Rational Software Corporation®

Rational® ClearCase®
Rational® ClearCase® LT

Introduction

VERSION: 2003.06.00 AND LATER

PART NUMBER: 800-026160-000

UNIX/WINDOWS EDITION

Rational®
the software development company

support@rational.com
http://www.rational.com

Contents

Figures

Tables

Preface

Rational ClearCase and Rational ClearCase LT are configuration management (CM) systems that manage multiple variants of evolving software systems. ClearCase is the Rational enterprise configuration management solution for large development teams working in parallel. ClearCase LT contains a subset of ClearCase features and is aimed at customers who do not need the scalability, flexibility, and robustness provided by ClearCase. Unless otherwise noted, ClearCase refers to both ClearCase and ClearCase LT throughout this manual. Rational ClearCase MultiSite is an optional add-on product that extends ClearCase by supporting parallel software development across geographically distributed project teams.

About This Manual

This manual is intended for evaluators and new users of ClearCase. It provides the following information:

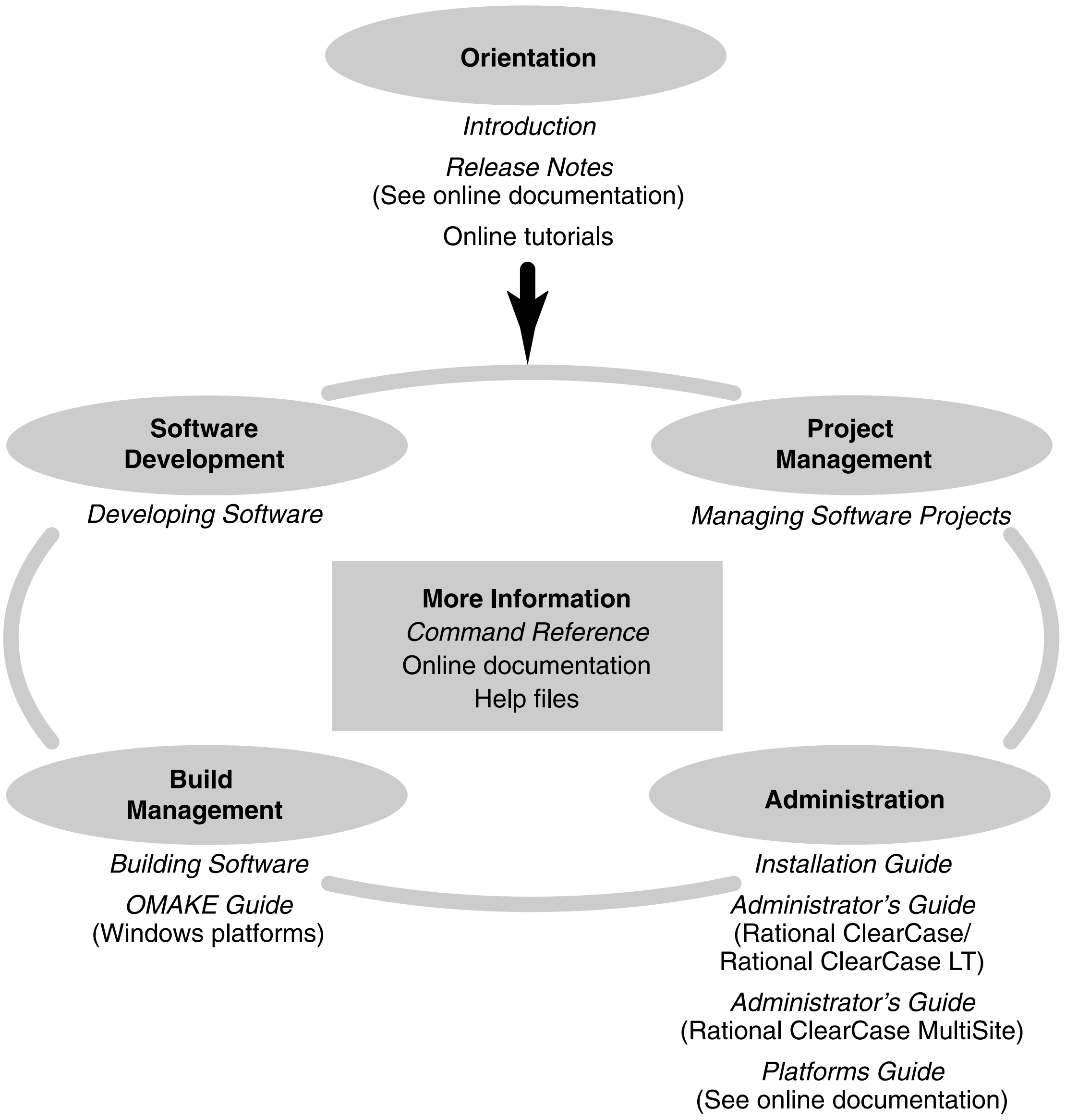
- Description of the product's features, concepts, and main user workflows
- Overview of the planning and installation process
- Overview of the product's main tools and user interfaces

User Roles and the ClearCase Documentation Set

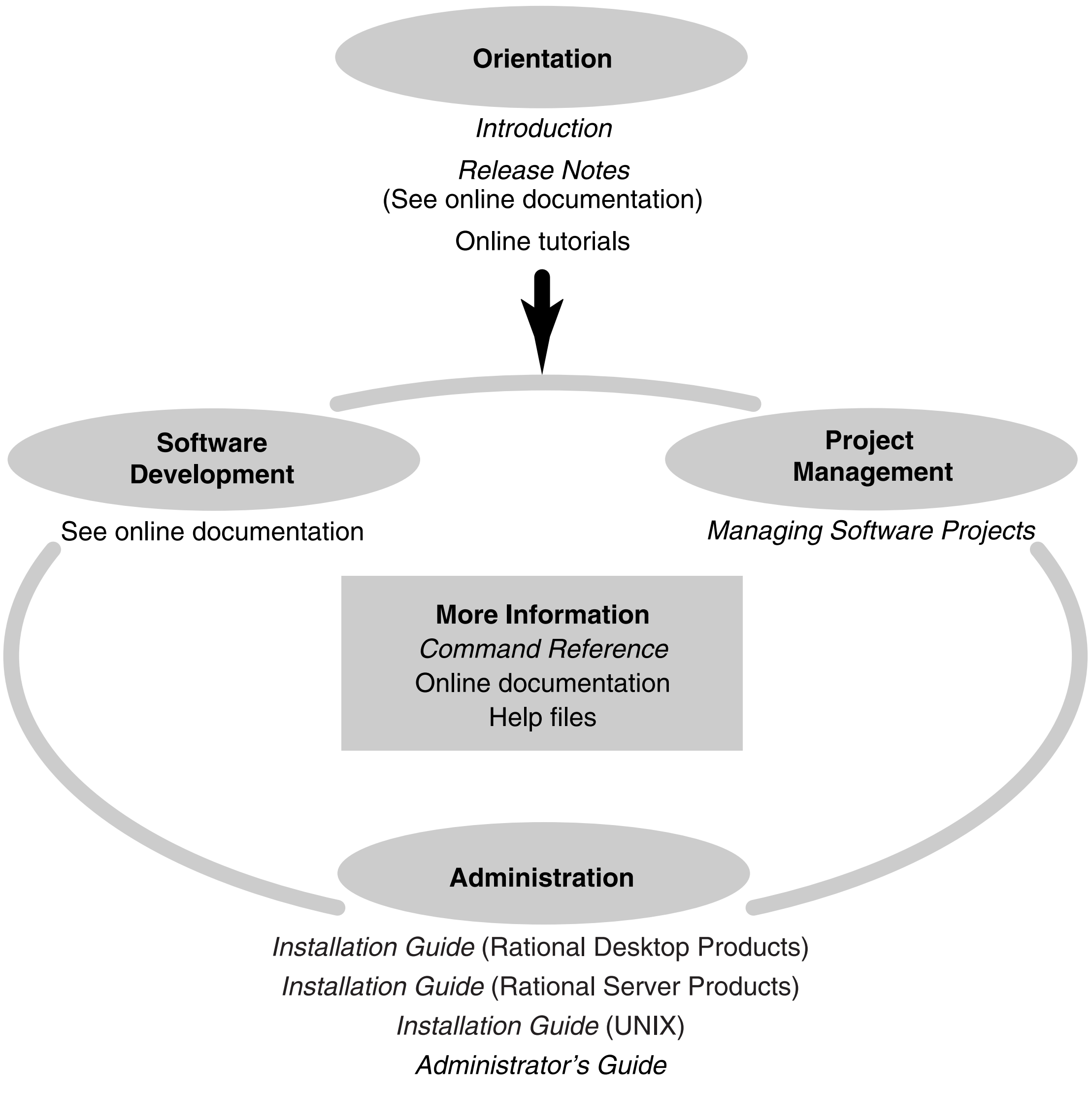
The documentation for ClearCase consists of printed and online task-oriented information, supporting ClearCase users acting in the following roles:

- **Project manager** — Defines, implements, and manages the objects, policies, and processes of a software development project
- **Developer** — Makes changes to the software configuration (that is, the files and directories) that belong to a software development project
- **Integrator** (also called build engineer or release engineer) — Builds and integrates the products of a software development project
- **Administrator** — Configures and maintains the ClearCase infrastructure, including ClearCase VOBs, views, servers, and clients, for part or all of your organization

ClearCase Documentation Roadmap



ClearCase LT Documentation Roadmap



ClearCase Integrations with Other Rational Products

Integration	**Description**	**Where it is documented**
Base ClearCase-ClearQuest	Associates change requests with versions of ClearCase elements.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
Base ClearCase-Apex	Allows Apex developers to store files in ClearCase.	*Installing Rational Apex* (UNIX)
Base ClearCase-ClearDDTS	Associates change requests with versions of ClearCase elements.	*ClearCase ClearDDTS Integration*
Base ClearCase-PurifyPlus	Allows developers to invoke ClearCase from PurifyPlus.	PurifyPlus Help
Base ClearCase-RequisitePro	Archives RequisitePro projects in ClearCase.	*RequisitePro User's Guide* RequisitePro Help
Base ClearCase-Rose	Stores Rose models in ClearCase.	Rose Help
Base ClearCase-Rose RealTime	Stores Rose RealTime models in ClearCase.	*Rose RealTime Toolset Guide* *Rose RealTime Guide to Team Development*
Base ClearCase-SoDA	Collects information from ClearCase and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
Base ClearCase-XDE	Stores XDE models in ClearCase	XDE Help
UCM-ClearQuest	Links UCM activities to ClearQuest records.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
UCM-PurifyPlus	Allows developers to invoke ClearCase from PurifyPlus.	PurifyPlus Help

Integration	Description	Where it is documented
UCM-RequisitePro	Allows RequisitePro administrators to create baselines of RequisitePro projects in UCM, and to create RequisitePro projects from baselines.	*RequisitePro User's Guide* RequisitePro Help *Using UCM with Rational Suite*
UCM-Rose	Stores Rose models in ClearCase.	Rose Help *Using UCM with Rational Suite*
UCM-Rose RealTime	Associates activities with revisions.	*Rose RealTime Toolset Guide* *Rose RealTime Guide to Team Development*
UCM-SoDA	Collects information from ClearCase and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
UCM-TestManager	Stores test assets in ClearCase.	*Rational TestManager User's Guide* TestManager Help *Using UCM with Rational Suite*
UCM-XDE	Stores XDE models in ClearCase	XDE Help
UCM-XDE Tester	Stores XDE Tester Datastores in ClearCase	XDE Tester Help

Typographical Conventions

This manual uses the following typographical conventions:

- *ccase-home-dir* represents the directory into which the ClearCase Product Family has been installed. By default, this directory is /opt/rational/clearcase on UNIX and C:\Program Files\Rational\ClearCase on Windows.
- *cquest-home-dir* represents the directory into which Rational ClearQuest has been installed. By default, this directory is /opt/rational/clearquest on UNIX and C:\Program Files\Rational\ClearQuest on Windows.
- **Bold** is used for names the user can enter; for example, command names and branch names.
- A sans-serif font is used for file names, directory names, and file extensions.

- **A sans-serif bold font** is used for GUI elements; for example, menu names and names of check boxes.
- *Italic* is used for variables, document titles, glossary terms, and emphasis.
- `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.
- Nonprinting characters appear as follows: <EOF>, <NL>.
- Key names and key combinations are capitalized and appear as follows: SHIFT, CTRL+G.
- [ ] Brackets enclose optional items in format and syntax descriptions.
- { } Braces enclose a list from which you must choose an item in format and syntax descriptions.
- | A vertical bar separates items in a list of choices.
- ... In a syntax description, an ellipsis indicates you can repeat the preceding item or line one or more times. Otherwise, it can indicate omitted information.

  Note: In certain contexts, you can use "**...**" within a pathname as a wildcard, similar to "*" or "?". For more information, see the **wildcards_ccase** reference page.
- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

Online Documentation

The ClearCase Product Family (CPF) includes online documentation, as follows:

Help System: Use the **Help** menu, the **Help** button, or the F1 key. To display the contents of the online documentation set, do one of the following:

- On UNIX, type **cleartool man contents**
- On Windows, click **Start > Programs > Rational Software > Rational ClearCase > Help**
- On either platform, to display contents for Rational ClearCase MultiSite, type **multitool man contents**
- Use the **Help** button in a dialog box to display information about that dialog box or press F1.

Reference Pages: Use the **cleartool man** and **multitool man** commands. For more information, see the **man** reference page.

Command Syntax: Use the **–help** command option or the **cleartool help** command.

Tutorial: Provides a step-by-step tour of important features of the product. To start the tutorial, do one of the following:

- On UNIX, type **cleartool man tutorial**
- On Windows, click **Start** > **Programs** > **Rational Software** > **Rational ClearCase** > **ClearCase Tutorial**

PDF Manuals: Navigate to:

- On UNIX, *ccase-home-dir*/doc/books
- On Windows, *ccase-home-dir*\doc\books

Customer Support

If you have any problems with the software or documentation, please contact Rational Customer Support by telephone, fax, or electronic mail as described below. For information regarding support hours, languages spoken, or other support information, click the **Support** link on the Rational Web site at **www.rational.com**.

Your location	**Telephone**	**Facsimile**	**Electronic mail**
North America	800-433-5444 toll free or 408-863-4000 Cupertino, CA	408-863-4194 Cupertino, CA 781-676-2460 Lexington, MA	**support@rational.com**
Europe, Middle East, and Africa	+31-(0)20-4546-200 Netherlands	+31-(0)20-4546-201 Netherlands	**support@europe.rational.com**
Asia Pacific	61-2-9419-0111 Australia	61-2-9419-0123 Australia	**support@apac.rational.com**

Overview of ClearCase

1

This chapter introduces the features of Rational ClearCase and Rational ClearCase LT and describes the different models for using the products. Unless otherwise noted, ClearCase refers to both ClearCase and ClearCase LT. This chapter also introduces Rational ClearCase MultiSite.

ClearCase offers two configuration management interfaces:

- Unified Change Management (UCM), which is an out-of-the-box, activity-based change management process.
- Base ClearCase, which is a set of tools that you can use to create a configuration management solution tailored to your development environment.

Benefits of Using ClearCase

Rational ClearCase is an extensive set of tools that you can use to manage and track software resources, whether they are Web pages, mission-critical data, documentation, or source code. Like other source control tools, ClearCase helps you do the following with your software resources:

- Create new versions of a software resource.
- Compare versions of a software resource.
- Merge changes from one version into another version.
- Control simultaneous changes to a software resource.
- Mark certain versions as stable sources to be used in builds.
- Determine the who, when, and why of a particular change.

ClearCase also provides some unique advantages:

- Supports team leaders who need to coordinate the activity of people developing products together.
- Gives project managers control over the extent and frequency with which team members synchronize their work.
- Supports parallel development: developers work in private areas and do not affect the work of team members.

- Assists integrators in combining the efforts of the team in a controlled manner.

Basic Terminology of ClearCase

Before you learn about the most important features of ClearCase, it is helpful for you to understand the following terms:

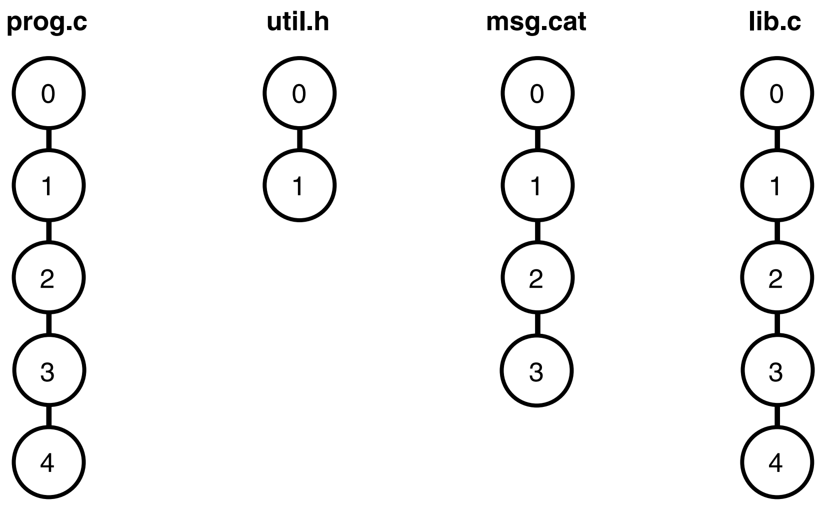
- File element
- Version
- Versioned object base (VOB)
- Check out-edit-check in model
- View

A *file element* is a file that contains software source code, a document, HTML code, XML code, or other data that can be stored in a file system.

A *directory element* contains file elements and other directory elements.

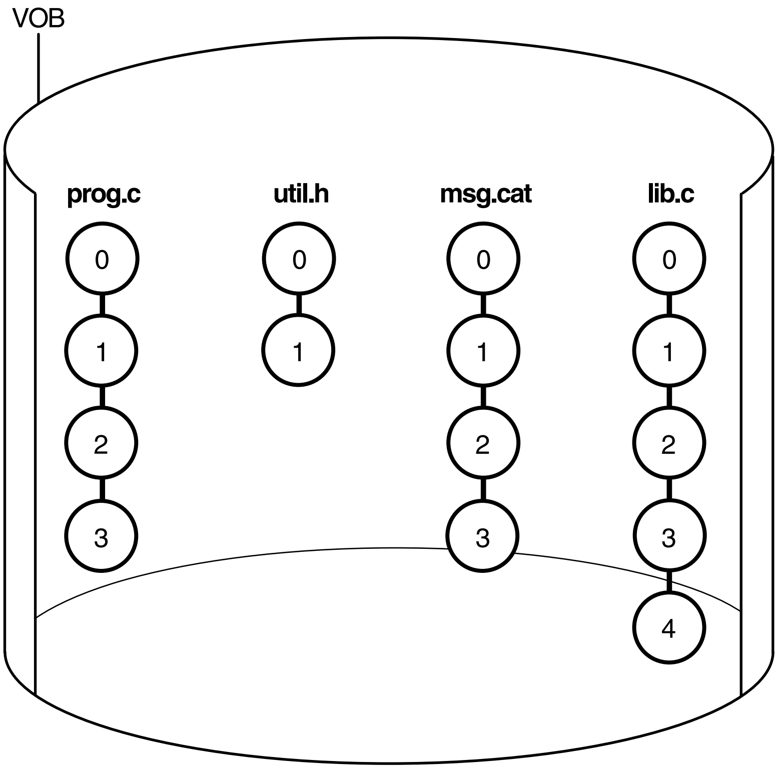
A *version* is a specific revision of an element. For instance, instead of overwriting the same copy of your draft each time you work on it, you store a copy of the first version, the second version, and so on (Figure 1).

Figure 1 Elements and Their Versions



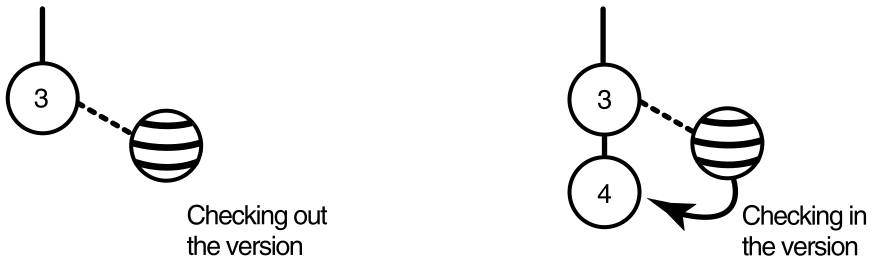
A *versioned object base*, or *VOB*, is a repository that stores versions of file elements and directory elements (Figure 2).

Figure 2 Storing Elements in a Versioned Object Base (VOB)



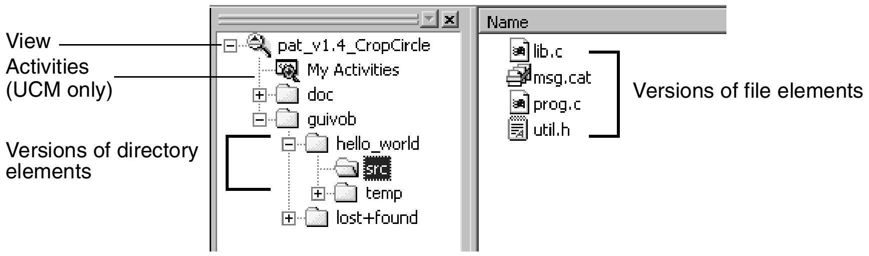
A *check out-edit-check in model* manages changes to your project. When you check out an element, ClearCase creates an editable copy in your view. When you check in an element, a new version of it is added to the VOB. For instance, you check out a chapter of a book from the documentation VOB. You work on a copy of the chapter in your view. When you want your changes to become a part of a new version of the chapter, you save the changes and check in the chapter (Figure 3).

Figure 3 Using the Check Out/Check In Model



A *view*, which is represented as a directory, provides access to a specific version of one or more elements in a VOB. It lets you select a set of versions of elements without having to specify the versions explicitly. It provides a workspace in which you can work on assignments in isolation from other developers (Figure 4). For example, as you work on the latest version of a Web page, no one can see the changes you made until you check in that work to the VOB.

Figure 4 Using a View to See Shared Elements



Differences Between ClearCase and ClearCase LT

This manual describes ClearCase and ClearCase LT. ClearCase is the Rational enterprise configuration management solution for large development teams working in parallel. ClearCase LT contains a subset of ClearCase features and is aimed at customers who do not need the scalability, flexibility, and robustness provided by ClearCase. Table 1 describes the main differences between ClearCase and ClearCase LT.

Table 1 Features in ClearCase but Not in ClearCase LT

Feature	**Description**
Dynamic views	ClearCase and ClearCase LT offer *snapshot views* as workspaces. Snapshot views work by copying versions of elements from VOBs to your computer. To see the latest versions of elements, you need to update your snapshot view periodically. An update operation copies the latest versions of elements from the VOB to your view. ClearCase also offers dynamic views. A *dynamic view* uses the *Multiversion File System* (MVFS) to provide immediate, transparent access to data stored in VOBs. When you work in a dynamic view, you do not need to copy data from VOBs to your view; you always see the latest versions of elements.
Multiple servers	ClearCase and ClearCase LT use a client-server architecture. A client process issues a request for a ClearCase operation, such as checkout or checkin. Server processes that run on the host where the VOB resides handle the request. ClearCase supports a multiple server configuration. You can locate VOBs on multiple, distributed hosts. ClearCase LT requires that you locate all VOBs on one host.

Table 1 Features in ClearCase but Not in ClearCase LT (Continued)

Feature	Description
Advanced build tools	ClearCase supports makefile-based building of software systems and includes tools (**clearmake**, **omake** on Windows, and **clearaudit**) that provide the following powerful enhancements: ▪ Build auditing, with automatic detection of source dependencies, including header file dependencies. ▪ Automatic creation of permanent bill-of-materials documentation of the build process and its results. ▪ Sophisticated build-avoidance algorithms to guarantee correct results when building in a parallel development environment. ▪ Sharing of binaries among views, saving both time and disk storage. ▪ Parallel building, applying the resources of multiple processors and/or multiple hosts to builds of large software systems. These enhancements are available only if you use dynamic views; therefore, they are not available with ClearCase LT.
ClearCase MultiSite	ClearCase MultiSite is an optional add-on product that extends ClearCase by supporting parallel software development and software reuse across geographically distributed project teams. ClearCase LT does not support MultiSite.

Using Unified Change Management

UCM structures the work of a project team into a defined, repeatable process. This section provides an overview of the key concepts and tasks involved in using UCM.

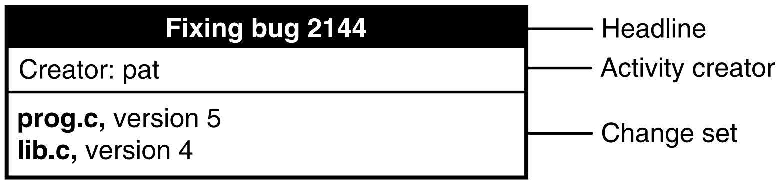
UCM Terminology

A project is a specific product of a development effort, such as a corporate Web site or an order fulfillment process for an e-business. In UCM, a *project* is an object that contains the configuration information (for example, components, activities, policies) needed to manage and track the work on a product. A typical UCM project in ClearCase consists of one shared work area and many private work areas (one for each developer).

A *component* is any group of source code and other relevant elements or files, such as a license module or a customer GUI, that the team develops, integrates, and releases as a unit. Components constitute parts of a project, and projects often share components.

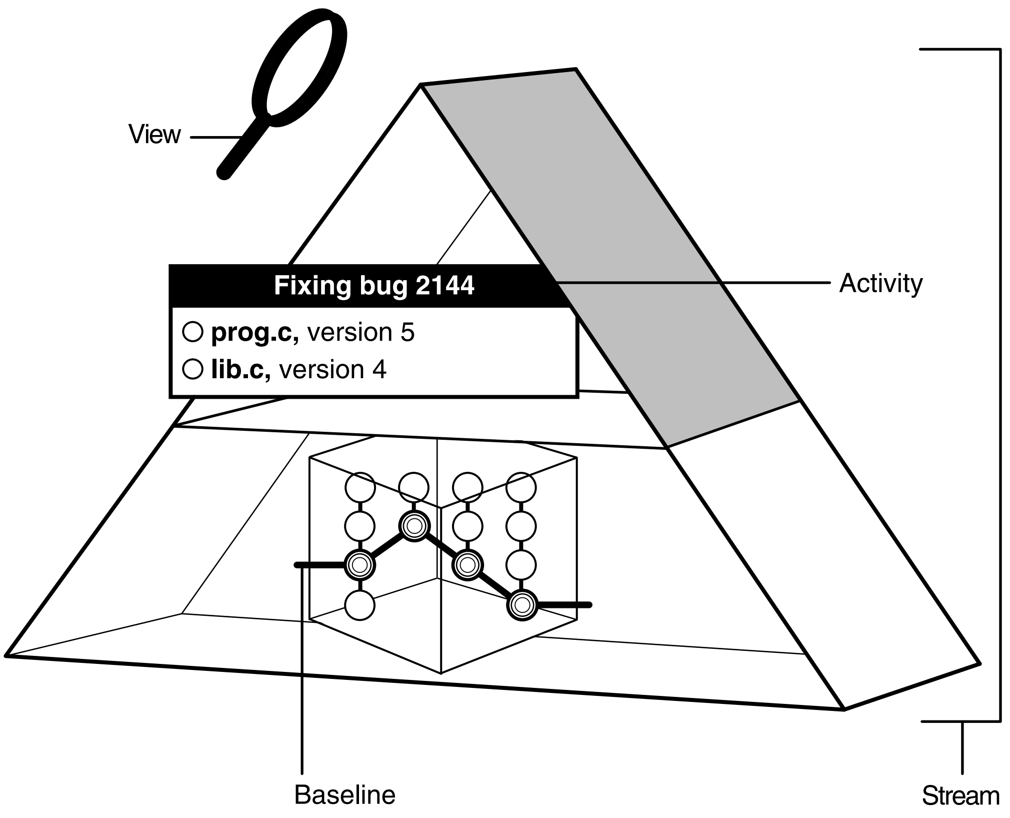
An *activity* is an object that records the set of files (*change set*) that a developer creates or modifies to complete and deliver a development task, such as a bug fix (Figure 5). Examples of other activities could include an update to a help file or the addition of a menu item to a GUI component.

Figure 5 An Activity



A work area consists of a view and a stream. As described in *Basic Terminology of ClearCase* on page 2, a *view* is a directory tree that shows a single version of each file in your project. A *stream* is a ClearCase object that maintains a list of activities and baselines and determines which versions of elements appear in your view (Figure 6).

Figure 6 A Stream



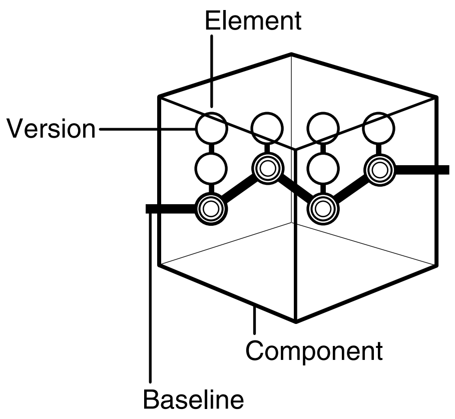
A project contains one *integration stream*, which records the project's baselines and enables access to versions of the project's shared elements. The integration stream and a corresponding integration view represent the project's primary shared work area.

Each developer on a project has a private work area, which consists of a development stream and a corresponding development view. The *development stream* maintains a list of the developer's activities and determines which versions of elements appear in the developer's view.

Although the integration stream is the project's primary shared work area, project managers can designate a development stream to be a shared work area for several developers who are working on the same feature.

A *baseline* identifies one version of each element in a component that represents the integrated or merged work of team members (Figure 7). It represents a version of a component at a particular stage in project development, such as the first draft of a book, a beta release, or a final product release. Throughout the project cycle, the project manager creates and recommends baselines and changes their promotion level attributes to reflect project milestones.

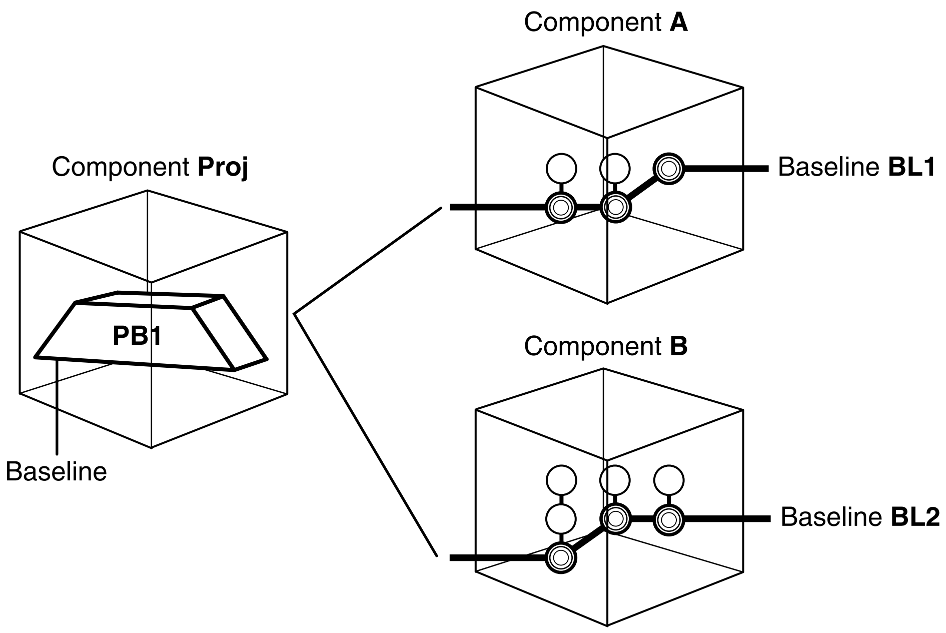
Figure 7 A Baseline



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.

If your project team works on multiple components, you many want to use a composite baseline. A *composite baseline* is a baseline that selects baselines in other components. In Figure 8, the **ProjBL1** composite baseline selects baselines **BL1** and **BL2** of components **A** and **B**, respectively. The **Proj** component does not contain any elements of its own. Its sole purpose is to contain the composite baseline that selects the recommended baselines of the project's components. By using a composite baseline in this manner, you can identify one baseline to represent the entire project.

Figure 8 Composite Baseline



Creating a Project in UCM

Before creating a project, the project manager maps files and directories in the product architecture to a set of UCM components. Using utilities provided in ClearCase, the project manager or administrator converts existing files and directories from third-party version control software systems to elements. See *Managing Software Projects* and *Administrator's Guide* for Rational ClearCase for details about importing existing files and directories into components.

The project manager creates a project, along with its baselines and integration stream, with the Project Explorer. You can create a new project from scratch or populate a new project with baselines from an existing project.

Note: When you install ClearCase, the Getting Started Wizard creates a *project VOB* (PVOB), an initial component VOB, an initial project, and an initial baseline. On Windows, it also asks whether you want to import data into ClearCase and runs the Import Wizard, if appropriate.

The project includes a private work area for each developer (Figure 9) and one shared work area (Figure 10). The project manager manages the shared work area. These areas are explained in more detail in *Joining the Project* on page 12 and *Delivering Work to the Integration Stream* on page 13.

Figure 9 Private Work Areas

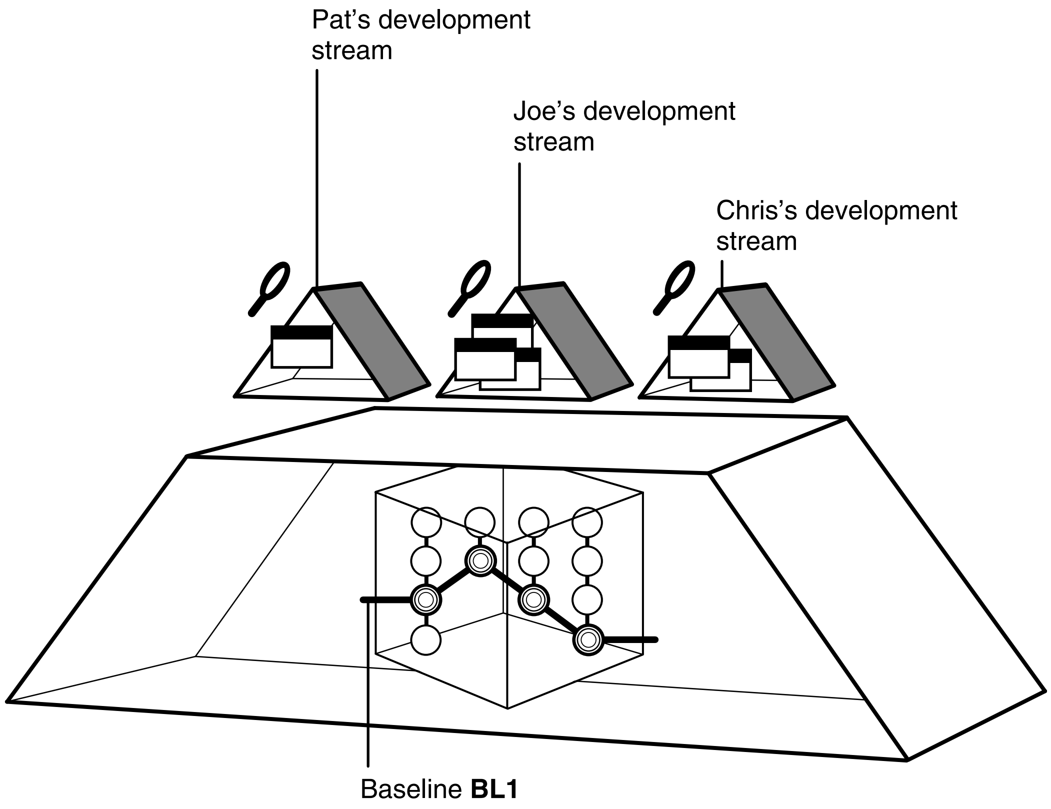
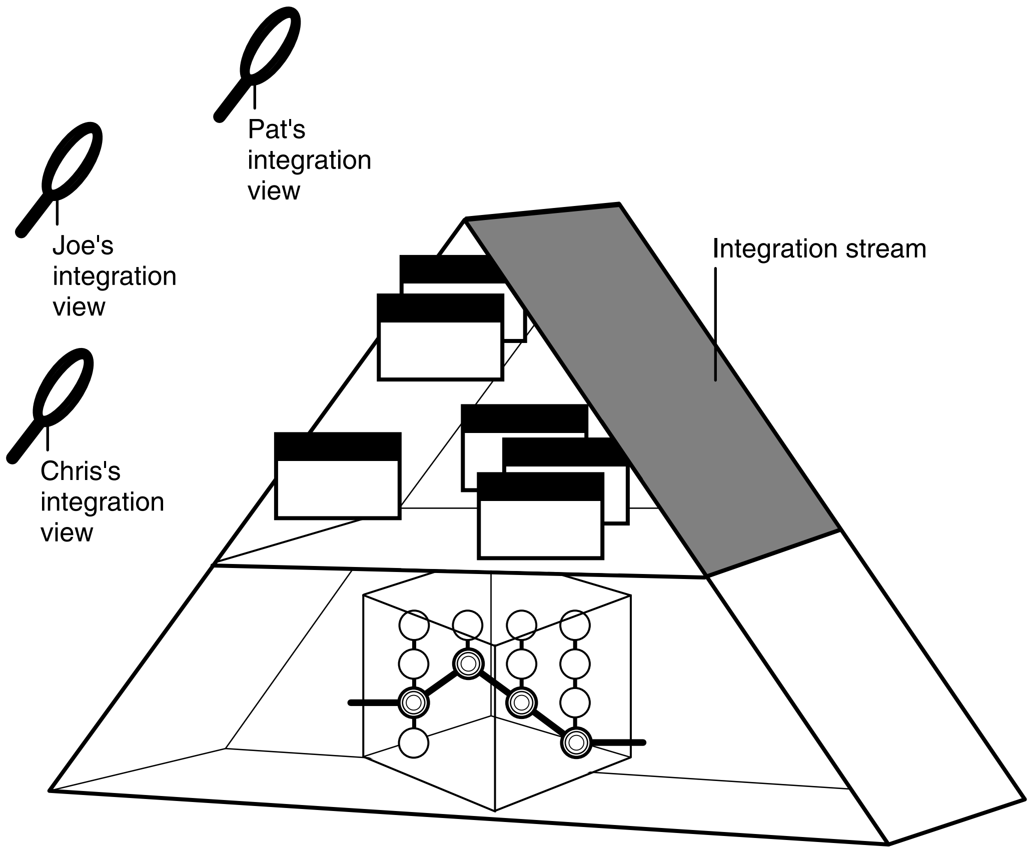


Figure 10 Shared Work Area



Joining the Project

To join the project, you use the Join Project Wizard to create a *development stream*, a *development view*, and an *integration view*.

The development stream and development view make up the private work area, where you work on activities. You use development views to access and change elements. The development stream determines which versions of elements appear in each development view.

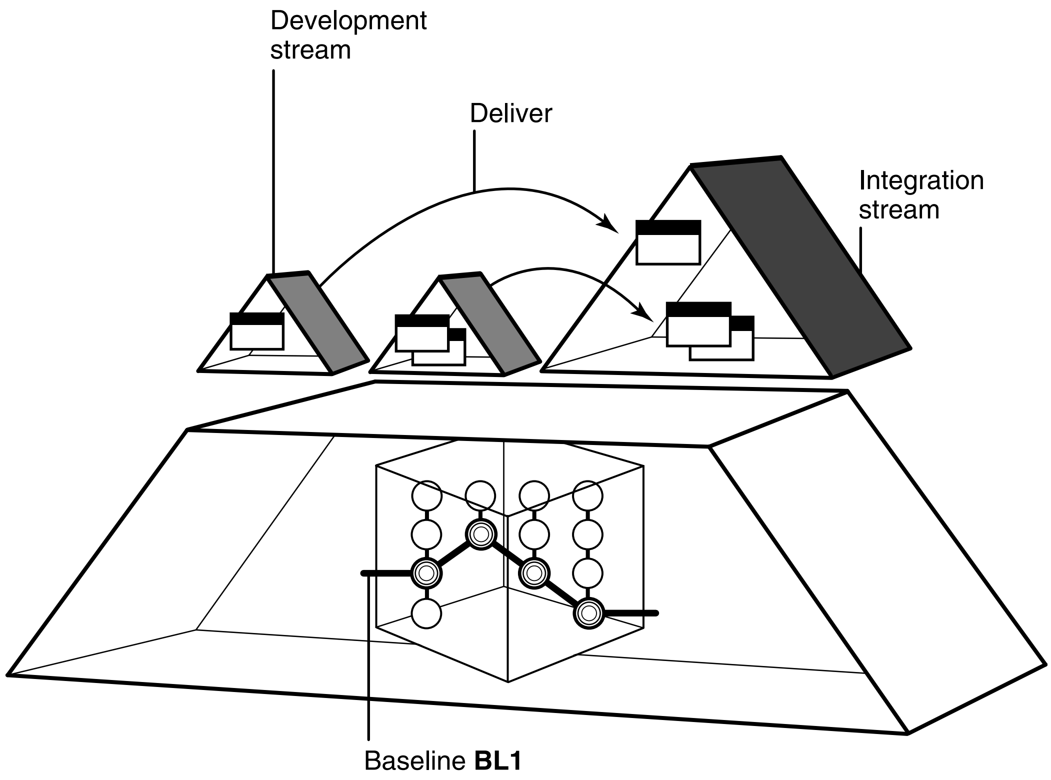
The integration view gives developers and project managers access to the work that has been delivered to the shared work area.

Delivering Work to the Integration Stream

When the project was created, the project manager created *one* shared work area, or *integration stream*.

When you are ready to integrate your work with the work of the team, you *deliver* that work from the development stream to the integration stream. The deliver operation may involve a merge if other team members have already delivered versions of one or more of the same elements (Figure 11).

Figure 11 Delivering Work to the Integration Stream



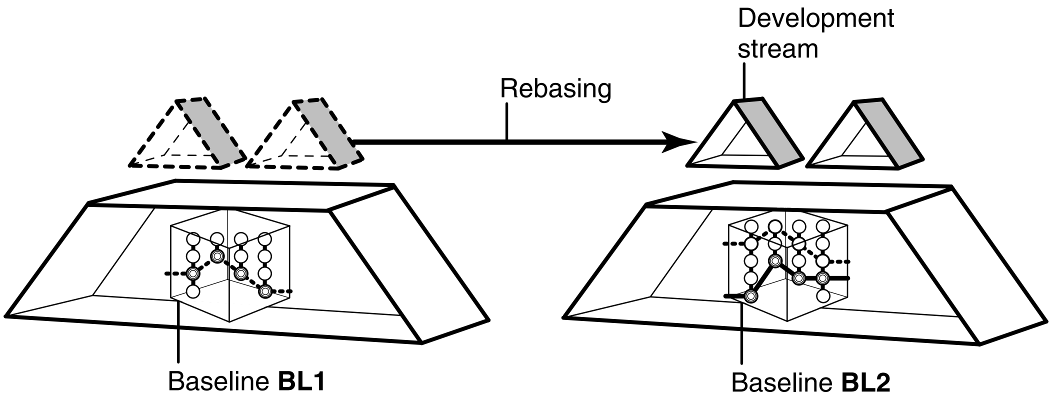
When the work is delivered and merged, you use the integration view to build and test work in the integration stream. The integration stream determines which versions of delivered elements appear in the integration view.

Rebasing Private Work Areas

Periodically, the project manager creates a new baseline, which includes the work delivered since the last baseline was created. When this new baseline passes a certain level of testing, the project manager promotes it, designating it as the project's *recommended baseline*.

As shown in Figure 12, you update, or *rebase*, your development stream to use the new recommended baselines. The rebase operation updates the development view so that it includes the versions from the new baselines. If the baselines include new versions of the same element that you checked in from a private work area (that is, another developer checked in and delivered a new version of the same element to the integration stream), the rebase operation merges the new version into your development stream.

Figure 12 Rebasing the Development Stream



Implementing Development Policies

UCM includes a set of policies that you can use on projects and streams to enforce development practices among members of the project team. By setting policies, you can improve communication among project team members and minimize the problems you may encounter when integrating their work. For example, you can set a policy that requires developers to update their work areas with the project's latest recommended baseline before they deliver work to the integration stream. This practice reduces the likelihood that developers will need to work through complex merges when they deliver their work. For a description of all policies that you can set in UCM, see *Managing Software Projects*.

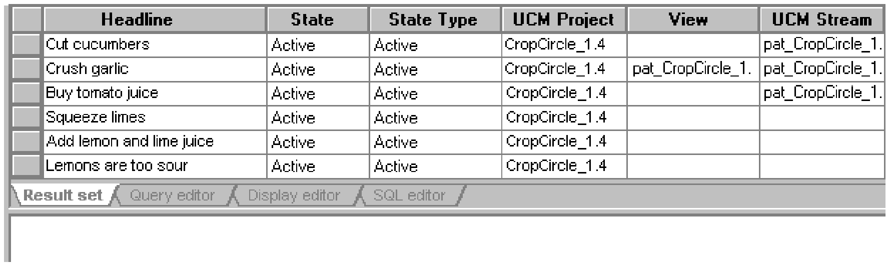
In addition to the set of policies that UCM provides, you can create triggers on UCM operations to enforce customized development policies. A *trigger* is a monitor that specifies one or more standard programs or built-in actions to be run whenever a certain ClearCase operation is performed. See *Managing Software Projects* for details about creating triggers.

Integrating UCM with ClearQuest

Rational ClearQuest is a change-request management application. Using ClearQuest, team members can submit change requests for their product, view and modify existing change requests, and create and run user- or site-specific queries and reports to determine project status.

The UCM-ClearQuest integration provides additional activity management capabilities by linking project activities in UCM to corresponding records in a ClearQuest user database. The integration lets the project team track an activity's progress from assignment through completion. Project managers can assign activities to developers, and developers can use ClearQuest queries (Figure 13) to find their assignments. Because activities have corresponding records in a ClearQuest user database, project managers can use all the query, reporting, and charting utilities in ClearQuest to help them manage projects.

Figure 13 Using a ClearQuest To-Do List to Find UCM Activities



Headline	State	State Type	UCM Project	View	UCM Stream
Cut cucumbers	Active	Active	CropCircle_1.4		pat_CropCircle_1.
Crush garlic	Active	Active	CropCircle_1.4	pat_CropCircle_1.	pat_CropCircle_1.
Buy tomato juice	Active	Active	CropCircle_1.4		pat_CropCircle_1.
Squeeze limes	Active	Active	CropCircle_1.4		
Add lemon and lime juice	Active	Active	CropCircle_1.4		
Lemons are too sour	Active	Active	CropCircle_1.4		

For information about planning and setting up the UCM-ClearQuest integration, see the Help and *Managing Software Projects.* For information about working on activities in a project that is linked to a ClearQuest user database, see *Developing Software* and the Help.

Using Base ClearCase

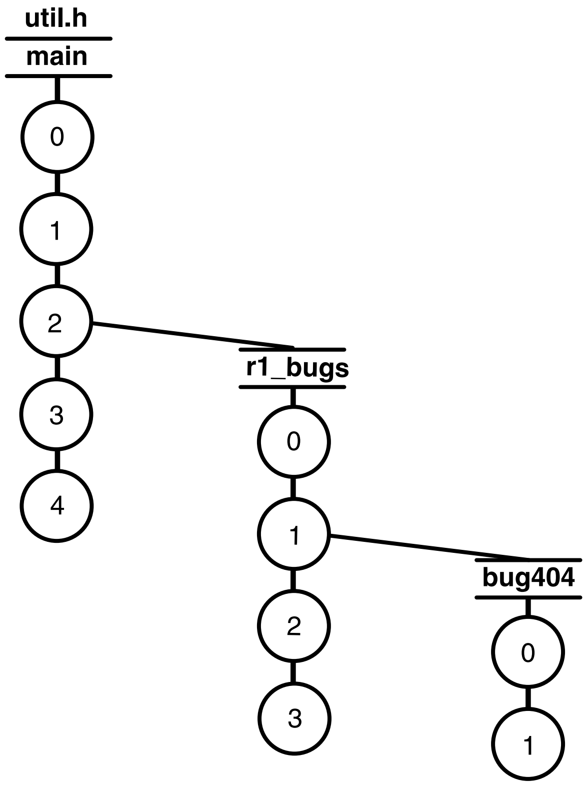
A project team can use ClearCase tools to develop its own system of managing and tracking software resources. This section describes the basic steps involved in setting up a work environment in base ClearCase.

For more information about creating a custom configuration management solution, see the Help, *Managing Software Projects*, and the *Administrator's Guide* for Rational ClearCase.

Base ClearCase Terminology

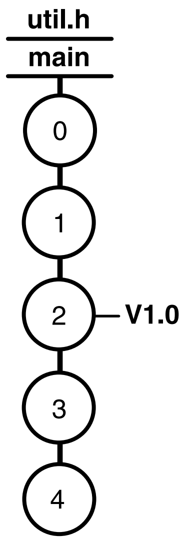
A *branch* is an object that specifies a sequence of versions of an element. Every element has one *main branch*, which represents the principal line of development, and may have multiple *subbranches*, each of which represents a separate line of development (Figure 14).

Figure 14 Branches



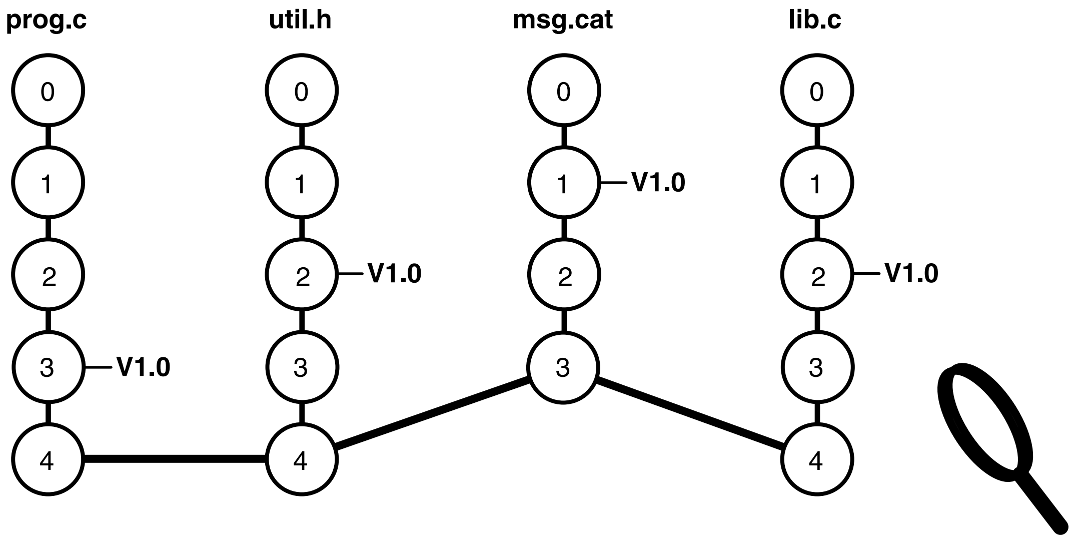
A *version label* can be attached to any version of an element to identify that version in an easy to remember way. A single version of an element can have several different labels. Labels are usually applied to a set of elements to mark important project milestones or the proposed starting point of a branch (Figure 15).

Figure 15   Version Label



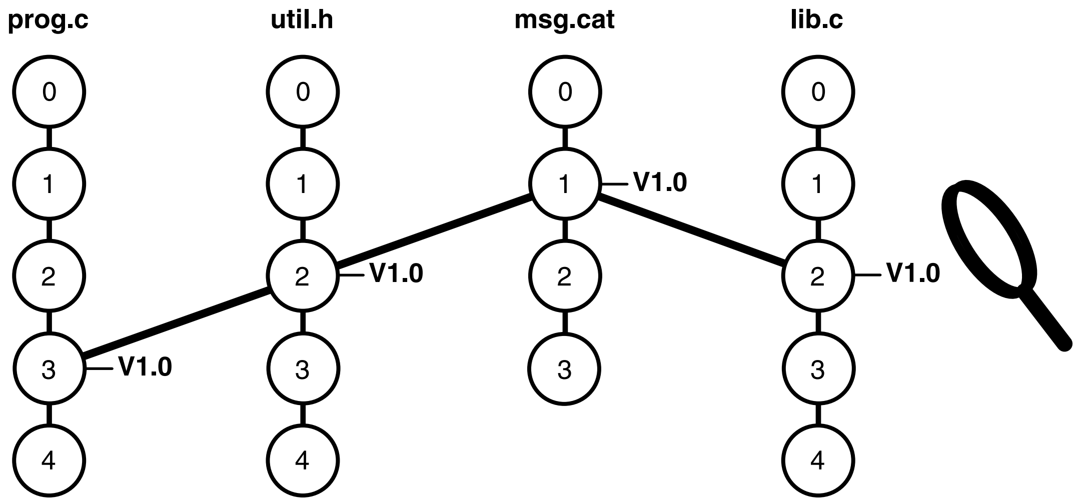
A *configuration specification*, or config spec, contains the rules used by a view to select versions of elements. The rules are very flexible, and you can use various specifiers to indicate which versions to select. For example, a config spec for a view used in ongoing development would contain rules that select the latest version on the development branch (Figure 16).

Figure 16 Selecting Versions Used in Development Work



To examine the versions that were included in a particular release, you would use a config spec that uses a label rule to select the versions that were labeled for that release (Figure 17).

Figure 17 Selecting Versions Included in a Particular Release



Setting Up a Project in Base ClearCase

When project managers set up projects in base ClearCase, they map files and directories in the product architecture to a set of ClearCase VOBs (just as project managers map files and directories to ClearCase components in UCM). For instructions on creating VOBs and populating them with files and directories, see the *Administrator's Guide* for Rational ClearCase.

After the files and directories are imported into VOBs, the project manager usually attaches labels to a set of versions of elements in those VOBs to designate the versions from which development should start.

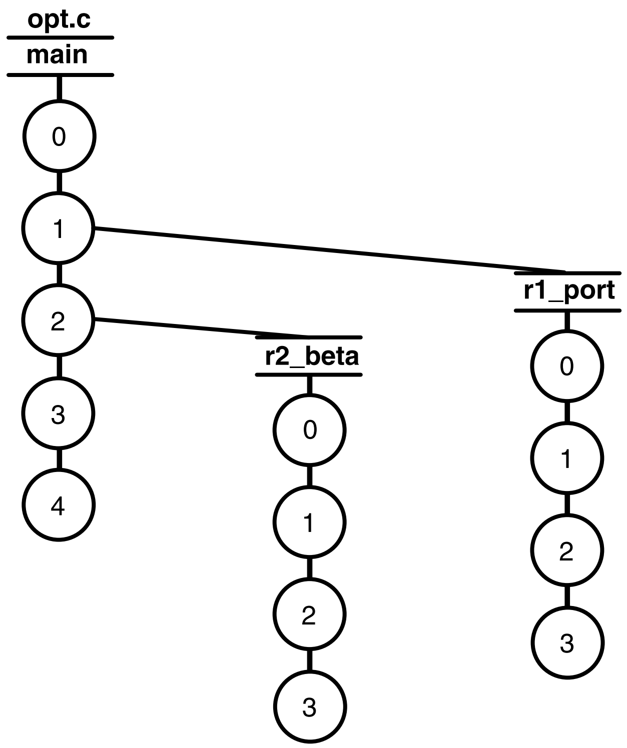
Planning a Branching Strategy

When you use base ClearCase, you must use branches directly to implement parallel development. (UCM uses streams to manage branches for you.)

This section provides a simple example of how to define a branching strategy. As you read the section, keep in mind that the possible branching and merging combinations are almost infinite.

A project team can use many branches concurrently. In Figure 18, the **main** branch represents the integration branch, an r1_port subbranch is a port to a new platform, an **r2_beta** subbranch contains work for the beta version of the next release, and so on.

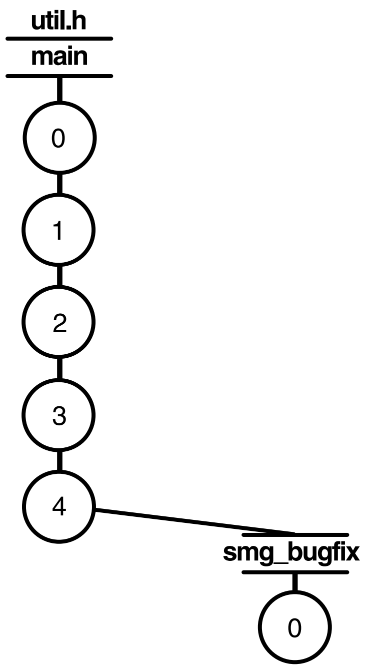
Figure 18 Branches in a File System Directory Tree



Using a Private Branch

At times, you may want to create a private branch based on the project branch. For example, you want to work on a bug fix without affecting the work that other developers are doing. To do this, choose the version from which you want to start. Then you can create a second config spec for a second view that will create a private branch starting at that version (Figure 19).

Figure 19 Working on a Private Branch



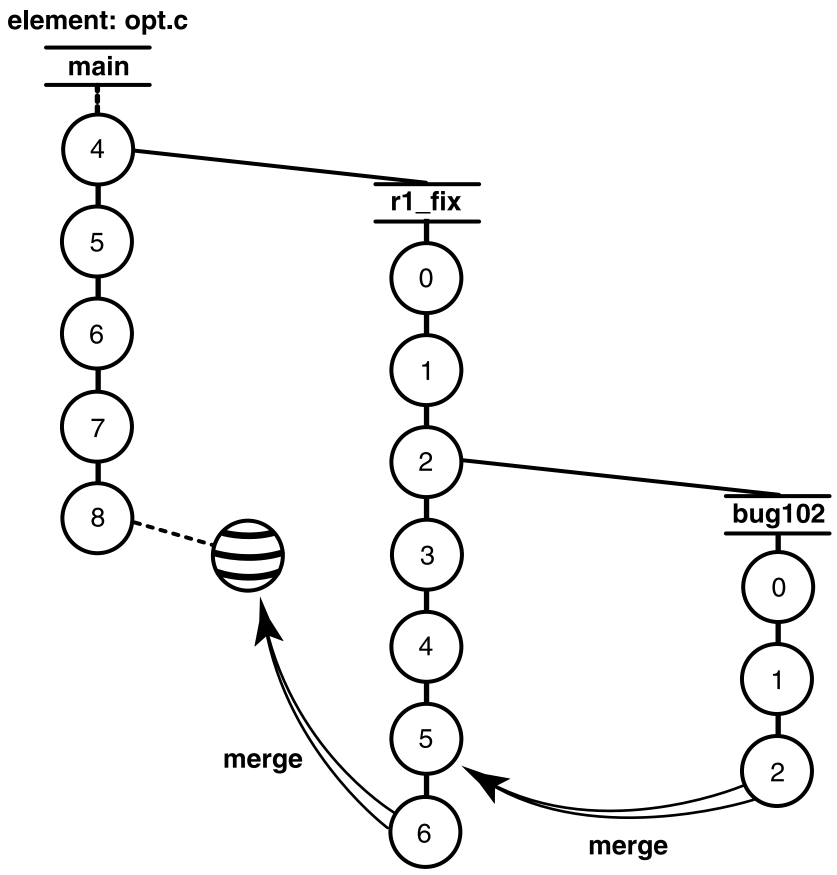
To incorporate your work into the project, merge the changes on your private branch to the project branch. If you decide to abandon your work, do not merge the changes. In either case, when you want to resume your work on the project branch, change your view back to use the project branch view.

For more information about creating private branches for development work and merging your work back to the project branch, see *Developing Software* or Help.

Merging Work Between Branches

To integrate work from one branch to another, you *merge* work from one subbranch to another branch. In Figure 20, all work must be merged to the **main** branch before it can be included in a product release. After the fix on the **bug102** branch is tested and approved for integration, you first merge the work to the **r1_fix** branch. At some point, you test all the work on the **r1_fix** branch; when that work is ready to be incorporated into the main project, you merge from that branch to the **main** branch.

Figure 20 Merging Branches



Creating Standard Config Specs

To ensure that team members are working on the correct branches (appropriate directory and file versions), the project manager creates a standard config spec that team members use in their views. For more information on creating config specs, see *Managing Software Projects*.

Creating Labels

A *label* in base ClearCase is a user-defined name that can be attached to a version. Labels are a powerful tool for project managers. By applying labels to groups of elements, you can define and preserve the relationship of a set of file and directory versions to each other at a given point in the development lifecycle. For example, you can apply a label to all versions considered stable after integration and testing. Use this baseline label as the foundation for new work.

Using Metadata to Implement Policies

In base ClearCase, project managers use *metadata* to define project policies and annotate different objects in the VOB. Metadata include the following:

- Attributes
- Branch types
- Hyperlinks
- Locks
- Triggers
- Version labels

Note: You can also use attributes, hyperlinks, locks, and triggers when you use UCM.

To annotate objects in the VOB, a project manager can use attributes, hyperlinks, and version labels:

- Attributes are name-value pairs that are attached to objects. One use for attributes is to attach bug numbers to versions; another is to identify which versions in a baseline have been tested.
- Hyperlinks connect two VOB objects. For example, you can define a hyperlink to connect a design specification to its associated executable.
- Labels identify specific versions of elements. Labels can be used in config spec rules to identify branching points or to mark all versions that were used in a build.

To define and enforce project policy, project managers can use locks and triggers. For example, an integration branch can be locked temporarily to prevent developers from merging work onto it. A project manager could also create a trigger that prompts developers to enter the bug number associated with a checkin and attaches that number to the version.

For details about using ClearCase metadata, see *Managing Software Projects* and *Administrator's Guide* for Rational ClearCase.

Integrating Base ClearCase with ClearQuest

Rational ClearQuest is a change-request management application. Using ClearQuest, team members can submit change requests for their product, view and modify existing change requests, and create and run user- or site-specific queries and reports to determine project status.

The base ClearCase-ClearQuest integration lets you to associate change requests with versions of ClearCase elements. The versions associated with a change request constitute a change set. In ClearQuest, a tab on the change request's record form displays the change set information.

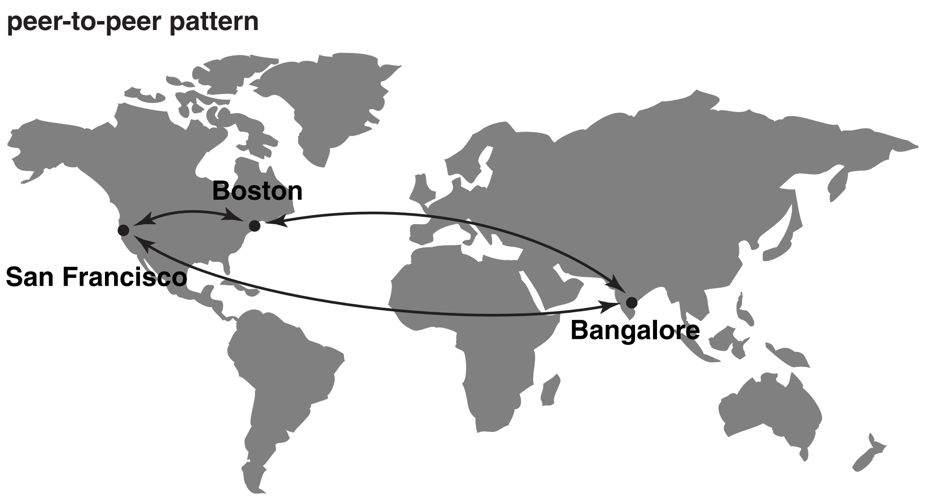
When you configure the integration you specify the conditions under which developers are prompted to associate versions with change requests. For example, you can require developers to specify a change request when they check out or check in a file, and you can have this requirement apply only to specific VOBs, branch types, or element types.

ClearCase MultiSite

Rational ClearCase MultiSite extends ClearCase by supporting parallel software development and software reuse across geographically distributed project teams.

ClearCase MultiSite enables developers at different locations to use the same VOB, as shown in Figure 21. Each site has its own copy, or *replica*, of that VOB. The set of replicas for a particular VOB is called a *VOB family*. At any time, a site can propagate the changes made in its own VOB replica to the other members of the VOB family, using either an automatic or manual synchronization process.

Figure 21 ClearCase MultiSite VOB Family



This manual describes ClearCase MultiSite only where it applies to a given ClearCase operation or concept. See *Administrator's Guide* for Rational ClearCase MultiSite for details about configuring, using, and administering ClearCase MultiSite.

Planning for and Installing ClearCase

2

This chapter identifies the major planning decisions you need to make and summarizes the basic steps required to install Rational ClearCase and Rational ClearCase LT, providing a high-level understanding of the installation process. It is not intended to be a complete installation guide. For details about installing ClearCase at your site, see one of the following manuals:

- For ClearCase, *Installation Guide* for the ClearCase Product Family.
- For ClearCase LT on UNIX, *Installation Guide* for Rational ClearCase LT
- For ClearCase LT on Windows, *Installation Guide* for Rational Server Products and *Installation Guide* for Rational Desktop Products

If you are a ClearCase administrator, charged with installing ClearCase at your site, read this chapter.

If you are a project manager, *Planning Issues* contains information about using UCM, base ClearCase, Rational ClearQuest, and Rational ClearCase MultiSite in your organization.

If you are concerned only with installing ClearCase on your client computer, start with *Installing ClearCase on Individual Computers* on page 31.

Planning Issues

This section describes some of the significant decisions administrators or project managers must make before you can begin installing ClearCase at your site.

Using Unified Change Management or Base ClearCase

UCM is an optional activity-based process of using ClearCase for version control and configuration management. Because UCM is layered on base ClearCase, it is possible to work efficiently in UCM without having to master the details of base ClearCase.

UCM provides the convenience of an out-of-the-box solution; base ClearCase offers the flexibility to implement virtually any configuration management solution that you deem appropriate for your environment. Think of base ClearCase as a feature-rich toolset and UCM as a process, based on recommended software development practices, that uses that toolset.

If your team will use UCM, you need to make the following planning decisions:

- Which components your team will work on?
- What stream hierarchy your project will use?
- Which development policies you will enforce?
- What naming scheme you will use for baselines?

If your team will use base ClearCase, you need to make the following planning decisions:

- What branching strategy your team will use?
- What configuration specification (config spec) rules developers will use to control the versions that are selected by their views?
- What label types and labels you will use to identify project milestones?

See Chapter 1 for an overview of the differences in configuring, working in, and managing software development projects between base ClearCase and UCM. However, for detailed information:

- See *Managing Software Projects* for information about creating and managing projects using UCM or base ClearCase, including the details about what you must consider for each before installing ClearCase.
- See *Developing Software* for information about how choosing UCM or base ClearCase affects how developers do their work in ClearCase.

Integrating with ClearQuest

In UCM, you can use ClearQuest to provide additional activity management features to your development environment, such as assigning activities, transitioning activities through states, and querying and reporting on activities based on state, user assignment, project, and so on. See *Managing Software Projects* for details about configuring ClearQuest and ClearCase to support UCM activity management.

If you are using base ClearCase functionality instead of UCM, you can associate ClearQuest change requests with ClearCase versions. See *Managing Software Projects* for details about configuring this integration.

Using ClearCase MultiSite

Before installing ClearCase MultiSite, you must resolve planning issues, such as which development artifacts to share across sites, how the various sites will access and change those artifacts, how to synchronize sites, and so on.

See *Administrator's Guide* for Rational ClearCase MultiSite for details about planning for and using ClearCase MultiSite.

ClearCase Site Preparation

The ClearCase administrator (possibly with advice from the ClearCase project manager) decides how to configure the ClearCase installation for the site. The administrator creates a shared release area from which users can install ClearCase on their individual computers. When creating the shared release area, the administrator specifies default values for installation parameters for the individual hosts based on the configuration decisions made for the site.

Check Hardware and Software Requirements

The *Installation Guide* contains information you need to know before installing ClearCase at your site:

- Supported platforms and file systems (including Windows/UNIX file access)
- Hardware and software requirements
- Platform-specific information pertaining to installation, such as disk space required, OS patches required, layered software packages required, and so on
- ClearCase and MultiSite patches incorporated into this release
- Issues with upgrading from a previous ClearCase release (both in general and pertaining to this particular release)
- Known issues pertaining to installation

Running ClearCase Site Preparation

The ClearCase Site Preparation program prepares the shared release area from which users can install ClearCase on their computers.

ClearCase Site Preparation configures the sitewide defaults that users see when installing ClearCase on their computers. This simplifies sitewide installation because the ClearCase administrator defines the ClearCase installation parameters for the site only once rather than relying on everyone making the appropriate choices when they install. It also simplifies installation, because users can accept the default values when prompted.

On Windows, running ClearCase Site Preparation requires at least local administrator privileges and often also requires network administrator privileges. On UNIX, running ClearCase Site Preparation requires root privileges.

See the *Installation Guide* for detailed information about running ClearCase Site Preparation.

Installing ClearCase Server Software

After running the ClearCase Site Preparation program, the administrator installs the ClearCase server software on one or more server host computers. ClearCase LT lets you install the server software on one host computer; ClearCase lets you install it on multiple host computers.

In addition to installing the ClearCase server software, the installation procedure creates some ClearCase objects (Table 2) and performs initial configuration to set up your team's development environment. On Windows, the Getting Started Wizard, launched at the end of the installation procedure, performs this work. On UNIX, the server setup part of the installation procedure performs it.

Table 2 ClearCase Objects Created by Installation Procedure

ClearCase Object	**UCM**	**Base ClearCase**
Project	X	
Project VOB (PVOB)	X	
Component VOB	X	
Administrative VOB		X
VOB		X

An *administrative VOB* contains global type objects, such as branch types, that are copied to client VOBs on an as-needed basis when users want to create instances of the type objects in the client VOBs.

The Getting Started Wizard performs the following additional tasks if ClearQuest is installed on the same computer:

- Creates a ClearQuest schema repository
- Creates a ClearQuest user database
- Configures the UCM-ClearQuest integration or the base ClearCase-ClearQuest integration

The Getting Started Wizard also lets you launch the Import Wizard, which imports files and directories that are not currently under ClearCase control into a VOB.

Installing ClearCase on Individual Computers

To install ClearCase on your computer, go to the release area created by your ClearCase administrator and run the ClearCase Installation program. (On Windows, this is **setup.exe**; on UNIX, it is **install_release**.)

Typically, you should accept the default installation parameters. See your ClearCase administrator before you override any default values.

See the *Installation Guide* for detailed information about installing ClearCase.

Working in ClearCase

3

This chapter describes the main tasks that project managers and developers perform and the tools that they use while working in Rational ClearCase. This chapter describes the UCM and base ClearCase environments on Windows and UNIX.

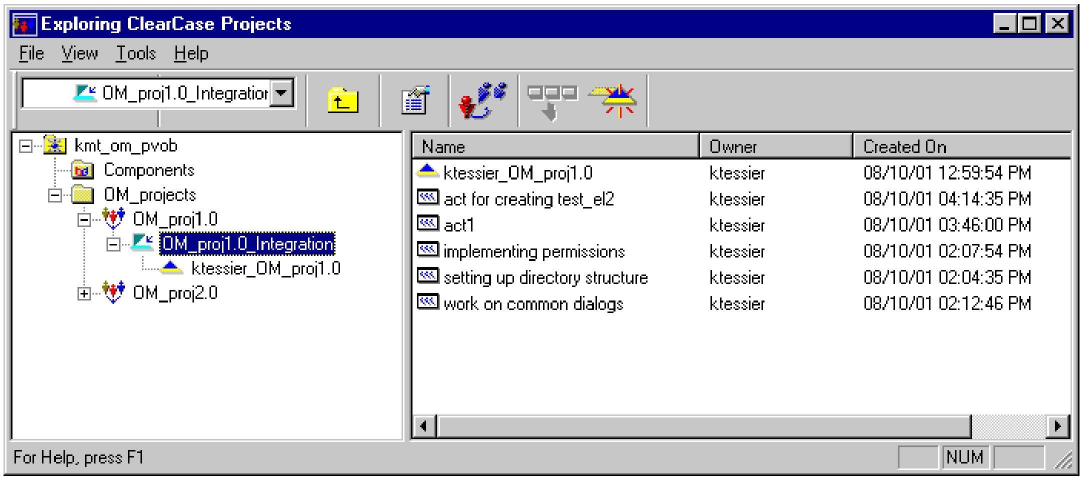
Project Managers

As project manager, you are responsible for creating and maintaining the development environment for your team. The tasks you perform and tools you use differ depending on whether you use UCM or base ClearCase.

Using UCM

The primary tool for managing UCM projects is the ClearCase Project Explorer, shown in Figure 22. The left pane provides a hierarchical view, starting with the PVOB, of UCM objects. The hierarchy displays components, folders, projects, and streams. The right pane displays the contents of the object that you highlight in the left pane. For example, Figure 22 displays the activities created in the integration stream.

Figure 22 ClearCase Project Explorer



Name	Owner	Created On
ktessier_OM_proj1.0	ktessier	08/10/01 12:59:54 PM
act for creating test_el2	ktessier	08/10/01 04:14:35 PM
act1	ktessier	08/10/01 03:46:00 PM
implementing permissions	ktessier	08/10/01 02:07:54 PM
setting up directory structure	ktessier	08/10/01 02:04:35 PM
work on common dialogs	ktessier	08/10/01 02:12:46 PM

The Project Explorer lets you perform the following tasks:

- Create projects, streams, and views
- Set development policies
- Make and recommend baselines
- View baseline histories and compare the contents of baselines
- Deliver to and rebase from streams in other projects

Using Base ClearCase

To manage a project in base ClearCase, you need to perform the following tasks:

- Define and implement a branching strategy
- Create a standard config spec for developers to include in their views
- Create label types and apply them to versions to identify milestones
- Merge versions from one branch to another branch

UNIX

On UNIX you can perform all of these tasks from the File Browser, which you invoke with the **xclearcase** command.

Windows

ClearCase on Windows provides several tools to help you perform these tasks:

- To create branch types and label types, use the Type Explorer.
- To create views, use the View Creation Wizard.
- To merge versions, use the Merge Manager.

To start these GUIs:

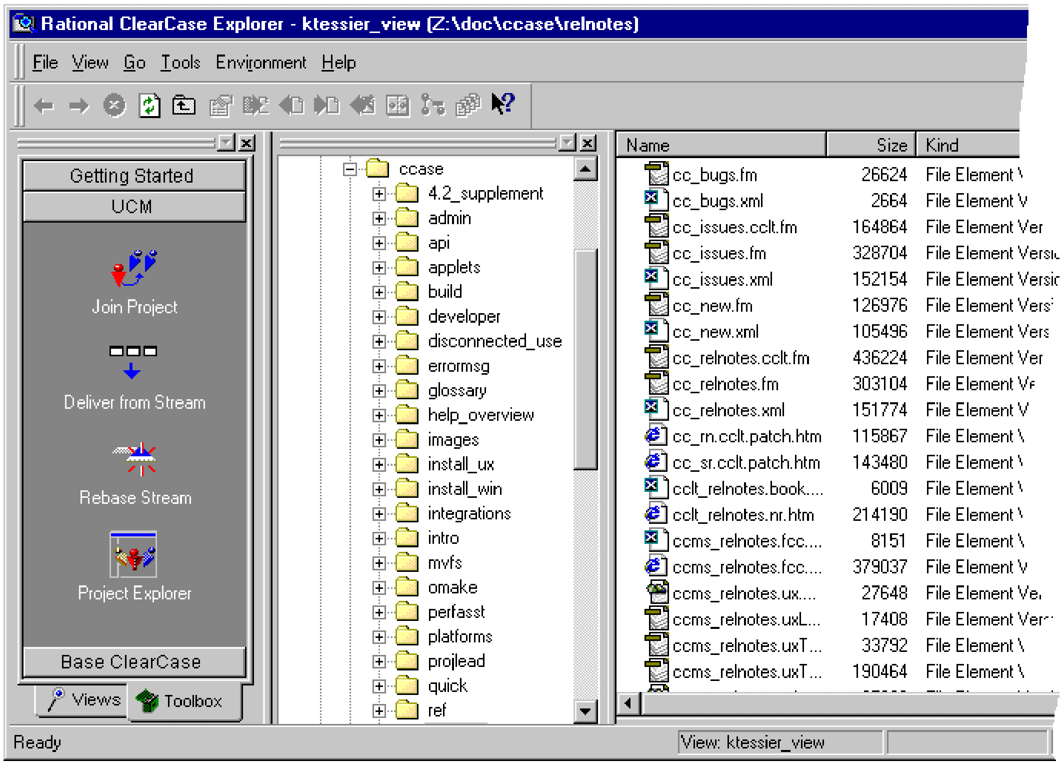
- In ClearCase, click **Start** > **Programs** > **Rational Software** > **Rational ClearCase** > *GUI name.*
- In ClearCase LT, click **Start** > **Programs** > **Rational Software** > **Rational ClearCase LT** > *GUI name.*

Developers Working on Windows

As a developer, you are responsible for setting up your work environment, creating and modifying files, and sharing those files with other members of your project team. The tasks you perform differ slightly depending on whether you use UCM or base

ClearCase; however, the tool is the same. ClearCase Explorer, as shown in Figure 23, is the primary tool for developers working in UCM or base ClearCase on Windows.

Figure 23 ClearCase Explorer



ccase
- 4.2_supplement
- admin
- api
- applets
- build
- developer
- disconnected_use
- errormsg
- glossary
- help_overview
- images
- install_ux
- install_win
- integrations
- intro
- mvfs
- omake
- perfasst
- platforms
- projlead
- quick
- ref

Name	Size	Kind
cc_bugs.fm	26624	File Element \
cc_bugs.xml	2664	File Element V
cc_issues.cclt.fm	164864	File Element Ver
cc_issues.fm	328704	File Element Versi
cc_issues.xml	152154	File Element Versi
cc_new.fm	126976	File Element Vers
cc_new.xml	105496	File Element Vers
cc_relnotes.cclt.fm	436224	File Element Ver
cc_relnotes.fm	303104	File Element Ve
cc_relnotes.xml	151774	File Element V
cc_rn.cclt.patch.htm	115867	File Element \
cc_sr.cclt.patch.htm	143480	File Element \
cclt_relnotes.book....	6009	File Element \
cclt_relnotes.nr.htm	214190	File Element \
ccms_relnotes.fcc....	8151	File Element \
ccms_relnotes.fcc....	379037	File Element V
ccms_relnotes.ux....	27648	File Element Ve
ccms_relnotes.uxL...	17408	File Element Ver
ccms_relnotes.uxT...	33792	File Element \
ccms_relnotes.uxT...	190464	File Element \

The ClearCase Explorer contains three main panes. The Shortcut pane, on the left, contains shortcuts for UCM and base ClearCase operations and your views. The **UCM** tab contains shortcuts for the following operations:

- Joining a project
- Delivering work to another stream
- Rebasing your stream to a new baseline
- Starting the Project Explorer

The **Base ClearCase** tab contains shortcuts for the following operations:

- Merging versions of an element
- Creating, starting, and removing views
- Updating snapshot views
- Editing a view’s properties
- Starting the ClearCase Report Builder

After you set a view context, the Folder pane, in the center, lets you navigate the hierarchy of folders within VOBs. The Details pane, on the right, displays the versions of elements within the folder you select. If you are using UCM, the Folder pane also contains folders for your activities.

Developers Working on UNIX

As a developer, you are responsible for setting up your work environment, creating and modifying files, and sharing those files with other members of your project team. The tasks you perform differ slightly depending on whether you use UCM or base ClearCase; however, the tool is the same.

The ClearCase File Browser is the GUI that provides you with access to most ClearCase operations. You can start the File Browser with the **xclearcase** command.

Working in UCM

The File Browser's **Project** menu contains selections for the following UCM developer operations:

- Joining a project
- Delivering work to another stream
- Rebasing your stream to a new baseline
- Starting the Project Explorer

Working in Base ClearCase

The File Browser's **View** menu contains selections for creating, setting, and updating your view.

The **Versions** menu contains selections for checking in and checking out files, comparing versions of elements, and merging versions.

Using cleartool

In addition to providing GUIs, ClearCase has a robust command-line interface utility, **cleartool**. Table 3 lists some of the commands for performing common operations. See the *Command Reference* for the full set of **cleartool** commands.

Table 3 Frequently Used cleartool Commands

cleartool Command	Description
checkin	Creates a new version of an element.
checkout	Creates a modifiable copy of a version.
deliver	Merges changes from one stream to another stream.
merge	Combines the contents of two or more versions of files or directories into a new version.
mkactivity	Creates a UCM activity.
mkbl	Creates a baseline.
mkbrtype, mkbranch	Creates a branch type, creates a branch in an element's version tree.
mkelem	Creates a file or directory element.
mklbtype, mklabel	Creates a label type, applies label to versions of elements.
mkproject	Creates a UCM project.
mkstream	Creates a UCM stream.
mkview	Creates a view.
rebase	Updates your stream's configuration with a new set of foundation baselines.

Working in an IDE

ClearCase supports integrations with a variety of interactive development environments (IDEs). This section describes integrations with two IDEs: Microsoft VS.NET and IBM WebSphere Studio Application Developer

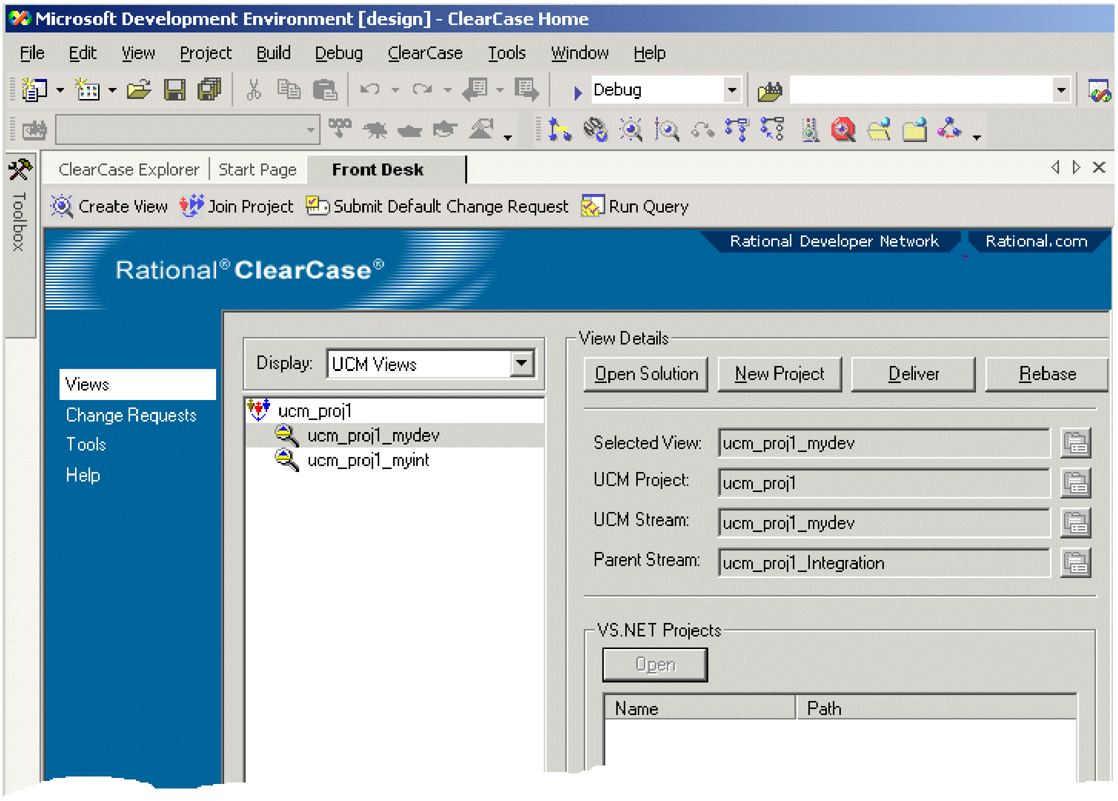
Microsoft VS.NET

The ClearCase integration with VS.NET lets developers perform all their development tasks from within one integrated GUI. Figure 24 shows the ClearCase **Front Desk** tab of the VS.NET GUI. Developers start by adding views to the VS.NET integration and then placing their VS.NET projects under ClearCase source control.

The **Front Desk** tab lets developers set their base ClearCase or UCM view context. In Figure 24, the developer has chosen to work in UCM and has navigated to the **ucm_proj1_mydev** stream within the **ucm_proj1** project. The **View Details** area displays the view attached to the selected stream.

After setting a view context, developers can bring up the ClearCase Explorer within the VS.NET GUI and perform ClearCase operations, such as checking out and checking in files and selecting UCM activities to work on.

Figure 24 ClearCase Integration with VS.NET



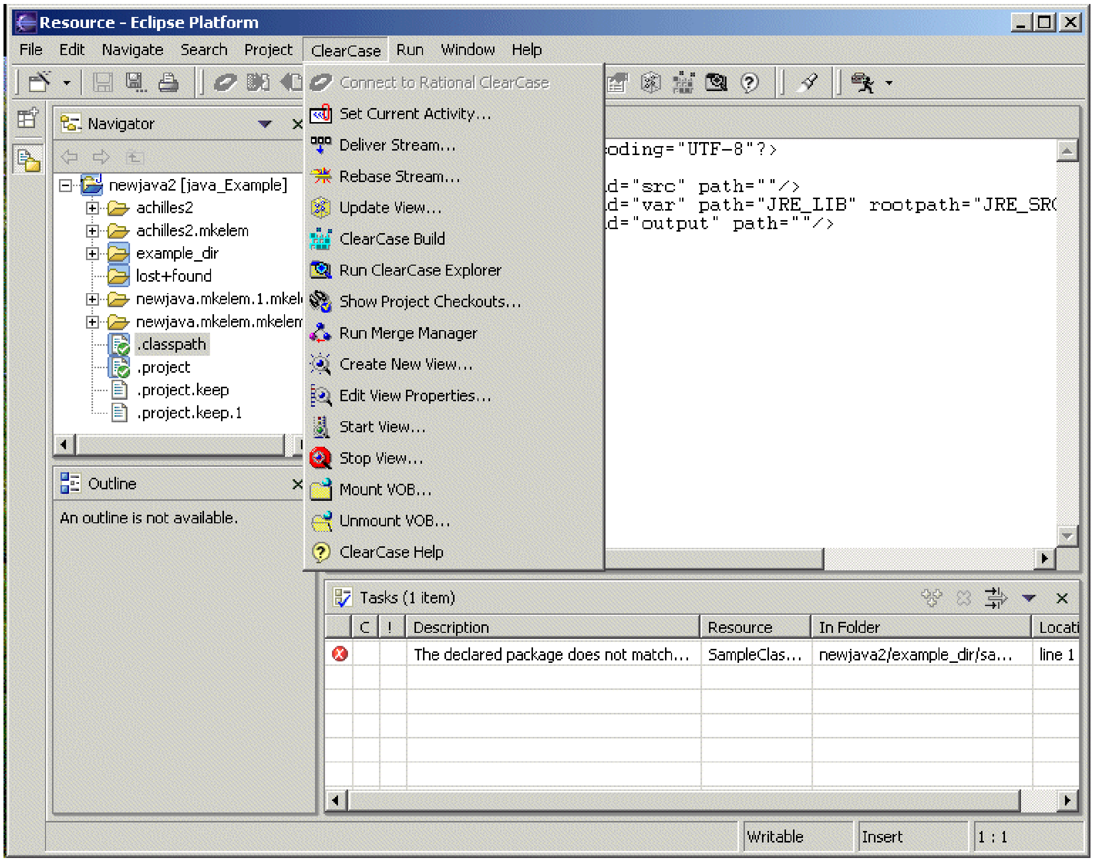
Selected View: ucm_proj1_mydev
UCM Project: ucm_proj1
UCM Stream: ucm_proj1_mydev
Parent Stream: ucm_proj1_Integration

IBM WebSphere Studio Application Developer

The ClearCase integration with Application Developer lets developers access most ClearCase operations from within the Application Developer GUI. As shown in Figure 25, the integration adds a **ClearCase** menu and toolbar icons to the Application Developer. Developers set up their integrated environments by associating their ClearCase views with Application Developer workspaces.

Developers can then perform checkin, checkout, and other ClearCase operations by clicking a toolbar icon or a context menu option or by starting ClearCase Explorer. Icons in the **Navigator** pane indicate which files are under ClearCase source control and whether they are checked out.

Figure 25 ClearCase Integration with Application Developer



```
oding="UTF-8"?>

d="src" path=""/>
d="var" path="JRE_LIB" rootpath="JRE_SR(
d="output" path=""/>
```

	C	!	Description	Resource	In Folder	Locati
⊗			The declared package does not match...	SampleClas...	newjava2/example_dir/sa...	line 1

Using the ClearCase Web Interface

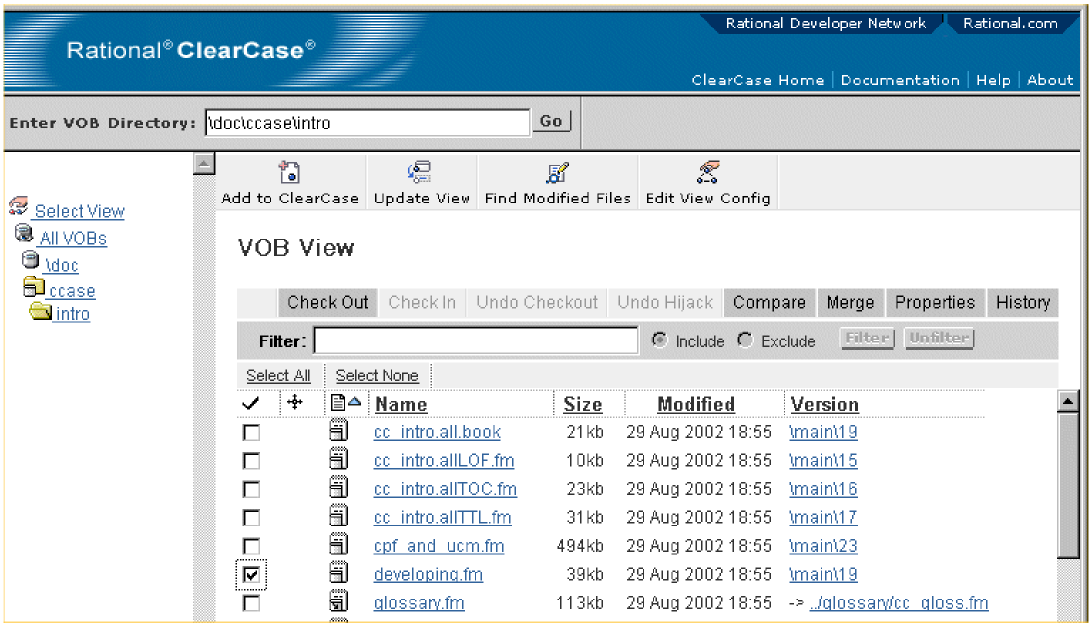
In addition to its native user interfaces, ClearCase provides a Web interface, shown in Figure 26. Developers can access ClearCase VOBs and perform the following operations through the Web interface:

- Create views
- Add files to source control
- Check out and check in files

- Compare and merge versions
- View properties and histories of elements
- Join UCM projects
- Deliver work from streams
- Rebase their streams to new recommended baselines

To use the Web interface, developers need only a Web Browser; they do not need to have ClearCase installed on their computers. For this reason, the Web interface is very useful for developers who work remotely or who need to access ClearCase from a computer that does not have ClearCase installed.

Figure 26 ClearCase Web Interface



Enter VOB Directory: \doc\ccase\intro

Filter: [✓ Include] Exclude

✓	Name	Size	Modified	Version
☐	cc_intro.all.book	21kb	29 Aug 2002 18:55	\main\19
☐	cc_intro.allLOF.fm	10kb	29 Aug 2002 18:55	\main\15
☐	cc_intro.allTOC.fm	23kb	29 Aug 2002 18:55	\main\16
☐	cc_intro.allTTL.fm	31kb	29 Aug 2002 18:55	\main\17
☐	cpf_and_ucm.fm	494kb	29 Aug 2002 18:55	\main\23
☑	developing.fm	39kb	29 Aug 2002 18:55	\main\19
☐	glossary.fm	113kb	29 Aug 2002 18:55	-> ../glossary/cc_gloss.fm

Glossary

ACTIVITY. A ClearCase UCM object that tracks the work required to complete a development task. An activity includes a text headline, which describes the task, and a change set, which identifies all versions that developers create or modify while working on the activity. When working on a version, developers must associate that version with an activity. If your project is configured to use the UCM-ClearQuest integration, a corresponding ClearQuest record stores additional activity information, such as the state and owner of the activity.

ADMINISTRATIVE VOB. A VOB that contains global type objects. Local copies of global type objects can be created in any VOB that has an **AdminVOB** hyperlink to the administrative VOB that defines the global type object. See also *auto-make-type*, *global type*, *local copy.*

BASELINE. A ClearCase UCM object that typically represents a stable configuration for one or more components. A baseline identifies activities and one version of every element visible in one or more components. Developers can create a *development stream* or *rebase* an existing development stream from a baseline.

BRANCH. An *object* that specifies a linear sequence of *versions* of an *element*. The entire set of versions of an element is called a *version tree*; it always has a single *main branch* and may also have subbranches. Each branch is an instance of a *branch type* object.

BUILD. The process during which a ClearCase build program (**clearmake**, **clearaudit**, or **omake**) produces one or more *derived objects*. This may involve actual translation of source files and construction of binary files by compilers, linkers, text formatters, and so on. A system build consists of a combination of actual *target rebuilds* and *build avoidance*. See also *express build*.

BUILD AVOIDANCE. The ability of a ClearCase build program to fulfill a build request by using an existing derived object instead of creating a new one by executing a build script. The build program can reuse a derived object currently in the view or *wink in* a derived object that exists in another view. The process by which the build program decides how to produce a derived object is called *configuration lookup*.

CHANGE SET. A list of related versions associated with a UCM *activity.* ClearCase records the versions that developers create while working on an activity. An activity uses a change set to record the versions of files that are delivered, integrated, and released together.

CHECKOUT/CHECKIN. The two-part process that extends a *branch* of an *element*'s *version tree* with a new *version*. The first part of the process, *checkout*, expresses your intent to create a new version at the current end of a particular branch. (This is sometimes called checking out a branch.) The second part, *checkin*, completes the process by creating the new version.

For file elements, the checkout process creates an editable version of the file in the *view* with the same contents as the version at the end of the *branch*. Typically, a user edits this file, then checks it back in.

For *directory elements*, the checkout process allows file elements, subdirectory elements, and VOB symbolic links to be created, renamed, moved, and deleted.

Performing a checkout of a branch does not necessarily guarantee you the right to perform a subsequent checkin. Many users can check out the same branch, as long as they are working in different views. At most one of these can be a *reserved* checkout, which guarantees the user's right to check in a new version. An *unreserved* checkout does not. If several users have unreserved checkouts on the same branch in different views, the first user to check in creates the next version.

CLEARCASE ADMINISTRATORS GROUP. (Windows platforms only) A special group, usually created in the Windows NT domain when ClearCase is installed. Only ClearCase administrative accounts and the login account for the ALBD Service should be members of this group.

CLEARCASE REGISTRY. A set of files on the *registry server host* that map logical VOB and view names (*VOB tags* and *view tags*) to physical storage locations (*VOB storage directories* and *view storage directories*).

COMPONENT. A ClearCase object that you use to group a set of related directory and file elements within a UCM *project*. Typically, the elements that make up a component are developed, integrated, and released together. A project must contain at least one component, and it can contain multiple components. Projects can share components.

CONFIG SPEC. A set of rules that specify which versions of VOB elements a view selects. The config spec for a *snapshot view* also specifies which elements to load into the view. See also *scope*, *version selector*, *version-selection rule*, and *load rule*.

DELIVER. A ClearCase operation in which developers merge the work from their own *development streams* to the project's *integration stream* or to a feature-specific development stream. If required, the deliver operation invokes the Merge Manager to merge versions.

DERIVED OBJECT (DO). An MVFS file produced by a **clearmake** or **omake** build or a **clearaudit** session. Each derived object is associated with the *configuration record* that is created by the ClearCase build program to document the build. A shareable DO can be winked in by other views; a nonshareable DO cannot be winked in unless you explicitly make it available.

DEVELOPMENT STREAM. A ClearCase UCM object that determines which versions of elements appear in a *development view* and maintains a list of a developer's activities. The development stream configures the development view to select the versions associated with the foundation baselines plus any activities and versions that developers create after they join the project or rebase their development stream.

DYNAMIC VIEW. A *view* that is always current with the VOB (as specified by the *config spec*). Dynamic views use the *MVFS* to create and maintain a directory tree that contains *versions* of VOB *elements* and *view-private files*. Dynamic views are not supported on all ClearCase platforms.

ECLIPSED. A VOB object that is not visible because another object with the same name is currently selected by the view.

ELEMENT. An *object* that encompasses a set of *versions*, organized into a *version tree*.

ELEMENT TYPE. A class of versioned file or directory objects. ClearCase supports predefined element types. Users can define additional types that are refinements of the predefined types. When an *element* is created, it is assigned one of the currently defined element types in its VOB. Each user-defined element type is implemented as a separate VOB object.

EXTENDED NAMESPACE. The ClearCase extension of the standard Windows or UNIX pathname hierarchy. Each host has a view-extended namespace, which allows a pathname to access VOB data using any view that is active on that host. Each VOB has a *VOB-extended namespace*, which allows a pathname to access any version of any element, independently of (and overriding) version selection by views. *Derived objects* also have extended pathnames, which include *DO IDs*.

FIRE A TRIGGER. The process by which ClearCase verifies that the conditions defined in a *trigger* are satisfied and causes the associated trigger actions to be performed.

FOUNDATION BASELINE. A property of a stream. Foundation baselines specify the versions and activities that appear in your view. As part of a *rebase* operation, foundation baselines of the target stream are replaced with the set of recommended baselines from the source stream.

HIJACKED FILE. A version in a *snapshot view* that is modified but not checked out. By default, a non-checked-out version in a snapshot view is given the file attribute of read-only. If you change this attribute and modify the file, you have hijacked the file by taking it out of direct ClearCase control.

HISTORY. Metadata in a *VOB*, consisting of *event records* for that VOB's *objects*. The history of a file element includes the creation event of the element itself, the creation event of each version of the file, the creation event of each branch, the attributes assigned to the element and/or its versions, the hyperlinks attached to the element and/or its versions, and so on.

LABEL. An instance of a label type object, supplying a user-defined name for a version. See also *object*, *metadata*.

LABEL TYPE. A *type object* that defines a version label for use within a VOB.

LOAD. To copy a version of an element to a snapshot view and keep track of the checkins, updates, and other ClearCase operations that affect the element.

LOAD RULE. A statement in the *config spec* that specifies an element or subtree to *load* into a *snapshot view*. Config specs can have more than one load rule. See also *version-selection rule*.

LOST+FOUND. A subdirectory of a VOB's top-level directory, to which elements are moved if they are no longer cataloged in any version of any directory element.

MAIN BRANCH. The starting branch of an element's *version tree*. The default name for this branch is **main**.

MASTER REPLICA. (MultiSite) The master replica of a ClearCase object is the only replica at which the object can be modified or instances of the object can be created.

MASTERSHIP. (MultiSite) The ability to modify an object or to create instances of a type object.

METADATA. The data associated with an *object* that supplements its file system data. Some of this data is created by users; some of it is created during ClearCase operations on the object.

MULTIVERSION FILE SYSTEM (MVFS). A directory tree that, when activated (mounted as a file system of type MVFS), implements a VOB. To standard operating system commands, a VOB appears to contain a directory hierarchy; ClearCase commands can also access the VOB's metadata. Also, *MVFS file system* refers to a file system extension to the operating system, which provides access to VOB data. The MVFS file system is not supported on all ClearCase platforms.

OBJECT. An item stored in a VOB. An object can be identified by an object-selector string, which includes a prefix that indicates the kind of object, the object's name, and a suffix that indicates the VOB in which the object resides. Examples: `lbtype:REL1@/vobs/vega` on UNIX and `lbtype:REL1@\vega` on Windows

OBJECT REGISTRY. A networkwide database that records the storage locations of all VOB storage directories and all view storage directories. The **mktag**, **rmtag**, **mkview**, **rmview**, **mkvob**, **rmvob**, **register**, and **unregister** commands add, delete, or modify registry file entries.

ORPHANED ELEMENT. An element that is no longer cataloged in any version of any directory. Such elements are moved to the VOB's `lost+found` directory.

PROJECT. A ClearCase UCM object that contains the configuration information needed to manage a significant development effort, such as a product release. A project includes one *integration stream*, which configures views that select the latest versions of the project's shared elements, and typically multiple *development streams*, which configure views that allow developers to work in isolation from the rest of the project team.

PROJECT VOB (PVOB). A VOB that stores UCM objects, such as projects, streams, activities, and change sets. Every UCM *project* must have a PVOB. Multiple projects can share the same PVOB.

REBASE. A ClearCase operation that makes a development work area current with the set of versions represented by a more recent *baseline* in another stream, usually the project's *integration stream* or a feature-specific development stream.

REPLICA. (MultiSite) An instance of a VOB, located at a particular *site*. A replica consists of the VOB's database, along with all of the VOB's data containers.

SCRUBBING. The removal of objects that are no longer used to free storage space:

- The **scrubber** utility discards *data container* files from *cleartext pools* and *derived object storage pools*.
- The **vob_scrubber** utility discards *event records* and MultiSite *oplog entries* from a *VOB database*.
- The **view_scrubber** utility removes *derived object* containers from the *view storage directory*.

SNAPSHOT VIEW. A *view* that contains copies of ClearCase *elements* and other file system objects in a directory tree. You use the Update Tool to keep the view current with the VOB (as specified by the *config spec*).

STREAM. A ClearCase UCM object that determines which versions of elements appear in any view configured by that stream. Streams maintain a list of baselines and activities. A project contains one *integration stream* and typically multiple *development streams*.

TRIGGER. A monitor that specifies one or more standard programs or built-in actions to be executed whenever a certain ClearCase operation is performed. See also *preoperation trigger*, *postoperation trigger*, *trigger type*.

TYPE. An object that defines a ClearCase data structure. Users can create instances of these structures: metadata annotations are placed on objects by creating instances of label types, attribute types, and hyperlink types. Each file and directory is an instance of an element type; each branch is an instance of a branch type.

UNIFIED CHANGE MANAGEMENT (UCM). A process, layered on base ClearCase and ClearQuest functionality, for organizing software development teams and their work products. Members of a project team use *activities* and *components* to organize their work.

VERSION. An *object* that implements a particular revision of an *element*. The versions of an element are organized into a *version tree* structure. Also: checked-out version can refer to the *view-private file* that corresponds to the object created in a VOB database by the **checkout** command.

VERSION TREE. The hierarchical structure in which all the *versions* of an *element* are (logically) organized. The version tree display also shows *merge* operations.

VIEW. A ClearCase object that provides a work area for one or more users. For each *element* in a *VOB*, a view's *config spec* selects one version from the element's *version* *tree*. Each view can also store *view-private files* and *view-private directories*, which do not appear in other views. There are two kinds of views: *snapshot views* and *dynamic views*.

VIEW-PRIVATE OBJECT. A file or directory that exists only in a particular view. View-private objects are not version controlled.

VOB (VERSIONED OBJECT BASE). A repository that stores *versions* of file elements, *directory elements*, *derived objects*, and *metadata* associated with these objects. With MultiSite, a VOB can have multiple *replicas*, at different *sites*.

VOB DATABASE. The part of a *VOB storage directory* in which ClearCase *metadata* and VOB objects are stored. This area is managed by the database management software embedded in ClearCase. The actual file system data is stored in the VOB's *storage pools*.

VOB FAMILY. (MultiSite) The set of all *replicas* of a particular VOB. All the replicas share the same VOB family UUID; each replica has its own VOB replica UUID.

WINK IN. 1) To cause a shareable derived object to appear in a view, even though its file system data is actually located in a VOB's *derived object storage pool*. 2) To convert a nonshareable derived object to a shared derived object.