

# ***Getting the Most Out of Your Change Control System Measuring Software Quality***

**By Pete Baxter**

## **Introduction**

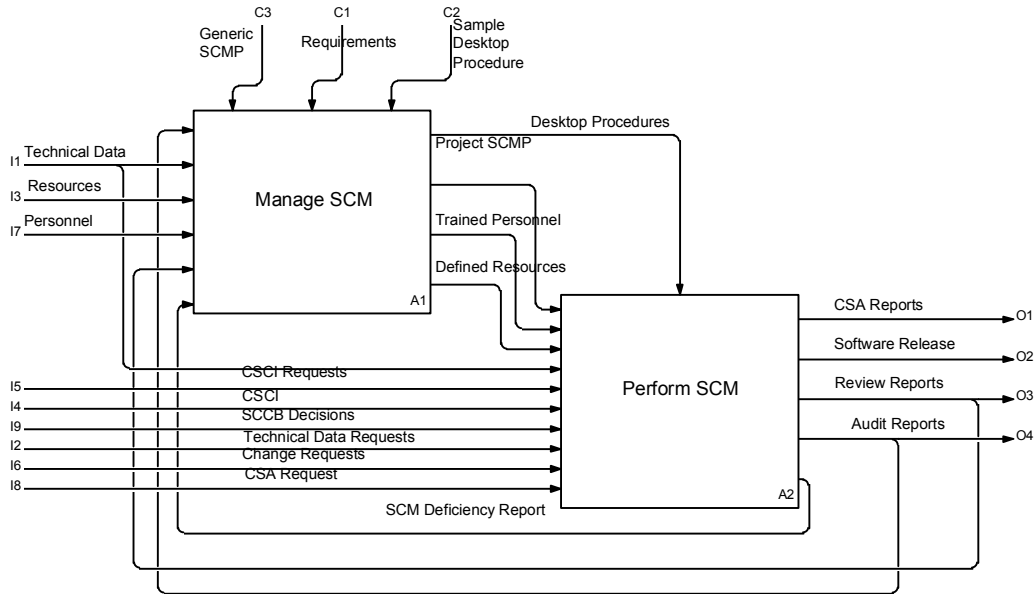
Configuration management (CM) tools are often cited as the first process automation implemented when establishing a software development environment. The only other tool more commonly found is the editor/compiler. CM tools that support the entire CM process will have two components, one for a file and configuration control part and a second for change management. The first component, the CM part, functions as a typical source control tool, handling such tasks as check-in, check-out, branching, versioning etc. The second component, a change part, provides a means to manage and track the change requests, usually either a defect or enhancement, which have been reported against the software.

Recent studies have shown that despite the availability of CM tools, software development projects continue to suffer from software quality problems caused by the inability to control the configuration of the product and changes made to it. In this paper, software measurement is used to gain a deeper and broader understanding of software product quality, starting early in the life cycle and continuing until successful completion.

## **Software Configuration Management**

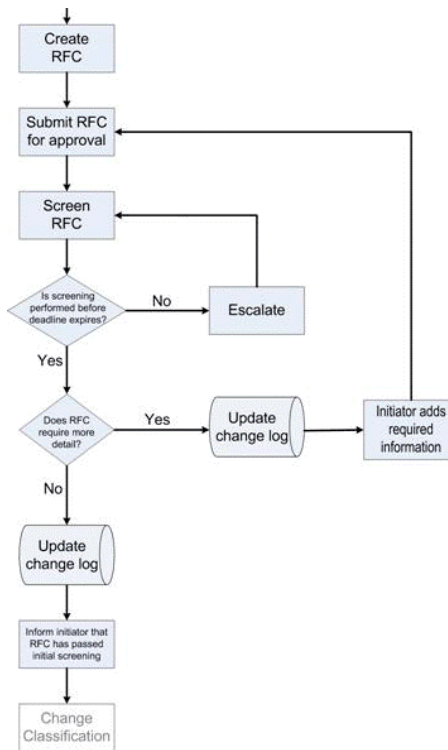
The purpose of Software Configuration Management (SCM) is to establish and maintain a product's integrity throughout the project life cycle. If a typical software process is being followed, SCM is used throughout all project phases, including analysis, requirements, design, implementation and testing/verification phases. Usually, SCM consists of two separate activities: 1) managing and 2) actually performing configuration management.

The following diagram, taken from a "Software Configuration Management Process Description" document developed by the U.S. Navy Space and Warfare Center, demonstrates the relationship between the management and performance of SCM. The diagram highlights the fact that someone typically reviews the candidate changes, selects all or a subset, assigns them to a project and then monitors the project to make sure progress is being made. The management aspect is done in parallel with what software developers do when they use their SCM tools to actually perform the SCM process.



The diagram (provided by Microsoft) at right presents a simplified view of the workflow for a change control process. Notice that some of the workflow at right is carried out in both the management activities and the perform activities.

The key point to any change control process is having a set of decisions where the flow of changes can be controlled. Management needs to review and approve change requests. Engineering needs to be aware of them and actually implement them. Other support activities, such as documentation, testing, customer communication and support need to accommodate change requests which are assigned to a product.



Notice in the above diagram that there are decisions for classifying a change request (called a Request for Change, or RFC), and then handling it. The figure demonstrates that not all change requests are accepted, and there are many reasons why it may not. Reasons to not accept a change request include: it is not valid, it was reported against a previous version and it has since been fixed, or that the user was not using the software correctly.

Once it has been determined that it is a valid change request, it must be classified, usually as a defect or enhancement and then given some type of disposition, meaning “what are we going to do with it?” The disposition could be that it is assigned to the current software release. If the release date, content and resources are fixed, then it might be deferred. In some cases, you might simply reject it.

Change management and change control are vital techniques for controlling project scope. The requirements process contributes (hopefully) one part of the content of a product – the other part comes from the change process. If you don’t manage change, you are admitting that you won’t be able to control project scope.

## **What Happens When Software Configuration is Not Managed?**

Before getting too far, let’s clarify the terms used in the SCM process. There is no common agreement on SCM terminology, so I provide the meanings appropriate for this white paper. (I don’t refute other definitions – I simply want to clarify the concepts being described.) The term “software configuration” means the complete set of files and resources required for a software product to be built and then to run on its target platform. The files required for the software configuration are tracked in a CM tool so that all files can be retrieved and changes reviewed. A software configuration will have one or more “baselines” where each baseline contains either a specialization or a subset of the software product. For example, a project may define three baselines to be built, one with 25% of the requirements, one with 60% of the requirements, and one with 100% of the requirements implemented. Note that, using these definitions, the software configuration does not change through the three baselines.

A Crosstalk article entitled “Demystifying Software Configuration Management” (printed in 1995) described the following scenario as an example of what happens when software configuration is not managed:

### **The Day Before a Progress Review with the Customer**

You are the manager of a software development project. You are gathering status data in preparation for your meeting with the customer. When you ask why a low-priority module has been changed so many times while a module that the customer is interested in has not been started, you discover that one of the

engineers has been trying to improve the module by making warning text appear in day-glow orange. Since the customer does not have color monitors, the engineer has wasted valuable time and money making unnecessary changes to the software.

The article has other, equally amusing, scenarios for poor SCM. But, the point of the scenarios is to demonstrate that there are a number of impacts to not having an SCM process, or not following (in the case where you have one).

The list of what goes wrong if you do not control the software configuration is long and painful. You may recognize these problems happening in your organization. They can be solved by simply having a change control tool and process. Some of the problems are the following:

- Changes made by different people collide and over-write one set of changes, causing the product to fail
- The product cannot be consistently and (ideally) automatically built
- The product does not install correctly because the shipped configuration does not match the tested configuration
- The documentation does not match the product because changes were made to one and not the other
- There is no plan or estimate for how many files, changes or lines of code are to be developed
- The rate of change cannot be managed

Many readers may argue that because they are a small company/organization, or because they are in a very small group/team, or because they don't use the same engineering process each time that the SCM process cannot apply to them. To this argument, I can only say "Huh?" A more complete response is that there are hundreds of analogies that demonstrate the ridiculousness of that argument, include: you use the same facility for each development, you use the same people for each development, you sit at the same desk using the same personal computer, you use the same basic engineering approach to each project, the organization provides the same supporting services, and on and on. Even if you have only one person, a CM tool can provide automation that makes that person more efficient. The argument to not use a CM tool and process, in this day and age, is justifiable in only a few rare circumstances.

## **What Happens When Changes Are Not Managed?**

Change control, like configuration control above, is an essential part of the SCM picture. Using change control is vital for keeping a project on track with a chance of making the expected ship date. The list of symptoms is long and the effects on the project painful. Hopefully, none of these are happening in your organization.

Some of the problems are:

- The scope of the project is greater than planned or expected (if there is no plan)
- Changes are still being made when system or integration testing starts
- You have to keep retesting because the code does not stop changing
- Regression tests fail
- Problems fixed in previous baselines appear in subsequent baselines
- Functionality is included in the product that was not needed (and for which no testing or documentation was planned)
- Windows or pages in the software change when there was no request to change it
- The software does not includes fixes or enhancements that have been promised to customers

## How To Start Managing Software Quality

First, make sure you define your software change and configuration process. Even a simple one will suffice, as in the case of the following process definition:

“The software project manager shall maintain the list of change requests, and shall either close or assign each one to a software product to be fixed through software or configuration changes.

Software developers shall store all files and resources required to build the software product on its target environment in the SCM tool.

Software developers shall utilize the SCM tool to store changes to files and resources required to build the product.”

The sentences above capture two essential ingredients: who does it and what they do. If you think this is too light, then come up with a better version for your organization. But, however you do it, write down the process. In the sentences above, there is no mention of branching and versioning, or of a change control board. There are a lot of things not covered that probably should be, but you have to walk before you can run. Don't try to implement every technique that you know about related to SCM, or to use every feature of your CM tool. Instead, grow into the process that your organization needs in steady, continuous steps. The first step is simply to track what you have.

The second step is to begin controlling and managing the change process. One of the biggest problems plaguing software development is requirements creep. You may wonder why I bring it up here. The answer is that software developers are a crafty lot, and by making changes to the software, they can include new functionality by simply checking out the code and making changes. So, unless you can start controlling the change process, you can't control requirements creep. In fact, you can't control the scope of software development without using change control.

Third, start measuring the process to see what is really happening. Start small. Start with basic measures of what is happening in your change control system. Make sure you

establish a plan for the number of defects and enhancements in the project. Make sure that managers know, at any given time, how many are left to do. Make sure that you have some mechanism to review the closed changes and make sure that the scope of the changes is reasonable.

## Summary

A software configuration management process, including both management and technical tasks, is an essential part of the software process. Selecting and managing change is as important as actually making the changes. Without establishing a plan and then monitoring changes during the entire software project, the chances for delivering required content by a specific date are slim. By using a measurement process, including a simple set of measures, you can start coming to grips with the scope and progress of change in your software project. You may find that with just a little process, you actually produce a better product faster.

## Contact Information

For more information, please contact Distributive Management at:

**Distributive Management, LLC**  
2300 Fall Hill Avenue, Suite 201  
Fredericksburg, Virginia 22401

(800) 779-6306  
(540) 372-6491 (fax)

[www.distributive.com](http://www.distributive.com)  
[sales@distributive.com](mailto:sales@distributive.com)