-engineering (type 1):** In order to use the potential of information technology, business re-engineering efforts proposed by [12] have to take into consideration. Experiences show that the existing information technology support has to consider when developing new processes and organizational structures in order to avoid a project failure because of restricted IT-development potentials [4]. The information needed for these decisions can be adopted from the impact analysis and the repository provided by a Y2K project.
- **CASE (types 1 and 4):** A successful introduction of CASE- technology allows a major improvement of software development productivity as well as the focus on business issues. In [13] an evolving role of CASE tools for software development is identified. The obstacle to the introduction of CASE technology is the initial effort need to build the repository and the inclusion of legacy systems. In [25], reverse-engineering is proposed as a solution to bring legacy systems and CASE together. The reverse-engineering activities and the repository within a Y2K project match these requirements.
- **Workflow Management (type 2):** Within the business re-engineering field, workflow management systems play a prominent role [24]. Important elements

of a workflow management environment are process description or handbooks [10], [7], workflow engine and existing operational applications. The goal is a flexible integration of existing and newly developed functionality for an optimal support of business process. In order to implement such architecture [15], [29], a profound knowledge of the existing functionality as gathered in a Y2K project is crucial. The impact analysis shows which applications or part of applications should be integrated into a workflow environment. The repository is very helpful to build interfaces between the workflow system and the existing applications.

- **Data Warehouses (type 3):** Whereas the work-flow approach focuses on processes on the operational level, data warehouses focus on information on the strategic level. In order to extract and transform the data from diverse sources into the data warehouse, detailed knowledge about the existing system is needed.
- **Cost Estimation Database (Type 4):** Panning future developments and estimating costs requires detailed information about the existing environment. This information can be extracted from the repository created within the Y2K project.

5. FUTURE WORK

The development of balanced and sound contracts with the computer industry available for use by federal agencies, and if such outside contractual assistance is needed, to assist such agencies in contracting for and effectuating Year 2000 compliance for current computer programs and systems as well to ensure year 2000 compliance for all programs and systems in the future.

REFERENCES

[1] D. Aebi, (April-1996), Re-engineering and Migration Betrieblicher Nutzdaten. PhD Thesis, ETH Zurich.

[2] D. Aebi, (1997), A Process Model for Re-engineering, Mi-gration and Multi-use of Business Data. In Euromicro 97, Proceedings of the First International Conference on Software Maintenance and Reengineering, 106-113. IEEE Computer Society Press.

[3] D. Aebi, (April-1997), Wer Das Problem Verdr*angt, st*urzt ab. Io Management, **66**(4), 44-47.

[4] D. Biemann, (April-1997), Wenn Informatik-engp*asse Die Reor-ganisation Gef*ahrden, Neue Technologien F*ur Un-ternehmen im Wandel, Io Managment, **66**(4), 48-51.

[5] N. Bruton, (1995), E_ective User Support: How to Manage the IT-help Desk, MacGraw-Hill.

[6] E. Codd *et al*., (1993), Providing OLAP (On-line Analytical Processing) to User Analyst: An IT Mandate, *http://www.arborsoft.com/OLAP.html*

[7] A. J. Cole, *et al*., (1995), A Computer-based Process Hand-book for a Engineering Business. In Proc. of the Seventh Int. Workshop on Computer-Aided Software Engineering (CASE), 172-181. IEEE Computer Society Press.

[8] P. De Jager and R. Burgeon, (1997), Managing 00, Surviving the Year 2000 Computing Crisis, Wiley.

[9] C. D. French, (1995), 'One Size Fits All' Database Architectures do not Work for DSS. SIGMOD Record, **24**(2), 449-450.

[10] J. Galler, (1995), Metamodelle Des Workow-managements, Technical report, Verno_entlichungen Des Instituts F*ur Wirtschaftsinformatik, Universitnat Saarbrnucken.

[11] H.-D. Gro_man, (1997), Das Data Warehouse Konzept, Theorie and Praxis Der Wirtschaftsinformatik, **34**(195), 8-17.

[12] M. Hammer and J. Champy, (1993), Reengineering the Corporation: A Manifesto for Business Revolution. Harper.

[13] C. Hardy, *et al*., (1995), A Comparison of Two Surveys on Software Development and the Role of Case in the UK. In Proc. of the Seventh Int. Workshop on Computer-Aided Software Engineering (CASE), 234-238. IEEE Computer Society Press.

[14] I. Hayes and W. Ulrich, (1997), The Year 2000 Software Systems Crisis: Challenge of the Century. Prentice Hall.

[15] D. Hollingsworth, (1994), The Workow Reference Model. *http://www.aiai.ed.ac.uk/project/wfmc/DOCS/ref-* model/rmv1-16.html

[16] C. Jones, (1997), The Global Economic Impact of the Year 2000 Software Problem. *ftp://ftp.spr.com/ar-*ticles/y2k52.pdf.

[17] J. Keogh, (1997), Solving the Year 2000 Problem. AP Professional.

[18] C. McClure, (1991), The Three R's of Software Automation: Re-engineering, Repository, Reusability. Prentice Hall.

[19] C. McClure, (1997), Value Added Year 2000: Harvest Components for Reuse. *http://www.reusabili* ty.com/papers1.html

[20] H. Mucksch *et al*., (1996), Das Data Warehouse-Konzept, ein Ueberblick. Wirtschaftsinformatik, **38**(4), 421-433.

[21] T. M. Pigoski, (1997), Practical Software Maintenance. Wiley.

[22] B. Ragland, (1997), Year 2000 Problem Solver. Mc Graw Hill.

[23] L. E. M. Richardson, (July/August-1997), The Sunny Side of a Year/2000 Project. Year/2000-Journal, **1**(4), 50-52.

[24] T. Sch*al, (1996), Workow-Management Systems for Process Organisations. Springer.

[25] H. M. Sneed, (1995), Reverse Engineering as a Bridge to Case. In Proc. of the seventh Int. Workshop on Computer-Aided Software Engineering (CASE), 304-317. IEEE Computer Society Press.

[26] C. Squire, (1995), Data Extraction and Transformation for the Data Warehouse. In Proc. of the Int. Conf. on Management of Data (SIGMOD), 446-447, ACM SIGMOD.

[27] U. D. Surajit Chaudhuri, (1997), An Overview of Data Warehousing and OLAP Technology, SIGMOD Record, **26**(1), 65-74.

[28] M. Tresch and M. Rys, (1997), Data Warehousing Architecture for On-line Analytical Processing. Theorie and Praxis Der Wirtschaftsinformatik, **34**(195), 56-75.

[29] K. Wallnau, *et al*. Towards a Distributed, Mediated Architecture for Workow Management. *http://lsdis.cs.uga.edu/activities/NSF-workflow*

[30] The Year 2000 Computer Problem.

Rajesh Verma
Asstt. Professor and HOD
Deptt. of Computer Science and Application
Guru Nanak Khalsa Institute of Tech. & Mgt. Studies
Yamuna Nagar, INDIA

R. K. Chauhan
Professor, Deptt. of Computer Science & Application
Kurukshetra University Kurukshetra, INDIA