Software Configuration Management

# Team Based Development for a Small Organization

## An effective guide for implementing Software Configuration Management

By Magesh M.

For additional information or assistance please contact Magesh at (+91)422-320-606. **www.thinkbn.com** • 127A, Trichy Road,
Coimbatore, TN 641045.

# Table of Contents

# Team Based Development for a Small Organization

## Software Configuration Management

## Introduction

Whatever be the size of a software organization the goal is and will always be a good reliable, bug free, good performing product complying with the requirements of the customers. A bigger organization achieves this by having separate department for configuration management other than the Quality Assurance team to take care that the product is well managed and controlled .A project should be controlled and managed at any point to ensure quality. However in smaller organizations having such a separate department will be bigger asks. The quality engineer also needs to play the role of a configuration manager. Under such situations how does one come up with a suitable configuration management system?  This article explains this and introduces the concept of team development and explains how a project can be kept under control and thus achieves a team development.

## What is Team Development?

### The Definition

*The term Software configuration management is a mechanism employed by tools that manage software application. Team development indicates the end result these tools along with the defect tracking tools should help us accomplish.*

A product cannot be completely managed and controlled by just controlling the software versions and configurations. What is needed is a way of managing the complexity of software projects while empowering team members to work effectively together.

In a nutshell, a team development solution should:

- Track changes to all project components
- Support parallel development (branches, differences, merges, distributed teams)
- Control the entire project and its evolution in time (releases and variants)
- Manage the approval of changes (promotion process)
- Manage builds and dependencies

## Software Configuration Management in Small Organizations

**What do we lack?**

*A small organization will be having in place a version control tool which will most probably be of the first generation i.e. a file based version management tool. It might also have a defect-tracking tool. The quality assurance group will be playing the roles of a build/installation engineer, configuration engineer, test engineer and a process engineer.*

They might not be having in place a process defined for configuration management and if at all it exists not base lined. They also will not have the metrics to establish the success of the process. To put in other words they lack historic data that is very essential to prove the effectiveness of a process.

**How to Improve**

The building blocks of functionality that should comprise team development solutions are

- Version Control
- Workspace and Release Management
- Build Management
- Process Management
- User Interfaces
- Architecture
- Integrations and Tool Interfaces

Although all these features may look reasonable and necessary, no single tool includes them all. However these shortcomings can be over come by making a brief study and survey of the environment and requirements of the organization in coming up with the products.

Given above is a pool of information that can be matched against the requirements to pick out which is relevant to us. For example having a first generation version tool might seem a setback to the configuration management at first but on analyzing the situation there might not be a requirement for a distributed development, the main issue addressed by second and third generation tools. The other functionalities like workspace insulation, build management can be effectively followed by defining proper procedures taking advantage of the small size of the organization.

Process should be defined to manage change request and metrics should be identified to measure the out come of the project. Process should also be established to collect data against these metrics.

## Managing Version Control

**The Codeline Policy**

*Define a codeline policy that specifies the use and check-ins of the codeline. A codeline can be seen as a canonical set of source files needed to produce the product. Examples of codelines can be development codelines, release codelines etc.*

A release codeline policy might specify that only bug fixes can be the check ins. In addition the developer can be provided with a private repository where the developer can check in the intermediate versions before checking them into the repository. By doing so the developer can be provided with a mechanism for check pointing changes at granularity they are comfortable with. The repository might not even be a separate project repository in the version control repository but can be as simple as a folder in the developer’s machine. At the same time periodic integration of the developer’s work with that of others is very important to maintain stability.

## Workspace Management

**Check in Periodically**

*As the user is provided with a separate repository for working on the files he is provided with a insulated workspace. Again as he checks in his files periodically the problem of workspace isolation is avoided.*

This is an important functionality provided by third generation version management tools. This functionality can be achieved using the first generation version management tool itself by following the procedure of checking in periodically.

## Build Management

**Avoid Big Bang**

*The only ingredients of the build should be the source codes and the tools to which they are input. Routine setup procedures can be automated to avoid human errors. A checklist can be maintained to check for the ingredients to completely build the software. Build outputs and logs should be achieved; this will be very useful when problems are encountered with new builds.*

The developers should be provided with a mechanism to build the system periodically to avoid unneeded surprises. Developers should be discouraged from maintaining long intervals between check-ins. This periodic build should be checked for interface compatibility (does it compile?) and testing (Does it still work?) Developers should be encouraged to build from files that are likely to be in the release (perhaps the newest code in the revision control system s trunk) to anticipate, and allow time to correct for, incompatibilities. The goal is to avoid a Big Bang.

## Other Functionalities

*Software configuration management does not manage the software product codes alone. It has to provide support and management for the process established to track the projects. In fact, equal priority should be given to the process artifacts and the subsequent changes to these artifacts need to be controlled.*

A baseline should be established for the artifacts and the changes that are done to these artifacts to tailor them to the organization needs should be tracked. Here a file-based repository fulfils the requirements and no sophisticated tool is necessary. Team development is not a stand-alone function. A good solution must work well with the other software development and maintenance tools that an organization employs. Hence policies should be established for maintaining the interface with other tools such as defect tracking.

## Conclusion

The process of implementing CM routines and procedures might seem straight forward but are very complex and time consuming. Continuous refining of the process and tailoring it to the organization is an iterative process.

Again identification of possible data to quantify improvement is difficult. Metrics needs to be defined from the existing information. The customer feedback should be taken as an important metric while tracking the effectiveness of the process in establishing the Configuration Management. Training needs to be provided at all levels and the role that an individual plays with respect to the system should be clearly specified. Thus, continuous tracking and refinement of the configuration management system can establish a good team development in small organization.

## References

I. “Team development Overview” by Alex Lobba

II. Configuration Management Patterns by Steve Berczuk

III. “High level best practices in Software Configuration Management” by Laura Wingerd and Chris toper Seiwald, Perforce Software, Inc

IV. Change Management for Software development by Continuous

V. White papers from SCM Labs, Inc
VI. The business benefits of software best practices – case studies
VII. Introduction of Software Configuration Management in very small organizations - ICONMAN
VIII. Introduction to Configuration Management – Gaining a competitive Edge-Datamat