

SFA Modernization Partner
United States Department of Education
Student Financial Assistance



Configuration Management Standards Using Rational ClearCase

White Paper
June 22, 2001



Table of Contents

<i>Introduction</i>	<u>3</u>
<i>ClearCase Configuration at Aerospace (Standard Configuration)</i>	<u>4</u>
<i>Using Clear Case</i>	<u>5</u>
Terminology	<u>5</u>
ClearCase Views	<u>5</u>
Checking Files In and Out	<u>9</u>
Creating Directories in the VOB	<u>10</u>
Removing Files and Directories from the VOB	<u>10</u>
<i>Installing ClearCase on Development Workstations</i>	<u>11</u>
<i>Installing ClearCase on SUN servers at VDC</i>	<u>12</u>
<i>Recommendations</i>	<u>13</u>



Introduction

ClearCase is a comprehensive software configuration management system. It manages multiple variants of evolving software systems, tracks which versions were used in software builds, performs builds of individual programs or entire releases according to user-defined version specifications, and enforces site-specific development policies.

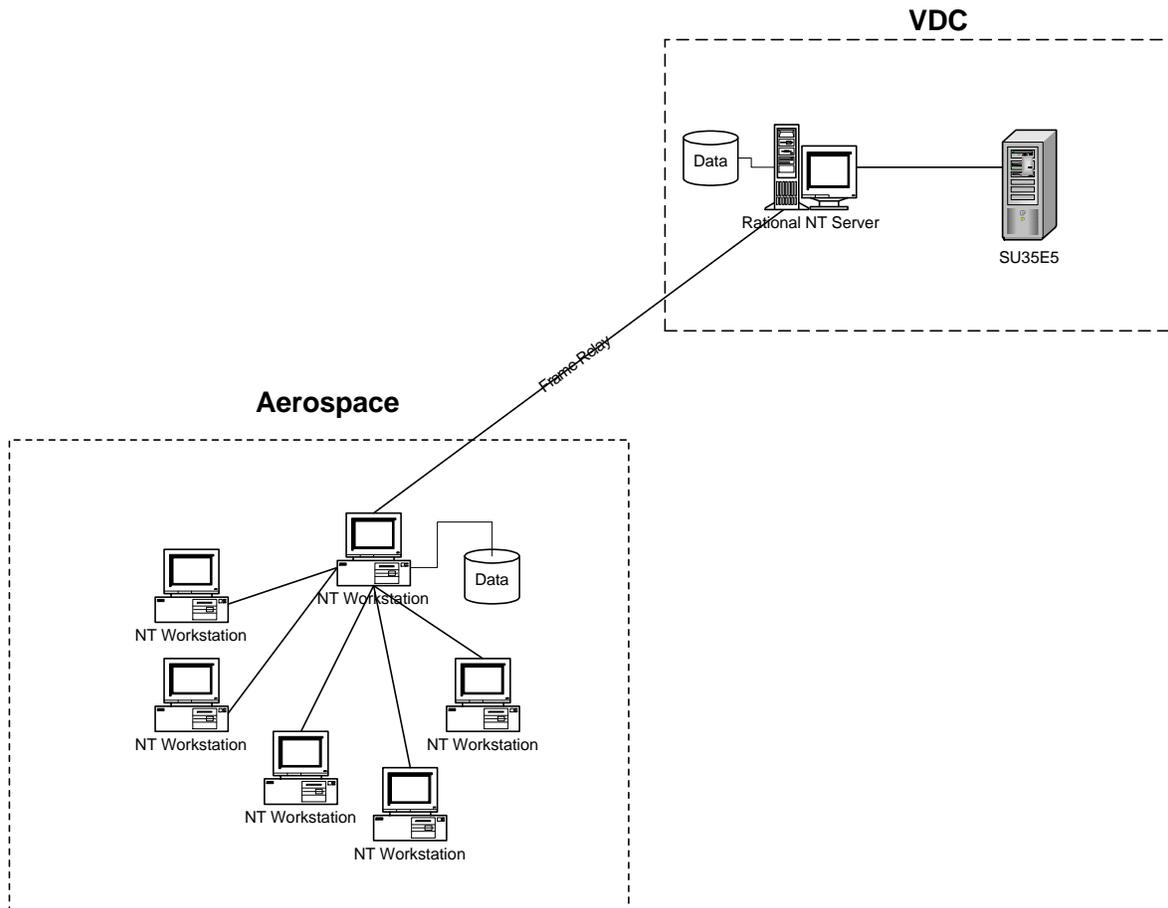
These capabilities enable ClearCase to address the critical requirements of organizations that produce and release software:

- **Effective development:** ClearCase enables users to work efficiently, allowing them to fine-tune the balance between sharing each other's work and isolating themselves from destabilizing changes. ClearCase automatically manages the sharing of both source files and the files produced by software builds.
- **Effective management:** ClearCase tracks the software build process, so that users can determine what was built, and how it was built. Further, ClearCase can instantly recreate the source base from which a software system was built, allowing it to be rebuilt, debugged, and updated ³/₄ all without interfering with other programming work.
- **Enforcement of development policies:** ClearCase enables project administrators to define development policies and procedures, and to automate their enforcement.



ClearCase Configuration at Aerospace (Standard Configuration)

Due to several technical issues with ClearCase be utilized through firewalls, a standard configuration management solution is needed. To resolve the issues, the architecture bypasses any dependencies on the firewall being opened to security risks. The resulting design is depicted in the diagram below. A local development environment has been built with a single workstation acting as the repository for ClearCase. All local workstations create development views from this repository. The local repository is replicated with the corresponding repository on the NT Server at the Virtual Data Center in Connecticut on a scheduled basis. This schedule is flexible within ClearCase. Once the VDC copy of the repository is up to date, a user must log onto the corresponding SUN server and manually update the repository there to reflect the latest changes in the code. As new projects enter the realm of development, this configuration can be utilized no matter where the development is occurring, as long as the location has a direct connection to the VDC via frame relay, T1, etc. With the introduction of ClearCase Multi-Site, development can occur from multiple locations on the same repository simultaneously. This tool is not needed with current development efforts but it could be useful for future projects. Below is a diagram of the current environment.





Using Clear Case

Configuration management tracks the evolution of a source file. Acting similar to that of a librarian, a configuration manager checks in and checks out files to developers, where they are free to perform any edits to the file. When files are checked in, the configuration manager tracks the changes made to the source file, the developer who performed the edits, the date/time of check-in, and a description of the changes made. By tracking the changes made to a source file, a developer is empowered with the ability to restore previous versions of that source file.

The ClearCase configuration management system will be responsible for all source code and objects. It uses a Versioned Object Base (VOB) virtual file system to track changes/histories to source files. ClearCase will be used to track all java source code and html files. Compiled code should not be versioned because it would cause a rapid increase of the size of the repository.

Terminology

- VOB - Versioned Object Base; the database on the ClearCase server which contains versioned files and relevant information.
- VIEW - The workspace that a developer sees when connected to a ClearCase server.
- ELEMENT - Any item that has been versioned in the VOB; typically, files and directories.
- VERSION - A specific instance of a versioned element.
- CONFIG SPEC - A set of predefined rules which determine which versions will appear in the view.
- VIEW SERVER - A process which communicates with the VOB to interpret the Config Spec, and displays the filtered database content into the view.
- VIEW PRIVATE - A file or directory which has not been introduced to the VOB. Such files cannot be seen by other users, unless they are using the same view as the user who created the view private object.

ClearCase Views

A view is a software development work environment that is similar to but greatly improves on a traditional "development sandbox". Each view can easily be configured to access just the right source data from the central repository:

- the up-to-date versions for development of the next major release
- the versions that went into the port of Release X.Y to hardware architecture Z
- the versions being used to fix bug #ABC in Release D.E

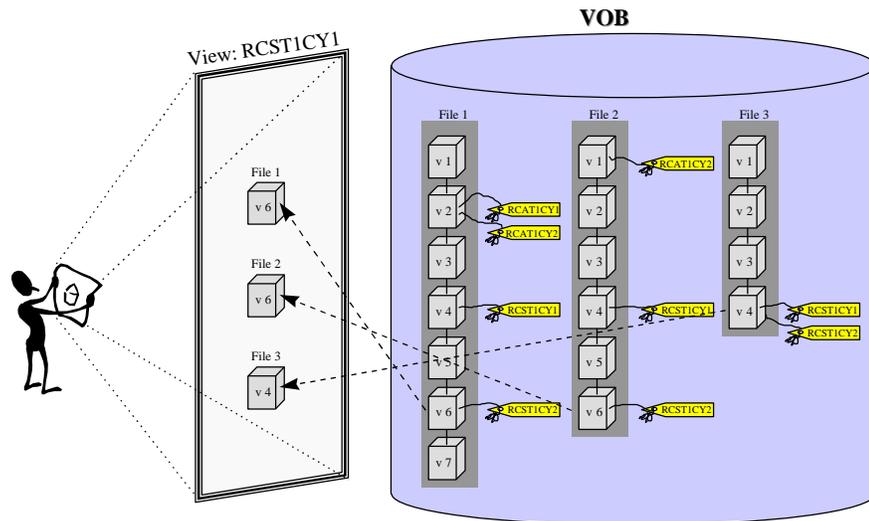
A view is an isolated "virtual workspace", which provides dynamic access to the entire data repository. The changes being made to a source file in a particular view are invisible to other views; software builds performed in a view do not disturb the work taking place in other views.



Working in views, ClearCase users access version-controlled data using standard pathnames and their accustomed commands and programs. The view accesses the appropriate data automatically and transparently.

A view's isolation does not render it inaccessible; a view can be accessed from any host in the local area network. A view can be shared by several users, working on a single host or on multiple hosts. One user might "peek" into another's view, just to see what changes are being made to a particular file.

Most of the time, you wish to see just one version of each of a VOB's files. (Often, it's the most recent version, but sometimes not.) Together, a consistent, matched set of versions constitute a particular configuration of the source tree. ClearCase includes a powerful and flexible tool for defining, viewing, and working with configurations -- the view. In essence, a view makes any VOB appear to be a standard directory tree, by selecting one version of each version-controlled object.



A view can be customized in many ways. For the most part, you will want to view the latest versions of all the elements (files) in the VOB. However, there will be those occasions when you want to view older versions of a particular file. In the picture above, the developer has chosen only to view those files with the 'RCST1CY2' label.

ClearCase supports the creation of any number of independent views. Each view is a flexible, resource-conservative workspace that combines these services:

- Access to permanent, shared storage: A view accesses the correct set of source versions for the task at hand (bugfix, port to a new architecture, new development, and so on). It automatically selects a particular version of each ClearCase element, according to user-specified rules. Together, a "matched set" of versions constitutes a particular configuration of the central data repository.
- Private storage: A view provides an isolated workspace with private storage, enabling individual users or small groups to work independently of each other.



A view can be completely private to an individual user, or shared among users. A view can be accessed on a single host, or from any host in a local area network. Views are compatible with all software development tools, not just ClearCase's own programs. Users working in views can use system-supplied programs, home-grown batch files, and third-party tools to process ClearCase data.

A view has some of the characteristics of a "snapshot" of a source tree that you've copied from some central repository to your own machine. Most importantly, it provides isolation: developers working in different views can modify files (perhaps even the same files) and rebuild software, without disturbing others' work.

It's useful to think of the view as being above the file system instead of within it, an omnipresent lens through which you can see all the available ClearCase data (that is, all the active VOBs) on your host.

Creating Views on NT Development Workstations

Open the ClearCase HomeBase under `Start/Programs/Rational ClearCase/`. You must be logged into the domain as yourself before executing this step.

- Click on the View tab. Then click create view button. Based on the user preferences, you can either create a dynamic view or a snapshot view.
 - Dynamic views are updated as other users check items out.
 - Snapshot views are local copies and can be controlled, but must be manually updated to remain in synch with the main repository.
- When it asks for a drive letter, use can use any available drives.
- Mount 'eai' (this will change based on the project).
- Please note that your view tag must be unique across the project, so personalize it with your initials and a short word describing the significance of this view. For example, JDP_view.

The view storage is located on your workstation, in a subdirectory of `M:/ccstg/VIEWS`. This directory must exist, and must be shared as VIEWS. Note that if the directory does not exist, the login script will create the directory and the share. If the share is removed, however, your views WILL NOT WORK.

Placing the view storage on the developer's local hard drive increases speed and reduces the chance of invoking excess processes on the server. However, this also means that hard drive space may be a factor.

Updating Views on Unix

To update the view on the SUN machine at the VDC, you must first have a logon for that machine. After logging on, you should only have to type `"/opt/clearcase/bin/cleartool updateview"` to update the code on the server. This will always have to be done manually due to the setup. Once the code is updated, it should be recompiled because compiled code will not be versioned by ClearCase. This is done to reduce the size of the repository. Once the code is compiled testing can begin. New views will not need to be created once the original view is completed.



Customizing your Views

Each view has a user-defined config spec (short for "configuration specification"), a set of rules for selecting versions of elements. Simply put, your configuration specification is stored in a text file. This file contains a list of sequential rules that defines your configuration specification. In UNIX, this file can be found in:

```
/users/<user_name>/<view_name>/config_spec
```

In NT, the file can be found in:

```
\views\<view_name>\config_spec
```

The default config_spec for the XXX environment is as follows:

```
element * CHECKEDOUT
element * /main/LATEST
```

This instructs the ClearCase view server to:

1. View all checked out files first
2. If the file is not checked out, then view the latest revision of the file.

Following are typical config spec rules, along with English translations:

```
element * ...\main\LATEST
```

For all file and directory elements, use the most recent version on the main branch.

```
element -eltype java_file *.java RCAT1CY1
```

For all elements of element type java_file whose file name suffix is .java, use the version labeled RCAT1CY1

Each view also has an associated server process, its view_server. Using the config spec rules, the view_server automatically resolves (converts) each reference to an element into a reference to a particular version of the element. The following steps summarize the view_server's procedure for resolving element names to versions:

1. User-level software (for example, an invocation of the java compiler) references a pathname. The ClearCase MVFS, which processes all pathnames within VOBs, passes the pathname on to the appropriate view_server process.
2. The view_server attempts to locate a version of the element that matches the first rule in the config spec. If this fails, it proceeds to the next rule and, if necessary, to succeeding rules until it locates a matching version.
3. When it finds a matching version, the view_server "selects" it and has the MVFS pass a handle to that version back to the user-level software.



This rule-based scheme for defining source configurations provides power and flexibility, supporting both "broad-brush" and "fine-tuned" approaches. The entire source environment can be configured with just a few rules, applied to all file and directory elements. When required, fine tuning can be implemented at the individual file or directory level.

For example, a user might discover that someone has created a new version of a critical header file (say, base.h@@\main\17) a change that causes the user's compilations to fail. A config spec rule can "roll back" the critical file (in that user's view) to a known "good" version:

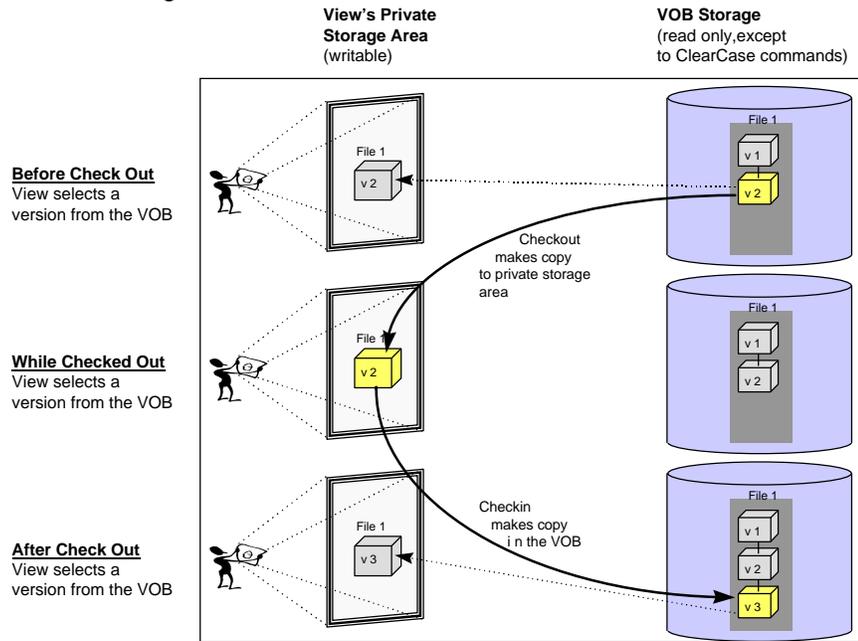
```
element base.h \main\16
... Or ...
element base.h \main\LATEST -time 11:30
```

The second alternative above illustrates use of a time rule, which can "roll back" one, many, or all source files to a known "good" state. This is very useful when an application compiles correctly before lunch break, but not after.

If no version of an element matches any of the config spec's rules, the element is suppressed from the view. This feature can be used to configure views that are restricted to accessing a particular subset of sources.

Checking Files In and Out

A user, working in a view, enters a checkout command to make a source file editable.



This seems to change a file element in the data repository from read-only to read-write. In reality, ClearCase copies the read-only repository version to a writable file in the view's private storage area.



This writable file, the checked-out version, appears in the view at the same pathname as the file element; the view accesses this editable, checked-out version until the user enters a checkin command, which updates the repository and deletes the view-private file.

Following are the checkin/checkout procedures using the ClearCase command line interface:

- To check a file out, right click on the file in ClearCase and click on checkout (this will also work in Windows NT Explorer). When prompted for a description, enter information about the modifications you are about to make. Other tools (VAJ) may have an integration with ClearCase to check a document out through an external controller.
- To check a file in, right click on the file in ClearCase and click on checkin (this will also work in Windows NT Explorer). The default description will be the comment entered when the file was checked out. Enter a description of what was changed. This may also be integrated with an external tool.
- To cancel a checkout and return the file to its pre-checkout condition, right click and choose undo checkout. This will create a .keep file if you use the defaults.
- When a new file is created in your directory, right click on it and choose add to source to allow ClearCase to control the versions.

NOTE: A user must be logged into the domain as their own ID prior to checking out.

Creating Directories in the VOB

The directory to be created must not exist prior to issuing the following command in cleartool. If a file or directory of the same name does exist, rename the original before creating the versioned directory.

```
mkvdir <dirname>
```

Removing Files and Directories from the VOB

The following scripts follow a specific procedure which allows a user to remove files and directories from the VOB. These scripts ensure that builds and releases which depend on previous versions will still be able to find the appropriate source files and directory structures.

```
rmvfile <filename>  
rmvdir <dirname>
```



Installing ClearCase on Development Workstations

When a new development workstation is added to the development environment several steps must be followed in order to enable it to work with ClearCase. Listed below are the steps to follow when a new user and workstation become available to perform development from Aerospace.

1. Set up an account for the new user on the primary domain controller and add them to clearcase_aerospace users
2. Set up account for workstation on the domain
3. Login locally as "administrator". Connect to EAIPDC under Network Neighborhood as the new user.
4. Connect to EAIPDC shared drive for clearcase. Open ClearCase Release Area.
5. Double click on setup.exe to initiate install
 - 5.1. All options should be default answers
 - 5.2. Make sure application is installed on D: drive (or whatever extra drive is named
 - 5.3. NOTE: Workstations must have "server" service running to allow for sharing. ClearCase requires sharing to be enabled.
6. After reboot, choose "yes" to begin storage configuration. Click "Next" to choose location of shared storage. Select "d" drive (or whatever named extra drive). Click "Ok".



Installing ClearCase on SUN servers at VDC

Listed below are the procedures to use when installing ClearCase on the UNIX machines located at the VDC.

1. Create /opt/ccrelease & /opt/clearcase directories on destination server
2. Log onto SU35E2 (4.20.15.132) with any ID that is available
3. Cd to /ftp-site/clearcase
4. ftp to destination server
5. cd to /opt/ccrelease
6. type bin
7. put ccase.tar
8. Log onto destination server now
9. Tar -xf ccase.tar
10. Run ./install_release from /opt/ccrelease/v4.1/sun5/install (answers to options listed below)
 - 10.1. 1 local
 - 10.2. 1 standar
 - 10.3. directory = /opt/clearcase (install must be in empty directory)
 - 10.4. [return]
 - 10.5. 8 full function
 - 10.6. f finish
 - 10.7. [return]
 - 10.8. [return]
 - 10.9. [return]
 - 10.10. no
 - 10.11. yes to continue installation
11. Look at .profile for migrat on E5 to put /opt/clearcase/bin in path and use ct as shortcut to cleartool
12. After updating profile run ". ./profile, then type set to reinitialize the profile (or logout & log back in)
13. Create a view "mkview -snapshot -tag <name of view>", name of view should be name of repository (need to verify this!)
14. Do a cleartool edcs. This will bring up vi session and you must put in "load \<name of VOB>". Save and exit.



Recommendations

As new projects are approved and development teams are formed, there will be an urgent need for standard version control as well as code repositories. Rational ClearCase product is a good tool that will satisfy all of SFA Modernization Partner's requirements. With the standard configuration described in this document, development teams will be able to make a single request to have a repository set up and development can begin. This could potentially save teams a considerable amount of time in getting a version control system established and getting started in development.

This document could provide a guide for all teams in using ClearCase for their purposes. As long as licenses are available from the Rational server at the VDC, teams can establish repositories locally, on the NT server, and on the SUN application servers. ClearCase is very flexible and compatible with tools like Visual Age for Java. Using ClearCase is quite intuitive to any developer. This will also allow for continuity across the Modernization Partner. As a result, Rational's ClearCase is a good choice for a configuration management tool within the SFA Mod Partner.