



IBM Software Group | Rational software

# IBM Rational ClearCase

*The Basics - Developing with ClearCase using UCM*

Rational software



@business on demand.

© 2009 IBM Corporation

Updated August 12, 2009

This module will cover an overview of ClearCase® terminology with respect to UCM.

## Introduction

- This module will cover an overview of ClearCase terminology with respect to UCM
- Prerequisites: Review of the ClearCase and ClearQuest® Tutorial for Developers of LSI Logic.

This module is associated with the ClearCase and ClearQuest Tutorial for Developers for LSI Corp. and is intended to help aid in the definition and description of various new terminology associated with ClearCase and ClearQuest.

## Module objectives

- The following topics are covered in this module:
  - ▶ What are VOBs and views?
  - ▶ What are elements, branches, versions, and version trees?
  - ▶ What is UCM?
  - ▶ What are components, projects, streams, baselines, and activities?
  - ▶ What are the deliver and rebase mechanisms?
- When you complete this module, you will be able to:
  - ▶ Successfully identify the terms that comprise of ClearCase with UCM development and their purpose.

This module covers basic and advanced ClearCase topics. It will cover base ClearCase concepts; vobs and views and what an element is. You will look at the structure of an element within the vob. You will also be introduced to the advanced concept of Unified Change Management or UCM for short. You will look at the various parts of UCM and how they are used. By the end of this module, you will be familiar with both base and UCM ClearCase concepts in addition to the terminology.

## What are VOBs and views?

- **VOBs**
  - ▶ A Version Object Base that is used to store source controlled data within ClearCase.
- **Views**
  - ▶ A mechanism that allows users to access and filter through the data within a VOB.



Here are the definitions of a VOB and view. They are very fundamental concepts in ClearCase. Understanding vobs and views will help you progress in using ClearCase.

## VOB

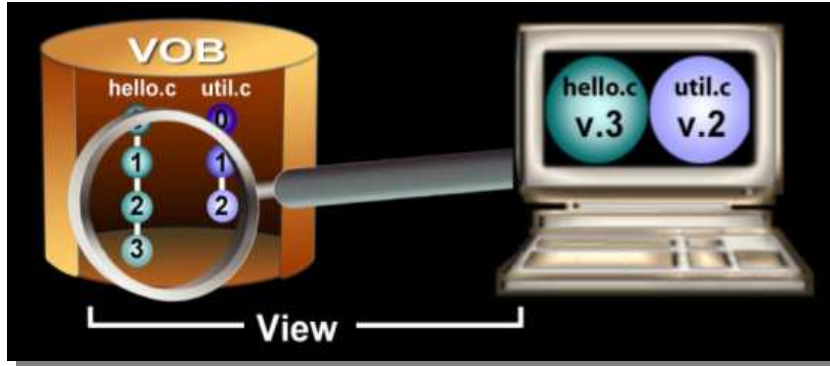
- A permanent, read-only data repository that:
  - Stores files, directories, and metadata
  - Stores version-controlled data
  - Displays its contents as files in a file system
  - Stores anything that can be represented as a file or directory
  - Can be replicated in two or more sites



A **VOB** or Versioned Object Base is a permanent, read-only repository that stores: versions of file elements, directory elements, derived objects (or, executables), version-controlled data, meta data associated with these objects, and virtually anything that can be represented as a file or directory.

## View

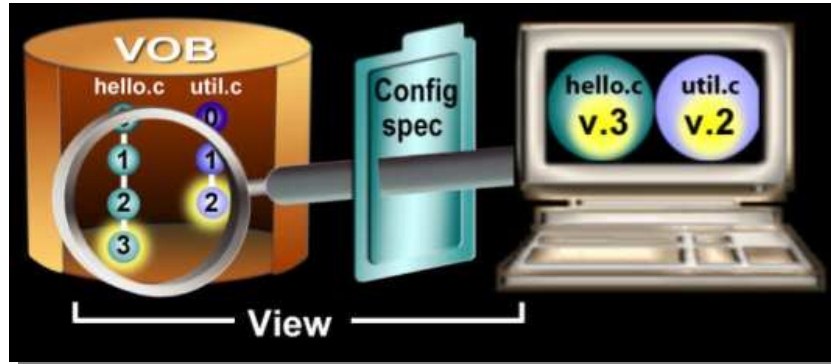
- A ClearCase mechanism that allows users access to versions of elements in VOBs
- An isolated workspace for a user or a group
- Enables users to work in parallel



A **view** is a ClearCase object that provides a work area for one or more users to edit and create versions, compile them into object modules, format them into documents, and so on. Users in different views can work on the same files without interfering with each other. To access elements in a VOB you have to use a view.

## View and configuration specification

- For each view, a configuration specification is a set of ordered rules used to select [at most] one version of each element
- A configuration specification determines which versions of an element are visible in the view



A set of rules called a **configuration specification** (or **config spec for short**), determines which files are visible in a view. A view's configuration specification selects one version from the element's version tree and displays it in the view. You can determine what elements are visible to the view by adjusting the configuration specification in ClearCase. In this example, only the versions highlighted above are visible in the view.

## Section

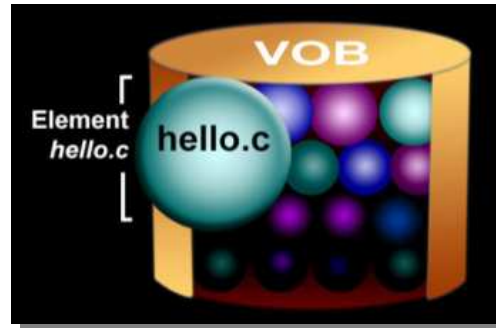
# ***What are elements, branches, versions, and version trees?***

This next section covers the concepts and terms elements, branches, versions and version trees. You will also see their structure within the vob.



## Elements

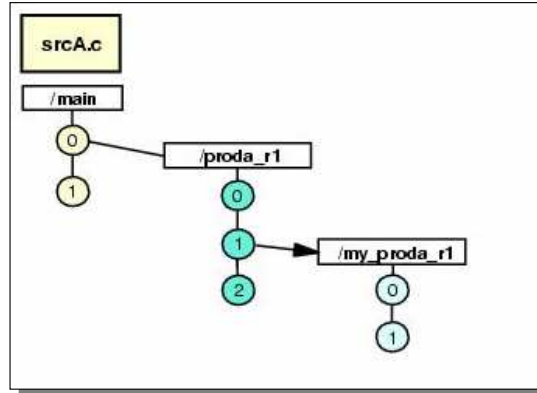
- A file or a directory, under source control, stored in a ClearCase VOB
- Can be any object that can be stored in a native file system, including:
  - Source files
  - Directories
  - Binary files
  - Object libraries
  - Documents



**Elements** can be either files or directories, under source control that are stored in a ClearCase VOB. A file element can be any object that can be stored in a native file system as shown here.

## Branches

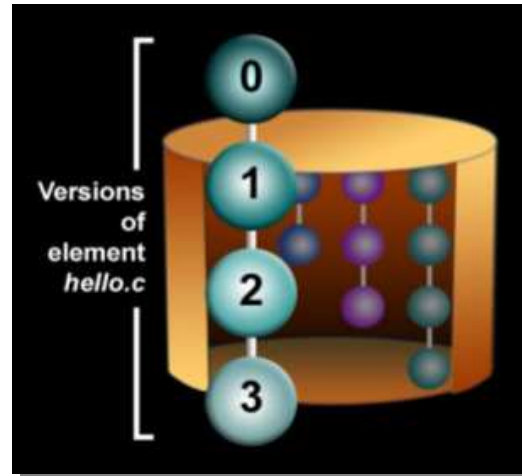
- A **branch** is an object that specifies a sequence of versions of an element
- Each element has a main branch
- Branching is done in order to provide work areas for developers



The concept of **branches** and branching is an important topic in ClearCase. A branch is an object that specifies a sequence of versions of an element. Branching is done in order to provide work areas for development. It is not uncommon to have dozens of branches in an element's version tree. Note that the ClearCase naming convention for branches is to use lower case alphabetic names. Because branching is done on a very low level, it also becomes very important to find a branching strategy that works from all levels of perspectives and abstractions. Some general rules that apply for branching to help with maintenance. First, the main branch should not be used for development work, but should be reserved for major milestones or released versions. Second, all major development is done on branches.

## Versions

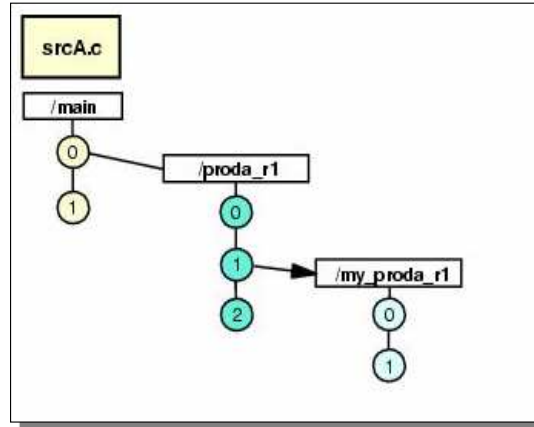
- An element consists of a set of versions, organized into a version tree
- Each version represents one revision of a file under source control
- Versions are displayed in a workspace or view



A **version** denotes a particular revision of an element. An element consists of a set of versions, organized into a version tree. Each time you revise and check in a file or directory, ClearCase creates a new version of it. The versions of an element are organized into a version tree structure as shown above – with numbered versions 0 through 3 of the element called “hello.c”.

## Version tree

- A graphical layout of the versions, branches, and attributes within an element.



The ClearCase version tree displays a graphical view of the version history for a ClearCase element. The different versions of an element in a VOB can be organized into a version tree. A version tree has branches, and each branch represents an independent line of development. Changes on one branch do not affect other branches until you merge. The initial version on the main branch always contains no data and is empty. Version 0 on any sub-branch always contains the same data as the version from which the sub-branch stems from.

## What is UCM?

- Unified Change Management
- Component-based configuration management system
- Covers version control, configuration control, and process management areas of SCM (software configuration management)
- Automatically associates an activity with its change set
- Easily identify activities included in each build and baseline



On this slide the IBM Rational® ClearCase Unified Change Management is introduced. UCM is a component-based configuration management system. When you specify the component architecture for a UCM project, it is useful to have a sense of how the number of components in the project affects the performance of the commands typically used by developers, project managers, and release engineers. ClearCase UCM covers the *version control*, *configuration control*, and *process management* areas of the SCM domain. UCM raises the level of abstraction to manage changes in terms of activities, rather than manually tracking individual files. UCM automatically associates an activity with its change set, which encapsulates all project artifact versions used to implement the activity. This enables you to easily identify activities included in each build and baseline.

## Section

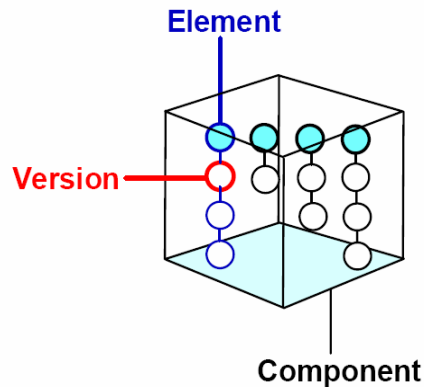
***What are components, projects, streams, baselines, activities, deliver and rebase?***

This section covers what components, projects, streams, baselines, activities, deliver and rebase are.

## UCM terms and concepts

### Component

- A group of file and directory elements
- Constitutes parts of a project
- Organizes elements into well-defined entities
- Two types of components used for UCM development:
  - ▶ Rooted component
  - ▶ Rootless component



A **component** is a group of files and directory elements (this might be a library, DLL, JAR, an executable, or other assets) that are released as a unit and are related by being located in a specific directory tree. Components provide separation of concern and organize elements into well-defined entities. A VOB can host one or more components, but a component without any elements does not have to be in a VOB. Similarly, components constitute parts of a project, and projects often share components.

There are two types of components in UCM: a rooted component and a rootless component. A rooted component is a UCM object linked to a VOB root or directory within a VOB. A rootless component is a UCM object containing no root directory (This type of component is used for composite baselines).

## UCM terms and concepts

### Project

- UCM object used to administer policy and design of UCM workflow.
- A Project contains configuration information for components, activities, policies, and so on
- One shared work area, many private work areas (for example, one for each developer)

### PVOB

- A PVOB is a VOB used to house all metadata for projects, streams, components activities, baselines in UCM environments.
- This VOB root folder is visible in the Project Explorer

For the first-time user of ClearCase UCM, the terminology can sometimes be a bit overwhelming but do not be intimidated by it. For every tool there is a learning curve, some steeper than others. The first step is to start with terminology and jargon used with UCM.

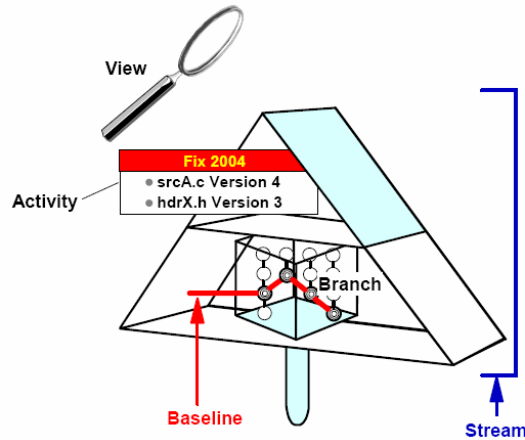
A UCM *project* is a logical unit that is mapped to the development structure of an application or system. A project contains the configuration information (for example, components, activities, policies) needed to manage and track the work on a specific product of a development effort. This can be an auction Web site or an order fulfillment process for an e-business. A basic UCM project in ClearCase consists of one shared work area and many private work areas. A Project VOB, or *PVOB*, is a VOB used to house all metadata for projects, streams, components activities, baselines in UCM environments. This VOB root folder is visible in the project explorer.



## UCM terms and concepts

### Stream

- UCM object that links to a Base ClearCase Branch Type.
- Maintains a list of activities and baselines
- This object lives in the PVOB and is hyperlinked to branch types existing in component VOBs.
- In UCM, streams are layered over branches.

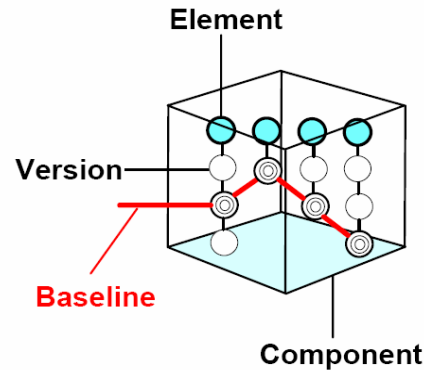


A **stream** is a ClearCase object that links base ClearCase branch types. Streams maintain a list of activities and baselines and determines which versions of elements appear in your view. This object lives in the PVOB and is hyperlinked to branch types existing in Component VOBs. In UCM, streams are layered over branches, so that you do not have to manipulate the branches directly. Note: This figure shows the ClearCase convention of representing streams.

## UCM terms and concepts

### Baseline

- Identifies one version of each element in a component that represents the integrated or merged work of team members
- Represents a version of a component at a particular stage in project development
- Baselines essentially allow projects to view and develop different versions of components.
- Baselines are component-specific



A **baseline** identifies one version of each element in a component that represents the integrated or merged work of team members. It represents a version of a component at a particular stage in project development, such as the first design, a beta release, or a final product release. Throughout the project cycle, the project manager creates and recommends baselines and changes their attributes to reflect project milestones. A baseline is the means of communication between team members, allowing them to share new changes developed in the development streams. For example, A UCM object linked to a ClearCase label tracks the development of a component.

When developers join the project, they populate their work areas with the versions of directory and file elements represented by the project's recommended baselines. Alternatively, developers can join the project at a feature-specific development stream level, in which case they populate their work areas with the development stream's recommended baselines. This practice ensures that all members of the project team start with the same set of files.

## UCM terms and concepts

### Activities

- A UCM object allowing users to record and track development on specific streams.
- UCM requires the use of this object when checking out or checking in on any stream.
- Activities are stream-specific; a view must be set to an activity before making modifications in a UCM environment.

Fix 2004
Creator: leif
<ul style="list-style-type: none"><li>• srcA.c Version 4</li><li>• hdrX.h Version 3</li></ul>

An **activity** is an object that records the set of files (*change set*) that a developer creates or modifies to complete and deliver a development task, such as a bug fix as shown here. UCM requires the use of this object when checking out or in on any stream. Keep in mind that activities are stream-specific; that is, a view must be set to an activity before making modifications in a UCM environment. Examples of other activities include: an update to a help file or the addition of a menu item to a GUI component. Note the activity title typically indicates the cause of the change (or is a link to ClearQuest).

## UCM terms and concepts (continued)

### Deliver

- Allowing work from one stream to be merged to a target stream.
- Commits changes to the shared project stream through the delivery of specific activities.
- Delivers must occur within a view context like that of a Base ClearCase Merge.

### Rebase

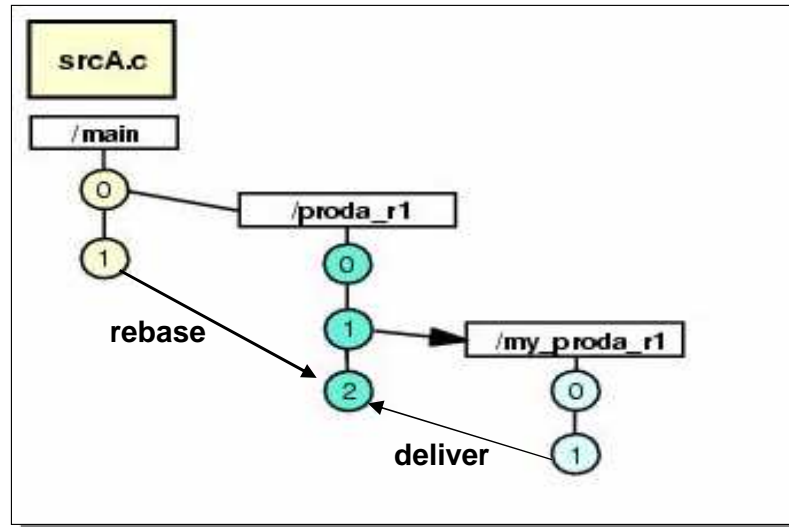
- Responsible for advancing streams to a baseline and regenerating the streams configuration specification
- Synchronizes working environment with the latest-best integrated configuration.



A **deliver** function is a UCM function allowing work from one stream to be merged to another or to a default target stream. Deliver commits changes to the shared project stream through the delivery of specific activities. Note that delivers must occur within a view context like that of a Base ClearCase Merge. A **rebase** function is a UCM function responsible for advancing streams to a baseline and regenerating the streams configuration specification. Rebase synchronizes working environment with the latest-best integrated configuration. Note that rebase, at times, require merging which must be done in a view context.

By performing a rebase before delivery, you reduce surprises at delivery time, which improves the stability of the integration environment.

## UCM terms and concepts (continued)



In simplest terms, a deliver is the mechanism for a developer to push their changes to the parent or integration stream from their own private or shared development stream. A rebase is essentially the opposite of a deliver and is the mechanism for a developer to pull changes from the parent or integration stream into their own private or shared development stream.

## Additional resources and references

- **Additional resources on ibm.com**

- ▶ Software Information Center

<https://publib.boulder.ibm.com/infocenter/cchelp/v7r1m0/index.jsp>

- ▶ IBM Education Assistant Modules

[http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp?topic=/com.ibm.iea.rcc/cc/RCCv70\\_Topic.html](http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp?topic=/com.ibm.iea.rcc/cc/RCCv70_Topic.html)



Additional resources can be found on ibm.com.

## Feedback

### Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

[mailto:iea@us.ibm.com?subject=Feedback\\_about\\_ClearCase\\_Terminology\\_UCM.ppt](mailto:iea@us.ibm.com?subject=Feedback_about_ClearCase_Terminology_UCM.ppt)

This module is also available in PDF format at: [../ClearCase\\_Terminology\\_UCM.pdf](http://../ClearCase_Terminology_UCM.pdf)



You can help improve the quality of IBM Education Assistant content by providing feedback.



## Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:  
ClearCase ClearQuest ibm.com Rational

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, Other Countries, or both.

Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.