

WHITE PAPER

Implementing CM Everywhere, the practicalities

Author: Tom Furukawa, Director, Technical Marketing
Telelogic Americas & Asia/Pacific

Renate Stücka, Director of Marketing
Telelogic Deutschland GmbH

Version: 1.0

Date: December 3, 2002

World Headquarters: Telelogic AB (publ.), PO Box 4128, SE-203 12 Malmö, Sweden, Phone: +46 40 650 00 00, Fax: +46 40 650 65 55
American Headquarters: Telelogic North America Inc., 9401 Jeronimo Road, Irvine, CA 92618, USA, Phone: +1 949 830 8022, Fax: +1 949 830 8023
Web: www.telelogic.com, E-mail: info@telelogic.com

Introduction

"80% of all software projects have exceeded time schedule, budget or have been even cancelled due to insufficient software configuration management". This is the conclusion arrived at by the market research company Ovum in the report entitled "Ovum Evaluates: Configuration Management" dated 2001. Although this study is based on a cross-section of software development projects throughout all application areas – there is no reason to assume that any restriction to projects within the automotive, military/aerospace or communications industry would bring about distinctly better results. Electronics in state-of-the-art systems such as automobiles, planes, missiles, and mobile phones fulfill tasks of rapidly increasing complexity, and indeed software complexity is increasing at the same rate, if not more. Intricate hierarchical or networked system architectures are created and have to be mastered.

Problem

Stringently administering and systematically maintaining these complex systems across several generations and variants is a task that is becoming increasingly more complicated. Conventional methods using simple version control software are no longer suitable to fulfill this task. Each new version and every single variant represents an additional factor to be considered; each functional unit in the company, from which change requests can be issued adds yet another dimension, and the same applies to each further development site in a distributed project. Thus, a multi-dimensional complex entity is created, which can hardly be controlled using heuristic methods.

The "Usability" Paradox

Configuration management has been developed during the course of the past 40 years, primarily as a method for the administration and integration of software modules of any granularity with the aim of being able to use and develop it further. Hand in hand with the increasing complexity of the requirements the user expects from the supporting tool, however, at the same time, the methodology in which this functionality is embedded is rejected by many users.

Configuration management cannot be viewed as an isolated "value in its own right": it encompasses core elements of the business process such as:

- organizational elements – administration of professionally and geographically distributed project teams

- quality elements – integrated process support and change management
- release management – support for concurrent development activities
- project organization – support for component-based development
- financial aspects – shorter time-to-market and lower operative costs

A high-performance configuration management system must address all these aspects. At the same time, this implies the existence of various "default" procedures for the correct process within the project: the tool must embed individual functions into processes that have been frequently tested, in order to guarantee a consistent result, e.g. in build management.

The "Usability" paradox is not an easy one to solve. On the one hand, there are the justified requirements posed by the user calling for a functionality, which matches the complexity of the task to be solved; on the other hand, generally a tool such as this can no longer be easily deployed: it becomes cumbersome, and the attendant procedures are then regarded as superfluous or as an inconvenient disciplining of one's own creativity.

There is a third dimension to the problem: the decision to use a high-performance configuration management system is an investment, which naturally should amortize – within an appropriate period – and thus contribute to a company's success. This is only achieved if:

- the functionality of the tool can be seamlessly woven into all processes, which come into contact with the software development projects. This also applies to more borderline activities such as the compilation of a manual, design of a packaging or planning of production, if it is a product combining software and hardware.
- all project members accept the tool, in order to be able to use it efficiently in their areas of responsibility.

One possible solution for the "Usability" paradox is based on the concept of rendering configuration management as transparent as possible – in the ideal instance the user should not even notice that a configuration management tool is being used in a project. In order to facilitate an easier comprehension of this approach, several basic configuration management concepts will be presented below, before we return to the "Usability" paradox.

What does "task based" actually mean?

State-of-the-art change & configuration management (CCM) systems provide a project environment, which structures this complicated interrelationship into measurable perspectives and simultaneously preserves the diverse cross relationships and correlations. That sounds good, but how is it supposed to work? An essential pre-condition for an efficient and effective functioning CCM System is given by the concept of a task-based approach. A change request (Change Request or CR) is viewed to be a summary of tasks, which have to be fulfilled for this request. It is irrelevant whether one or several tasks come into question – the change request is not judged to have been fulfilled until all attendant tasks have also been fulfilled (see Figure 1).

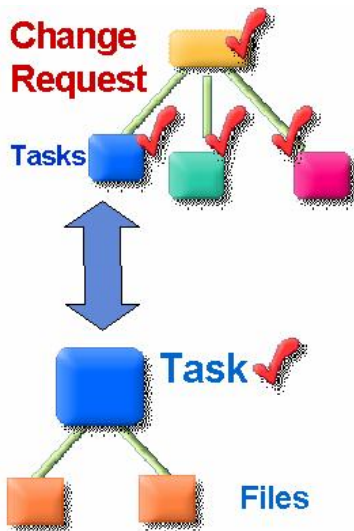


Figure 1 - Context of change requests, tasks, and files

One example: the change request asks for the introduction of a new feature into an application's web surface. Tasks, which can be assigned to this CR, are implementations of the feature, extension of the web surface, augmentation of the user manual and quality assurance measures for the new feature. Each task is assigned to a project member for execution: the software developer looks after the implementation, the web designer adds to the web surface, the technical author incorporates the new features into the user manual, and the quality engineer is responsible for inspecting the results of the tasks. The change request, the tasks and the project members responsible for them are documented in the system, to this end a database is generally used. Each employee then turns to his/her individual task.

Files, which have to be changed as part of fulfilling a task, are marked accordingly and automatically versioned. This is where the actual domain of

configuration management starts: it has to ensure that any changes made to any project artifact, whether it is source code, images, documentation, etc., are saved to the database as a new version and associated with the assigned task to which the change pertains to. It also needs to reflect whether it is currently being worked on or completed and ready to be shared with other team members. Otherwise the system under construction will in all likelihood be incapable of running. This differentiation between "finished" modules and ones, which are "work in progress", is known by all conventional version management systems. Frequently however, the project members regard the correct use of such systems to be cumbersome – the consequence being that the necessary discipline is lacking in system usage, and thus the version management rendered useless

Additional essential modules in configuration management are shown in Figure 2. Decisive here is that change management is understood to be a fully integrated element of configuration management. The automatic linking of the task and files in question ensures that the status and implication of a change request is administered by the system and rendered transparent at any given time, without requiring any manual action for safeguarding and updating this information. A special challenge is presented when various tasks act on the same file, something that can occur frequently even in projects of medium size: hardly any company can afford to dispense with the need for parallel development at various sites. Very often conventional version management systems fail in such instances. A significantly advanced configuration management tool however offers comprehensive support for the necessary merge processes, which run partly automatic and partly interactive. This is how merge processes are accelerated, and code duplications are not required.

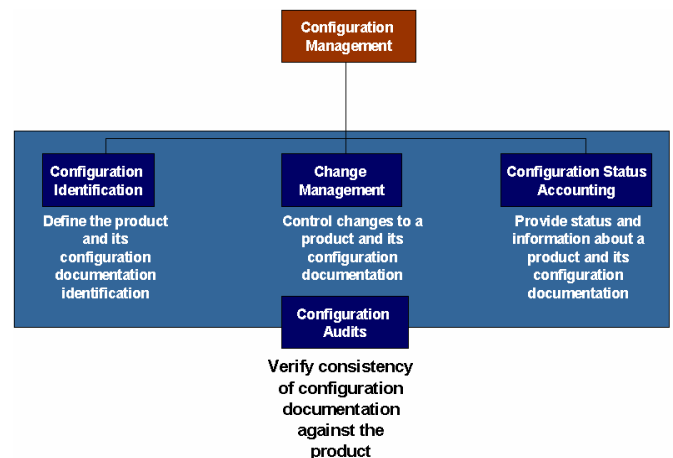


Figure 2 - Configuration management tasks

What do change and configuration management systems have to render?

The example has served to clearly highlight several basic requirements. The task-based approach conceals advantages both for “productive” project members as well as for management and administration. The project members can retrieve their intuitive approach model of requirements and resulting tasks, and the current status is fully transparent for project management at any given time. A further advantage is the embedding of change management into the overall view of the development process: from the start to the end of a product’s life cycle the project runs through several, in part interlocked iterations (see Figure 3).

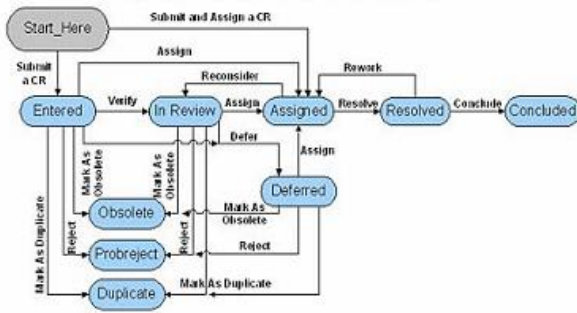


Figure 3 - Change management in the development process

In this complex scenario, keeping track of operational versions and variants is only possible with the aid of a tool, which reflects and supports process-inherent structures and procedures. Ideally, the tool should be web based so that all groups within the company who are involved in the project can benefit from it – regardless of where they are geographically located and the tasks that they are responsible for. All information, which is required for monitoring change requests and all reports, can be accessed via the web interface and can be matched to requirements for each specific project. Accurate online aids and a selection of pre-defined templates render the intuitive use very simple: without any protracted training, users are in a position to initiate change requests, assign tasks, and generate individual reports. The need for extensive administration of dedicated clients for change management can be dispensed with.

The monitoring of change requests and generation of representative reports are also part of the pre-conditions for practical deployment of a change management tool. To this end various points of view should be available as well as an opportunity to compile individually arranged reports in varying degrees of itemization. Different forms of

representation for the best possible visualization are also indispensable.

The execution of changes in a project always leads to the creation of a new variant, which exists parallel to an existing one or a new version, which supersedes an existing one. For this reason a professional change management without the seamless integration with configuration management is unthinkable. For a configuration management tool particular attention should be paid to providing a capability to support large projects. Initially, this would appear to be a matter of course, however this includes various properties that are not provided by simple tools. Thus the underlying database should in any case be capable of scaling, and the use of several database servers should not lead to inconsistent conditions. The same applies to offline work: project members or managers, who are not always online, must also be capable of working with the tool. This is the only way of reliably avoiding the classic cases of double or fragmentary development and interface conflicts. Although it is critical for the tool architecture to be able to support large amounts of data, the other less obvious capability is its interface and features. With large projects come large amounts of information, all of which need to be accessed and understood quickly and easily. Therefore, it is also a critical requirement for the change management tool to support interfaces and features that are designed to support large amounts of information.

The principle of a task-based system enables an approach which is easy to comprehend: a new configuration is generated from a base version plus a defined quantity of changes (as a consequence of executed change requests), that are linked with a list of tasks, which for their part are linked to a number of objects (files), which can also be referred as object versions. The path back to a known basic version can be taken at any given time – risky changes shortly before a schedule release can be reversed without any side effects and scheduled for a subsequent version. Particularly in the case of build management’s support for the building of a running system, there are distinct differences between the tools. Weak or lack of build support can lead to incalculable risks. One minor piece of negligence, which can easily occur from a manual approach, can lead to dramatic consequences. It may be that a variant from a fundamental product release that is significant for business success can no longer be reliably reproduced because the components are not documented in a non-ambiguous way.

Practical support for build management starts a long time before the build process. Through task-based check-in of “finished” modules, any risk of missing or incorrect modules being “checked in” to the database

are minimized. The intelligent localization of possible conflicts due to conflicting or incomplete changes is also included here. By using the attribute of a module or object the build manager can determine at any time, whether this module has been modified for a specific change and thus manage the build process accordingly.

2nd generation configuration management

Despite all the inherent advantages, the use of such a tool always requires a certain amount of self discipline on the part of the project member: change and configuration management cannot function without complete relinquishment of manual check-in and check-out as well as other administrative tasks which have to be performed manually. This is the "Usability" paradox: the user, who requires the utmost functionality from his/her tool, also expects at the same time that use of this tool is as simple as possible and that the actual daily work is interrupted as little as possible.

How can this paradox be solved? The challenge is to find ways of implementing all the complicated and timesaving functionality into the tools while simultaneously, drastically simplifying their handling. There are three basic strategies to help achieve this:

1. Visualization

The visualization of data is one of these strategies. Tools must be capable of clearly differentiating between relevant data and distracting data. For everyone in the company who is specifically involved in the project, the important data must be visible, at the appropriate time, with the appropriate detail and context. This applies to all participating hierarchical levels and specialist groups. This is achieved by structuring the information on various levels, so that each user can access the data that he/she requires very quickly.

Each practical visualization concept must also consider an additional aspect, one which is frequently ignored: the concealment of data. Everything which is irrelevant at a specific time, should be hidden. By hiding unimportant data, the user is screened from "ground noise" and can thus absorb the decisive data much quicker.

2. Role-based user surfaces

When designing the user interface, if various optional roles for project members are also taken into consideration, the role-based visualization can be implemented in a much more thorough and consistent manner. This does not represent a new approach if one views individual tools. What has to be done is to push forward this approach, in order to enable role-based user interfaces to also be used in

a typical project environment, in which several different tools can be used.

One example: a test engineer usually uses a test management tool as a primary working environment. However, in order to be capable of productively and efficiently contributing to a specific project, and to fulfill his/her tasks completely, the test engineer also requires access to a requirement management tool and the system, which monitors and administers the change requests. The most elegant way is to provide the test engineer with this functionality integrated seamlessly as part of his/her primary working environment – the test management tool. Ideally the role-based user surface should go even further; it should not only provide, what the user has to do, but rather take into consideration exactly what the engineer hopes to achieve by doing so. The main task of the user is never to use a tool correctly: the user's task is to perform a job in the project.

3. Automation

This third basic strategy is also the key for successful implementation of the first two strategies. The tools must automate as many steps and processes as possible. One of the reasons why a user frequently regards a tool to be extremely cumbersome impedence to their usual course of action lies in the fact that the majority of tools are exactly that! Time and time again they interrupt the user with fundamental interactions thus breaking up his/her flow of work and thus concentration (see Figure 4).



Figure 4 - Traditional "Passive" CM System



Figure 5 – Transparent “Active” CM System

A useful tool should automate everything which does not genuinely require any intellectual effort. In the majority of cases this can and must be context and time sensitive (see Figure 5). Depending on a specific process and the point of time, the automated operations are generally different.

This means that the only steps that remain are those for manual intervention, ones which cannot be decided automatically – thus from the point of view of the user a reasonable interaction with the tool is given.

The main purpose of automation however should not be to solely automate certain operations in a tool application. A quantum leap is achieved, if the entire process is automated. The entire current of company activities, including the software development process is automated to as great an extent as possible. Today in many companies, the same phenomenon as mentioned above in relation to tools is given with regard to processes: the "Usability" paradox. There exists an underlying understanding that processes are necessary and practical, although their introduction and implementation is difficult. Exhaustive training is required; in spite of this, however, there will be wrong deployment patterns. This will lead to an impression whereby the process represents little more than a serious impediment to the individual performance capability. And even for the stakeholders who might champion processes, without automation, the process visibility is rarely accurate and certainly not up-to-date, as process events are not tracked consistently. Additionally, accurate measurement of the process is close to impossible; therefore the organization cannot identify best practices and best technologies.

Summary

As change and configuration management tools matured over the years, it is now becoming critical to

business success and profitability. Companies have realized that by making change and configuration management an integral part of application development, it gives them competitive advantage by allowing them to increase productivity and quality, while reducing time to market.

Although this trend is very positive, change and configuration management tools are susceptible to the "Usability" paradox. Tool vendors need to redirect their research and development efforts. Instead of just adding more and more functionality to tools, thought must go into how this affects the end user. Decisive here is the fact that the tool "understands" which goal the user is aiming for in his/her specific role at any given time. To this end decision-relevant data must be made available, whereas routine tasks can be automatically processed without any manual intervention.

The key here is to automate the mundane and allow people to focus on their work. Change management is about managing the workflow of how changes get captured, approved, assigned, and completed. During this process all relevant information must be captured and properly linked together. It is not enough to just document the history of change, it also needs to be related to other artifacts such as work orders, assigned tasks, and software changes. Advanced change management systems must automate as much as possible, the information capturing and linking in order for the users to not get distracted from their work.

The next generation of software configuration management systems (SCM) must be as transparent to the user as possible. Its purpose is to accelerate the development process by enabling a smooth development process. However, as a software developer, the last thing you want to have to worry about is breaking the build because you forgot to check in a file. The ideal SCM interface is a transparent one that does not require mass amounts of training, prevent software configuration mistakes and will not disrupt your software development activities.

With an automated change and configuration management system, the "Usability" paradox is thus solved. User acceptance increases tremendously and the daily use is automated so that all productivity advantages for such a solution can be fully utilized. An automated solution simplifies the decision for supporting migration to an enterprise change and configuration management system, especially for companies who are "growing out" of their existing solutions.

1st BCS CMSG Conference 2003
Implementing CM Everywhere, Change, Configuration & Content Management

References

- (1) Ovum, Ovum Evaluates: Configuration Management, April 2001

About the authors



Tom Furukawa is the Director of Technical Marketing for Telelogic Americas and Asia/Pacific. With more than ten years of experience in the development and management

of advanced software products and solutions for the commercial, financial, and automotive industries, Mr. Furukawa is responsible for product evangelism for Telelogic's change and configuration management product line and integrated product solutions. This includes market, quantitative and competitive analysis; developing content and messaging for promotional communications and public relations activities. In this capacity, he also drives product innovation, provides technical field sales assistance, creates sales tools, training and channel support. He has been instrumental as a thought leader in driving the development, packaging and marketing launch for breakthrough technologies such as Telelogic CM Synergy™ with ActiveCM™.

Previously, Furukawa served as product manager, development manager, and senior software developer for the Telelogic Synergy product line and application lifecycle management tools. Prior to joining Telelogic in 1993, he has held several positions in the software development and marketing teams for a document imaging/workflow company, major defense contractor, and a leading systems integrator. He has a bachelor's degree in Information and Computer Science from the University of California, Irvine.



Renate Stuecka is the Director of Marketing for the area of automotive and embedded systems at Telelogic. She is responsible for increasing awareness and improving perception of Telelogic as a strong partner of the automotive industry. Mrs.

Stuecka's tasks include definition and execution of a segment focused market communication strategy to address the automotive industry, management of strategic partners and alliances for this market, as well as analyzing market feedback and research data to ensure compliance of Telelogic's product strategy with the automotive market needs.

Mrs. Stuecka joined Telelogic in November 2000, bringing to the company more than 15 years of experience in marketing, sales and development of high-tech products in the software and communications industry. Prior to joining Telelogic, she was Director of Product Planning for a leading global provider of telecommunications equipment. In this role her responsibilities included product management and product marketing for ISDN solutions and Voice-over-IP technology. Previously, Mrs. Stuecka was in a Sales position where she successfully established an international network of distributors and OEM customers for a fast-growing German provider of ISDN solutions. She started her career as a software engineer and project team leader at a large German software and system company. Mrs. Stuecka has a university degree in Computer Science from the University of Dortmund, Germany.



World Headquarters: Telelogic AB (publ.), PO Box 4128,
SE-203 12 Malmö, Sweden, Phone: +46 40 650 00 00, Fax:
+46 40 650 65 55

American Headquarters: Telelogic North America Inc.,
9401 Jeronimo Road, Irvine, CA 92618, USA, Phone: +1 949
830 8022, Fax: +1 949 830 8023

Web: www.telelogic.com, E-mail: info@telelogic.com