# Reality Configuration Management

Donald E. Casavecchia
*ACS Defense, Inc.*

*You are not alone if you have found that in your job as configuration management (CM) lead, you are given less than optimal support for your task, or are asked to scale back your CM goals. This author faced these dilemmas in his CM position at a small systems development facility. Here is how he adjusted his CM practices based on facility resources and management's commitment to CM.*

Do you work for a small systems development facility? Does management profess a desire to implement the Software Engineering Institute's (SEI) Capability Maturity Model® [1] (CMM®) by next fiscal year? Are you the one they hired to miraculously transform their hobby shop into the lean mean systems generating machine they envision? Are you getting something less than the 100 percent support you were originally promised?

Three years ago, I joined a small systems development facility as configuration management (CM) lead, a newly created position, and was initially tasked with getting their software under control. This is a government facility with engineering contractors supplying the labor, and government engineers filling management positions as technical advisors.

Once past the security clearance barrier, I determined the facility was consistently in the process of developing some 20 separate projects simultaneously, each with six or less project members, with start to finish schedules ranging from three months to two years. As soon as one project ships its deliverables, another proposal is turned into real work, and a new project is kicked off. Post-delivery system support ranged from no support to the full operations and maintenance (O&M) regiment. Project team members are a mix of seasoned engineers and technicians that build complex systems entirely behind closed doors with project-obtained resources.

Prior to my arrival, senior management provided formal CMM training for every employee. One year later, project managers and technical advisors were required to attend repeat CMM training sessions. The facility chief and deputy appeared to want repeatable process-oriented systems development for their facility. They failed, however, to set forth the policy and direction to accomplish it. Their desire to step up a level from producing ill-managed prototypes to cost- and schedule-driven first articles with detailed build-to documentation was not being realized.

They held an *all-hands* on-site briefing to emphasize improvement goals. Unfortunately, they directed their frustrations down their own organization rather than coordinate across the customer base for better quality assurance requirements and adherence to stricter standards. Without customer requirements for repeatable processes with meaningful milestone reviews, the underlying work ethic remained "do only what it takes to get the job done."

> *"Instead of several CM tools from competing vendors, one was selected as the center's standard, and training became an across-center effort instead of each project sending their engineers for vendor-supplied training."*

By interacting with project members, I was able to identify the following reoccurring CM deficiencies:

- Vague, often undocumented requirements.
- *Rough-order-of-magnitude* proposals with cost and schedule estimates usually provided before requirements were firm.
- Follow-up project plans that failed to provide enough detail for project members to understand what it was they were building.
- No commitment to make project plans living documents.
- Chassis, cable, printed wiring board (PWB), mechanical, and schematic drawings too loosely controlled, with far too many redline variations that contributed to *best guess* build-to documents.
- Lack of rigid inspection checkpoints on drawings and PWB build-ups.
- Minimal software design documentation and few written unit test plans.
- No agreed-upon milestone identified for starting formal change management.

After several sessions with the lead system engineer during several months, I concluded that our small systems development facility, with less than 70 contractor and 15 government employees, could not dedicate the resources to establish a systems engineering workgroup and charter it with developing and implementing center policies, processes, and procedures. I witnessed our lead engineer receiving even less upper management support than I received. Eventually, he was dismissed from the program (and not replaced). It was apparent that if I wanted to improve CM practices, it would require a *grassroots* approach.

My first three months were spent developing a *makefile* [2] build system and converting a project's CM system from a homegrown source code control system (*sccs*)-based system [3] to one based on a commercial off-the-shelf (COTS) CM product. This quickly established me as a hands-on team player and gained me the support of key engineers and managers.

With one *fire* extinguished, I still had 19 other projects in need of CM improvements. With nobody yelling fire, I persuaded management to let me design and establish a local area network (LAN) to install and maintain a common set of engineering tools to be used across projects. Instead of each project purchasing, installing, and maintaining their own development environment on stand-alone workstations or makeshift workgroups, we pooled selected project products, switched from node-locked to floating licenses where possible, and established a

| Categorized | Project Type | Risk |
|---|---|---|
| Class 1 | Concept Design | High |
| Class 2 | Development Design | Moderate to High |
| Class 3 | Integration Design | Moderate |
| Class 4 | Application Enhancement | Low to Moderate |
| Class 5 | Application Maintenance | Low |
| Class 6 | Production | Low |

Table 1: *Classes of Projects Defined*

development infrastructure.

Previously isolated workspace offices now received LAN drops, meaning workstations could utilize the LAN to allow project developers to access infrastructure products. By providing a common infrastructure product base for Windows and Unix platforms, management could budget and coordinate training targeting infrastructure products and set policy and procedures for project teams to follow. Instead of several CM tools from competing vendors, one was selected as the center's standard, and training became an across-center effort instead of each project sending their engineers for vendor-supplied training.

After using this evolving, controlled infrastructure for three years, our center defined six classes of projects (Table 1).

We found that we often pursue a concept design project that later spawns separately funded integration design and/or production projects. Fielded systems often call for incremental advancement of a design (Class 2) or major enhancement (Class 4) projects. Each new request for a proposal is now categorized as Class 1 through 6, and each class carries predefined level-of-effort disciplines like CM, quality assurance, and documentation support.

Most of our Class 1 and 2 projects fall under the general descriptions of *proof of concept*, *investigate leading edge technology*, *rapid development of a prototype*, *conduct a trade study on xyz technology*, etc. These projects are usually short-scheduled with limited funding. Often, they are meant to only convince the customer that we could exploit the technology and deliver a system. Because we often build a *working model* or *prototype* and often write white papers as part of these project executions, capturing the more important parts of the project is all CM is able to achieve; often, we receive a media with the soft copy deliverables.

Sometime later (weeks, months) we may get tasked with revisiting the earlier effort and building a first article to demo. The follow-up task is a new project, separately funded with additional requirements. Since we have proved the concept, it is now less a risk and more an existing technology Class 3 or 4 project.

The following four sections in this article depict CM methods available to our project managers to satisfy CM requirements for the six classes of potential projects with which this center is involved. When asked to quickly (less than 90 days) produce a narrowly defined system (Class 2 project), the Archival Method is appropriate. The Archival Method is selected for requests of additional copies of a system we designed and built 18 months ago as an integration design project using the Open Repository CM Method (the additional copies would be categorized as a Class 6 project).

The Open Repository Method is usually appropriate for a concept design (Class 1) or development design (Class 2) project where schedule is usually longer than three months, and deliverables are often prototypes or working models.

The Focused Repository Method is always appropriate for our *bread-and-butter* systems that have proven themselves and when a hardware and/or software enhancement (Class 4 project) is requested. The Focused Repository Method is usually selected when problems are reported (Class 5) on fielded systems for which our center is on the hook for lifecycle support.

With the bulk of our delivered systems living short life cycles (mostly due to technology advancements), overCMing can be a real cost and schedule issue. If/when a fielded system (Class 3 project) exceeds expectations and takes on a long life cycle (greater than four years) with requests for additional copies with expanded functionality, we sometimes have to resurrect an Archival Method repository and bring it up to Focused Repository levels of CM resource commitment.

We have yet to achieve a system that provides enough metrics to seriously examine and tune our CM processes (Optimized Repository). I look forward to that day. The "CM Discipline Progression" depicts our least restrictive to our most restrictive CM method. I would love to report that every time our center has selected minimal CM (Archival Method) for a project, we have not regretted it. Likewise, we have gone all the way with the Focused Repository Method only to watch our system sit on the shelf with no takers.

## Archival Method

The Archival Method (characterized as a *capture* technique) is selected when minimal CM is appropriate. The program manager (PM) specifies the schedule milestone(s) at which the baseline will be archived and identifies the set of system components for capture. Minimum CM occurs when the selected milestone is System Acceptance Test (SAT) and an O&M phase is not specified. When multiple milestones are designated, or an O&M phase is required, all soft copy files associated with the milestone should be placed into a project repository and *labeled* with the milestone acronym. CM technicians work closely with project members to catalog system components, down to lowest replaceable units (LRUs), comprising the project at the specified milestone.

### *CM Requirements*

The PM is responsible for identifying the set of system components to be archived. Software system components, comprised of source files (no intermediate or build product files), are isolated from project work areas (preferably placed on a transfer media) after the following is verified:

- Builds cleanly, without errors.
- Successfully executes.

Each software system component's build and execution (run-time) environment must be documented to ensure its reproducibility. The following are the minimum details to include:

- Development and target platform nomenclature (if appropriate), including identification of any special boards, cables, peripherals, and drivers.
- Operating system version and list of patches.
- COTS and/or government off-the-shelf (GOTS) version, installation order, feature selections, configuration files, patches, and integration code.
- Compilers, linkers, and loaders, including their version and switch settings.
- Environment variables and their settings.
- Dependencies on any third party libraries, identify source and version, and use restrictions.
- Actual license certificates, keys (dongles), and maintenance agreements.
- Test tools, either internally developed

or commercial.

Each soft copy document turned over to CM should be saved in a format that turns off any tool propriety revision display feature[4]. This is especially important for drawings from computer-aided design COTS packages.

Hardware turned over to CM (usually for transfer to an external O&M facility) will be appropriately identified and classified. Bill of materials (BOM) must be detailed down to line replacement units (LRU). Any special handling or environmental stowage requirements must be made known at the time of turnover.

#### *Strengths*

- Simplifies project member's work environment.
- Allows staffing the project with less experienced workers.
- Minimizes impact to project members.
- Reduces training needs.
- May result in shorter system development timelines.

#### *Weaknesses*

- Places most of the responsibility for executing CM onto CM technicians who least understand the project's organization, goals, and deliverables.
- With respect to version control, this method merely captures a snapshot set of system components corresponding to the designated milestone. Component versions created between archived snapshots are lost.
- With respect to change management, when issues/problems are not documented or processed via a review/approval process, management also has no insight as to the number of problems fixed (product quality) or the way that problems are fixed (design quality).
- With respect to configuration control, this method frequently requires an inordinate effort from CM to configure the baseline, i.e., understand the project components hierarchy (software file system restructuring is perhaps the worst case) and identify and apply a meaningful software labeling scheme.
- With respect to status accounting, little or no metrics are available or collected. It is difficult to associate between changes to components and the driving requirement, e.g., no way to track revision three to system component X with the corresponding issue/problem report.
- With respect to auditing, no formal baseline exists until a *capture* milestone is executed. Customer insight to completed work is not verifiable. Change implementation is invisible.

## Open Repository Method

The Open Repository Method (characterized as a *contribution* technique) is selected when management desires closer control and progress review capabilities. Project members are tasked with routinely submitting soft copy versions of their work into a project repository. More dynamic and comprehensive than the Archival Method, the Open Repository Method ensures that aggregate changes to a component are contributed to the repository as an identifiable version. Typically, an *add to* or *check-in/check-out* interactive exchange is employed to mature the repository from project start-up through all phases of the project's life cycle. Management review of a project is greatly simplified when every project member is conscientiously contributing to and/or entering changes into the repository at predefined milestones.

#### *CM Requirements*

It is necessary to comply with each Archival Method requirement in addition to the following:

- Pre-coordination and agreement between CM and the project of a file system structure to accommodate all known system components and their interfaces.
- As much advanced notice as possible on project selected development tools to allow for interoperability evaluation to determine the best way to save and stow soft copy files from these tools into the repository.
- A mandatory repository check-in comment that appropriately describes the aggregate of changes to a component since last check-in.

#### *Strengths*

- Anyone with permission to access the repository can monitor when project components enter the repository as change sets at designated milestones.
- Work is centralized to a single file system simplifying backup procedures.
- Facilitates project member communications because everyone knows where to look for project items.
- Provides integration between development tools (like Microsoft's Word and Visual C++) and the repository for direct check-in/check-out processing right from the tool they are using; with some tools, compare and merge capabilities can exist.
- Increases the opportunity for *common* software and software reuse.
- Provides management with meaningful metrics on project and component size and complexity. The number of versions for an item may convey the level of development difficulty or number of problems overcome.
- Reduces the data entry workload on CM by distributing the repository entry responsibility to each project member.

#### *Weaknesses*

- With respect to version control, although the open repository method may produce file versioning, without an organizational policy that mandates following a change-management procedure for each file update, discrete version control is not being exercised.
- With respect to change management, although the open repository method allows a comment to be entered when a repository file is added, modified, or replaced, without a formal issue management process with board adjudication, change management is not being exercised. Repository changes are not subject to formal review.
- With respect to configuration control, without formal change management, controlling a project's configuration by defining labels that correspond to schedule milestones and manually apply them is about the best you can achieve. The open repository method does not achieve verifiable baseline advancement. An opportunity exists for unsolicited enhancements.
- With respect to status accounting, version metrics and associated comments are available and can be reported, but correlation of problems fixed to specific files changed is still missing.
- With respect to auditing, it provides both management and customer the opportunity to review soft copy files (including schedule updates) in the repository, but fails to provide issue-management metrics (action item, issue, problem report, engineering change proposal, engineering change notice, revision-level change, etc.).

## Focused Repository Method

The Focused Repository Method (characterized as a *directed* technique) is selected when management has CM policies/procedures institutionalized within their organization and the intent is for full control of workflow processes for the project[5]. Project members are indoctrinated on CM policies, issue documenting, and

change management procedures. The PM is indoctrinated on management review policies and the various metric reports available from status accounting.

The Focused Repository Method is a disciplined process-oriented approach to achieving CM during project execution. All project members have access to the issue-tracking tool and all issues are documented when they become known. The timely review and disposition of every issue conforms to the organization's change management process. For our organization, two Configuration Control Boards (CCBs) handle the disposition of issues. A project-level CCB handles all issues that do *not* affect schedule, cost, or design. A program-level CCB dispositions all schedule-, cost-, and design-related issues. Management overview of project execution is near real-time because issues are immediately surfaced and dealt with.

### *CM Requirements*

It is necessary to comply with each Archival and Open Repository Method requirement in addition to the following:

- Project plans must identify major system components and supply support documents that fully describe:
  - Hardware components detailed to subassemblies, LRUs, with drawings, board layouts, and accurate BOM and formal build-to documentation.
  - Drawings and layouts comply with IPC-A[6] revision standards.
  - Firmware, vendor supplied or custom developed, maintenance plan.
  - Bundled COTS, GOTS, version, license and distribution agreements.
  - Software components, build order, build mechanism, build environment, run-time environment, release strategy, version description, and maintenance plan.
- Project members must attend CM training sessions covering tools, processes, and procedures.
- Project engineers responsible for tool selection must coordinate with CM on tool integration and upgrade tasks.

### *Strengths*

These include all the strengths listed under the Open Repository method plus the following:

- Verifiable change management.
- All repository changes are subject to formal review.
- All changes are captured.
- Project issues are documented, adjudicated, and dispositioned resulting in traceable system component changes within the maturing baseline.
- Provides for accurate file compare capability, a single change set that directly correlates with a single issue.
- Management has full insight into the number of problems fixed (product quality).
- Management can review exactly how an issue was fixed.
- Reduces CM technician's involvement with repository input.

### *Weaknesses*

- Requires organizational policy, procedure, and training programs, each subject to a continuous improvement effort.
- Customer buy-in, including compiling adequate cost and schedule metric briefings to convince customers that implementing effective CM gets them better products at cheaper prices within shorter development cycles.

*"A small systems development facility can achieve limited CM proficiency by selectively implementing CM disciplines across their business lines."*

## Optimized Repository Method

The Optimized Repository Method (characterized as a *tuning* technique) is selected for a project by management only after an acceptable number of projects using the Focused Repository Method have been completed and thoroughly evaluated. Isolated areas with weak processes and procedures, insufficient metric collectors, inadequate change-tracking information, forms and route slip inadequacies, high quality control failure areas, and high percentage test failure components are targeted for having their workflow processes fortified. Fortifications include the following: more stringent reviews, tighter version management, better testing (additional regression tests), additional required fields for capturing metric data, and a higher degree of system decomposition. Management targets a specific project for observation and evaluation using the improved CM practices.

### *CM Requirements*

It is necessary to comply with each Archival, Open Repository, and Focused Repository Method requirement in addition to the following:

- Project members must attend CM training sessions covering tool, process, and procedure enhancements or replacements.
- Project members must comply with entering additional form and route slip inputs.
- Project members must supply greater detail to required comment fields when assigned issue and change resolution tasks.

### *Strengths*

These include all strengths listed under the Open and Focused Repository Methods plus the following:

- The project is first to try out new tools, processes, and procedures.
- Increased metric data usually results in more accurate cost and schedule reporting.
- Companies executing disciplined systems development have an advantage when it comes to attracting good engineers.
- Companies executing disciplined systems development have the advantage over undisciplined companies that cannot bid (SEI/CMM competition).

### *Weaknesses*

- The project is first to try out new tools, processes, and procedures.
- May result in a longer system development timeline.
- Tendency for higher project cost associated with implementing process changes.

## Conclusion

A small systems development facility can achieve limited CM proficiency by selectively implementing CM disciplines across their business lines. As management realizes benefits from the relatively small resource investments associated with the Archival Method, a natural progression to Open and Focused Repository Methods becomes almost automatic as customers communicate their desire or need for system advancements.

Conversely, when the deliverable is a working model that proves a concept, or a design, cost, and schedule are the paramount requirements, Archival Method processes may be just the right amount of

CM. Government contracts awarded to the *low bidder* demand cost effective proposals with the basis of estimates receiving careful scrutiny. Minimizing CM often becomes a target for cost savings for small developing facilities facing the reality of having to present a winning proposal.◆

## Notes

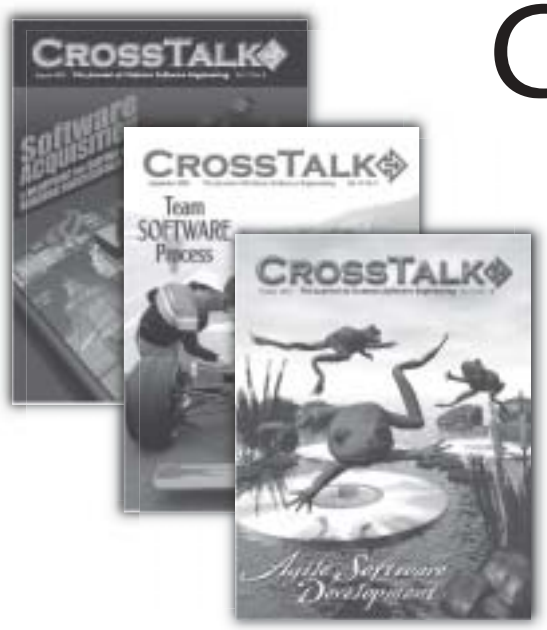
1. Substitute ISO9001, Capability Maturity Model® Integration℠, or Malcolm Baldrige Award, as applicable.
2. Makefile: A manually generated, specially formatted input file to the *make* utility, which contains information about what files to build and how to build them. The *make* utility streamlines the process of generating and maintaining object files and executable programs. It helps to compile programs consistently, and eliminates unnecessary recompilation of modules that are unaffected by source code changes.
3. A source code control system (SCCS) allows you to control write access to source files and to monitor changes made to those files. Allows only one user at a time to update a file, and records all changes in a history file. SCCS is a bundled Unix utility.
4. Our center's CM tool is integrated with our desktop office package to automatically display two versions of a non-binary file in revision mode.
5. Our center selected Rational's ClearCase product as its standard for creating and maintaining individual project repositories because ClearCase allows engineers, technicians, and managers to work directly in the project repository.
6. IPC: From 1957-1999, IPC stood first for *Institute for Printed Circuits*; later it became *Institute of Interconnecting and Packaging Electronic Circuits*. In 1999, IPC changed its name to just plain IPC, which has its offices at 2215 Sanders Road, Northbrook, IL 60062-6135. The IPC provides the following information:
   - Standards to facilitate communications between suppliers and customers.
   - Guidelines with current industry positions on a wide range of subjects.
   - Research to solve industry problems.
   - Correlation of industry test methods.
   - New developments in interconnection technology.
   - A monthly publication called Relay.
   - Provides training (and certification) at U.S. and overseas sites.
   - Maintains a web site at <www.ipc.org>.

   For example: IPC-A-610C – Acceptability of Electronic Assemblies: This standard is a collection of visual quality acceptability requirements for electronic assemblies.

## About the Author

**Donald E. Casavecchia** is director, configuration management/quality assurance (CM/QA) with the Warrenton Operations Strategic Planning Group of ACS Defense, Inc. He is a hands-on CM manager with more than 20 years experience with the Department of Defense and government projects, including assembly, BASIC, FORTRAN, C, and script programming, setting up and maintaining software development environments, and supporting embedded systems development. Casavecchia believes in implementing practical CM/QA solutions.

**ACS Defense, Inc.**
**P.O. Box 700**
**Warrenton, VA 20188**
**Phone: (540) 349-3762**
**Fax: (540) 349-3517**
**E-mail: donc@wtc.gov**




# Call for Articles

If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are especially looking for articles in several specific, high-interest areas. Upcoming issues of CROSSTALK will have special, yet non-exclusive, focuses on the following tentative themes:

**Commercial and Military Applications Meet**
*June 2003*
Submission Deadline: January 20, 2003

**Defect Management**
*August 2003*
Submission Deadline: March 17, 2003

**Information Sharing/Data Management**
*September 2003*
Submission Deadline: April 17, 2003

**Please follow the Author Guidelines for CROSSTALK, available on the Internet at: www.stsc.hill.af.mil/crosstalk**
**We accept article submissions on all software-related topics at any time, along with Open Forum articles, Letters to the Editor, and BackTalk submissions.**