

**Chapter
2**

**ClearCase
Concepts**

*Get on the
Fast Track!*

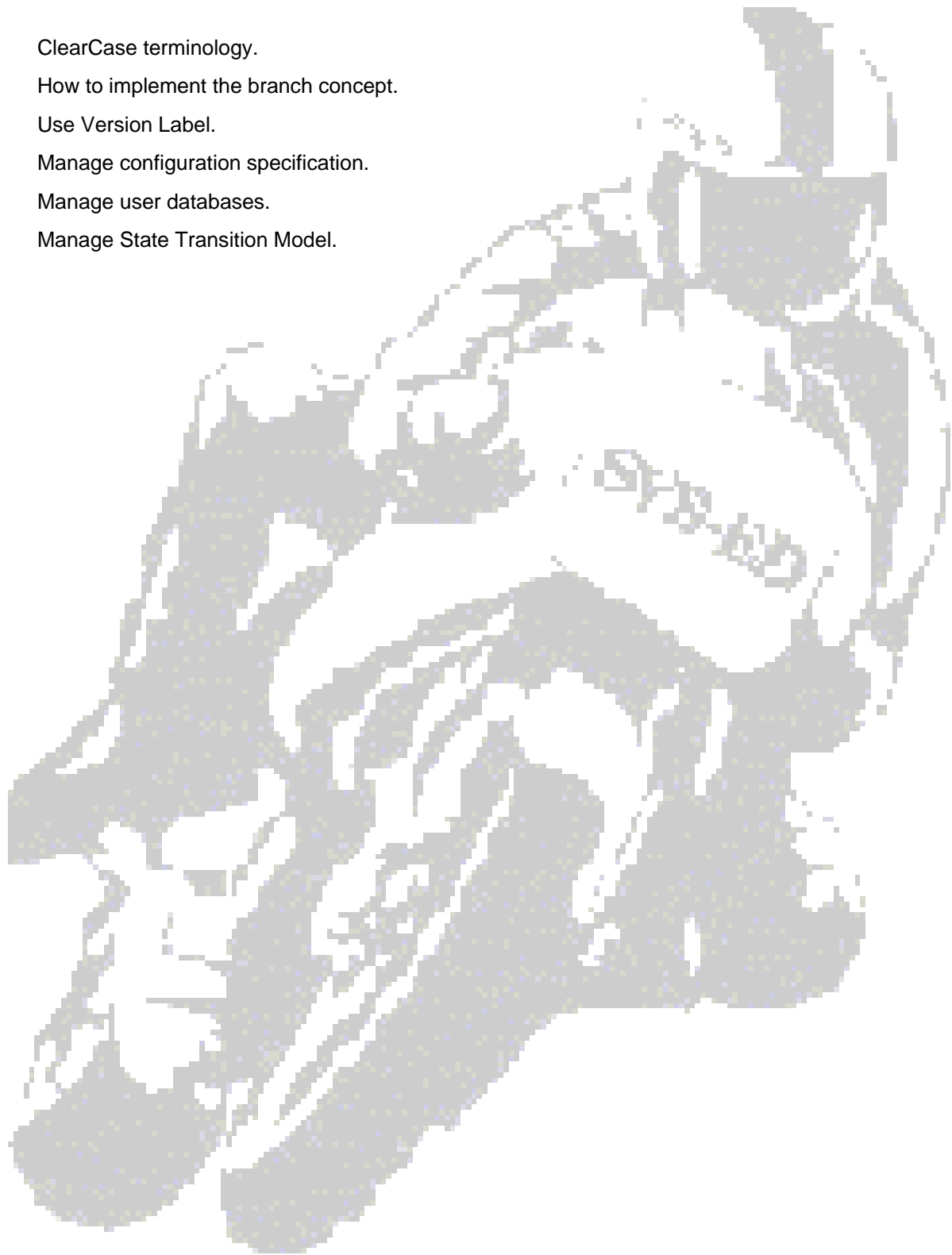


**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- ClearCase terminology.
- How to implement the branch concept.
- Use Version Label.
- Manage configuration specification.
- Manage user databases.
- Manage State Transition Model.



1 ClearCase Terminology

The following terms are central to utilizing the features of ClearCase.

- Element
- Version
- VOB: Versioned object base
- View
- Checkout model

1.1 Element

Elements are control objects that include a version tree.

ClearCase distinguishes between two types of elements:

Element	Description
File	Any file that can be stored in a file system
Directory	Contains file elements and other directory elements.

1.2 Version

A version is a specific revision of an element. Versioning makes copies of the data at a meaningful point in order to be able to return to that point at a later stage.

1.3 VOB: Versioned Object Base

A VOB: versioned object base is a repository that stores versions of file elements, directory elements, derived objects, and metadata associated with these objects.

- A VOB database is 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 VOB storage pools.
- VOBs support the notion of triggers, which are attached to elements and specify one or more standard programs or built-in actions to be executed whenever a certain ClearCase operation is performed on the element.
- A PVOB - Project VOB is used when UCM is implemented as the software configuration management process. Every UCM project belongs to a PVOB and multiple UCM projects can share a PVOB.
- In ClearCase MultiSite, a VOB family is the set of all replicas of a particular VOB.
- In ClearCase MultiSite, a VOB can have multiple replicas¹, at different sites.

1.4 Views

A view is represented as a directory; it provides access to a specific version of one or more elements in a VOB. Utilizing a simple rule-based approach, a view selects a set of versions of elements without having to specify the versions explicitly.

ClearCase offers two view concepts:

<p>Snapshot</p>	<p>Workplace created on the local computer which is created by copying versions of elements from VOBs to the computer.</p> <p>With snapshot views, it will be necessary to update the view periodically in order to be able to see the latest versions of elements. An update operation copies the latest versions of elements from the VOB to the local snapshot view.</p>
<p>Dynamic</p>	<p>Provides immediate, transparent access to data stored in VOBs, based on configuration specification selection rules. When working in a dynamic view, data does not have to be copied from VOBs to the view; the latest versions of elements can be seen.</p> <p>Dynamic views are accessed using the MVFS: multiversion file system, a ClearCase provided file system driver. They are not available in ClearCase LT.</p>

1.5 Checkout Model

A checkout model provides the capability for a private and editable copy of a specific element, and manage changes to projects. When an element is checked out, ClearCase creates an editable copy in the view. When an element is checked in, a new version of it is added to the VOB.

Some versioning tools use a checkout model where the checked out element is locked during the checkout; this is known as a reserved checkout model. This model works fine in serial development, where the likelihood of more than one person checking out a specific element at the same time is very small.

ClearCase supports concurrent development and provides a checkout model which allows for more than one person to check out a specific element at the same time.

A checkout model can be either optimistic or non-optimistic:

Model	Description
Optimistic	A first-come-first-served policy is used, and the order of checkout is irrelevant.
Non-optimistic	The order of checkout determines the order of checkin. Whoever checked out the specific element first, gets a reservation for first checkin.

ClearCase provides non-optimistic checkout by default, but can be configured to be optimistic per file or by default.

2 Branch

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.

Branching is done in order to provide work areas for development. It is not uncommon to have dozens of branches active at a time. The ClearCase naming convention for branches is to use lower case alphabetic names.

Since branching is done on a very low level, it also becomes very important to find a branching strategy that works from all level of perspectives and abstractions.

There are some general rules that applies for ClearCase:

- The main branch should not be used for development work, but should be reserved for major milestones or released versions.
- All major development is done on branches.

ClearCase branches can be used non-exclusively for various purposes:

- Physical-for components and subsystems.
- Functional-for patches, fixes, releases, enhancements, and features.
- Environmental-for different operating systems, platforms, and tools.
- Organizational-for teams, programs, projects, work activities, and roles.
- Procedural-for processes, policies, and states.

Each branch in ClearCase is an instance of a branch type object. This affords the capability to attach attributes to the branch types, which can then be used to mark the purpose of the specific branch, and ensure that the right type of branches are being used.

3 Version Label

A version label can be attached to any version of an element to identify that version.

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 starting point of a branch.
- A label type is an object that defines a version label for use within a VOB.
- Labels can be queried, and are useful in tracking which version of an element goes with versions of other elements.
- Labels can only be attached to an element version, and by default, only one instance of a label type may be attached to any version of an element tree.
- A label type can be defined in order that one instance of a label type can appear once on each branch of a tree.

Labels work best as a snapshot of the system; they should be more or less static in nature. Once attached to a particular object, they should not be moved.

3.1 Drawbacks: Multiple Use of the Same Label Version

There are drawbacks to using the same version label several times in the same element tree.

- It is potentially confusing.
- In a version-extended pathname, a full branch pathname must be included along with the version label, which overrides the whole purpose of using a label in the first place.

4 Configuration Specification

A configuration specification, or config spec, contains the rules used by a view to select versions of elements.

The rules are flexible, and various specifiers can be used for indicating which versions to select.

A view has exactly one config spec.

4.1 Element

Elements are control objects that include a version tree.

ClearCase distinguishes between two types of elements:

Element	Description
File	Any file that can be stored in a file system.
Directory	Contains file elements and other directory elements.

A version is a specific revision of an element. By versioning, copies are made of the data at a meaningful point in order to be able to return to that point at a later stage, if necessary.

A view, which is represented as a directory, provides access to a specific version of one or more elements in a VOB. With a simple rule-based approach, a view selects a set of versions of elements without having to specify the versions explicitly.

4.2 Schemas

A ClearQuest schema is a complete description of the process models for all the components of a user database.

The descriptions include:

- States and actions of the model.
- Structure of the data that can be stored about the individual component.
- Hook code or scripts that can be used to implement business rules.
- Forms and reports used to view and input information about the component.

ClearQuest provides out-of-the-box schemas that can be customized for a client installation.

Schema Repositories

ClearQuest stores schemas in a special type of database called a schema repository, which is also sometimes referred to as a master repository or a master database.

A schema repository can store multiple schemas.

5 User Databases

A user database in ClearQuest is a collection of user data for one process model. Each user database is associated with a specific version of a specific schema.

The schema defines the way the data is stored and changed in that database. The database is an instance of a version of a schema. Users change data in the databases when they add or modify information about change requests. These changes have no effect on the schema.

Databases contain a record for each change request. As the change request moves through its lifecycle, the data stored in this record changes accordingly.

5.1 Database Sets and Connections

A database set (dbset) is one specific schema repository and all of its associated user databases.

A connection is the set of credentials that allow access to the database set.

6 State Transition Model

A state transition model, is a systematic representation of the possible steps in one change request lifecycle.

The state transition model represents a change management process model. The model defines the change request lifecycle in terms of the states that the change request passes through, the actions that are taken to move between the states, and the rules that define when and how the actions can be taken.

The state of a change request is its current status. Typical states include submitted, assigned, opened, postponed, duplicated, resolved, and closed.

An action is an activity that moves a change request from one state to another. Typical actions include assign, reject, open, postpone, duplicate, validate, resolve, and close.

Rules define when and how an action can be taken.