

IBM

IBM Software Group | Rational software

IBM® Rational® ClearCase® Unified Change Management (UCM)

Module 1: Introduction to UCM

Rational software

@business on demand.

© 2007 IBM Corporation
Updated July 12, 2013

The slide features a blue background with a circular pattern of dots. The IBM logo is in the top right. The title and subtitle are centered in the white area. The Rational software logo is in a blue box. A horizontal bar with various icons is below the title. The @business on demand. logo is in the bottom right. Copyright and update information are at the very bottom right.

This presentation will cover Unified Change Management, or UCM.

Course objectives

- The following topics are covered in this course:
 - What is UCM?
 - UCM terms and concepts
 - Developing in UCM
 - UCM projects
 - Project VOB
 - Project structures
 - Project folders
 - UCM internals
 - Project internals
 - Stream internals

This Introduction to UCM course will begin with an introduction to UCM, then cover UCM terms and concepts, followed by Developing in UCM. The next module covers UCM projects including the project VOB, project structures, and project folders. The last module covers UCM internals including project internals and stream internals.

What is UCM?

- 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

IBM Rational ClearCase Unified Change Management 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.

UCM terms and concepts (1 of 7)

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)

vob

- Versioned object base

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.

Project

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, such as 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.

Objects under version control in ClearCase are stored with their histories in repositories called VOBs. VOB stands for versioned object base.

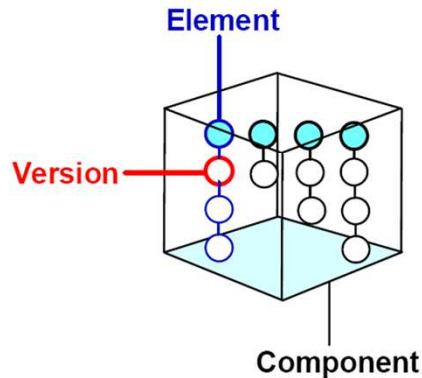
pvob

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.

UCM terms and concepts (2 of 7)

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



Component

A **component** is a group of files and directory elements (this might be a library, DLL, JAR, an executable, or any set of 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.

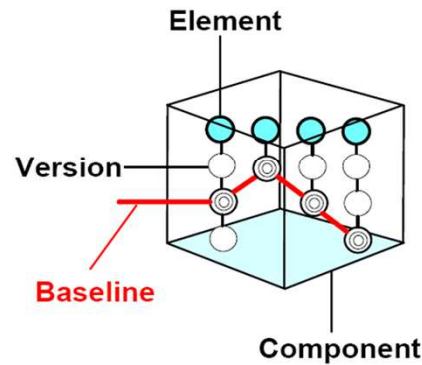
A Rooted Component is a UCM object linked to a Vob Root or Directory within a Vob.

A Rootless Component is a UCM object containing no Root Directory. (This type of Component is used for Composite Baselines)

UCM terms and concepts (3 of 7)

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.



Baselines

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 (4 of 7)

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.

Deliver

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.

Rebase

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 rebases, 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 (5 of 7)

Activity

- 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">• srcA.c Version 4• hdrX.h Version 3

Activity

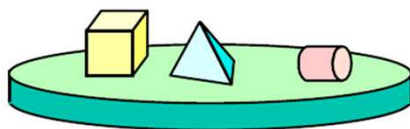
An **activity** is an object that records the set of files (*change set*) that a developer creates or modifies to complete and deliver a development task, such as a bug fix as shown here. Second, UCM requires the use of this object when checking out/in on any stream. Third, keep in mind that activities 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).

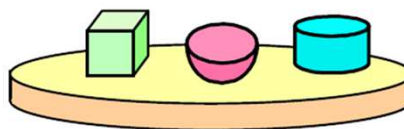
UCM terms and concepts (6 of 7)

Work Area

- Development area associated with a change.
- In base ClearCase, a view is a directory tree that shows a single version of each file in your project, and the view is your *work area*.
- In UCM however, a *work area* consists of a view and a *stream*.



My work area



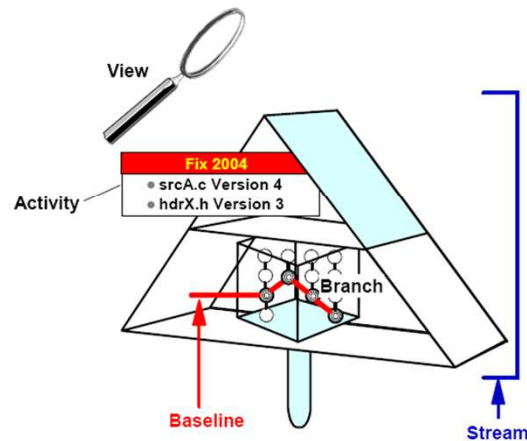
Your work area

A **work area** is a development area associated with a change. In base ClearCase, a view is a directory tree that shows a single version of each file in your project, and the view is your *work area*. In UCM however, a *work area* consists of a view and a *stream*.

UCM terms and concepts (7 of 7)

Stream

- UMC 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.



Stream

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 - Project explorer

The screenshot shows the 'Exploring ClearCase Projects' window. The left pane displays a tree view with the following structure:

- pvob
 - Components
 - Project
 - Project_Integrationm_Stream
 - Project_dev_Stream

The right pane shows a table with the following content:

Name
Activity1

Callout boxes provide definitions for the following terms:

- pvob**: The Pvob root folder containing all project information. This is the physical repository for UCM metadata.
- Components**: The component folder organizes and contains all components created in this Pvob.
- Project**: A specific project configured for development.
- Project_dev_Stream**: A Development Stream based on the Projects Integration.
- Activity1**: An activity available to views on the Project_dev_Stream to make modifications to data in the available component root directories.
- Project_Integration_Stream**: The Projects Integration Stream. This stream will make components available for development in the Project.

To help you put all of the terms together, this is an example of a ClearCase Project Explorer window highlighting the concepts and terms covered in the prior slides.

The **ClearCase Project Explorer** is the GUI interface to the UCM metadata. Project Explorer looks the same on UNIX and Windows, though Windows provides some additional functionality (a component browser and some drag-and-drop features).

The callout boxes shown here for pvob, components, project, project dev stream, activities, and project integration stream, provides additional detail about how each component fits into the project explorer window.

(leave a pause for 10 or more seconds)

Developing in UCM (1 of 4)

- Variables that factor into UCM design decisions:
 - **Components** - Decomposing an application into configuration objects
 - **Projects** - Organizing the development and release efforts
 - **Streams** - Organizing the workflow within a project
- Involve your product architect
 - Get your product architect involved up front to help define the configuration components for your applications

UCM design overview

Now that you understand the basic UCM objects, you probably have more questions, like: What is the best way to put the objects together into a usage model? How many components should I use and at what level should they be defined? Or how many projects should I have?

Well, the answer is: It depends.

This section will break down the variables that factor into UCM design decisions by taking a closer look at the objects that most determine your usage model: Components, projects, and streams.

Components, projects, and streams define the structure of the configuration and how, at a high level, the team works together during development.

Make sure to get your product architect involved up front to help define the configuration components for your applications!

Developing in UCM (2 of 4)

1. Developing in UCM starts with the creation of a project VOB
 - Use the `-ucmproject` flag in the `mkvob` command
 - Create a project under the root folder of the Pvb
 - The project must contain streams
2. Integration streams
 - Create an integration stream
 - Contains the configuration of the project
 - A streams configuration is a list of specified baselines for components
 - The specified components can be made modifiable at the project level
3. Project policies
 - Make modifiable specific components
 - Modification of component is controlled by project policies

Developing in UCM starts with the creation of a project VOB (or pVOB). Creating a Pvb requires the use of the dash `ucmproject` flag in the `mkvob` command. You then need to create a Project beneath the Root Folder of the Pvb. Note that the project must contain streams in order for development to be possible

Next, you must create an integration stream. The Integration Stream will contain the Configuration of the Project. A Streams Configuration is a list of specified Baselines for components. This will make the components available to the project. Once the Integration Stream is configured, the specified components can be made modifiable at the Project level. Modification of Component is controlled by Project Policies.

Developing in UCM (3 of 4)

4. Development streams

- How many development streams (child streams) will you need?

5. Associated views

- Views will need to be associated with the streams for work to be done
- Views will be configured by a configuration specification stored on the stream the view is created on
- Add the “view” shortcut in ClearCase explorer

Next, consider the development streams. After configuring the Projects Modifiable component list, you can then decide how many development streams (child streams) you will need.

Once Development streams are created views will need to be associated with the streams for work to be done. Views will be configured by a configuration specification stored on the stream the view is created on. You can then add the “view” shortcut in ClearCase explorer.

Developing in UCM (4 of 4)

6. Make an Activity

- Once setting to the view you will then need to make an activity
- The view must be set to the new activity to begin development (or, check out and check in), run:

```
cleartool mkact (activityname)
```

- To see what activity is set in a particular view, run:

```
cleartool lsact -cact
```

Make an Activity. Once setting to the view you will then need to make an activity. The view must be set to the new activity to begin development. Once the activity is set in the view, then you can begin developing in UCM. To see what activity is set in a particular view, run the command: `cleartool lsact dash cact`

This command lists the current activity for that view.

Module summary

- In this module, you have learned about:
 - What UCM is
 - UCM terms and concepts
 - How to begin developing in UCM

Contributors: Marcus Matic, Alex Grillakis, and Will Frontiero

In this module, you have learned about, what UCM is, UCM terms and concepts, and how to begin developing in UCM.

At this stage, you have enough background information to begin developing in UCM. The next module will cover how to create your UCM Projects, including Project VOBS, Project Structures, and Project Folders.

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
RCCv7 UCM Module1 UCMIIntroduction.ppt](mailto:iea@us.ibm.com?subject= Feedback about RCCv7 UCM Module1 UCMIIntroduction.ppt)

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

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

ClearCase IBM Rational

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

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.

Information is provided "AS IS" without warranty of any kind. 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 NON-INFRINGEMENT. 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 2007. 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.

