

Virtual Configuration Management

By Kent Gladstone, MSSE, MSIM, PMP, SAIC and Hayden Rolando, Editor

Executive Summary

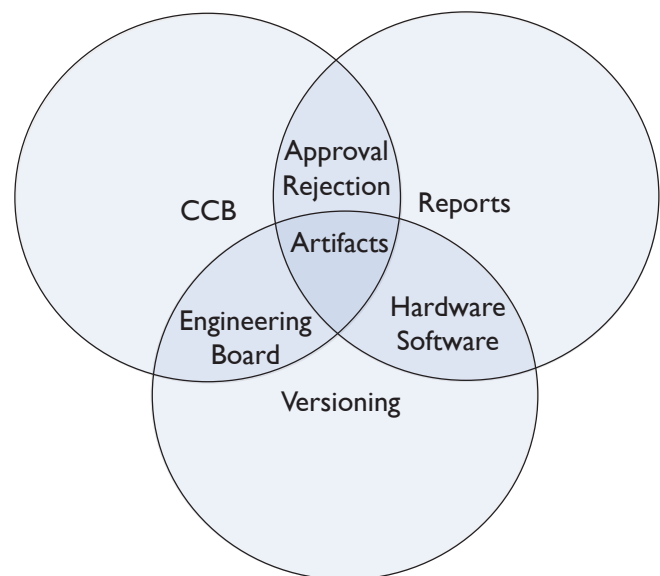
Configuration management (CM) is more than keeping a project inventory. It is the basis of project communication and controlling changes that affect the project, such as requirements or design change. With globalization, traditional CM is no longer a viable option. This paper explains the importance of CM to the project manager, and offers suggestions on how to digitize this discipline.

Configuration management (CM) is a key component of project management, giving the project manager a center of communication and a view of project status. Because CM allows the processes and the history of changes within a project to be electronically accessible, the project manager, whether leading a geographically dispersed team or a collocated team, can use CM to view the health of the project's system. Unfortunately, when there is a need to make budget cuts, many project managers tend to drop configuration and quality management before considering alternatives. This problem was perhaps most evident during the Y2K effort and continues to be pervasive (Starbuck, 1999). Several studies show that the lack of CM significantly increases the risk of project failure (Weatherall, 2006).

Configuration management traditionally has three components: version control, change control and reports (audits), all of which include human interaction, documentation and the option to be stored digitally. The three components are managed and coordinated electronically with little human involvement except in entering and extracting information.

Most people mistake CM for an inventory system, but ideally it is a communication venue that enables the team to track each others' work and to convey progress to the project manager. Figure 1 demonstrates how reports can show the status of coding progress through versioning as well change control board (CCB) outcomes.

Figure 1: CM components



Virtues of Configuration Management

Several principles and practices are associated with CM. These are well defined in "Configuration Management Principles and Practices," a whitepaper written by Kandt (n.d.). The premise is that CM, when applied correctly, avoids duplicating artifacts (e.g., ordering two copies of the same version of software when the project needed only one) and encourages correctness (e.g., by ensuring that the correct version of software is installed into a server). It is good practice to put work packages and

their related activities and documentation into CM, but it has been my experience that project managers often include only customer deliveries. Despite this, CM provides an opportunity for the customer and the project manager to communicate clearly the status of those deliverables.

By tracking configuration management metrics, a project manager can identify progress and balance the triangle of cost, schedule, and quality. In software development, for example, units of code are not put under version control until they have been unit-tested and the developer is satisfied that they meet the requirements. Units of code can be equated to a set of activities; these activities can be rolled up into a work package that appears in the work breakdown structure, as well as the schedule. The project manager can check the progress of the work package by way of CM reports, identifying newly delivered code in the system environment. The project manager can also check the actual delivery against the scheduled delivery of that package. Another benefit of CM is its use in tracking requirements changes. If these changes are not tracked, the project manager will lack data to determine if the project is going well or to know if it is behind schedule due to the volume of changes requested; failure to track requirements changes is frequently blamed for failed projects.

In “What We Can Learn About Productivity and Effectiveness Metrics from Software Design and Development, Training, and Marketing Communications,” the authors laud the use of metrics to measure the health of the schedule, to ensure that the quality management steps to produce code have been taken, and to meet requirement expectations (Carliner, Constantinides, St. Amant, & Walstad, 2002).

Most CM uses different models to meet the unique project management requirements. The best known is ISO/IEC 15504, developed by the Joint Technical Committee of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The basis of this model is:

1. Identifying, defining, and baselining all relevant items generated by the process or project
2. Controlling modifications and releases

3. Recording and reporting the status of the items and modification requests
4. Ensuring the completeness and consistency of the items
5. Controlling storage, handling and delivery of the items

Another model more commonly used in the United States is the capabilities maturity model (CMM) (“A Framework for Software,” n.d.). In contrast to other models such as the

ISO/IEC 15504, the CMM uses specific processes or standards, and includes:

1. The set of artifacts (configuration items) under the jurisdiction of CM
2. How artifacts are named
3. How artifacts enter and leave the controlled set
4. How an artifact under CM is allowed to change
5. How different versions

- of an artifact under CM are made available and under what conditions each one can be used
6. How CM tools are used to enable and enforce CM

CM reports can, and do, report statistics. These statistics, like earned value management, show trends and predictive behavior. In a white paper by Crespo, Matrella, and Pasquini (1996), it is stated that “the current software engineering techniques, and, in particular, the use of configuration management techniques and tools, allow the manager to keep track of the evolution of a program during its life cycle and of the faults that have been found during its testing and operational usage.” The project manager can use CM-related statistics and apply them to project management activities such as scheduling, enabling him to enhance the project’s planned value data. Based on the results, he is better able to predict the behavior of the project based on past performance and completed (or not completed) tasks.

Virtual Configuration Management

Given that artifacts (code, documents, and lists of hardware with their configuration and location) are, in theory, already filed electronically, it seems only natural that human interaction with these would evolve to be automated as well. The new capabilities coming out in the market today make virtual CCBs doable. In 1999 when I was a project manager being assessed at Software Engineering Institute CMM Level 3, I had everything digitized. To prepare audits

“ By tracking configuration management metrics, a project manager can identify progress and balance the triangle of cost, schedule, and quality. ”

and save time, I had any documents that required signatures scanned into a centralized repository. Our team was located in McLean, Virginia, while the assessment team was in San Diego, California. We gave them access and passwords to our server, and even cut CDs for their use. At first they refused to believe that we digitized everything and insisted that we print everything and fax it to them. It was only after they received over 1,000 pages of data from us, saw all the paper and interviewed the team that the assessors realized that digitization was possible. From then on digitization was commonplace throughout the organization.

Virtual Change Control

I came on board a project that comprised more than 200 people across the United States, four different development areas, and several commercial products. From the outset, there were several lessons to be learned.

Our first order of business was to develop plans, including the configuration management plan (CMP). The CCB section of the plans included a list of people who would participate in the CCB and the reporting requirements. The CCB included a core set of team leaders. The concept was to bring in participants depending on the topic of the change request. What the program manager failed to direct was that functional leads were considered optional participants as well. Further changes were incorporated to include the test leads for the different functional areas that might be affected. The list of people grew from a core five people with five people revolving in and out of the CCB depending on the functional area in question.

Our next step was to develop procedures for change, including a list of configuration item artifacts (documents, code and hardware), along with expected delivery dates. The difficulty in this step was twofold. First there was the issue of collecting the configuration items. CM depended on functional and development leads to provide this information. Several meetings led to agreements on how these items would be identified, how they would be put into configuration management, and who would be the contact person to track changes to these items. Second, we realized the need for four different procedures to accommodate the life of the artifacts:

- Design documents and the requirements, which would most likely affect cost and schedule, followed formal CCB procedures.
- Developmental artifact (code, procedures, and database scripts). Adding these artifacts into configuration management using the version control tool associated with the application (Perforce for Java

code, and PVCS for Oracle scripts) and notifying the appropriate people of this event.

- A process to approve or reject changes, or fixes, to these development items.
- A process for system administrators to be notified of new uploads, system testing, or any other activities that would affect the system or the users.

Although initially the CCB tried to meet at a specific time using teleconference, this proved difficult because of time zone differences. After two failed attempts, the configuration management team looked into collaboration tools. The requirements included:

- Allowing links to a document, rather than disseminating copies to each participant when CM used e-mail
- Electronic voting capabilities for the change request to each member of the CCB
- A discussion or comment field for each member of the CCB for the change request
- Enabling a deadline on the voting period and automatically disallowing votes after the allotted time
- E-mail notification to CM when a vote or comment was submitted
- Having the option to make votes available to the public, or not

No single tool provided the CM with all the metrics needed. Other requirements included:

- A change request form that included the originator of the request, a change control number, open and close dates
- Ability to track the number of open or closed request changes
- Ability to categorize the configuration items and count those that were open or closed request changes

The tool selected for CCB activities allowed members to review submitted change requests at their convenience. The central point of contact was the configuration manager; any questions or discussions were directed to this person, who in turn forwarded these to the appropriate members of the CCB. All documents associated with the change request were linked into the request so that only one copy of the document existed electronically. The use of links proved to be an efficient way of avoiding having too many versions of a document floating among the team members. The configuration manager set a voting deadline for each

request. If a member did not vote, it was assumed that he or she consented to the change. There were two benefits to this method: 1) changes could be reviewed at any time by the board members, and 2) people requesting the changes could view comments, discussions and the final tally of the vote. All transactions became part of historical files. Figure 2 shows the process flow we used for the project. This process allowed the CCB to quickly make decisions in a fast track project regardless of time or geography.

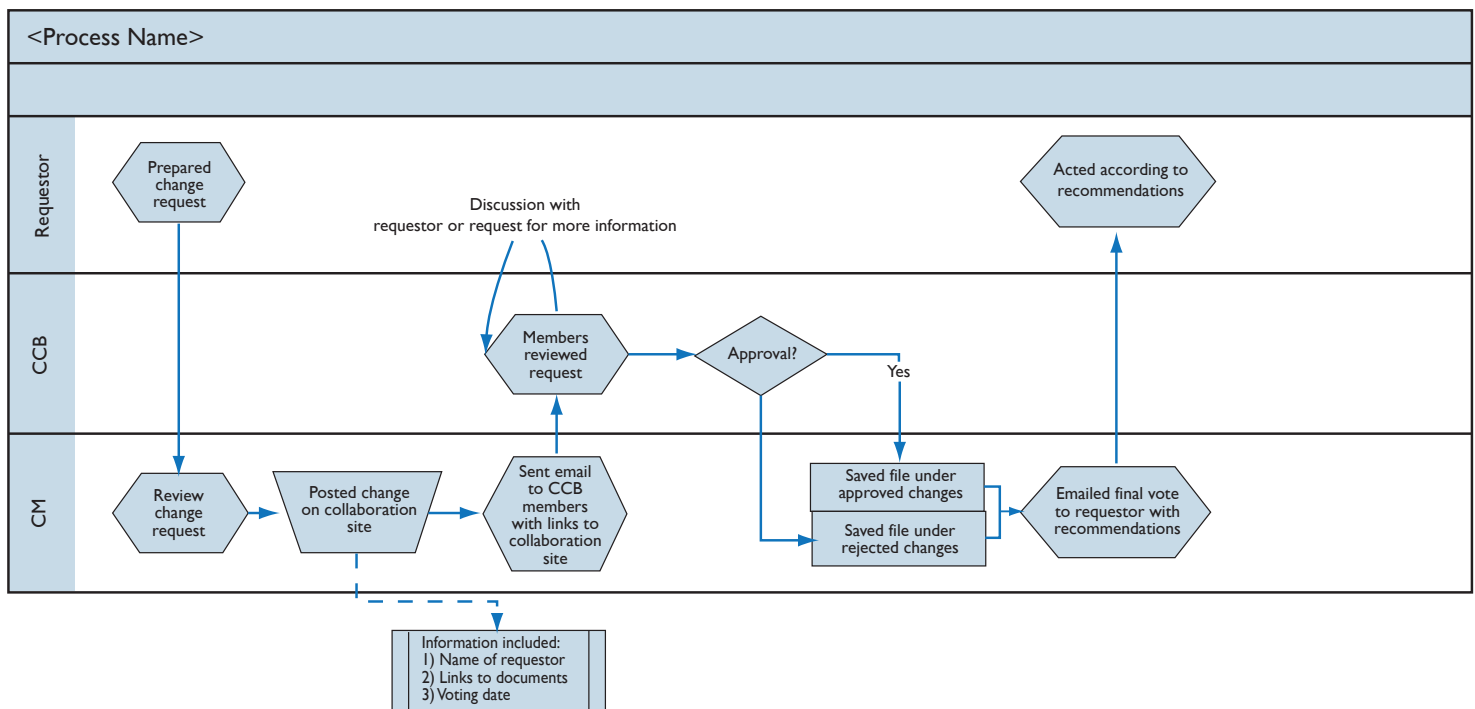
The metrics tool that was selected doubled as the testing tool. Developers mediated the testing versus the change requests through drop-down lists that identified the entry. The categories were “Test” and “Change Requests.” The change request form had more categories, which identified the type of document and the functional area of the change. These criteria allowed weekly CM reports identifying the number of change requests that were open or closed by category. The reports also showed the length of time between opening and closing a completed change after the request was approved.

For example, the product design document sections were stored as separate files within a folder in the collaboration tool. While working on these documents, an analyst would find missing, wrong, or duplicate information. The analyst would then “check in” the document into the folder of the collaboration tool. Users

used the testing tool to create the change control ticket and notified CM. Unfortunately the testing tool did not allow external links to go directly to the change request ticket. CM recreated the ticket in the collaboration site, added a brief description of the request, added the link to the document found on the same collaboration site, and set the final voting date. Members of the CCB reviewed the document and cast their vote. In the meantime CM monitored the vote or requests for more information or discussions. When the final vote was cast, CM updated the status of the change request in the testing tool and notified the requestor of the decision by e-mail. All e-mail pertaining to the change request was filed as one package into either an Approved or Rejected folder.

The project also had a test control board tasked with mitigating identified code and data defects. They used the testing tool allowing the lead tester to convene with the development leads to review defects, enhancements and completed repairs. These daily reports showed histograms identifying areas with the highest number of code and associated defects. This testing tool also helped the project manager ensure that enough resources were available for the critical path. These reports were summarized for the executive committee. If a problem could not be resolved within the scheduled delivery date, the test lead created a change request form and followed the CCB process.

Figure 2: CCB process used



Recommendations

According to the 1998 Bull Survey, 57% of projects fail because of miscommunication or bad communications (“Failure Causes,” n.d.). Twenty percent of projects fail because of progress mismanagement. According to the Computing Technology Industry Association (CompTIA), nearly 28% respondents attributed poor communications as the cause of project failure (“Poor Communication,” 2007). The Standish Group’s survey of 2003 (BNet) shows 66% of 13,522 IT projects failed, whereas more than 50% were over schedule or had costs over-runs due to poor planning, unclear or changing goals, areas that entail communication (Taimour, 2005). These surveys show that no improvement in the number of project failures has occurred in the past 12 years.

Communications is paramount to the success of a project, and configuration management, through CCBs, ensures awareness of the project’s health. CM communicates the health of the project by reporting defects and changes that are not meeting schedule, and by identifying when there is a lack of (or an overabundance of) resources for a given task of the project. The project manager can convert these numbers in terms of cost and schedule. Smart project managers take advantage of CM activities to coordinate and communicate within and outside the project through CCBs or CM reporting.

Most CM activities can be performed electronically. Several software tools on the market today make this viable. Virtual CCB is especially beneficial for projects having geographically dispersed teams. These tools allow the configuration manager to post the request and links going directly to supporting documentation. Many systems are available that allow discussions to be forwarded to parties outside the CCB, thus keeping a record of issues, questions and answers. Finally, these tools allow the CCB members to vote at their convenience.

Engineering control boards are by their very nature more unwieldy because of their need for real-time discussion. Although the item for discussion remains electronically available, the nuances are more difficult to moderate. If a teleconference is held to discuss the item, a recording feature can be used. The configuration manager, who should be in attendance, manually enters the disposition of the item when the meeting is done, and then tracks the final decision to closing.

The configuration manager creates reports for different stakeholders and coordinates project meetings that use these reports. These reports can be project management—level reports showing the number of changes throughout the life cycle of the project, task lead reports on the status of a defect, or a “bill of laden” for system administrators to

move components from development, to testing to production. Configuration management is the central host for communication on a project.

“ Twenty percent of projects fail because of progress mismanagement. ”

References

- BNet. (2003, March 25). Latest Standish Group CHAOS Report shows project success rates have improved by 50%. *Business Wire*. Retrieved January 2008, from http://findarticles.com/p/articles/mi_mOEIN/is_2003_March_25/ai_9916967
- Carliner, S., Constantinides, H., St.Amant, K. & Walstad, C. (2002). What we can learn about productivity and effectiveness metrics from software design and development, training, and marketing communications. Retrieved January 2008, from <http://saulcarliner.home.att.net/idbusiness/litreview.htm>
- Crespo, A. N., Matrella, P., & Pasquini, A. (1996, November). *Sensitivity of reliability growth models to operational profile errors*. Proceedings, Seventh International Symposium on Software Reliability Engineering. White Plains, NY.
- Failure causes. IT Cortex. (n.d.). Retrieved January 2008, from <http://search.yahoo.com/search?p=1998+Bull+Survey+57%25&ei=UTF-8&y=Search&fr=yfp-t-324&cxargs=0&cpstart=1&b=11>
- A framework for software product line practice*, version 5.0. (2008, February). Carnegie-Mellon Software Engineering Institute. Retrieved January 2008, from http://www.sei.cmu.edu/productlines/frame_report/config.man.htm
- Kandt, R. K. (n.d.). Configuration management principles and practices. NASA *Jet Propulsion Lab Technical Reports*. Retrieved July 2008, from <http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/10507/1/02-2525.pdf>
- Poor communication top cause of project failure. (2007, March 14). Retrieved February 2008, from Projects@Work website: <http://www.projectsatwork.com/content/articles235492.cfm>

Starbuck, R. (1999, March). Using CM to recapture baselines for Y2K compliance efforts. *Crosstalk*, pp. 7-10.

Taimour, A. N. (2005, November). *Why IT projects fail*. Retrieved February 2008, from Project Perfect website: http://www.projectperfect.com.au/info_it_projects_fail.php

Weatherall, B. (2006, December 19). The Business of Software Development. *CM Journal*. Retrieved March 2008, from <https://www.cmcrossroads.com/content/view/7415/264>

About the Author

Kent Gladstone, PMP, has been in the IT industry for 20 years. She has risen through the ranks from junior analyst to senior software and system engineer. Her work has included programming, database administration, configuration management, quality assurance, SEI assessor and project manager. She holds a Masters of Science in Systems Engineering and Information Management, and acquired the Project Management Professional Certification in 2007. gladstonek@saic.com