Configuration Management Extensions for Millennium Compliance: An Experience Report[1]

Alain April
Technology Plus - Batelco
P.O. Box 32594
Isa Town
State of Bahrain
+1 973 600 247
aapril@Batelco.com.bh

Alain Abran
University of Quebec in Montreal
Software Eng. Management Lab.
P.O. Box 8888, Succ. Centre Ville
Montréal, Québec, Canada
+1 514 987 3000 (8900)
abran.alain@uqam.ca

Ettore Merlo
École Polytechnique
Département de Génie Informatique
C.P. 6079, Succ. Centre Ville
Montréal, Québec, Canada
+1 514 340 4711 (5758)
merlo@rgl.polymtl.ca

Abstract

During 1997, the Information System (IS) Division of the Bahrain Telecommunications Company (Batelco) implemented a millennium compliance program. This experience report presents additional configuration management requirements implemented to manage the millennium project associated with the IBM-MVS applications. Included is a definition of compliance for Year 2000 projects, conversion approaches, additional configuration management requirements, Year 2000 Components Tracking System (Y2KCTS) process overview followed by the lessons learned. At the time of writing this paper there is 1 year 8 months and 24 days left before the new century. The year 2000 project is a significant undertaking and an absolutely no-choice project for Batelco.

Keywords

Configuration Management, Year 2000, Millennium Compliance, Software Quality.

1 Introduction

Year 2000 will necessitate changes to databases, calculation algorithms, batch programs, utilities (Sorts, Selcopy and Procs) and information systems documentation. In addition, the integrated nature of Management Information Systems (MIS) will require careful project management to properly coordinate the changes and impact to the systems interfaces to ensure interoperability. Batelco (Bahrain Telecommunications Company) operations depend on two major Cobol subsystems, namely CS and AR. These systems are considered legacy systems as they have been conceived during the 70's, with an approximate size of 50,000 function points. The size increases slightly yearly when new products/services are introduced to offer new telecommunications services or further mechanize the administration of the corporation. The major part of this software is MIS, with a majority of the systems using Cobol-CICS on an IBM-MVS platform. Beginning November 1997, IT Applications on this platform have been inventoried and assessed to address the Year 2000

[1] April, A., Abran, A., Merlo, E., *Configuration Management Extensions for Millennium Compliance: An Experience Report*, SoST '98 - Symposium on Software Technology, Buenos Aires, Argentina, 1998.

problem. Furthermore, to ensure timely conversion, a project management structure was put in place (see Figure 2) and a release approach selected (see Figure 1):

Phase 1: Inventories of applications;
Phase 2: Choice of Strategy and segmentation in releases;
Phase 3: Conversion (7 sequential releases):
Phase 4: Contingency Plan.

Figure 1: MVS-Y2K high level project life-cycle

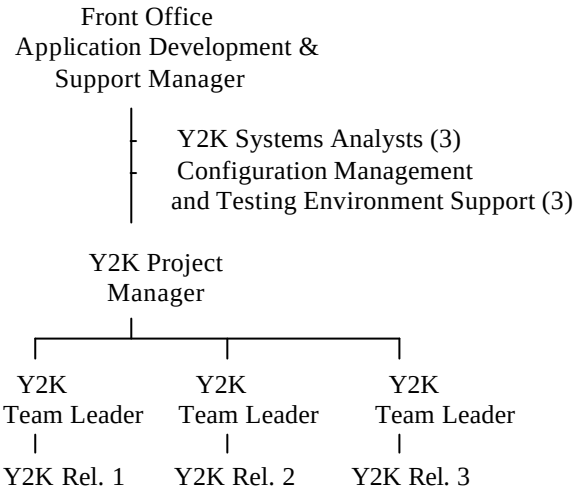

Figure 2 – Project Organization

Resulting Year 2000 management models had to be developed to support our conversion efforts and to comply to the British Standards BSI-DISC PD2000-1 "A Definition of Year 2000 Conformity Requirements" [4]. To ensure that every software component of the existing subsystems be controlled and tracked through the four-phase Year 2000 project life-cycle described in Figure 1, a software component management tracking system had to be designed and its key requirements are discussed in this paper. Section 2 presents the compliance statement and standards issues, section 3 describes the Y2K conversion strategy and the requirements for additional configuration management practices and finally section 4 describes our approach to fulfilling these requirements.

This paper concludes with a report on experience gained in the release 1 of the Year 2000 project and researching software improvement models of the SEI-CMM [8] and Trillium [1] and software quality processes described in previous work [2], the ISO 10011 [10] and IEEE 1028 [11] standards.

2 Compliance Statement

2.1 Lack of widely accepted date standards

Currently in the Software industry there is no formal usage of an accepted International Standard on date representation. Attempts have been made since the mid 1970s to set such standard, and even as recently as the late 1980's the ISO-8601 [9] standard advocated the "yyyymmdd" physical format for dates. Federal Information Processing Standards proposed recently a standard for use in interfaces between information systems [5], it has not yet gain wide market acceptance and most IT departments have never heard of them.

With no widely accepted standard deployed to the industry, management and IT staff is still left to work to whatever suit them locally. With no widely accepted single date standard over the past decades, it is no wonder that information systems will fail to handle a date for the turn of the next century.

2.2 Compliance statement and rules

To address this problem, during 1997 the key issues were identified first and this led to the formulation of a corporate compliance statement, together with the selection of local standards to address this issue.The Year 2000 compliance rules identified were derived from an analysis of the following sources: BSI-DISC PD2000-1 [4], the GTE Proposed Criteria for Century Compliance [7] and Cable and Wireless guidelines [3].

In summary, the compliance issue requires that neither performance nor functionality should be affected by dates prior to, during and after year 2000. In particular, the following five rules must be met:

Rule 1: No value for current date will cause any interruption in operation;
Rule 2: Date-based functionality must behave consistently for dates prior to, during and after year 2000;
Rule 3: In all interfaces and data storage, the century in any date must be specified either explicitly or by unambiguous algorithms or inferencing rules.
Rule 4: Year 2000 must be recognized as a leap year
Rule 5: Contingency planning sets out the target completion date of the project six months previous to Year 2000.

The compliance statement supports the project management of our project by establishing a baseline for unit testing, system testing, integration testing and validation as well as setting quality objectives. The next step consisted of choosing a conversion approach and technical solutions to the year 2000 problem that suited our environment and resources.

3 Conversion strategy and configuration problem

3.1 What conversion approach is taken by the team?

A number of approaches are available to solve the year 2000 problem:

{Re-Engineering, Data & Files date expansion, Program encapsulation, Windowing, Year compression, Date compression, Century Flagging, Date encoding, Ordinal dates and Date stamps}

All of them require the technical team of IT to modify existing data files or/and source code components. Every approach has its advantages/ disadvantages. Each strategy has a direct impact on:

{Implementation approach, Programming and testing effort and Configuration management of source components }

Since at Batelco the existing operational files contain records no older than 1973 the windowing technique was chosen by management. Readers unfamiliar with the windowing technique should refer to the glossary for a detailed explanation of this technique.

Other factors that were considered, at the time, were that the strategy was adapted for CS and AR and technical resources available. The windowing approach forces a software development life cycle using ‘releases’ that implies tighter management of the status of each source code component (also called configuration management). An evident benefit of this approach is associated with the fact that input/output screens and reports have no conversion need and only files which have a date as part of the key need to be addressed. Logic, date routines and utilities have to be investigated and converted.

3.2 Why is additional configuration management needed for Y2K windowing?

Management decided to adopt the conversion in several sequential releases (nine releases). To scope the effort associated with release 1 Business Analysts investigated the first major sub-system of CS and researched all the files being used by that sub-system. Then each file was investigated for dates as part of a key field.

This activity identified all the files that required a modification. This first study is done in a top down approach looking at the call graph of each program. The second study consists in a bottom up study that investigates all the software components that access the impacted files.

This is also done using the call graph of the programs. Once a set of software components is defined there is a specific investigation of all the date routines, logic using dates and utilities using dates. The resulting investigations are drafted into a specification document associated with the first release of the Year 2000 project.

Since no call graph technology was available locally we first investigated if the commercial configuration management tool on MVS could handle the search capabilities for both the top down and bottom up investigations. It appeared quickly that a major documentation effort would be required to establish the software component relationships into the commercial tool. Tests later revealed that the limited query ability of the tool would not be suitable for our needs.

To support the windowing release approach some software component status tracking process is required to complement the commercial configuration management tool. It became urgent to address the issue, as the release 1 team did not keep track of all the items that were identified as requiring no change.

We also discovered that software components might have to be analyzed and converted more than once during the full life cycle of the project. As an example program PES64A is impacted by year 2000 in release 1 because of file DELCO which has a date as part of the key and also because it calls a date routine PRCODAT that needed to be changed.

After the conversion effort of release 1 this software component was made Year 2000 compliant. Release 2 analysis identified PES64A into its scope also ! This time because of three files (CIRCUI, EQUIPL and EXCIR) that have dates as part of keys. Therefore all the logic and date subroutines were already fixed for this software component. This needs to be tracked especially for planning purposes. There is also a need to find out when PES64A will be Year 2000 compliant !

With 9000 software components to be converted the complexity of keeping track of the status of each software component and their dependencies became an issue for management. This difficulty led to the design and development of Y2KCTS “Year 2000 Software Components Tracking System”.

4 Y2K Components Tracking System

4.1 The Y2KCTS Process Overview

Year 2000 Components Tracking System and Processes have been designed to keep up-to-date conversion status information on each software component and their dependencies. Two Y2KTS sub-systems have been developed:

- A software component relationship sub-system for the status and relationships browsing/ updates;
- Databases compatible with Microsoft Query and ACCESS to provide the ad-hoc reporting.

The following 3 processes are associated with this tracking system. The Y2KCTS Process is composed of three main sub-processes:

Development and Support: The main objective of this sub-process is to ensure that the Y2KCTS

Databases are kept up to date during the Y2K project life cycle. A Y2KCTS MVS and a PC administrator have been appointed and their activities are described in the project plans. The development and support of the IBM-MVS mainframe library data extraction was assigned to a senior team leader in order to extract the component relationship model, see Figure 3. This function is referred to as the Y2KCTS mainframe administrator. The development and support of the PC subsystem and Y2K Status Databases has been also defined.

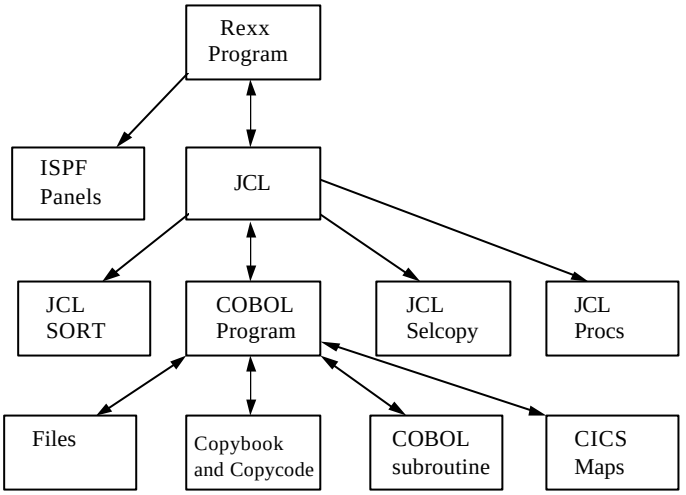

Figure 3: Software Components relationships model

Data Configuration Management: The main objective of this sub-process is to update the Y2KCTS PC Databases with changes that occur to the production environment during the Y2K project. Change is inevitable and needs to be managed during the Y2K project. This process uses inputs from the Software Configuration administrator and ensures monthly updates to the existing Y2KCTS Databases. Monthly, the Software Configuration Manager issues a list of changes to production libraries. For each of those changes the Y2KCTS IBM-MVS manager extracts the relationship files using syntax analyzers extraction programs built on MVS. The Y2KCTS PC manager updates the Y2KCTS PC Databases using the relationship files. These relational databases are more flexible for Y2K software components tracking and query purposes.

Release Component Updates & Reporting: The main objective of this sub-process is to ensure that the component compliance to Y2K is captured and issues are documented for each Y2K release. This process produces a status report for management and control of the Y2K project. Update of the Y2KCTS data is required during the initial analysis of each release. Ad-hoc reports can be produced via Microsoft SQL, Microsoft ACESS and IBM-MVS scan utility to ensure that the release list of modules and files to be tracked is accurate.

4.2 What does the Y2K Component Tracking System provides?

Each of the software components of Figure 3 have been assigned two Y2K statuses. The first is called the conversion status and the second is called the dependency status. The first Y2K status is currently under the control of the Y2K analyst. The status can be updated to any of the following selections as the project progresses:

Anal-req:	To be analyzed
Anal-und:	Analysis underway
Need-conv:	Analyzed and needs Y2K conversion
No-Impact:	Analyzed, not impacted by Y2K
Part-Conv:	Partial conversion done

Full-Conv:	Conversion done
Tested:	Tested, certified for Y2K

Figure 4: Component Status

Initially all the components have been assigned a status of 'Anal-req' which means that a Y2K impact analysis is required on each software component. Notice the conversion status in Figure 4 that are in boldface.

The Y2K analyst will ultimately change the conversion status of a software component to 'No-impact' and 'Full-conv' as the work progresses. These two specific states are used to update automatically the dependency status of each component. The following two rules must be **TRUE** in order to trigger an automatic update of the dependency status:

Rule #1: If all the dependents of a component have either no Y2K impact ('no-impact') or have been fully converted ('Full-conv') the dependency status of that component will be considered done ('Dep-done') which means dependents of this component are Y2K compliant.

Rule #2: If all the dependents of a component have been addressed they will have a dependency status of Y2K conversion done or No dependent or have been identified as having no Y2K impact.

When these two rules are TRUE the dependency status of that component is set to 'Dep-done' which means that its dependents are Y2K compliant. Initially all the components have been assigned a dependency status of 'Dep-ToDo' (see Figure 5) which means that a Y2K impact analysis is required on the dependents. Only where a component does not have any dependent we have set the dependency status to 'No-Dep' which means no dependent.

Dep-ToDo:	Dependent to be evaluated/converted;
Dep-done:	Dependent(s) have been addressed;
No-dep:	No dependent(s) exist;
No-Impact:	Dependent analyzed and not impacted by Y2K.

Figure 5: Dependency status that trigger an automatic update

5 LESSONS LEARNED FROM THE FIRST RELEASES

As this first release is now being moved to production we can report on the following lessons learned:

- Seven Hundred software components were too large a release for a 4-member team;
- Additional configuration management practices are required to track the status of each software component;
- Unit testing should be done formally.

A good size for our environment is no more than 500 software components by release this being of a good manageable size for our teams. It also became apparent that a status repository was required to keep track and control which software component had been analyzed, partially or fully converted and document the Y2K readiness based on software components/files readiness.

Finally we are doing formal unit testing and quality assurance for each software component. Release 2 assigned a dedicated test manager and we are still experiencing much rework in the areas of:

- Running the Batch runs (tests) successfully the first time;
- Aging the data to test the 1999 and 2000 boundaries is time consuming and can introduce defects;

Acknowledgments

Dr. Peterson of the University of Bahrain reviewed this paper. His contribution is gratefully acknowledged.

References

1. Alain April, Francois Coallier, Trillium: A Customer-Oriented Assessment Method for Software System Development Capability, Proceedings Quebec-German Workshop on Software Measurement, Bonn, October 1995.
2. Alain April, Alain Abran, Ettore Merlo: Process Assurance Audits: Lessons Learned, ICSE'98, Kyoto, Japan, April 1998.
3. S.Bansal, Millennium Call Center booklet, Cable and Wireless, 1998.
4. BSI DISC PD2000-1, A Definition of Year 2000 Conformity Requirements, BSI, 1998.
5. Federal Information Processing Standards (FIPS)4-1: "Representation for Calendar Date and Ordinal Date for Information Interchange," March 25, 1996.
6. Gary E. Fisher, Test Assertions For Date and Time Functions, National Institute of Standards and Technology (NIST), February 3, 1998.
7. GTE Government Systems Corporation, Proposed Criteria for Century Compliance, PA96014, Waltham, MA, 1996, March 25, 1997.
8. W.S.Humphrey, Managing the Software Process, SEI series in Software Engineering, Addison Wesley, 1989.
9. ISO 8601:1988, Date/Time Representation, International Organization for Standardization, Case postale 561, rue de Varembé, CH-1211 Genève 20 Switzerland.
10. ISO/IEC JTC1/SC7, 10011-1 and 2:1991, Guidelines for auditing quality systems, International Organization for Standardization, Case postale 561, rue de Varembé, CH-1211 Genève 20 Switzerland.
11. IEEE Computer Society, IEEE Standard for Software Reviews and Audits, IEEE Std 1028-1993, IEEE, New York, New York.

GLOSSARY

compliant: "compliant" system's dates are stored, manipulated (including, but not limited to calculating, comparing, and sequencing), exchanged, and displayed in a way that cannot be misinterpreted and are not ambiguous. "Compliant" system's hardware and software products' correctly process date and date related data individually and in combination in both the 20th and 21st Centuries. Finally, "compliant" systems have no extended semantics, calendar errors, date overflow, and inconsistent semantics.

contingency plan: a plan for responding to the loss of system use due to a disaster such as a flood, fire, computer virus, or major software failure. The plan contains procedures for emergency response, backup, and post-disaster recovery.

conversion: the process of making changes to databases or source code.

database: an aggregation of data; a file consisting of a number of records or tables, each of which is constructed of files or a particular type, together with a collection of operations that facilitate searching, sorting, recombination, and similar operations.

enhancement: new functions added to an existing system to meet new or revised requirements

function point: Functional size of software as prescribed by the International Function Point User Group (IFPUG)

interface: a boundary across which two systems communicate. An interface might be a hardware connector used to link to other devices, or it might be a convention used to allow communication between two software systems. This is to include interfaces internal to the system, its applications and programs, to other internal or external systems.

integration: two or more software applications that must run on the same physical processor(s) and under the same operating system.

integration testing: testing to determine that the related information system components perform to specification.

interoperability: (1) The ability of two or more systems or components to exchange data and use information (IEEE STD 610.12) (2). The ability of two or more systems to exchange information and to mutually use the information that has been exchanged.

legacy system: an existing automated information system (AIS) or application which will be replaced in whole or part in the future. Those systems are typically contained to limited enhancements and investments.

software component: a sub-element of a software program i.e.: a program, a subroutine, a file, a JCL SORT, a JCL SELCOPY, a CICS MAP, a copybook, etc.

system testing: testing to determine that the results generated by the enterprise's information systems and their components are accurate and the systems perform to specification.

unit testing: testing to determine that individual program modules perform to specification.

validation: the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

windowing: This approach to solving the year 2000 problem works in the following manner: When year 2000 begins, computers will interpret '00' in their two-digit dates format and current logic as 1900 instead of 2000. This is mainly because the century is not part of the logic and date format. Since there is no date preceding 1973 in the current files, it can be assumed implicitly that the range from '00 to 72' can be used in the future as 2000 to 2072. The remaining range 73 to 99 is implicitly 1973 to 1999. So with the windowing approach, at some point in time from 00 that represents implicitly 2000 there must be an assumption that the century goes back to 19 instead of 20. This year is called the window pivot year. We have chosen 1973 as this window pivot year. Taking this pivot window pivot year any year from 00 to 26 will be 2000 to 2026, but any year greater than 73 as an example 81 will be considered 1981. This approach only works for business applications whose dates are within a 100-year range.

Year 2000 compliant: information systems able to accurately process date data--including, but not limited to, calculating, comparing, and sequencing--from, into, and between the twentieth and twenty-first centuries, including leap year calculations.

Year 2000 problem: the potential problems and their variations that might be encountered in any level of computer hardware and software from microcode to application programs, files, and databases that need to correctly interpret year-date data represented in 2-digit-year format.