Software Configuration Management

Configuration Management for Distributed Development

By Nina Rajkumar.

For additional information or assistance please contact Nina at (+91)422-320-606. **www.thinkbn.com** • 697A, Trichy Road, Coimbatore, TN 641045.

Table of Contents

Configuration Management for Distributed Development

Software Configuration Management

Introduction

Configuration Management (CM) is an "umbrella" activity that is applied throughout the Software Engineering process. Configuration management (CM) includes synchronizing and supporting developers in their common development and maintenance of a system.

In order to utilize skilled personnel despite geographical location, groups of developers are now working all over the world on the development of common systems, a situation called *distributed development.*

This paper presents different cases and architectures with respect to distributed development and their demands on Configuration Management Tools. This paper also presents the features of some of currently available CM tools that support Distributed Development.

Configuration Management

The Concept

Configuration Management (CM) is a discipline within software engineering with the aim to control and manage projects and to help developers synchronize their work with each other. This is obtained by defining methods and processes to obey, making plans to follow and by using CM-tools that help developers and project leaders with their daily work.

CM has, during the past years, been considered more and more important due to several reasons. One reason is the influence of the well-known SEI Capability Maturity Model (CMM), which has pointed out CM as an important ('key process') area to achieve level 2 on its 5 level scale. Another important reason is the fact that software is getting larger and more complex and needs the support from CM. Strategic Planning is a process to guide the members to envision the future and develop the necessary procedures and operations to achieve that future.

Distributed Development

A Scenario

A significant segment of today's software industry is moving toward a model of project organization that involves the use of multiple engineers at multiple sites working on a single software system or set of highly interdependent software systems. An example of this kind is that the trend that developers are getting more dispersed, although still working on the same system.

CM Tools were originally developed earlier under the assumption that the people as well as the files are situated at the same geographical location. When this assumption no longer is true it creates new demands and tools and processes needs to be re-evaluated.

Some aspects, such as file access, have been partly solved by supporting server replication of the files. Other aspects, such as the creation of a general picture and a context, and the communication between developers and groups, have remained manual and without direct tool support. When people working closely together are geographically dispersed, we need to consider how these additional aspects may be supported in the work method and in tool support.

Irrespective of reason many companies have found that the methods and tools used do not fully support their current situation, and that it will be even worse in the near future. Therefore they now intend to develop such support. We can also see a trend where tool vendors focus more on support for distributed development and for teams in general. Regardless of the Team size and geographical distribution, software development teams need a powerful software solution that allows them to work together efficiently and productively.

Cases of Distributed Development

The different cases that have been identified are:

- Distance working.
- Outsourcing
- Co-located groups
- Distributed groups

The above cases occur individually or in combinations. For instance there may be groups which are normally connected but which may occasionally be distributed.

Distance Working

This kind of distant work is brief work being performed elsewhere than the usual place of work. For example working from home.

When developers work at home (or elsewhere) on a more regular basis or for longer periods of time, a situation similar to that for distributed groups arises. A limited computer utility and a relatively slow means of communication with the world around is characteristic of distance working. Despite this, there is a desire to be able to start working quickly, as the total working time on each occasion is short (typically a few hours in the evening), which means that it must be possible to set up the working environment quickly.

Remote login to the place of work and the home computer is being used as a terminal. In this case of distributed development, the developers work should be in sync with the work done by other people on the same files. This completely relies on the CM tool used.

Outsourcing

Instead of developing everything by yourself or buying existing components, you can engage a third party to develop them for you. This is usually called outsourcing and gives, a greater control of the development of the component, albeit at a higher price.

1. Outsourcing is based on a close collaboration between the supplier and the purchaser.
2. Consequently it is often possible for the supplier to test the component in an environment similar to the target environment prior to delivery. The purchaser then usually provides the test environment.
3. The purchaser is ultimately responsible for the product and possible error/change management can be reflected in changed demands on the component towards the supplier.

4. As with any order, it must be clear what should be delivered, but in this case it is further complicated by the fact that the demands as well as the environment may change. In this case of distributed development, the purchaser must be able to integrate new versions of the component into the product, which itself may have developed since the latest release of the component.
5. The supplier should be able to manage the updating of the developments.

Co-located Groups

Developers at different affiliated companies usually belong to local groups or projects. The division of the work has already been determined at the structuring of the project/product to prevent too much dependency between the different groups. The product is divided up into sub-products, which can be developed by different project groups. This division makes it possible to do most of the development locally within the groups without the requirement of much communication with other groups.

Within the group and between groups in the same place, the situation is the same as with local development. Groups in different places normally only have access to the latest stable versions produced by the other groups. Due to the geographical distance, potential problems will inevitably be more difficult to solve. Therefore, updating and distribution between the groups requires more effort and administration, these may be considered as internal deliveries and therefore tend to come more infrequently.

Cooperation between the groups may be facilitated if the work is planned in phases of which every-one is aware. In this case, when the locations are permanent, each local group should be able to work within a complete development environment and with the possibility of testing and added to this change management of common components, such as interfaces, is of particular importance.

Distributed Groups

Distributed groups with members at different locations means that the members of the group are also distributed, i.e. that the people working in the same project, perhaps even in the same files, are geographically dispersed. The possibility of daily communication by formal as well as informal meetings is lost.

Projects working towards the same product usually use some common libraries or components. Changes in these are unusual, but sometimes inevitable. If group members at different places want to make changes simultaneously, they face a situation similar to that for the updating of interfaces where there are “connected groups” but in this case the problems apply to all files. The obvious example is

when people included in one group, have to travel around to other groups for various reasons.

Of course there is a desire to be able to continue working with the usual project, this will then be done as a distributed group. A similar situation arises when staff is moved to new projects but often need to be consulted on the old project.

From a CM perspective, it is important that the members of the group receive information about what the others in the group are doing, how the project is developing, its status, which changes have been done and by whom etc. It is important to support the division of files and concurrent, simultaneous changes. Solutions using "locking" and exclusive access to files work poorly, as it is difficult to solve situations where group members, located at different sites, must wait for each other.

Architecture

We have seen the different cases of distributed development. To meet the demands arising from these different situations, we can locate workstations and repository servers in these places in different architectures. In this case a geographic place equipped with repository/server is called a "site". Developers with workstations but without a repository/server are therefore not a site. Some CM tools for Distributed Development are based on these architectures.

The different architectures being discussed are:

- Remote login.
- Several sites by Master-Slave connections.
- Several sites with differing areas of responsibility.
- Several sites with equal servers

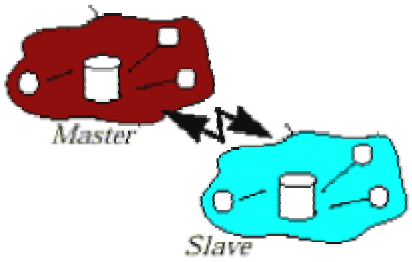
Remote Login

Simply put, everyone logs into and use a single server. Those situated at locations other than where the server is located, log into the server by remote login, telnet, or other similar protocols.

Technically a developer then works as if situated locally but is limited by a slower (and possibly a less reliable) connection, for instance over a modem or Internet. The same way a laptop can be remotely connected to the server. Some of the product's files are copied to be then worked on locally. The CM tool should support updating and synchronization of the files.

Several sites by Master-Slave connections

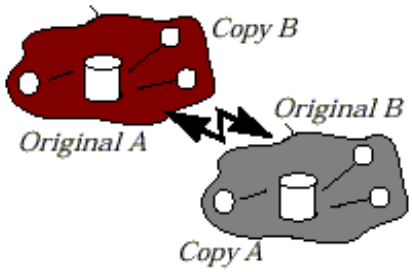
A version of a connected sub-system is copied from a master to another (slave) server where it is further developed. The CM tool supporting this architecture



eliminates complicated merge situations. A situation like this may occur with outsourcing for instance.

Several Sites with differing areas of responsibility

Different sites are responsible for different sub-systems. The division can be based on the responsibility for certain files. The variant concept must be the same for all of the files on the servers. For those parts that a site is not responsible for, the information can only be read.

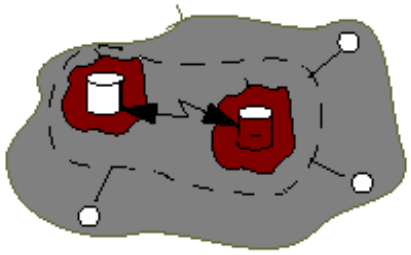


Synchronization is achieved by the changes in the original being transferred to the copy. It depends on the protocols supported by the CM tool. Synchronization is often done automatically and at close intervals. It should be noted that a site could have the original of one sub-system and at the same time have copies of others. This means that updating can occur in both directions between servers holding the

originals for different sub-systems. Compared to the master-slave architecture, this is a more permanent division and the synchronization is usually done automatically and therefore more frequently.

Several Sites with Equal Servers

This is an architecture where several equal servers are located at different sites. These are automatically synchronized at close intervals (hours, minutes, and seconds) and all of the servers have (with very little delay) the same information.



The result is that a developer can work at any site (towards the server at that site) without noticing a difference.

Configuration Management Tools

Rational ClearCase MultiSite®

www.rational.com

Rational Clear Case MultiSite® is a product option of Rational Clear Case. A component of Rational's integrated change management solution, it enables parallel development across geographically distributed teams.

Rational Clear Case MultiSite supports team members down the street or across the world, delivering automated, error-free replication of project databases and transparent access to all software artifacts.

The features and benefits are,

- Provides the safest, most reliable means to exchange multi-site Rational Clear Case repository-based information.
- Automatically resends information during network failures. Recovers repositories in the event of system failure.
- Scales easily to support projects regardless of size of team, location of members, or platform.

- Saves time and network resources by efficiently transmitting only incremental changes that appear in Rational Clear Case's project repositories.

PVCS Version Manager

www.merant.com

PVCS Version Manager organizes, manages and protects software assets, supporting effective software configuration management (SCM) across your entire enterprise.

The intuitive, graphical user interface, based on Windows® Explorer™ concepts, is easy to use, and encourages team members to apply SCM practices consistently and effectively. Your choice of tailored clients for Integrated Development Environments (IDEs) or Web development platforms extend Version Manager functionality to developers from within preferred development environments, or from Project Command Line Interface (PCLI). Application quality is improved and productivity soars as Version Manager automates common tasks, enables safe code reuse and protects against lost changes, overwrites and content errors. PVCS Version Manager Plus adds the power of PVCS VM Server™, enabling distributed teams and remote developers to work collaboratively via the Web, while sharing protected and centrally managed software archives.

Continuus CM Synergy DCM

www.continuus.com

CM Synergy DCM enables geographically distributed development teams to work together. Continuus Software's Distributed Change Management capabilities provide the industry's first comprehensive solution for managing development processes in decentralized and distributed team environments. Continuus CM Synergy with the DCM component supports the entire development process across multiple locations or across multiple developers working together at a single location.

For any enterprise software development team, regardless of its size or geographic distribution, Continuus CM Synergy provides the following benefits:

- Complete team development capability. Continuus CM Synergy is a task-based, workflow-oriented environment where logical changes flow easily between groups regardless of their location. It includes all the tools necessary for comparing and merging code created by different developers, and for the immediate sharing of the newly merged code among teams.
- Automatic notification of status changes. This provides development team members complete visibility to all changes in status relevant to their elements of a project regardless of their geographic location.
- Automation of predictable change management processes. This capability enables timesaving concurrent development through distributed teams.

- Development team effectiveness. Continuus CM Synergy promotes an independence that liberates developers from the mundane, repetitive, error-prone tasks associated with traditional software development processes.
- Flexibility for supporting any type of project. Continuus CM Synergy can manage any level of "granularity" or configuration in code being passed among development teams, ranging from objects and components to entire applications.
- Process flexibility. Teams can adapt to changes in the composition of the software organization, new processes and/or policies as well as temporary or emergency situations that may arise.
- Reduced administrative overhead. For administrators, the system simplifies and consolidates administrative tasks. Through an easy-to-use graphical interface for configuration and set-up, the entire change management send and receive process can be automated if desired, resulting in minimal administrative overhead.
- Reduced developer overhead. Distribution of software to multiple locations is a simple straightforward process performed by the change management administrator and can be fully automated. This means that the use and management of DCM is seamless to the developer.
- Heterogeneous platform support. Continuus CM Synergy allows for transfers of code between teams using a combination of UNIX and Windows NT servers and workstations.

Conclusion

The architectures discussed in this paper can serve as a guide when planning the introduction of distributed development, and as a basis for analyzing the consequences and limitations of different solutions. It will never be very easy to use the architecture until there is complete tool support. It is concluded that some CM tools of today support distributed groups extensively, but there is scope to develop CM Tools which enables a highly interactive distributed development.

References

Software Engineering, "*A Practitioners Approach", Roger.S.Pressman*

A Generic, Peer-to-Peer Repository for distributed Configuration Management, *Andre van der Hock, Dennis and Alexander L.Wolf.*

Configuration Management for Distributed Development, *Lund Institute of Technology – Practice and Needs*

Change Sets Revisited and Configuration Management of Complex Documents by *Stephen A. MacKay.*

Successfully managing the complexities of Today's Distributed development by *Continuus Software Corporation*