Configuration Management – an automated solution

R.A. Green

Colton Danste Ltd, Princes House, Princes Street, Swindon, Wiltshire, SN1 2SF, UK

Abstract

Historically, the performance of Configuration Management tools has not been particularly impressive. Their roles have usually been restricted to various forms of version control, coupled with some control mechanisms for managing releases and controlling the application of changes, but few have incorporated more elaborate configuration control functions, such as partitioning, parallel development and document cross-referencing.

Further, in common with their Project Management and CASE tool counterparts, Configuration Management tools have been singularly inadequate in addressing the two essential attributes of functionality and usability, which are of key importance in any computer-based development support product. This is particularly true in the area of build management, where parallel development environments, problem tracking and correction and reversionary build functions may need to be controlled concurrently.

These tools have not properly addressed the definition and management of process in the context of IT development activities. Traditionally, the tool has assumed that all relevant processes were either fully defined, or that they had no significance for their use.

However, over recent years, the development of C.M. tools has resulted in a much more complete coverage of the required scope in supporting project environments, including distributed computing architectures such as Client Server systems. Also, tools are now available which are based on a process-driven approach, but which do not restrict the type of process model adopted.

This paper will address the required characteristics of a Configuration Management tool, particularly in distributed computing environments, including an outline of the approach to the implementation of a CM product, discussing the issues associated with the importance of managing the migration from the current environment to one controlled by the tool.
1 Introduction

Although Configuration Management has been around for many years, and its essential elements broadly understood, the true significance of the impact of C.M. on the I.T. project execution environment has only been recognised relatively recently.

Traditionally, the implementation of C.M. has been largely ‘people and paper’ based, involving a considerable level of bureaucracy where complex projects were being undertaken, and with many opportunities for disaster due to failure of the design or operation of the C.M. procedures. Thus in many cases the root cause of many failures could be traced to C.M. activities, whether it be due to their absence, ineffectiveness, or lack of application.

Another parameter central to the issue of effective C.M. is the definition of processes, which tools must address to ensure coverage of the required functions in supporting project environments, including distributed computing architectures such as Client Server systems. Before discussing the specific requirements for a C.M. tool, the basic essential elements of such a tool will be reviewed.

2 The Elements of Configuration Management

Configuration Management is a discipline with an inherently hierarchical nature, since the objects being controlled are part of an organisational hierarchy. In this context, it can be considered to consist of the following five elements, namely:

Version and Variant Control
Configuration Control
Build Management
Change Control
Problem Tracking and Task Management

The following definitions are provided as a guide to understanding the operation of a C.M. tool in providing the business and project management benefits, rather than a definitive discussion of their nature and application.

Version and Variant Control

This function is concerned with the control of distinctive instances of Configuration Items or Objects, which represent the lowest level of hierarchy in C.M. activities. Objects at this level may typically be software files, (source, object, library, etc.) or individual documents, or any other type of defined Configuration Object.

The control of variants of Configuration Objects is treated in the same way as for versions, but has greater implications for parallel developments, since variants are usually ‘parallel’ instances of a Configuration Object at a particular version status.
**Configuration Control**
This function is concerned with the control of combinations of Configuration Objects, which are defined as being part of a group for a specific purpose. Configurations form a significant proportion of a C.M. database, since they may include Configuration Objects of any defined type, including hierarchical types such as configurations of configurations.

**Build Management**
The management of builds are distinguished from Configuration Control; they are not usually hierarchical, they form a unitary level in the C.M. architecture, and have a special importance in the life cycle of many I.T. projects. Builds are usually undertaken at key points in the project, either to establish a new baseline, or to undertake validation or acceptance testing, or possibly to produce an instance of the system with specific functions or characteristics. This approach may often be used in incremental or prototyping project life cycle models.

**Change Control**
This function covers the control of changes to any defined Configuration Object at any level in the C.M. system. Proper Change Control operation should include procedures for the assessment, approval and authorisation of all changes, any specific testing requirements, and should also include dependency / impact analysis of the effects of making the change(s) requested.

**Problem Tracking and Task Management**
This is distinguished from Change Control by the circumstances in which it occurs, rather than by any differences in operation, which are similar to those for Change Control. Problems by definition mean that a defect or error has been detected, whether by a member of the project team, or by a user or customer, perhaps via a Help Desk report, rather than in response to changes in specifications or design. Dependency or impact analysis are often implemented as part of escalation procedures.

Task Management relates to the control of the (often multiple) activities undertaken to implement the necessary changes to fix the problem. In large projects or systems, and where distributed development environments exist, the identification and control of several changes applied by different people at different times is essential in managing the risks which may arise.

The above definitions will be assumed in the following sections when discussing the role and performance of C.M. tools in the I.T. project environment.

### 3 The Requirements for a C.M. Tool

**General**
The impact of C.M. applies throughout the project life cycle, regardless of which model is used, and consequently the implementation of an automated C.M. system is equally profound in its effects, whether good or bad.

This must be appreciated when contemplating the procurement and installation of a C.M. tool, since it will be reflected not only in the expectations of the tool’s performance, but also in the required commitment of resources in
identifying its scope of application within an organisation, the analysis of C.M. processes and in planning and executing the migration to its use.

Failure to analyse and plan in this way has probably been responsible for many implementation failures, in common with other automated techniques such as CASE tools.

The Role Of Process

As discussed in detail by Cagan[1], the concept of process is crucial to the understanding of the requirements for an effective configuration management system. This is particularly important when considering the distributed development environments which are becoming increasingly common in IT organisations. Processes may be distributed over a single machine or over a network between several developers, and may involve several procedures and activities.

In general, IT organisations will have some awareness of their C.M. processes, and will have documented current ‘local practice’ as deemed appropriate. However, many are experiencing significant pain from their current C.M. system, either through inadequacy or absence, and this is hindering them from improving product quality, and hence their business performance.

A comprehensive definition of the C.M. process model for a project or organisation is an essential pre-requisite to defining the required features and facilities of the desired C.M. tool. This is because the tool should support and enforce all of the processes defined in the model. It is also necessary to enable an implementation plan to be developed which includes the key activities of management and resourcing.

It should also be noted that C.M. processes will map across many types of Configuration Object, and vice versa. Thus a given process might involve source files, library files and associated documents, for example. Therefore the tool should be capable of managing many different types of Configuration Object, and ideally should be able to handle abstract types such as meta-data (methods and techniques) forming part of the development environment.

The Features and Facilities of a C.M. Tool

Before discussing the approach to planning the adoption to a C.M. tool, it would be useful to identify the necessary features when evaluating the products available in the context of the environment in which the selected tool will have to operate.

The following list of features may be used to form the basis of a ‘checklist’ of required or desired attributes of a tool to meet the requirements for C.M. in a particular organisation. They focus on the importance of Functionality and Usability as two key reference attributes, since these are the attributes which are most prominent in adopting and utilising the tool, and will most strongly influence its successful implementation. However, the segregation of the features listed below between these two attributes is not intended to be rigorous.
Functionality  The tool should offer the following facilities

a. support and implement all of the five C.M. elements described in Section 2.

b. provide a wide range of Configuration Object type definitions

c. provide dependency/impact analysis information when processing change requests

d. support the promotion or ‘journey’ of Configuration Objects through their respective development life cycles, with appropriate controls and security

e. be capable of supporting multiple C.M. process models concurrently, whilst enforcing a single process model for a given Configuration Object type

f. provide a customisable Report Generator facility for activities and processes

g. the file structure presented to the user must be configurable by the organisation, e.g. on a project basis, and these structures should be of any type and of unlimited size

h. incorporate adequate security and access privilege measures to enforce the segregation of user roles in the C.M. system, e.g. as developer, tester or administrator

i. support distributed code management, for those organisations using this environment, such as Client Server systems

j. be able to support parallel, remote or distributed builds, with automated build dependency analysis, for distributed development situations

Usability  The tool should offer the following facilities:

a. have a fully customisable user interface which provides the maximum information and flexibility, such as an industry standard GUI, to meet the particular needs of an organisation

b. possess a transparent file system, rather than require all Configuration Objects to be loaded into a product-specific filestore, to simplify implementation and allow file operations without involving the tool

c. have a comprehensive Configuration Object type definition facility, in order to create and customise object types interactively

d. be able to reproduce builds and to identify all configuration objects and their dependencies, for regression testing or legacy system management purposes

e. where appropriate, support command line operations, if this is desired for executing certain operations, without compromising the GUI interface

f. support integration with other products, to provide the user with a fast and efficient method of linking C.M. tasks with other activities
Software Quality Management

106

g. be able to collate and present metrics data for report purposes, and to provide a comprehensive querying facility

h. provide a fully integrated problem tracking capability which provides the links to associated C.M. activities

i. be able to communicate with database systems other than its own, and based on other operating systems, to reduce the problems of transfer and control

j. provide a comprehensive tutorial capability to assist users with problems and provide a degree of on-line training

The above list is not intended to be exhaustive, but is an indication of the range and nature of features, and the corresponding C.M. issues, which would typically need to be considered when reviewing C.M. tool capabilities.

It is suggested that useful independent evaluation information may be obtained from reports produced by organisations such as OVUM[2], who specialise in carrying out comparative assessments of products within a particular sector. Their reports are comprehensive, and provide a pointer to currently marketed products and future developments in C.M. tools.

It is important to stress at this point the strong need to allocate adequate resources to the planning and analysis activities associated with C.M. system requirements and tool evaluation and selection, since the implementation of an automated C.M. solution represents a strategic I.T. commitment for an organisation.

4 Managing the Migration to Automated C.M.

Introduction

From the foregoing discussion, it will be seen that the issue of adopting the tool, or Migration Management, is going to be a key factor if a successful implementation, leading to the expected benefits to the organisation, is to be achieved. In general, the requirements placed by the organisation for such a tool will be substantial, and this is reflected in the list of expected features given above.

However, it should also be regarded as a unique opportunity for an organisation to address its C.M. problems, and to enable process improvements to be achieved, leading to better management of the development environment and project data, with consequential benefits to both deliverables and to the business. These quality management benefits should not be overlooked.

As with a quality management system, implementing an effective automated C.M. solution will inevitably require changes within the organisation, because of the global nature of C.M. on development activities. An extended discussion of the issues to be addressed in contemplating such changes is given by Dart[3], in which the critical subject of process is emphasised.

Thus it is crucial to a successful implementation to understand the importance of the role played by the supplier in supporting migration to the C.M. tool, in addition to supporting the product itself. Suppliers which offer a
complete migration support service and promote a ‘partnership’ ethic are likely to contribute most to a successful adoption of the tool.

**Management and Planning Issues**

Because of the strategic impact of C.M., it is vital that those management issues which need to be addressed at an early stage are given proper attention. Within the scope of this paper, I shall outline a **Migration Model** which could be used as the basis for planning the adoption of a C.M. tool.

In managing and planning the adoption of the tool, it is necessary to understand the scope of issues which have to be addressed. Although not all organisations will face all of these, they do represent a fairly complete list of the most common problems and difficulties to be solved.

These issues can be grouped into distinct categories, as described below.

a. Technical - how the tool works, its hardware platform, operating system requirements, network requirements, and the level of customisation

b. Managerial - the level of planning, monitoring, resource and task management, and the impact on project schedules

c. Organisational - business process issues, company infrastructure, roles and responsibilities and communications

d. Cultural and political - how the company operates and motivates, how will necessary changes affect those involved, how will resistance be managed and career expectations resolved

e. Risks - what are the potential unknown factors arising from decisions already made, and problems associated with legacy systems, etc.

A diagrammatic representation of the categories of activity and their relationship to the migration management organisation is given in Figure 1, with particular emphasis on the interactive nature of the Migration Model.

In addition, there are certain complexity factors which will need to be included in the agenda of issues to be resolved during the planning activities. These factors are summarised in the list below:

a. Scale of I.T. applications to be managed

b. Distributed development and databases

c. Legacy systems with long lives

d. Heterogeneous hardware and operating systems

e. Network architecture problems

f. Distributed build activities

 g. Parallel developments and change activities
h. Unrealistic project schedules

i. Unknown or inadequate processes

**The Project Approach**

In order to be able to manage all of the issues and activities described above in terms of goal setting and resource management, the logical approach to adoption/migration management is to plan it as a project.

All of the established elements of project management, including planning aids such as network and activity charts and monitoring and reporting mechanisms, should be employed.

The project phases which comprise the Migration Model are identified as follows:

Phase 1 Preparation and Planning
Phase 2 Process Definition
Phase 3 Pilot project implementation
Phase 4 Roll-out across the organisation
Phase 5 Process Improvement

It should be noted that phase 5 is not strictly part of the adoption project itself, but rather an on-going refinement of the C.M. framework, using the performance metrics and management data as input to the review activities.

One of the most important additional issues, which is frequently underestimated or overlooked, is education, both of staff and management, and the timing of training activities must be planned carefully to ensure that those involved in the adoption project have the necessary skills when they are needed.

**5 Summary**

Configuration Management is a large and important subject, and the aim of this paper has been to provide an outline of the issues and problems associated with the task of selecting and implementing an automated solution to Configuration Management for an I.T. organisation. Product-specific information can be obtained from appropriate CM tool vendors, and from independent evaluation sources such as the OVUM[2] reports.

The indexed lists provide condensed information with the aim of enabling those charged with the responsibility of implementing a CM tool to develop checklists for planning and reference purposes.

As an ‘aide memoir’ for those involved in the selection and implementation of a CM tool, there follows an eight point summary checklist of management issues, reflecting and summarising the discussions in previous sections. These are as follows:
1. Manage it as a project
2. Allow sufficient time for migration/adoption of the tool
3. Define roles and responsibilities - build consensus in the organisation
4. Define C.M. processes fully - develop a process model
5. Remember education and training, this can be time-consuming
6. Select the migration path (pilots, etc.) carefully
7. The highest risks are associated with organisational and cultural issues
8. The complexity factors - make sure they are catered for in the plan

Conclusion
With the rapid growth of distributed I.T. environments, driven by increasingly stringent requirements for maintaining effective control of development activities, the need for a tool to provide an effective automated C.M. solution which possesses the capabilities described in this paper is more urgent than ever.

It is the author’s belief that the latest generation of C.M. tools, as represented by the currently available leading products, can go some way to meet these requirements, and I.T. organisations of all sizes now have the opportunity to bring this critical Configuration Management function under control, perhaps for the first time.
Figure 1 The Elements of Migration Management

Note.

The significance of two-way arrows lies in the interactive nature of the adoption project, and the importance of monitoring and review procedures.

6 References

