



WebSphere Education



Collaboration support for version control

Unit 13

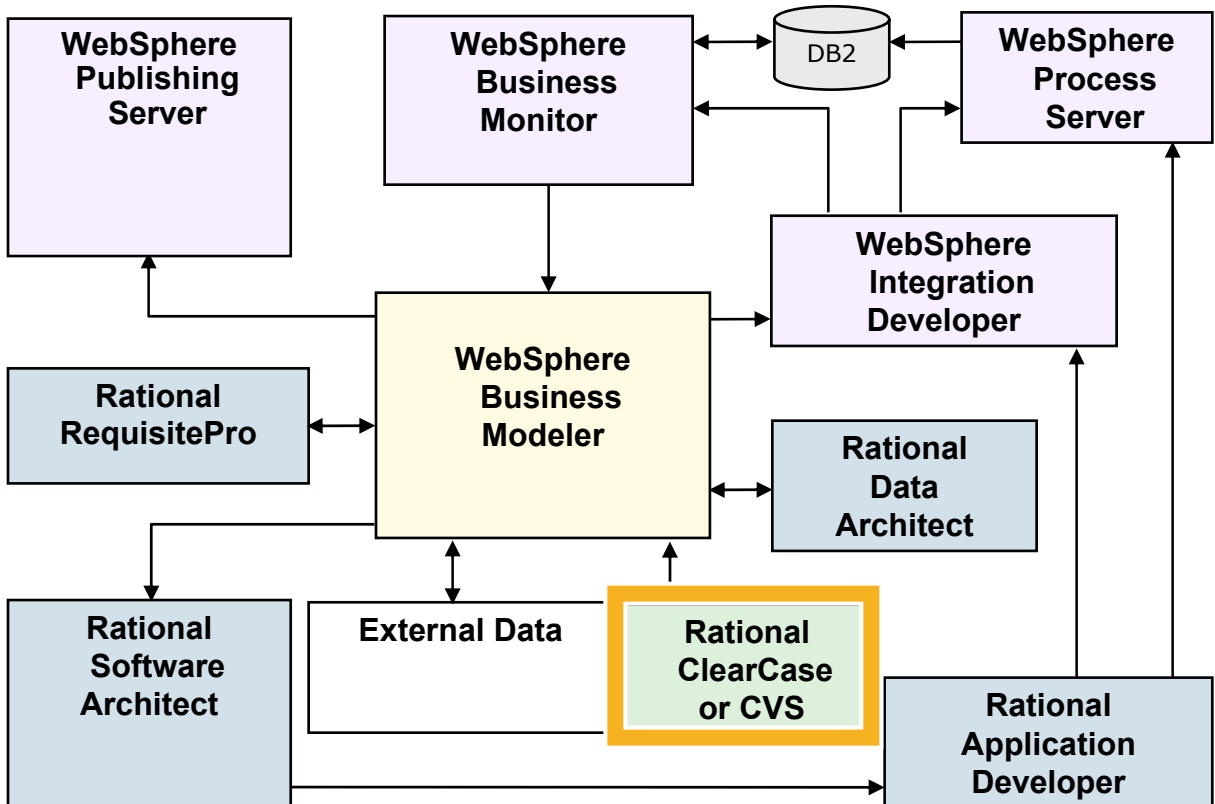


Unit objectives

After completing this unit, you should be able to:

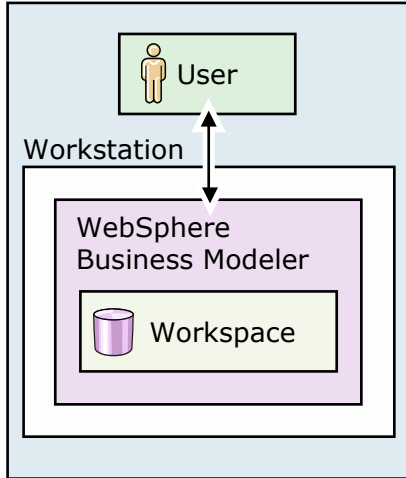
- Explain the need for project versioning
- Describe Concurrent Versions System (CVS)
- Describe IBM Rational ClearCase
- List project versioning steps
- Explain the importance of version control
- Describe development using project versioning
- Explain best practices and deletion strategy

Model versions are maintained by another product

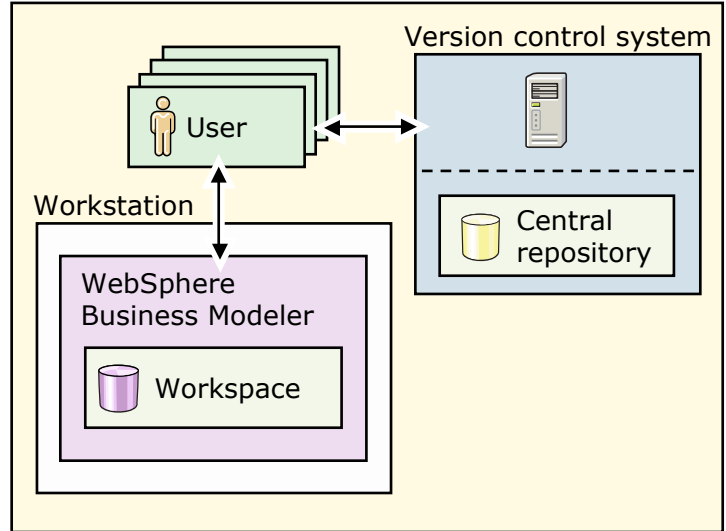


Environments for modeling

Single-user environment



Multiple-user environment

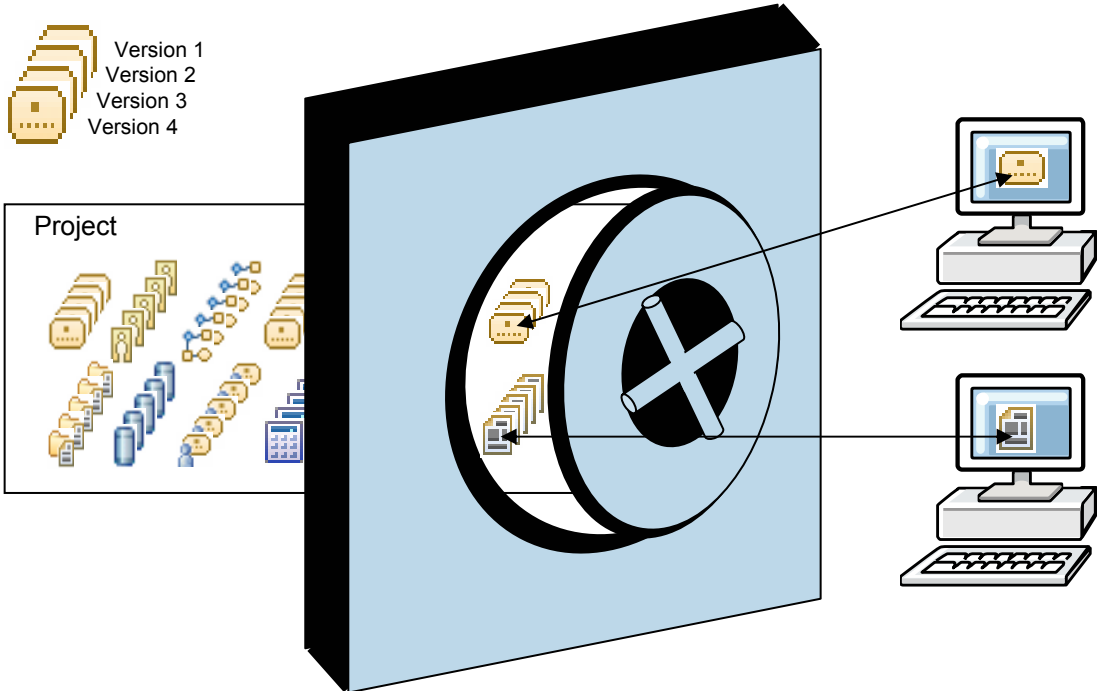


Need for project versioning

- To distribute the effort of modeling or modifying an entire project among multiple team members.
 - Members can view and post project artifacts to a version control system.
 - Check out processes and create their local versions.
 - Submit their changes back when done
- Modeler can use IBM ClearCase or Concurrent Versions System (CVS) to access a version control repository on a server.
 - Maintain secured version control of project data in the repository
 - Each modification of a project element (such as process, resource, or catalog) stored as a distinct version of the original item
 - Post business modeling projects to share, view, make copies of the shared projects, and save the copies to their local machine
 - View the history of project element modifications
 - Compare two or more versions of the same item

Controlling elements as if they were in a vault

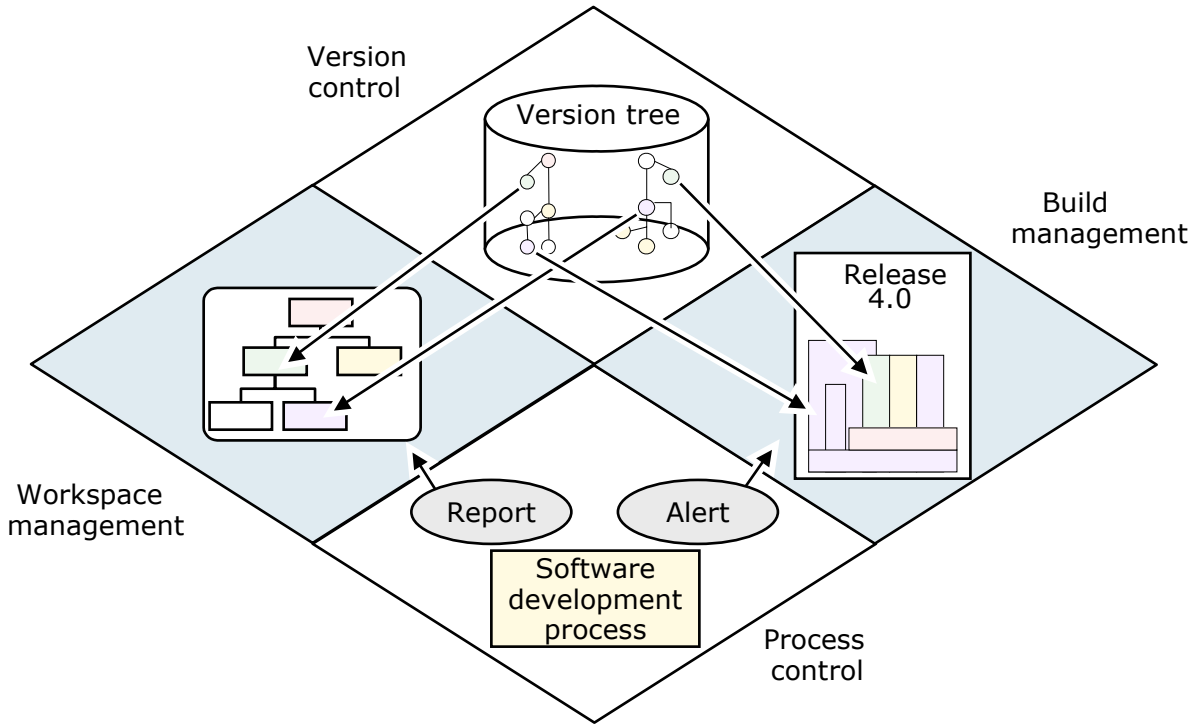
- A version control system securely manages all the elements in a model individually, not as one big document.



Concurrent Versions System (CVS)

- Open-source version control system primarily used for development
 - Keeps track of files and changes
 - Allows collaboration among developers in different geographic locations
 - Used to address code versioning issues
- Most commonly used code versioning and storage system today
- Common uses
 - Enable collaboration on medium to large projects distributed across geographies
 - Provide court evidence to demonstrate original work
 - Store source code version history

IBM Rational ClearCase (1 of 2)



Software configuration management

IBM Rational ClearCase (2 of 2)

- The ClearCase design:
 - Creates versions of all types of files and directories
 - Records and reports the actions, history, and milestones
 - Reproduces accurately every release
 - Traces and reproduces builds
 - Assures the integrity of all software elements
- ClearCase is an integrated system, but it is useful to group its features into four functional groups:
 - Workspace management
 - Version control
 - Build management
 - Process control

Reasons for using versioning tools

- Efficient, reliable, and layered security
- Diversity of support for multiple platforms
- Central and distributed code
- Backup and multiple layers of saved changes
- Only differences are stored on server — not copies
- “Concurrent” access to the same file by multiple users
- Keeps track of items checked in and checked out
- Locks elements to prevent other team members from checking in versions (ClearCase only)

Project sharing

- To share a local project:
 - Establish the connection to the version control system.
 - Create the structure to store the project's data.
- Once connected, send (commit) project data to the version control system.
 - Copy the committed data from the workspace on your local machine to the version control repository.
- Users do not have to commit all of the project's data (model elements).
- Local machine keeps all of the project data whether some or all of the data are committed.
- All authorized team members can access the committed data.
 - Check it out of the version control system using the project versioning component of WebSphere Business Modeler.

Modeler version control terminology

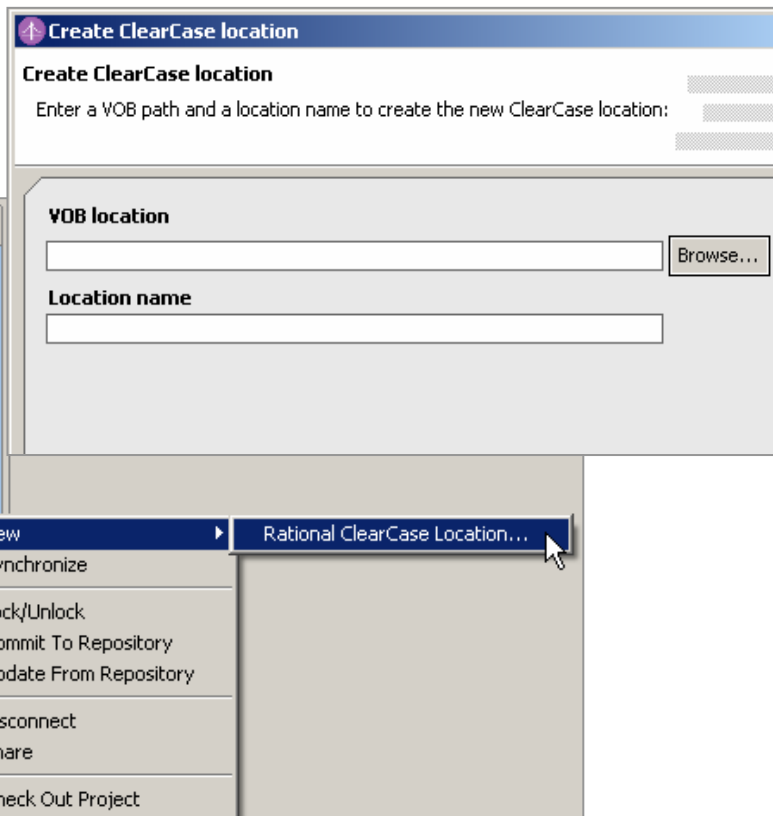
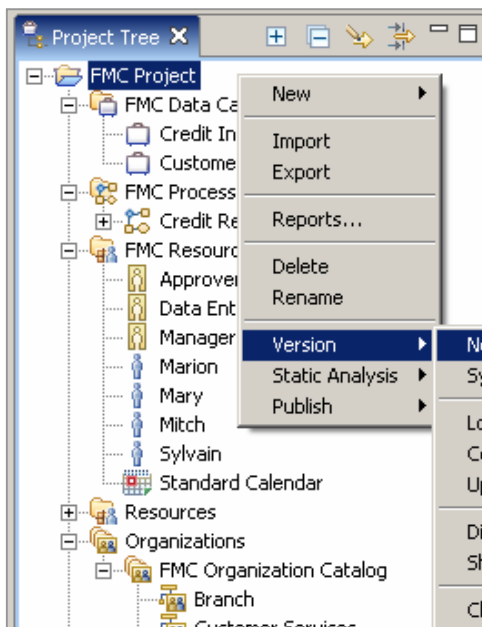
- **Share** (connect) a project to a version control system
- **Disconnect** (stop sharing) from a version control system
- **Commit** (check in) project data to the repository first time
- **Check out** (copy to different local machine) to make changes
- **Update** to receive changes from the repository
- **Synchronize** to determine the differences between your local copy and the central repository

Typical sharing sequence

User modeling action	User A	User B
A creates a model and connects to repository	Share	
A stores the model in the repository	Commit	
B wants to change the model, has to connect		Share
B copies model down to the workstation		Check out
B makes changes to the model		
B checks for conflicting changes		Synchronize
If no conflicting changes, B sends changes		Commit
A has been adding to the model		
A checks for conflicting changes	Synchronize	
If no conflicting changes, A sends changes	Commit	
A gets B's changes	Update	
B checks for A's changes		Synchronize
B gets A's changes		Update

Adding a ClearCase repository

- Right-click the Project Tree
 - Select **Version** → **New** → **Rational ClearCase Location**



Adding a CVS repository

The screenshot shows the IBM Rational IDE interface. On the left, the 'Select Perspective' dialog is open, with 'CVS Repository Exploring' selected. Below it, the 'CVS Repositories' view is visible, showing a 'New' button and a 'Repository Location...' button. On the right, the 'Add CVS Repository' wizard is open, showing fields for 'Location' (Host and Repository path), 'Authentication' (User and Password), and 'Connection' (Connection type and Use Default Port). A yellow callout box points to the 'New' button in the 'CVS Repositories' view, containing the text: 'Open the CVS Repository Exploring perspective by selecting **Window** → **Open Perspective** → **Other**'.

Select Perspective

- Business Modeling (default)
- CVS Repository Exploring**
- Debug
- Generic Log Adapter
- Java
- Java Browsing
- Java Type Hierarchy
- Plug-in Development
- Profiling and Logging
- Resource
- Team Synchronizing
- Test

CVS Repositories CVS Annotate

New Repository Location...

Refresh View

Add CVS Repository

Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host:

Repository path:

Authentication

User:

Password:

Connection

Connection type:

☒ Use Default Port

☐ Use Port:

☒ Validate Connection on Finish

☐ Save Password

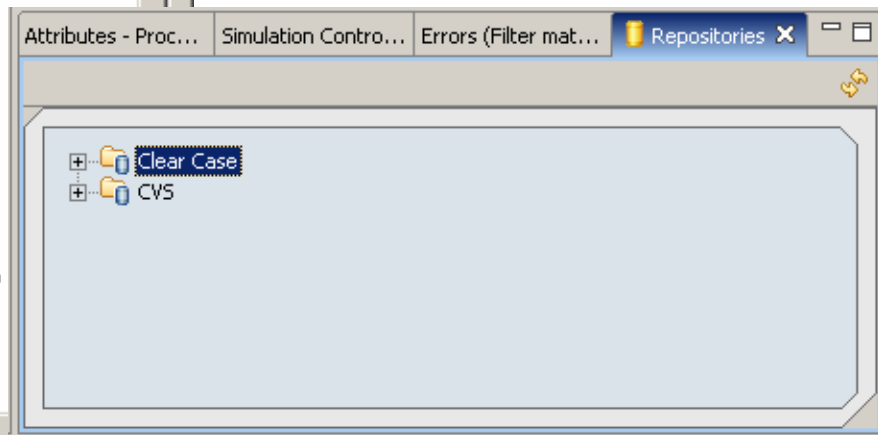
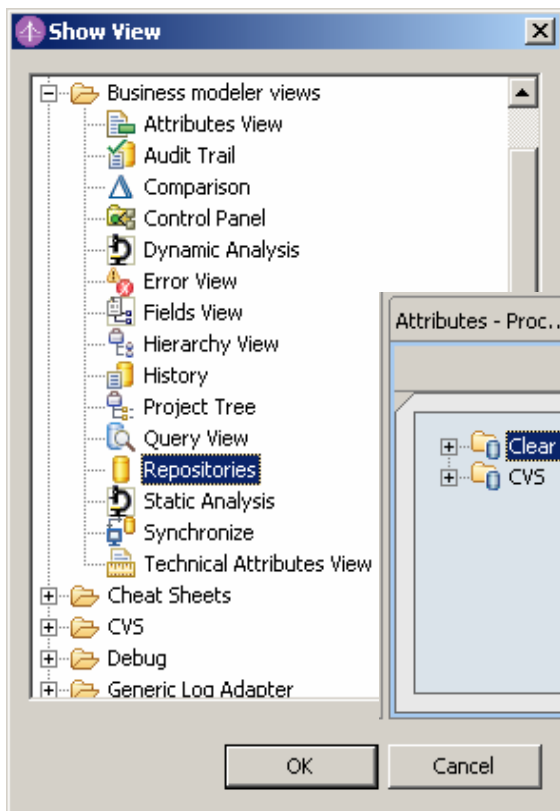
Open the CVS Repository Exploring perspective by selecting **Window → **Open Perspective** → **Other****

Other

Saved passwords are stored on your computer in a file that's already protected, for an intruder to read.

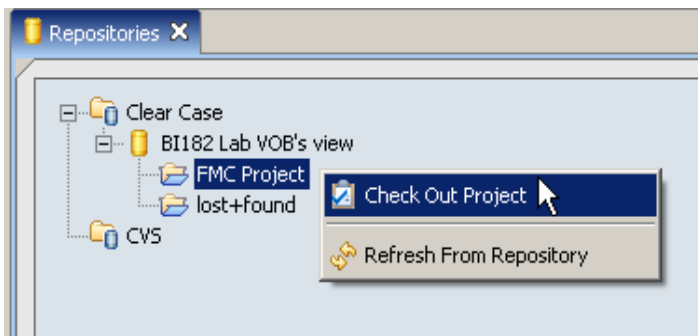
Repository view

- To view Repositories
 - Select **Window** → **Show View** → **Business Modeler Views** → **Repositories**



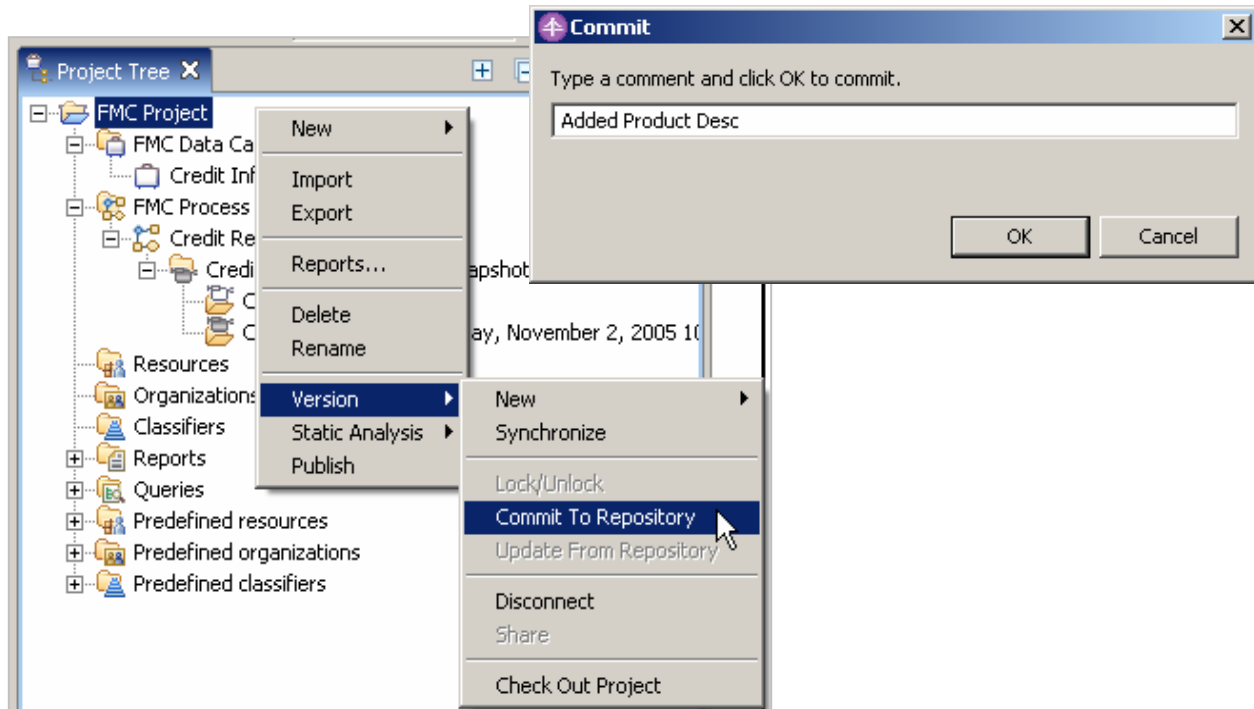
Checking out a project

- From the **Repository view**
- Any registered repository server will be displayed in the view
- Right-click the desired project
- Select: **Check Out Project**
 - Cannot check out a project that already exists in the Modeler workspace



Commit to repository

- Once the modification is made, commit changes back to the repository.

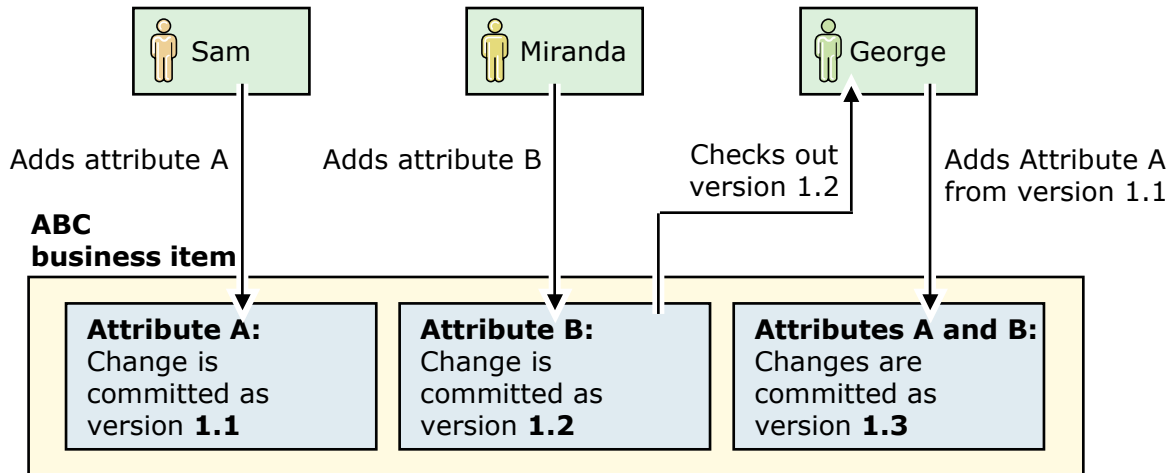


Synchronize project data

- Synchronize view displays:
 - Elements that have changes to receive from the version control repository (incoming)
 - Elements that have changes to send to the version control repository (outgoing), or
 - Elements that have both incoming and outgoing changes
- Synchronize with one or more of the following options:
 - Commit To Repository
 - Update From Repository
 - Overwrite and Commit To Repository
 - Overwrite and Update From Repository

Synchronization is important

- After the initial commit, model elements are kept synchronized
 - The same elements in both local and repository locations
- The repository increases the version number every time someone commits a change to the model element
- The workspace maintains the same version number until the user commits a change to the model element or updates it from the repository



Resolving conflicts

- A conflict occurs when:
 - Changes to an older version of an element are made
 - Team members work on the same version of project data simultaneously
- Elements with incoming and outgoing changes may have a conflict
- Users must explicitly overwrite one of the versions to achieve synchronization
 - If the overwritten version contains information the current version needs, the user needs to reconcile the two versions.
 - A user should view the difference between current and the overwritten version.
 - The user then makes the appropriate changes to the current version and commits the changes as a new version.

Locking and unlocking elements: ClearCase

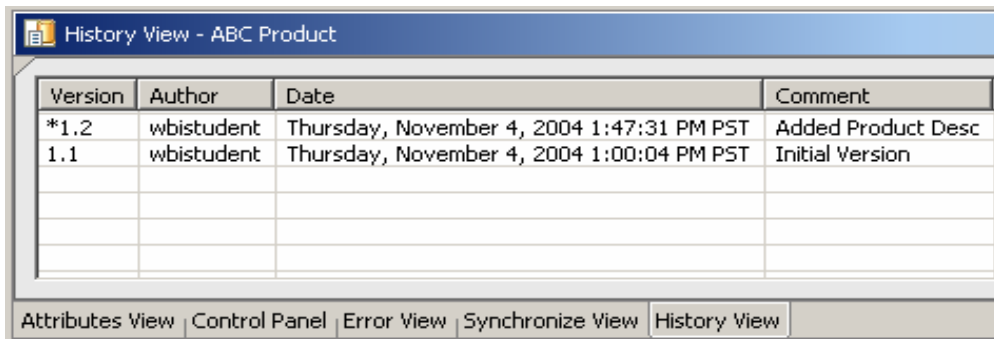
The screenshot shows the Rational ClearCase Project Tree with the 'Credit Information' element selected. A context menu is open, and the 'Lock/Unlock' option is highlighted. A 'Lock' dialog box is also open, showing the 'Lock' option selected. Two callout boxes provide additional context.

- Lock elements to prevent other team members from checking in versions.
- Unlock elements that no longer need protection.

- The ability to lock and unlock elements is only available to projects that are in a Rational ClearCase repository.

Viewing the history of a component

- After modifications to a project are committed to a repository, a comparison can be made between the local workspace and the repository original
- Project versioning function can restore previous versions of components
 - A persistent undo function
- Can compare multiple versions
 - Right-click a component and select **Version** → **Show History**



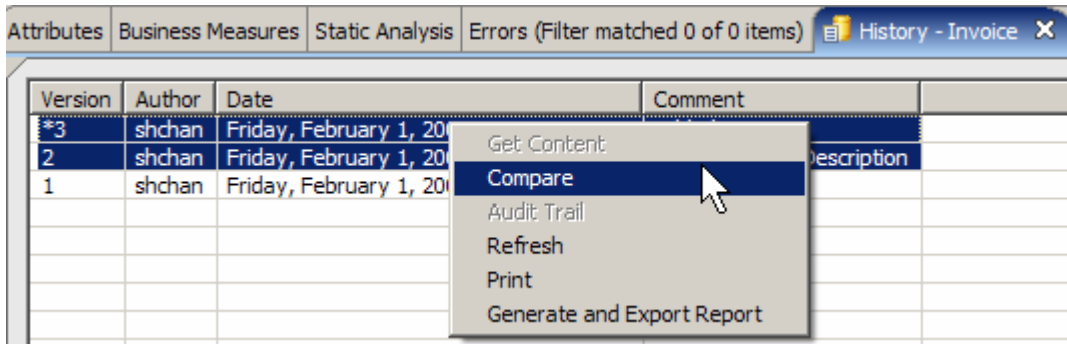
The screenshot shows a window titled "History View - ABC Product". Inside the window is a table with four columns: "Version", "Author", "Date", and "Comment". The table contains two rows of data. The first row shows version *1.2, author wbistudent, date Thursday, November 4, 2004 1:47:31 PM PST, and comment Added Product Desc. The second row shows version 1.1, author wbistudent, date Thursday, November 4, 2004 1:00:04 PM PST, and comment Initial Version. Below the table is a navigation bar with five buttons: "Attributes View", "Control Panel", "Error View", "Synchronize View", and "History View". The "History View" button is currently selected.

Version	Author	Date	Comment
*1.2	wbistudent	Thursday, November 4, 2004 1:47:31 PM PST	Added Product Desc
1.1	wbistudent	Thursday, November 4, 2004 1:00:04 PM PST	Initial Version

Attributes View | Control Panel | Error View | Synchronize View | History View

Comparing versions

- **History view** displays all revisions of the component.
- Select two versions, and right-click for the context menu to compare.
 - If three or more versions are selected, Audit Trail becomes active.
 - If only one version is selected, the Get Content command is enabled.



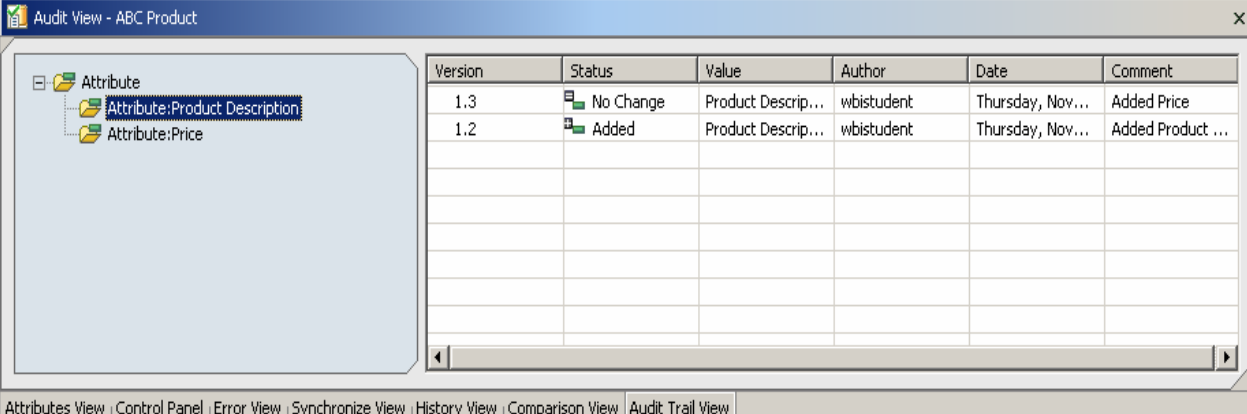
Comparison view

- The Comparison view will show the information of the two versions.
 - Date: date change took place
 - Author: user who made the change
 - Version: version number
 - Value: actual data for each version
 - Status: status of change (for example, Added or Updated)
- The value “Not Available” with “Added” status indicates that the “Product Description” was a new item that did not exist before.

Comparison View - ABC Product - (1.2 1.1)			
Product Description	Value	Version	Author
			Date
	Product Description	1.2	wbistudent
	Not Available	1.1	wbistudent
	Status	Added	

Audit trail

- To display additional details of versions in an audit trail:
 - In the Business Modeling perspective, right-click the element to display
 - Select **Team** → **Show History** on the context menu for **History view**.
 - Select three or more versions, and then right-click one of them to view history.
 - Click **Audit Trail** on the context menu for the **Audit Trail view**.
 - In the **Audit Trail view**, right-click the newer version.
 - Select **Show Details** for the **Comparison view**.

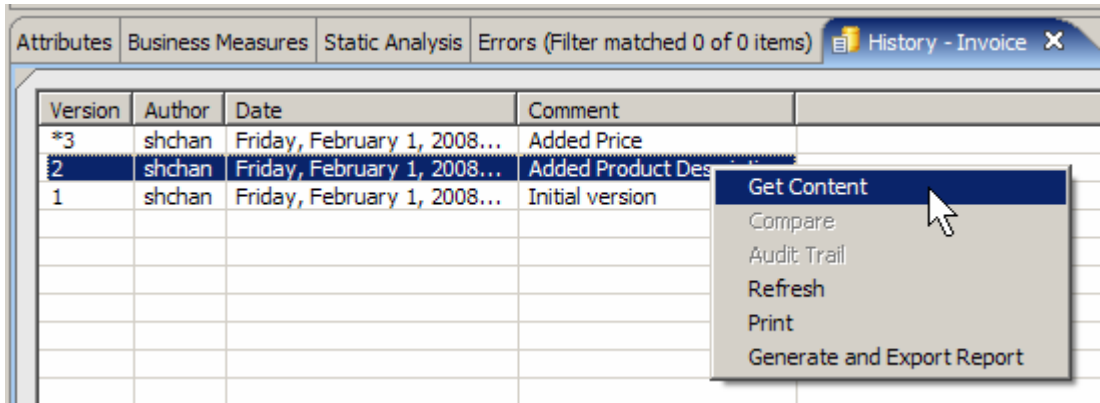


Version	Status	Value	Author	Date	Comment
1.3	No Change	Product Descrip...	wbistudent	Thursday, Nov...	Added Price
1.2	Added	Product Descrip...	wbistudent	Thursday, Nov...	Added Product ...

Attributes View | Control Panel | Error View | Synchronize View | History View | Comparison View | **Audit Trail View**

Get content

- The Get Content command will retrieve a specific version of a component to the local workspace and overwrite any local changes.
- Local changes must be committed to the repository before issuing this command to save the changes.



Disconnect projects

- A user can stop sharing a project by closing the connection between the local machine and the version control system.
 - Closing a connection does not remove the project from the version control system.
 - Other team members can still access the repository copy.
- Disconnect does not remove the project from your workspace.
- To reconnect to the repository copy:
 - Remove the project from local workspace.
 - Check out the project from the repository to the local machine.

Best practices for version control (1 of 2)

- Small team with no version control system
 - Importing and exporting the project (MAR file) may be sufficient to exchange models or elements.
 - A real version control system may be an excessive burden on the project and the team members.
- Large team with version control
 - Someone has to perform the role of model administrator.
 - This person resolves conflicts that arise during development.
 - Access may be limited to some projects and model elements.
- Large team with version control
 - Everyone is required to synchronize their local workspace and the version control repository before committing any changes.
- Large team with version control
 - Synchronize and choose updating and committing from the Synchronize view rather than directly updating through navigation.

Best practices for version control (2 of 2)

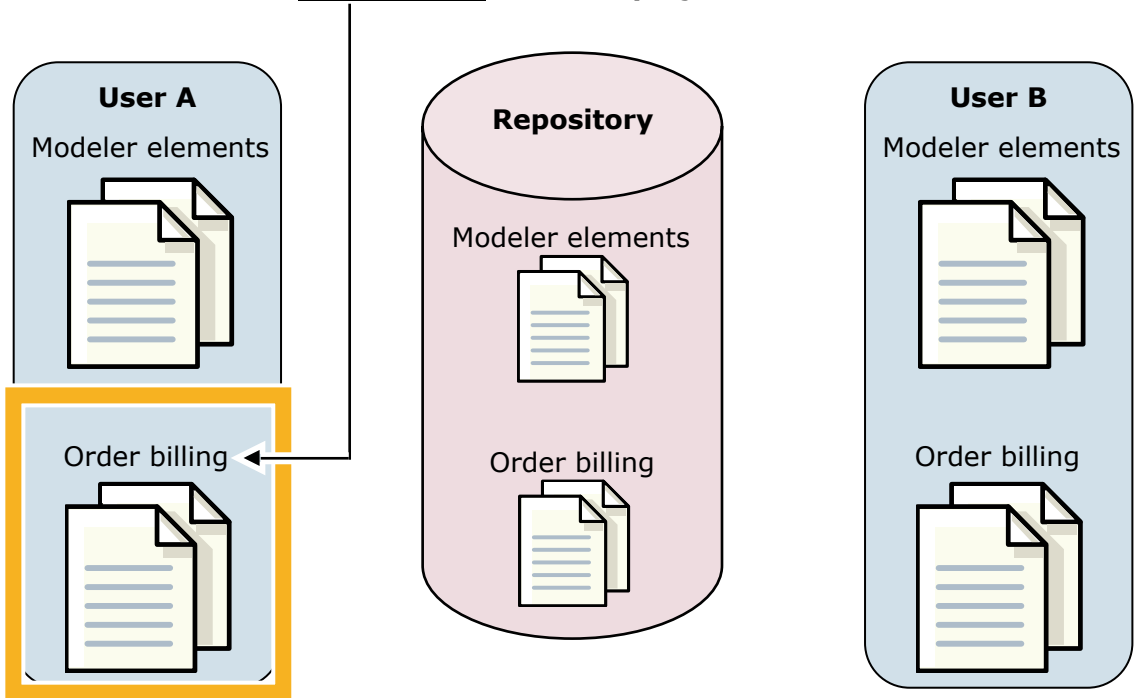
- Large team with version control
 - Use lock capabilities provided by the version control system when available.
 - Understand and test what happens once the lock is released.
- Large team with version control
 - Develop a process to remove elements from shared models.
 - If you remove an element such as a business item or process from a shared project in the Project Tree view, other users sharing the project must delete the element from their local workspace.
- Large team with version control
 - Develop a shared element removal naming convention.
 - It is important to establish a naming convention for flagging elements to be removed.
 - Someone must take on the role of physically removing elements flagged for removal at agreed upon times.

Best practices for deleting elements

- Identify element to be removed
- Flag element to be removed
- Commit renamed element
- Update all projects
- Notify and gain agreement
- Remove and commit removal
- Everyone deletes from their project
- Projects are now synchronized

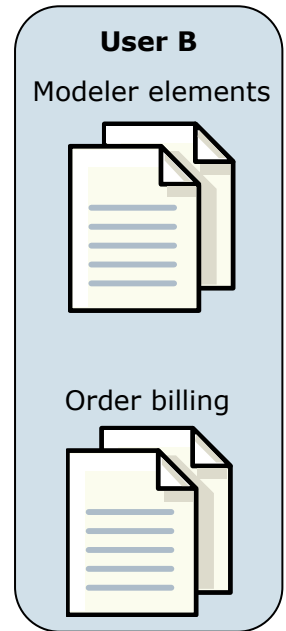
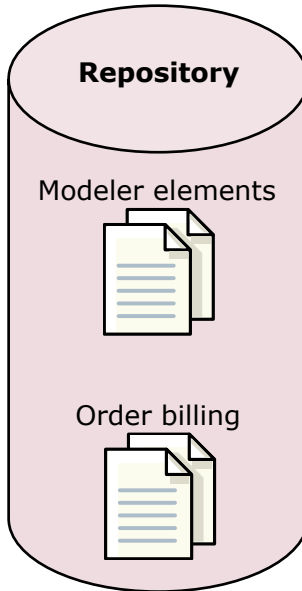
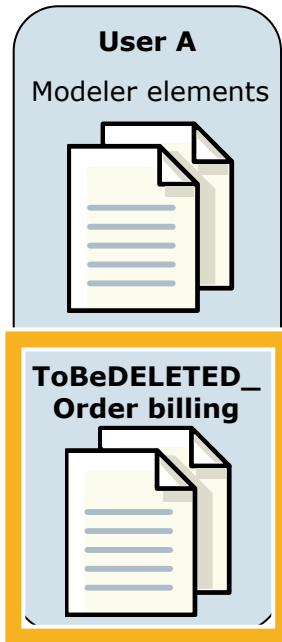
Identify element to be removed

Need to remove Order billing from the project and all workstations.



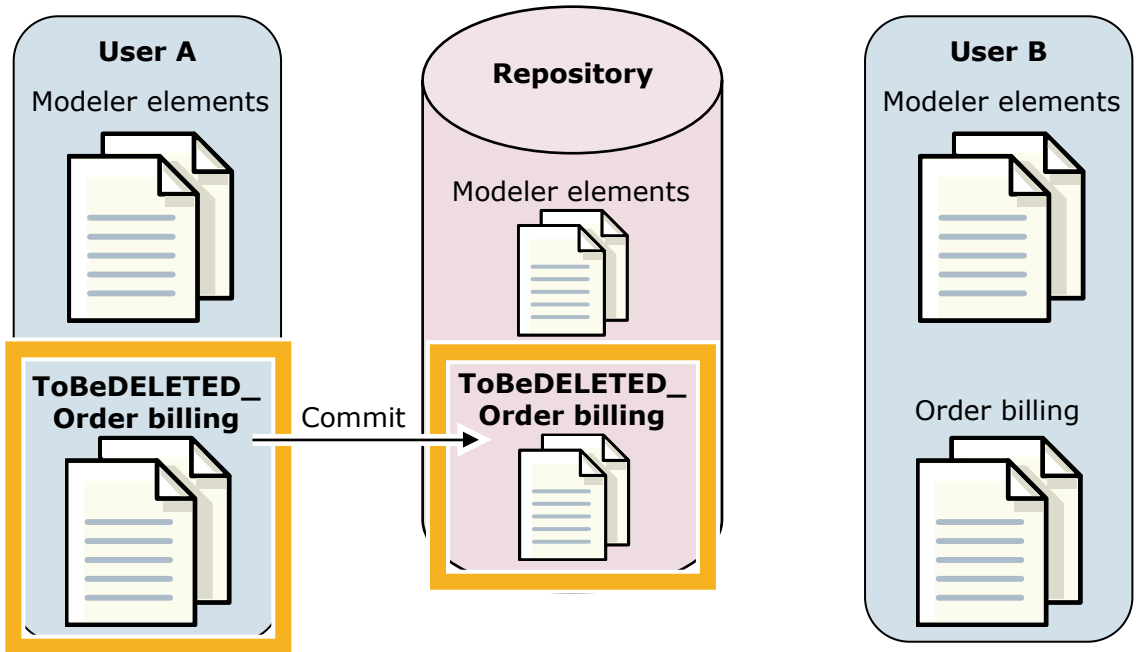
Flag element to be removed

Flag Order billing for removal by renaming it.



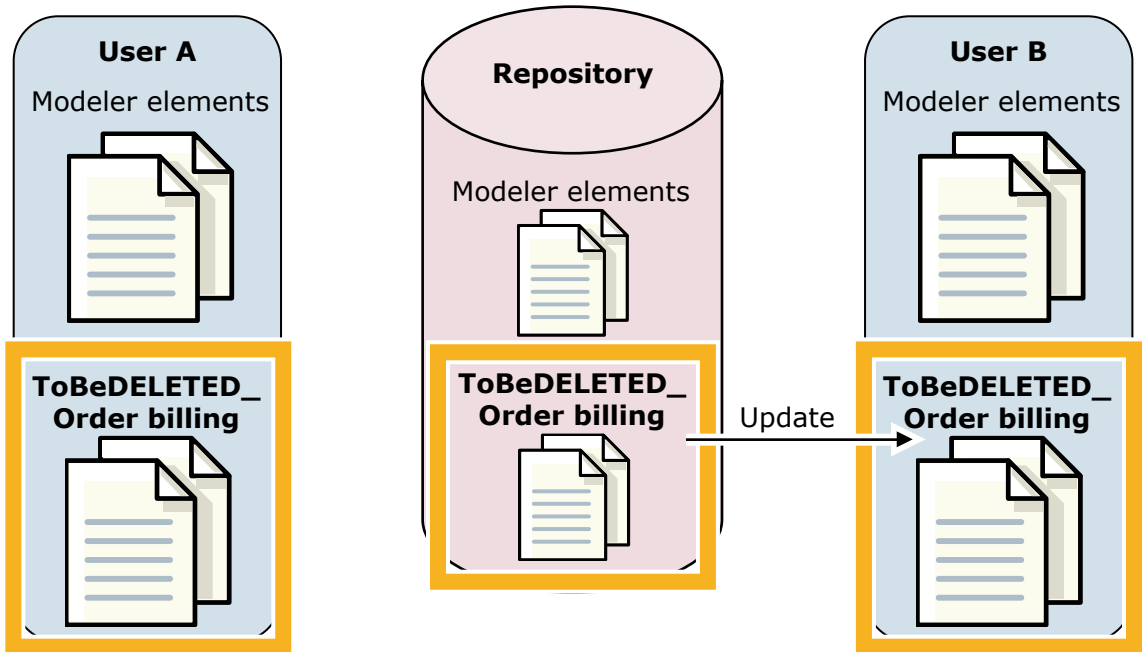
Commit renamed element

Commit changes to the repository.



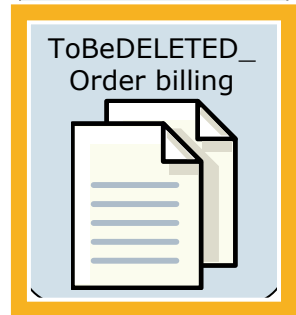
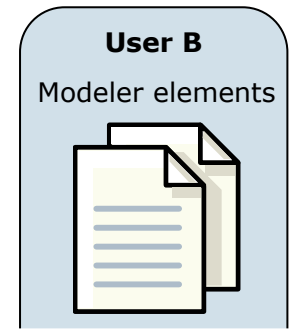
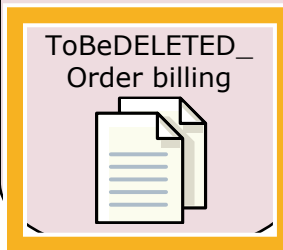
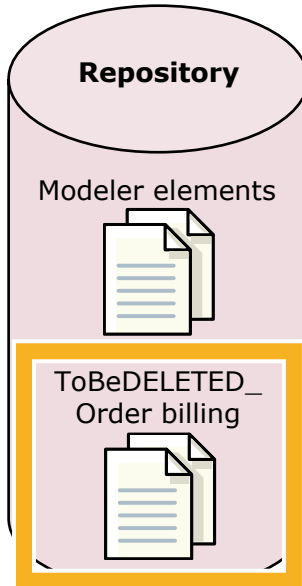
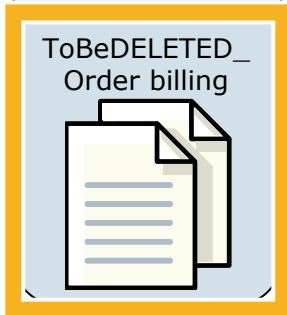
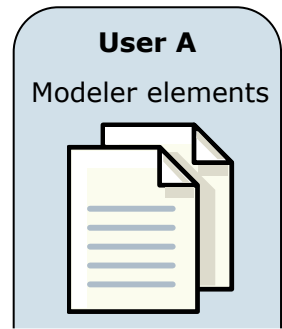
Update all projects

All users update their project to receive the renamed element.



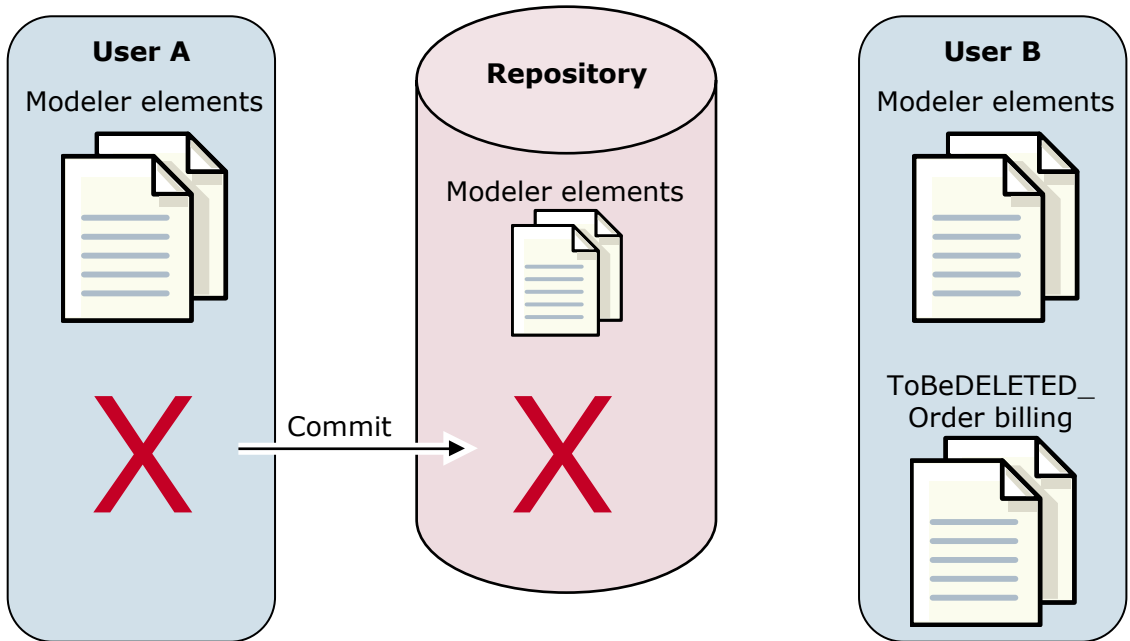
Notify and gain agreement

Notify everyone and wait until everyone agrees the element can be removed.



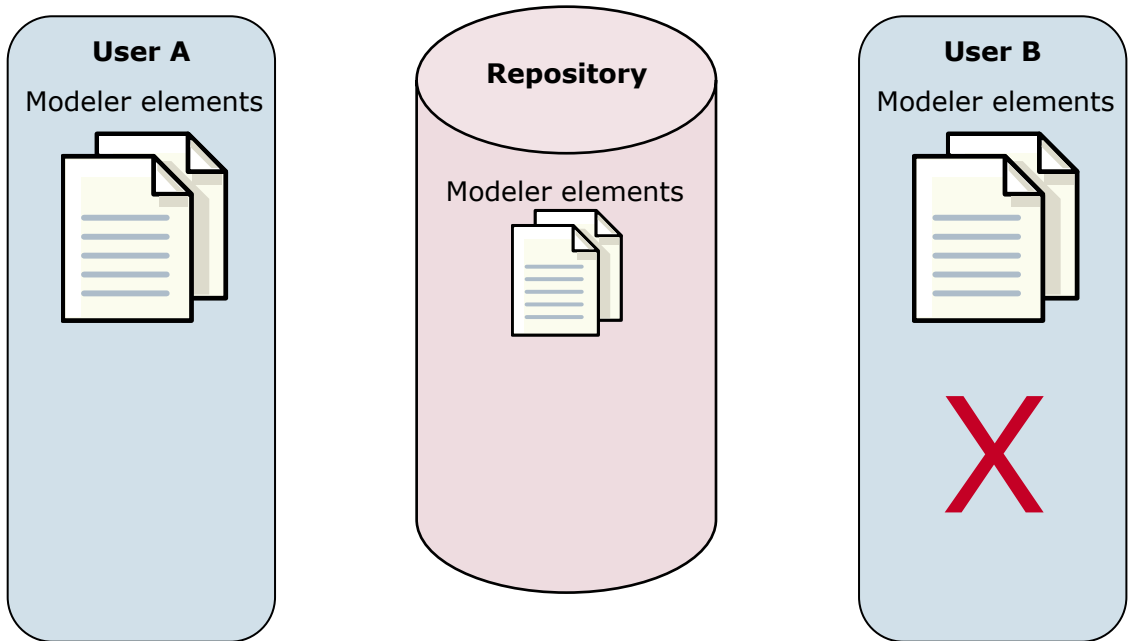
Remove and commit removal

Remove, commit to repository, and notify everyone to delete.



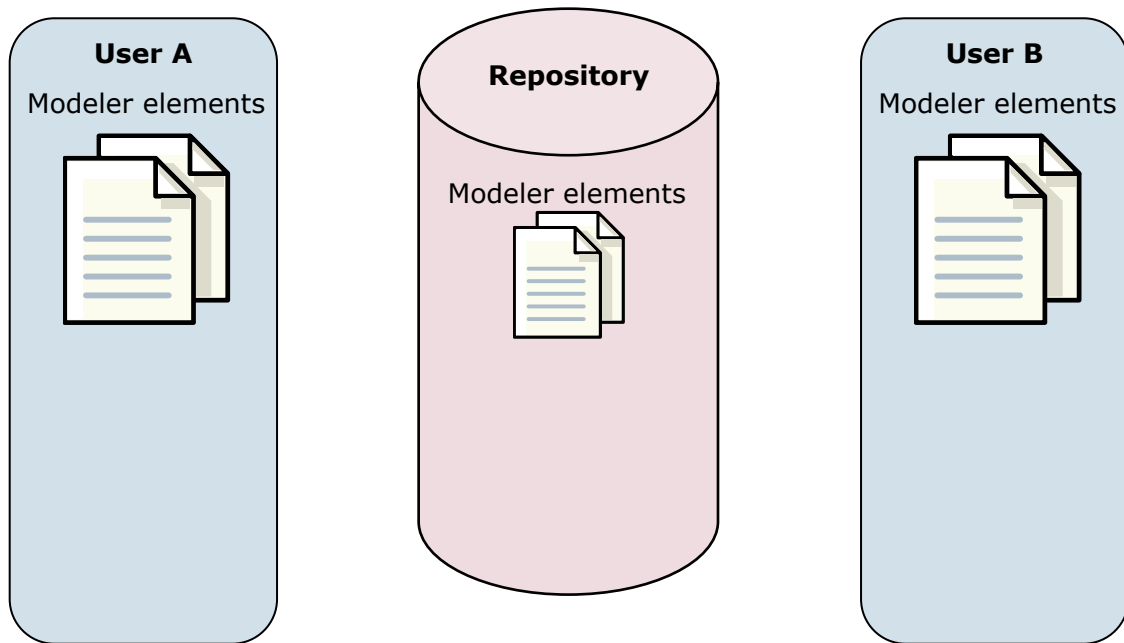
Everyone deletes from their project

Everyone deletes the element that needs to be deleted. Failure to do so could result in the element being committed and restored on the repository again.



Projects are now synchronized

Projects are synchronized once the elements are deleted from the repository and all local projects.



Checkpoint: Version control

Your instructor will review these questions with you as a group. If time permits, the instructor may provide you time to answer the questions on your own before the group discussion.

1. WebSphere Business Modeler supports which two version control systems?
2. What is the difference between “commit” and “check out”?
3. In which version control system are users able to lock and unlock elements in WebSphere Business Modeler?
4. What is the difference between “update” and “synchronize”?
5. What is the function of “get content”?

Checkpoint solutions: Version control

1. IBM Rational ClearCase and Concurrent Versions System (CVS)
2. “Commit” synchronizes your local workspace and the version control system’s repository by sending the changes you made. “Check out” enables you to work with and modify a local copy of the project.
3. IBM Rational ClearCase
4. “Update” retrieves the latest version of the elements in the catalog or project from the repository. “Synchronize” compares the local contents of the catalog or project with the contents of the repository.
5. The “get content” command retrieves a specific version of a component to the local workspace and overwrites any local changes.

Unit summary

Having completed this unit, you should be able to:

- Explain the need for project versioning
- Describe Concurrent Versions System (CVS)
- Describe IBM Rational ClearCase
- List project versioning steps
- Explain the importance of version control
- Describe development using project versioning
- Explain best practices and deletion strategy

Exercise overview

In this exercise, you will:

- Use Rational ClearCase for version control
 - Rational ClearCase server is installed on your VMware image
- Add a project to Rational ClearCase
- Make and managing changes in your project
- Update a project from the repository
- Work in a multiple-developer environment