COGILITY
SOFTWARE

Change Management in Cogility Studio

Cogility Studio version 7.0

Document version number 7.0

Cogility Software Corporation
111 N. Market St. #888
San Jose, CA 95113

support@cogility.com

Printed in the United States of America.

Change Management

Contents

Preface

Chapter 1 Change management

Chapter 2 User modifications

Chapter 3 Versions

Chapter 4 Model comparison

Chapter 5 Differences and conflicts

Preface

Cogility Studio provides the tools to create a model for an enterprise information system, and deploy it as a J2EE application. The Cogility Studio documentation provides support for this endeavor.

Cogility Studio documentation

Cogility Studio comes with several volumes of documentation to help you.

- *Installing and Configuring Cogility Studio* describes the installation and configuration of your application server, database and Cogility Studio.
- *Getting Started with Cogility Studio* is a brief overview of Cogility Studio.
- *Modeling with Cogility Studio* tells you how to build a model-driven enterprise application using Cogility Modeler and associated tools.
- *Using Action Semantics with Cogility Studio* provides a reference to modeling action semantics for use with Cogility Studio.
- *Change Management in Cogility Studio* describes the change management system for models and model artifacts.
- *Model Deployment & Execution in Cogility Studio* is a guide to application monitoring, maintenance and migration, and describes the utilities that you can use to test and monitor your model deployed as a enterprise application.

Several white papers on various topics are also available to further your understanding of enterprise application integration, business process management, model driven architecture and other related topics. See the Cogility website:

http://www.cogility.com.

1

Change management

Following the best practices of standard software configuration management (SCM), Cogility Studio has a range of tools that track and manage the changes to your models as they evolve. You can create and compare versions, resolve conflicts, and merge changes.

In an organization with a small number of people working on their own installations of Cogility Studio, each person may maintain a version of the model in a local authoring repository. The base-line or master model is held in the organization's software configuration management system (SCM) such as Perforce or CVS. To update the base-line model, a developer imports the base-line model into his local authoring repository, compares his local model to the base-line, merges his changes with the base-line using Cogility Modeler, then checks the base-line back into the organization's SCM system. All of the model's versions are updated. Other developers working on the model go through the same process, developing the model iteratively. In this scenario, the base-line model is considered "locked" and may not be worked on by more than one developer at a time. For larger organizations a more elaborate structure for managing collaborative development may be necessary.

Cogility Modeler's change management system does not manage changes between different versions of the base-line model between installations, only between the imported base-line model and the working model on the local installation. The change management system works only on a local installation of Cogility Modeler for the models stored in the local authoring repository. It compares model objects in the current context (see "Current CM context" on page 16 of the guide, *Modeling with Cogility Studio*) with model objects in the authoring repository.

Base-line model

The first version of your integration model serves as the base-line for all subsequent versions. For any one repository, you might start building your model from scratch or you might import a model to serve as the base-line. A newly created repository remains empty except for the meta-model until you turn over the model you created from scratch or imported.

"Model container" on page 21 of the guide, *Modeling with Cogility Studio* describes how to start developing an integration model from scratch. Once you have created the model, you place it under change management with a process called turn over. See "Turn-over version" on page 23 for more information. Only a model that includes a model container may be versioned or turned over because the configuration map for that model takes its name from the model name.

The model you want to use as a base-line may already exist as a file. If so, it was first created in Cogility Modeler and exported. See "Model export" on page 25 for more information.

If you want to import the base-line for comparison against an updated model in the current context, see "Model import" on page 27.

Importing a base-line for updating

To import a model from a file into the current context:

1. If you have Cogility Modeler running and a model loaded into the current context, exit Cogility Modeler (from the **File** menu, choose **Exit**).

   Note: To avoid importing a model into the current context, potentially corrupting an existing model, it is recommended that you exit Cogility Modeler and log in with an empty repository.

2. If you do not have a new, empty repository, create one as described in "Loading the repository" on page 21 of the guide, *Modeling with Cogility Studio*.
3. Run Cogility Modeler (see "Logging into the current context" on page 17 of the guide, *Modeling with Cogility Studio*).
4. Import the model from the shared store.

   See "Model import" on page 27. How you import the model depends upon the format you have used to store it and how you have configured Cogility Modeler.
5. If you want to reset all of the version name labels on all of the artifacts, version the model with forced versioning.

   See "Forcing versioning" on page 19.
6. Place the model under change management.

   See "Placing a model under change management" on page 6.
7. Exit Cogility Modeler (from the **File** menu, select **Exit**), then log in again with the same repository into which you imported your model.

   You do this to re-establish the configuration map for the model. Until you log out, the default deployment model configuration map is the map under which all modifications and versions are created. When you log back in, the configuration map for the imported and turned over model becomes the current configuration map. For more information, see "Turn-over version" on page 23, below.

This model is now the first version of the model in the current context. You can change, add to or remove from this model and create new versions of either separate model objects or the entire model.

Placing a model under change management

Before the change management system can track changes to your model, you must explicitly create a configuration map and version for the model. Placing the model under change management accomplishes this. For more information about the map and version, see "Current CM context" on page 16 of the guide, *Modeling with Cogility Studio.*

To place a model under change management:

1. With the model container selected, click the Place Model Under CM button .
2. In the dialog that appears, for **Turnover Version Name**, enter a descriptive name.
3. In the **Comments** field, describe the version.

   The model now assumes the turn-over version name label you entered, and this appears next to the model container label in the tree view, as shown below.

Change management preferences

You can set several preferences for change management, including the following:

- Report content differences of a diagram as physical conflict
- Warn before applying other version's changes to mutable version in context
- Separate potential (logical) and physical conflicts
- Do not report potential conflicts
- Clear all in-memory resolution tracking objects after every turnover
- Treat 'key' and 'alias' as one single attribute called 'Name'

Setting change management preferences

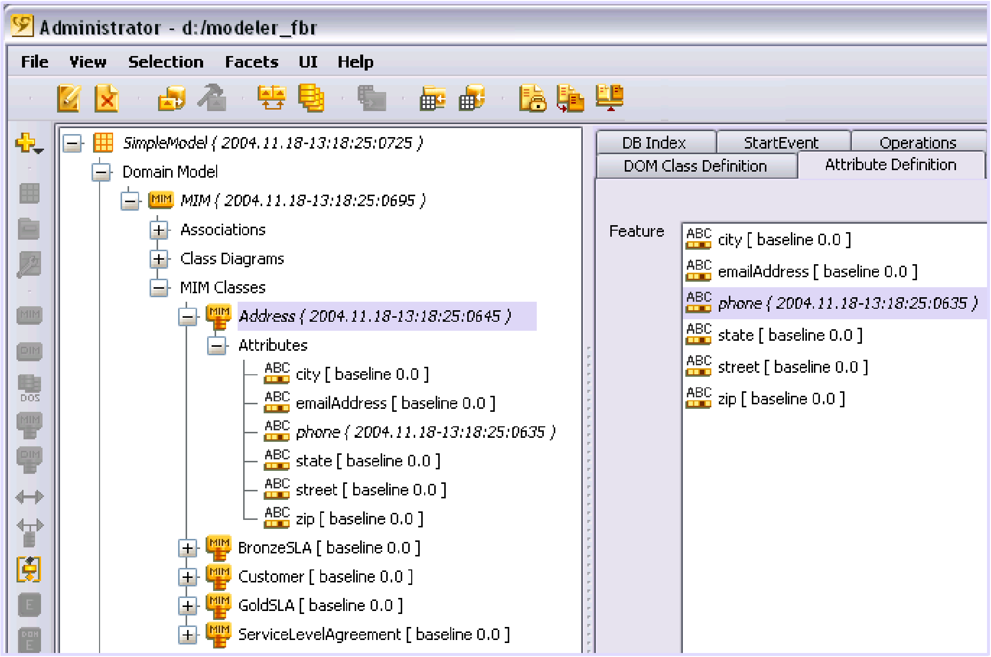
To set change management preferences:

1. In Cogility Modeler, from the **Selection** menu, choose **CM Operations > CM Preferences**.
2. Select the preferences you want and click **OK**.

2

User modifications

As you make changes to model artifacts, their labels in Cogility Modeler's tree view appear in italics. In the picture below, the Address class been changed, the new phone attribute was added to it. So the phone attribute, the Address class, the MIM, and the SimpleModel model container labels now all appear in italics.

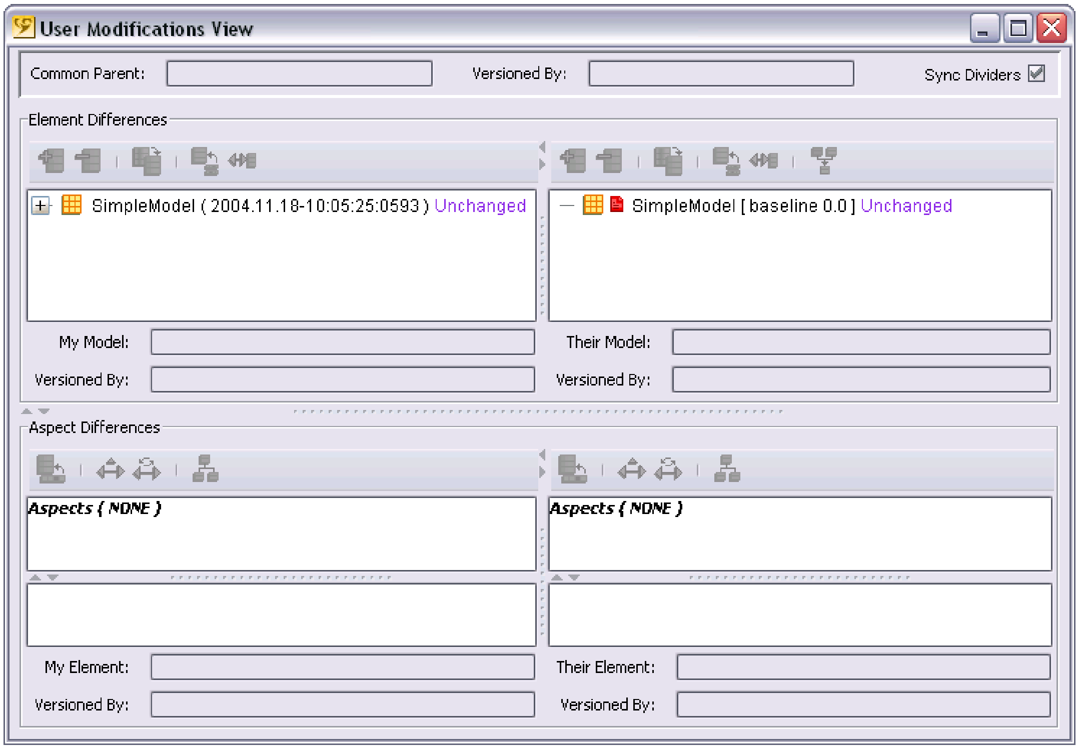

Administrator - d:/modeler_fbr
File
View
Selection
Facets
UI
Help
SimpleModel (2004.11.18-13:18:25:0725)
Domain Model
MIM (2004.11.18-13:18:25:0695)
Associations
Class Diagrams
MIM Classes
Address (2004.11.18-13:18:25:0645)
Attributes
city [baseline 0.0]
emailAddress [baseline 0.0]
phone (2004.11.18-13:18:25:0635)
state [baseline 0.0]
street [baseline 0.0]
zip [baseline 0.0]
BronzeSLA [baseline 0.0]
Customer [baseline 0.0]
GoldSLA [baseline 0.0]
ServiceLevelAgreement [baseline 0.0]
DB Index
StartEvent
Operations
DOM Class Definition
Attribute Definition
Feature
city [baseline 0.0]
emailAddress [baseline 0.0]
phone (2004.11.18-13:18:25:0635)
state [baseline 0.0]
street [baseline 0.0]
zip [baseline 0.0]

This means that each of these artifacts is now *mutable*. A mutable artifact is one for which there are pending changes. A mutable artifact may not be exported or imported until it is versioned or its model is turned over.

As soon as you turn over a model, all of the artifacts in it become *immutable*. All pending changes to the artifact are actualized and no longer may be reverted. All versions are still available through the Conficts and Differences view, however, for comparison. See "Version comparison" on page 32. Also, you can view the details of changes to turn-over version elements through the element revision history window. See "Viewing turn-over version element history" on page 23.

You can view and work with mutable artifacts in the User Modifications view. The User Modifications view lets you revert back to last immutable turn-over version of an element or element aspect. Note that this is different from an "un-do" operation wherein the last change to an element would be reverted to the element's state prior to that change. Regardless of how many intervening changes have occurred to an element, reverting the latest change always restores the element to its state as of the last turn-over version.

Initially, the User Modifications view shows, in the upper left pane, the changed artifact's top-level container. In the picture below, this is the SimpleModel model container. Notice that the label includes a time stamp, in parentheses, that describes when the artifact was last modified. The model container itself is unchanged, as shown, but it contains changed elements, as evidenced by the plus sign to the left of the object. Displayed in the upper right pane is the original artifact before it became mutable. Notice that its original version label, "baseline 0.0" is displayed.



SimpleModel (2004.11.18-10:05:25:0593) Unchanged

SimpleModel [baseline 0.0] Unchanged

Aspects (NONE)

Aspects (NONE)

When you select the changed artifact's top-level container and click the plus sign, the elements of the top-level container object are listed in a hierarchy. Elements may be changed, added or removed. Additionally, the Common Element, Versioned By, Current Model and Other Model fields are populated. Because the model artifacts in the current context are mutable, they are being compared to their immutable versions that preceded the changes. The Common Element field describes the original artifact from which both versions descended. The Versioned By field shows the login ID of the user who made the change. This information is more appropriate when comparing dissimilar versions, as described in "Element differences" on page 37.

The specific changes to the elements are described in the Aspect Differences panes. See "Aspect differences" on page 13.

Viewing user modifications

To view and work with user modifications:

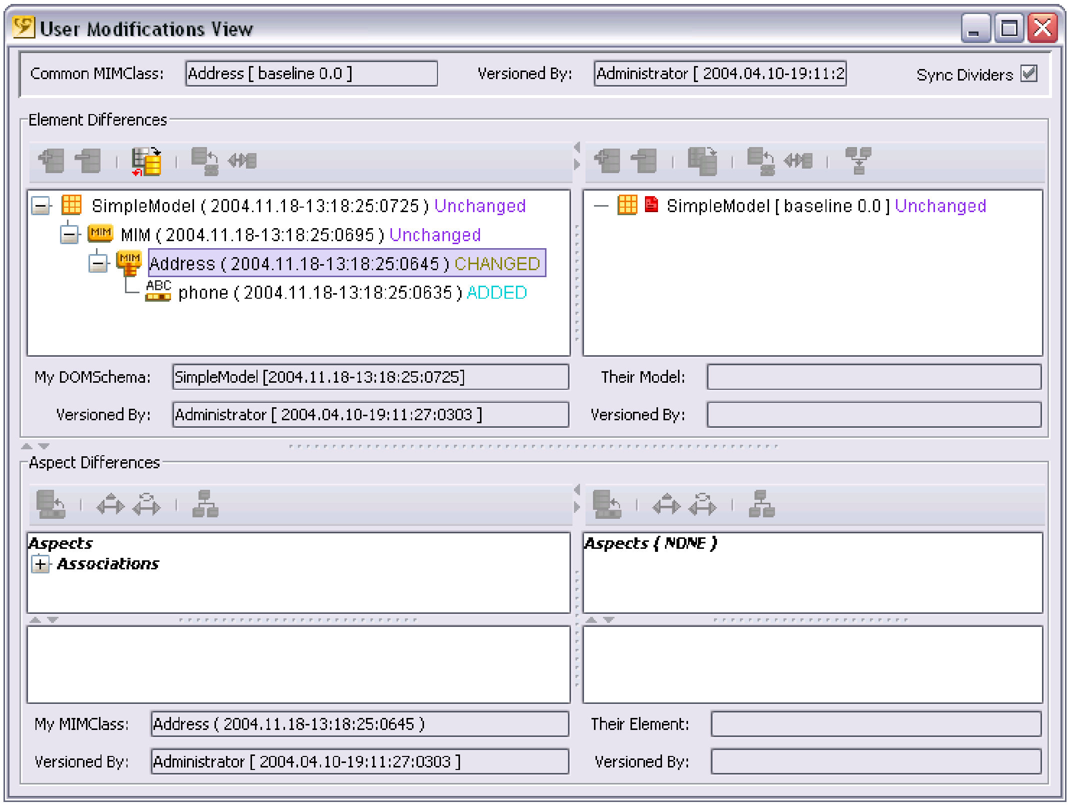
1. In Cogility Modeler's tree view, select a mutable artifact.
2. From the **Selection** menu, choose **CM Operations** > **Show User Modifications**, or click the **Display Pending Changes** button .

   The User Modifications view displays.

3. In the upper left pane, select the changed artifact's top-level container and click the plus sign.

Element differences

The top panes of the User Modifications view describe element differences within model artifacts. In the left pane, the changed elements are listed within their model's hierarchy, including the elements' container objects. In the following picture, the phone attribute was ADDED to the Address class, which is now CHANGED because of the addition.



User Modifications View

Common MIMClass: Address [baseline 0.0]
Versioned By: Administrator [2004.04.10-19:11:2
Sync Dividers ☑

Element Differences

- SimpleModel (2004.11.18-13:18:25:0725) Unchanged
  - MIM (2004.11.18-13:18:25:0695) Unchanged
    - Address (2004.11.18-13:18:25:0645) CHANGED
      - phone (2004.11.18-13:18:25:0635) ADDED

- SimpleModel [baseline 0.0] Unchanged

My DOMSchema: SimpleModel [2004.11.18-13:18:25:0725]
Versioned By: Administrator [2004.04.10-19:11:27:0303]
Their Model:
Versioned By:

Aspect Differences

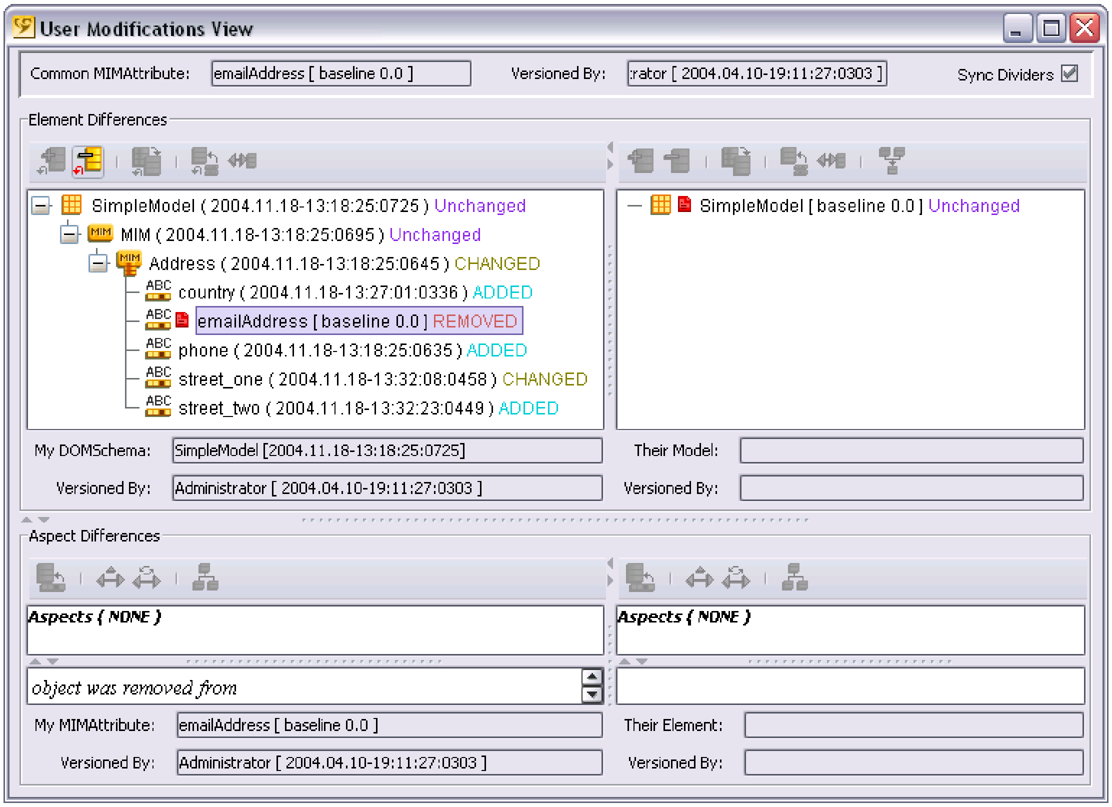
Aspects
+ Associations

Aspects { NONE }

My MIMClass: Address (2004.11.18-13:18:25:0645)
Versioned By: Administrator [2004.04.10-19:11:27:0303]
Their Element:
Versioned By:

When you add or remove an element, such as an attribute to or from a class, you can revert the addition or removal on a per-element basis. So if you have made several additions to a class or removed several elements from it, you can choose which ones to revert. Any changes you revert restore the element to its state as of the last turn-over version. With each specific element you can revert the changes as described in "Reverting elements" on page 12.

When you want to revert all of the changes to the model artifact, you can revert to the parent version, the artifact as it existed following the last turn over. This reverts all changes to all of the artifact's elements and re-establishes the artifact as immutable. If the artifact is a container, all additions, removals and other changes to the elements in that container are reverted. See “Reverting to the parent version” on page 12.



User Modifications View

Common MIMAttribute: emailAddress [baseline 0.0]   Versioned By: :rator [2004.04.10-19:11:27:0303]   Sync Dividers ☑

Element Differences

- SimpleModel (2004.11.18-13:18:25:0725) Unchanged
  - MIM (2004.11.18-13:18:25:0695) Unchanged
    - Address (2004.11.18-13:18:25:0645) CHANGED
      - country (2004.11.18-13:27:01:0336) ADDED
      - emailAddress [baseline 0.0] REMOVED
      - phone (2004.11.18-13:18:25:0635) ADDED
      - street_one (2004.11.18-13:32:08:0458) CHANGED
      - street_two (2004.11.18-13:32:23:0449) ADDED

- SimpleModel [baseline 0.0] Unchanged

My DOMSchema: SimpleModel [2004.11.18-13:18:25:0725]   Their Model:

Versioned By: Administrator [2004.04.10-19:11:27:0303]   Versioned By:

Aspect Differences

Aspects (NONE)   *Aspects (NONE)*

object was removed from

My MIMAttribute: emailAddress [baseline 0.0]   Their Element:

Versioned By: Administrator [2004.04.10-19:11:27:0303]   Versioned By:

Changes to elements may be further described by aspects. You can review the changes in the aspects pane. See “Aspect differences” on page 13.

Reverting elements

To revert changes to specific elements:

1. Select the changed elements (hold down the Ctrl key to select multiple elements) and click the appropriate revert button:

   - Revert removal button for REMOVED elements
   - Revert addition button for ADDED elements

   The following buttons become active for changes to elements that have aspect differences. See “Aspect differences” on page 13.

   - Revert all attribute value applications button for CHANGED elements
   - Revert all association targets button for CHANGED elements

Reverting to the parent version

To revert to the artifact’s parent version:

1. Select the artifact and click the revert to parent version button .

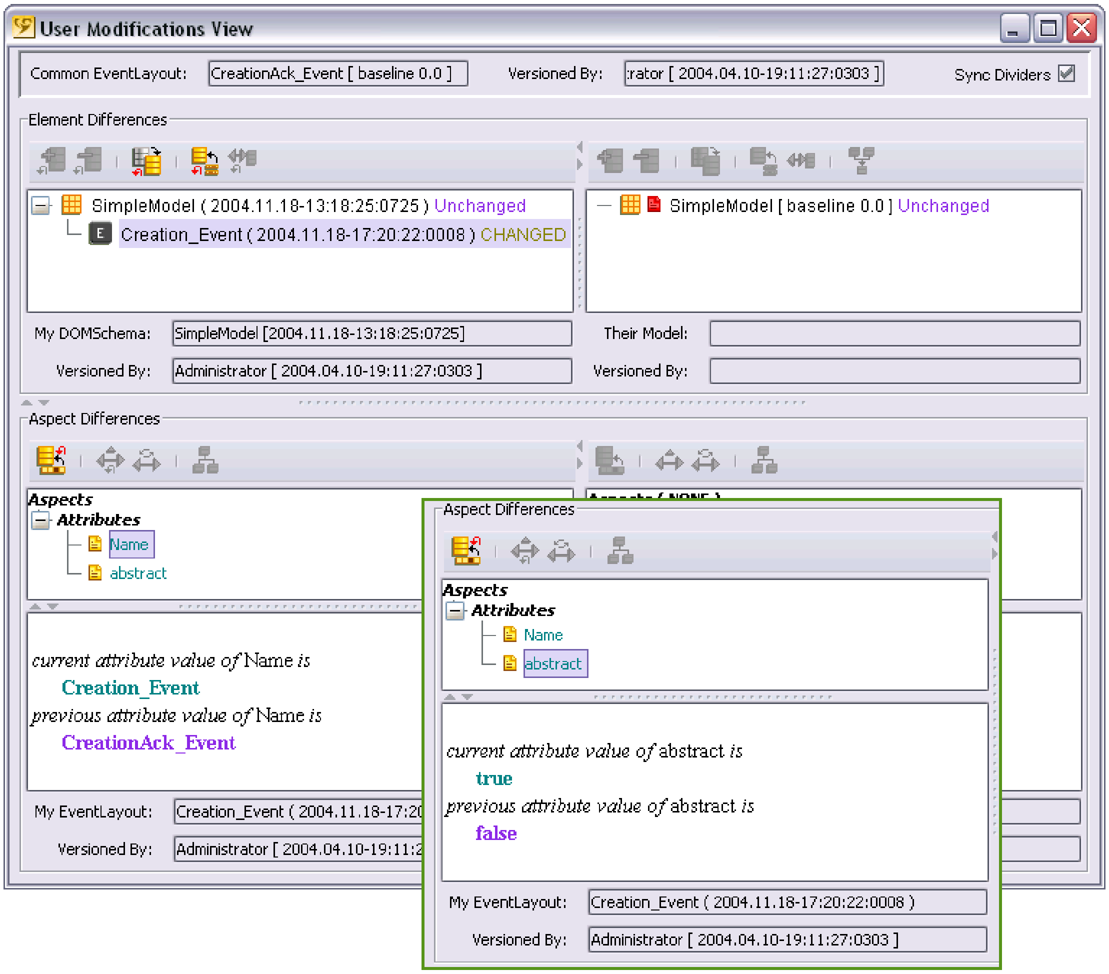
   You can select multiple artifacts by holding down the Ctrl key while selecting.

Aspect differences

Changes to elements are described by *aspects*. In the Aspect Differences panes of the User Modifications view you see four types of element aspects: associations, attributes, contents and superclass. For a changed element, the aspects of that change describe how the model artifact was changed in the current context.

Attribute change aspects

Because Cogility Modeler artifacts are UML entitites, they have attributes. For example, an event artifact, like all model artifacts, has a name. If you change that name, the aspect of that change (from the original name to the new name) displays in the aspects pane of the User Modifications view. In the following picture, the event's name has changed from CreationAck_Event to Creation_Event. When you click on the attribute in the aspects pane, the change is described in the pane below it. Also, the event's abstract attribute (a Boolean that indicates whether the event is an abstract event definition) has been changed.



User Modifications View

Common EventLayout: CreationAck_Event [baseline 0.0]
Versioned By: :rator [2004.04.10-19:11:27:0303]
Sync Dividers ☑

Element Differences

SimpleModel (2004.11.18-13:18:25:0725) Unchanged
Creation_Event (2004.11.18-17:20:22:0008) CHANGED

SimpleModel [baseline 0.0] Unchanged

My DOMSchema: SimpleModel [2004.11.18-13:18:25:0725]
Versioned By: Administrator [2004.04.10-19:11:27:0303]

Their Model:
Versioned By:

Aspect Differences

Aspects
Attributes
Name
abstract

current attribute value of Name is
Creation_Event
previous attribute value of Name is
CreationAck_Event

My EventLayout: Creation_Event (2004.11.18-17:2
Versioned By: Administrator [2004.04.10-19:11:2

Aspect Differences

Aspects
Attributes
Name
abstract

current attribute value of abstract is
true
previous attribute value of abstract is
false

My EventLayout: Creation_Event (2004.11.18-17:20:22:0008)
Versioned By: Administrator [2004.04.10-19:11:27:0303]

You can revert both changes to the last turn-over version by clicking the revert all attribute value applications button under Element Differences, as described in "Reverting elements" on page 12. However, if you want to revert one attribute change while keeping another, you use the buttons under the Aspect Differences pane.

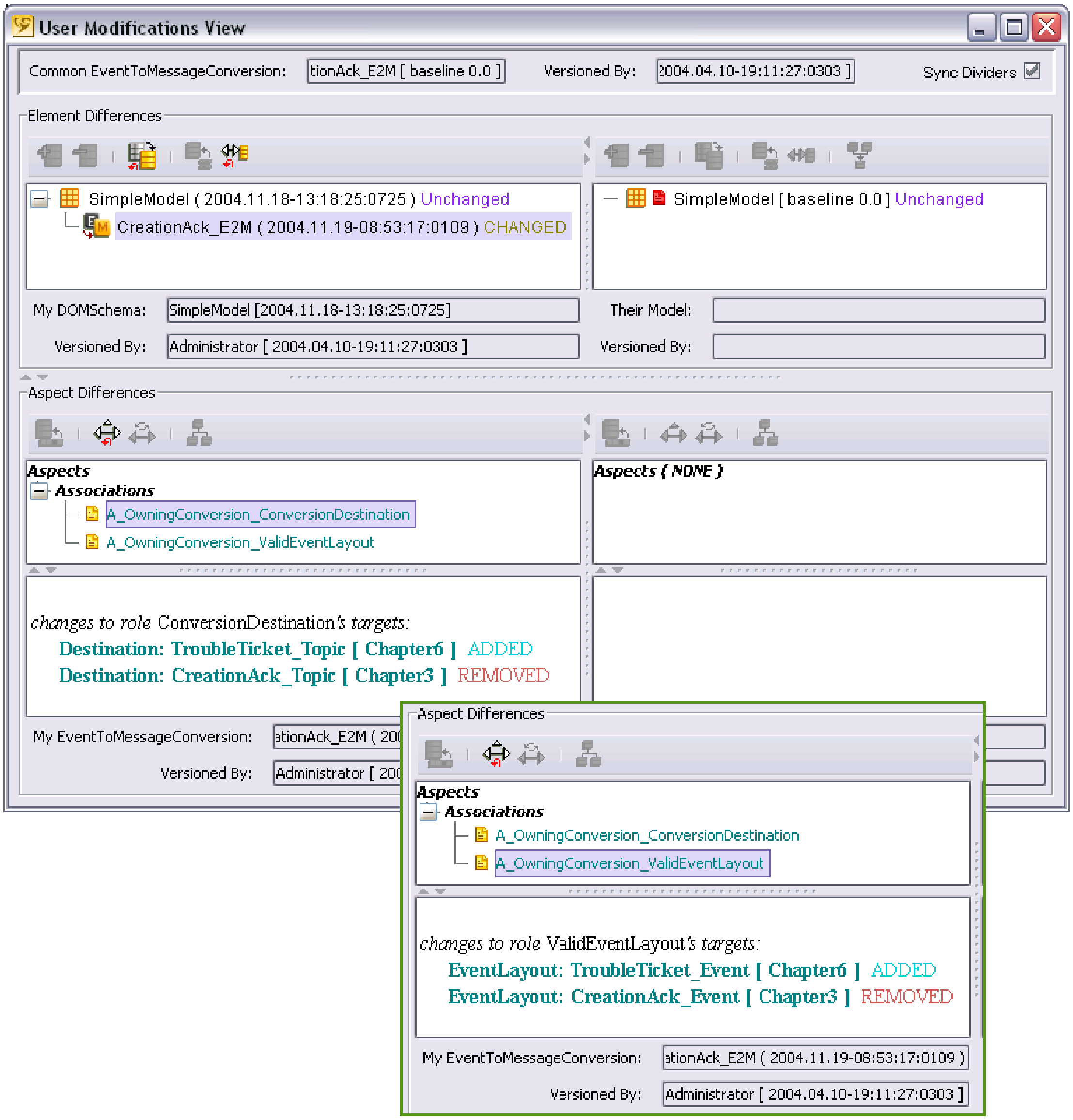
Reverting attribute aspect changes

To revert changes to specific element attributes:

1. Select the element attribute and click the revert attribute value button .

Association change aspects

Cogility Modeler artifacts are associated with other artifacts by means of a UML association. When you change an artifact, you may also change its association with other artifacts. For example, an event to message conversion artifact is associated with a message destination artifact. If you change the message destination, you are changing the association, and this shows up in the aspects pane when you open the User Modifications view. In the picture below, the CreationAck_E2M conversion's message destination has changed from CreationAck_Topic to TroubleTicket_Topic, as described by the ConversionDestination association in the Aspect Differences. Also, the conversion artifact's associated event has changed from CreationAck_Event to TroubleTicket_Event.



User Modifications View

Common EventToMessageConversion: tionAck_E2M [baseline 0.0]
Versioned By: 2004.04.10-19:11:27:0303]
Sync Dividers ☑

Element Differences

SimpleModel (2004.11.18-13:18:25:0725) Unchanged
CreationAck_E2M (2004.11.19-08:53:17:0109) CHANGED

SimpleModel [baseline 0.0] Unchanged

My DOMSchema: SimpleModel [2004.11.18-13:18:25:0725]
Versioned By: Administrator [2004.04.10-19:11:27:0303]

Their Model:
Versioned By:

Aspect Differences

Aspects
Associations
A_OwningConversion_ConversionDestination
A_OwningConversion_ValidEventLayout

Aspects (NONE)

changes to role ConversionDestination*'s targets:*
Destination: TroubleTicket_Topic [Chapter6] ADDED
Destination: CreationAck_Topic [Chapter3] REMOVED

My EventToMessageConversion: ationAck_E2M (20
Versioned By: Administrator [20

Aspect Differences

Aspects
Associations
A_OwningConversion_ConversionDestination
A_OwningConversion_ValidEventLayout

changes to role ValidEventLayout*'s targets:*
EventLayout: TroubleTicket_Event [Chapter6] ADDED
EventLayout: CreationAck_Event [Chapter3] REMOVED

My EventToMessageConversion: ationAck_E2M (2004.11.19-08:53:17:0109)
Versioned By: Administrator [2004.04.10-19:11:27:0303]

You can revert all association changes to the last turn-over version by clicking the revert all association targets button under Element Differences, as described in "Reverting elements" on page 12. However, if you want to revert one association change while keeping another, you use the buttons under the Aspect Differences pane.

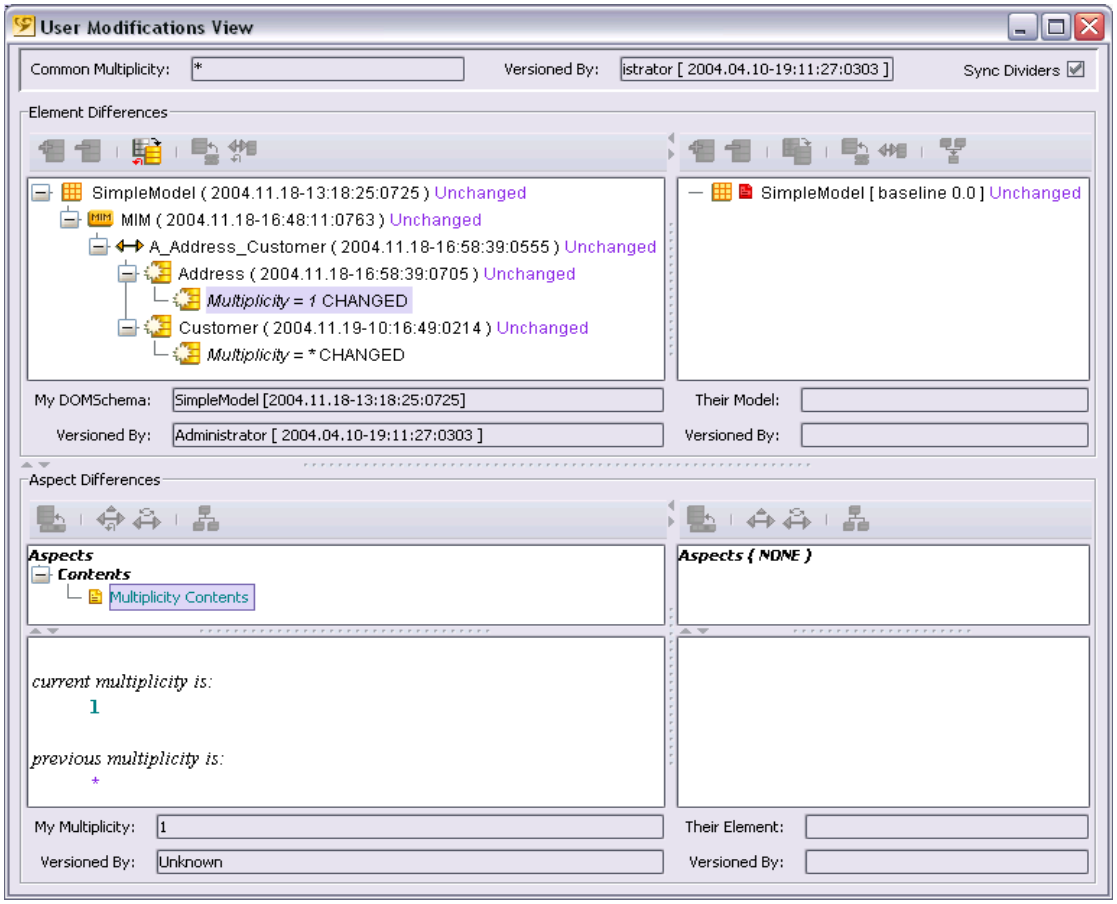
Reverting association aspect changes

To revert changes to specific element associations:

1. Select the element attribute and click the revert association target button .

Content change aspects

Changes to Cogility Modeler artifacts' contents are also described in the Aspect Differences. For example in an association object, the target and source multiplicities are considered contents of that artifact. In the following picture, the A_Address_Customer association's Address target multiplicity has changed from * (many) to 1 and the Customer source multiplicity has changed from 1 to * (many).



Common Multiplicity: *

Versioned By: istrator [2004.04.10-19:11:27:0303]

SimpleModel (2004.11.18-13:18:25:0725) Unchanged
MIM (2004.11.18-16:48:11:0763) Unchanged
A_Address_Customer (2004.11.18-16:58:39:0555) Unchanged
Address (2004.11.18-16:58:39:0705) Unchanged
Multiplicity = 1 CHANGED
Customer (2004.11.19-10:16:49:0214) Unchanged
Multiplicity = * CHANGED

SimpleModel [baseline 0.0] Unchanged

My DOMSchema: SimpleModel [2004.11.18-13:18:25:0725]

Versioned By: Administrator [2004.04.10-19:11:27:0303]

Their Model:

Versioned By:

Aspects
Contents
Multiplicity Contents

Aspects { NONE }

current multiplicity is:
1

previous multiplicity is:
*

My Multiplicity: 1

Versioned By: Unknown

Their Element:

Versioned By:

Each target and source are elements of the association, and as such changes to them are treated in the element differences pane. The Aspect Differences pane can only describe the aspects of the change to each element.

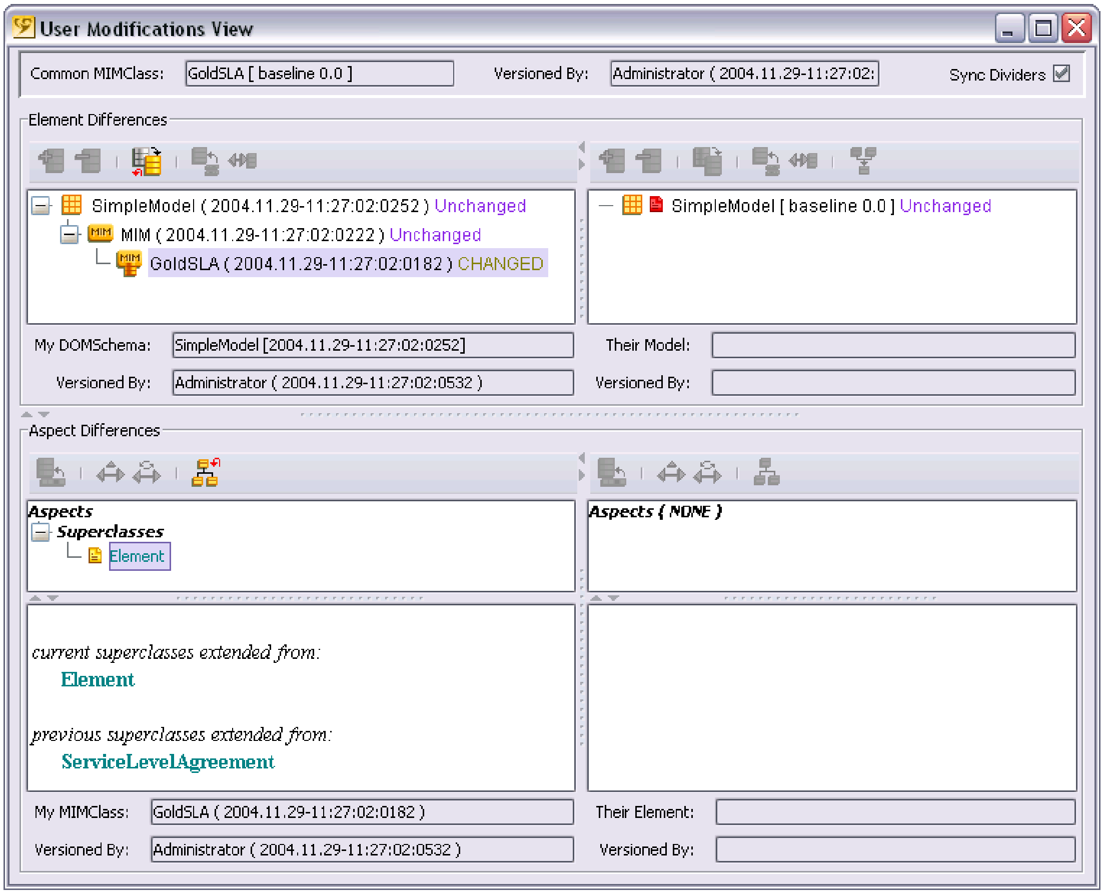
Reverting content aspect changes

To revert the changes to an element's content:

1. Select the element and above the element differences pane, click the revert to parent version button .

Superclass change aspects

A superclass aspect difference occurs when an element's superclass changes. In the picture below, the GoldSLA class' superclass has changed from ServiceLevelAgreement to Element.



Common MIMClass: GoldSLA [baseline 0.0]   Versioned By: Administrator (2004.11.29-11:27:02:   Sync Dividers ☑

SimpleModel (2004.11.29-11:27:02:0252) Unchanged
MIM (2004.11.29-11:27:02:0222) Unchanged
GoldSLA (2004.11.29-11:27:02:0182) CHANGED

SimpleModel [baseline 0.0] Unchanged

My DOMSchema: SimpleModel [2004.11.29-11:27:02:0252]   Their Model:
Versioned By: Administrator (2004.11.29-11:27:02:0532)   Versioned By:

Aspects
Superclasses
Element

Aspects (NONE)

current superclasses extended from:
Element

previous superclasses extended from:
ServiceLevelAgreement

My MIMClass: GoldSLA (2004.11.29-11:27:02:0182)   Their Element:
Versioned By: Administrator (2004.11.29-11:27:02:0532)   Versioned By:

Reverting superclass aspect changes

To revert the changes to an element's superclass:

1. Under Aspect Differences, select the superclass aspect and click the unapply resolution button .

   You may also revert the entire element, as follows:

2. Under Element differences, select the element and click the revert to parent version button .

3

Versions

As you work with a model, you may want to isolate a group of changes so that if you want to revert to the version of the model before the changes, you can do so. Versioning the model or specific model artifacts freezes the model or those artifacts under a specific version label.

Versions created with the versioning operation are called working model versions. They pertain to a model in the current context, and are saved under a turn-over version when the model is turned over. See "Turn-over version" on page 23. Versions created with the version operation are not available upon log-in, but rather from within the model in the current context. You can view version history (see "Version history" on page 21) and refer back to these versions for comparison against the model in the current context (see "Version comparison" on page 32). You may elect to create a version of any particular model artifact following a change to it, or you may version the entire model.

When you version the model in the current context, the new version is created for the configuration map under which you logged into Cogility Modeler. If you loaded a newly created, empty authoring repository, the configuration map under which you logged in is the default configuration map. If you version the model in the current context, the version is saved under the default configuration map, regardless of whether you have already turned over previous versions of the model. Versioning always applies to the model as it was loaded into the current context. The following scenario illustrates this point:

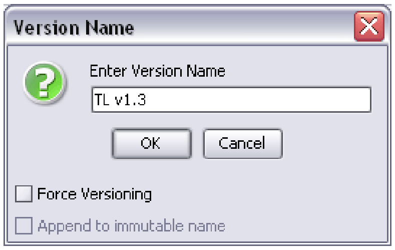
- Load a new, empty authoring repository into the current context.
- Import a model named NewModel.
- Make changes to the NewModel and turn it over.
  - ❑ A new turn-over version of NewModel is now created under the NewModel configuration map.
- Make changes to the NewModel, select the model container and version the entire model.
  - ❑ A new version of NewModel is now created under the default configuration map. Note that this is not a turn-over version, but a modified version, as described below.

This happens because the default configuration map is the original configuration map under which you logged on. Whereas turning over a model explicitly assigns the version to the latest defined configuration map, versioning the model only assigns that version to the configuration map under which you last logged on.

Any specific artifact or any of its elements may have several versions. Also, you can select several artifacts and version them together.

Versioning a model (as opposed to turning it over) or versioning a specific artifact is treated as any other change to the model prior to turning it over. If you version a model or an artifact without turning it over, then log out, the model is treated as a modified model, described in "Modified model" on page 20.

In a collaborative development scenario, where there are several people working on the model, a best practice is for each person to label his changes with his initials, as in Figure 3-1.



Enter Version Name: TL v1.3

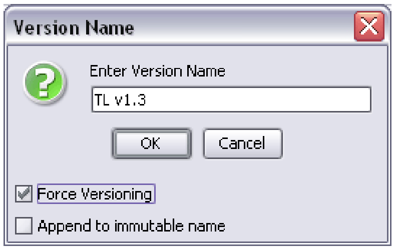
☐ Force Versioning

☐ Append to immutable name

Figure 3-1: Version name

The Force Versioning checkbox applies to container artifacts. See "Forcing versioning" on page 19. You can preserve the current version name and append to it the new name for those artifacts for which you are forcing versioning.

Ordinarily, you don't version immutable artifacts. When you select an immutable artifact and try to version it, the version name associated with that object won't change unless you force versioning. So forcing versioning is useful if you want to change the version name of an artifact without actually changing the artifact. To force versioning, check the Force Versioning checkbox in the Version Name dialog as shown in Figure 3-2.



Enter Version Name: TL v1.3

☑ Force Versioning

☐ Append to immutable name

Figure 3-2: Force versioning

When you version a container artifact, forcing versioning applies the version name to the selected container artifact and all the artifacts it contains, regardless of whether they are mutable or immutable. Otherwise, if you do not check this box, only the mutable artifacts contained in the selected artifact are given the version name. You can preserve the current version name and append to it the new name for those artifacts for which you are forcing versioning.

Versioning specific artifacts

To version specific artifacts:

1. In Cogility Modeler's tree view, select the objects you wish to assign to the version and click the version selected objects button .
2. In the **Version Name** dialog, enter a version name and click **OK**.

Note that after versioning specific artifacts of a container, those artifacts are shown as immutable. However, their container artifacts are still shown as mutable. Thus, you could still revert the changes in the User Modifications view (see "User modifications" on page 9).

Forcing versioning

To force versioning on all artifacts in a container:

1. In Cogility Modeler's tree view, select the artifact that contains changed (mutable) objects and click the version selected objects button .
2. In the **Version Name** dialog, enter a name for the version.
3. Check the **Force Versioning** checkbox.

   The new version name replaces the original name. You can preserve the original version name and append the new name to it in the next step.
4. To append the new version name, check the **Append to immutable name** checkbox.

   In the picture below, the Customer object and all of the attributes it contains have a new version name following the force version operation where the Append to immutable name checkbox is unchecked.

Note that although the Customer object does not appear in italics, indicating that it is no longer mutable, the changes to it may be reverted in the User Modifications view (see "User modifications" on page 9).

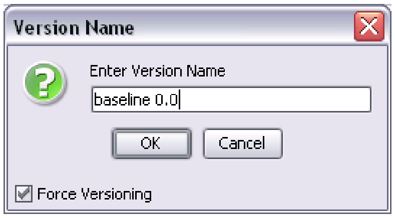
Versioning an entire model

Versioning an entire model is no different than versioning any other container artifact. It is useful if you have imported a model that you want to use as the baseline for a new generation of versions. Versioning an entire model resets the version names for the artifacts in the model.

To version an entire model:

1. In Cogility Modeler's tree view, select the model container and click the version selected objects button .
2. In the Version Name dialog box, enter a version name.
3. Check the Force Versioning checkbox.

   See "Forcing versioning" on page 19.



Version Name

Enter Version Name

baseline 0.0

☑ Force Versioning

4. Click **OK**.

All of the artifacts in the model now have the same version name.

Revising version information

This operation returns the selected element to the mutable state and places the default time stamp in the version label.

To revise version information:

1. In Cogility Modeler's tree view, select the element or container.
2. From the **Selection** menu, choose **Versioning > Revise**.

Duplicate elements and containers

This function duplicates an element or a container and all enclosed elements. The duplicated elements are connected by relationships that match those of the originals. Associations from the originals to objects not duplicated are recreated for the duplicated objects, and in-bound aggregates are recreated (so the new object is owned by the same object as the original). The duplicated element receives the same name as the original with "Copy" appended to the name. Note that this appendage is separated by a space, making the element unusable in action semantics. You must change the name in order to use it.

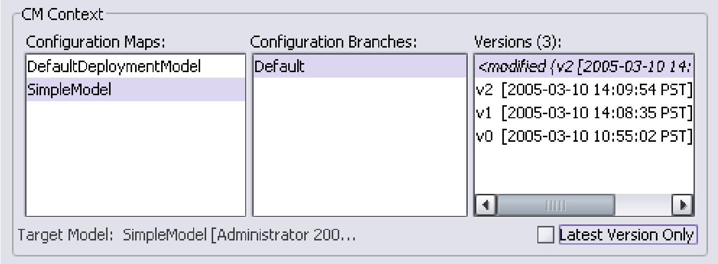
Duplicating elements or containers

To duplicate an element or container:

1. In Cogility Modeler's tree view, select the element or container.
2. From the **Selection** menu, choose **Versioning > Duplicate**.
3. In the editor view, in the **Name** field, change the name.

Modified model

You may log out of Cogility Modeler at any time. If you have made changes to the model in the current context without turning it over, those changes are effectively "saved" for you when you next log in. The changed model is listed as a version with the word *modified* in italics at the top of the Versions list of the log in screen.



CM Context

Configuration Maps:
DefaultDeploymentModel
SimpleModel

Configuration Branches:
Default

Versions (3):
<modified {v2 [2005-03-10 14:
v2 [2005-03-10 14:09:54 PST]
v1 [2005-03-10 14:08:35 PST]
v0 [2005-03-10 10:55:02 PST]

Target Model: SimpleModel [Administrator 200...

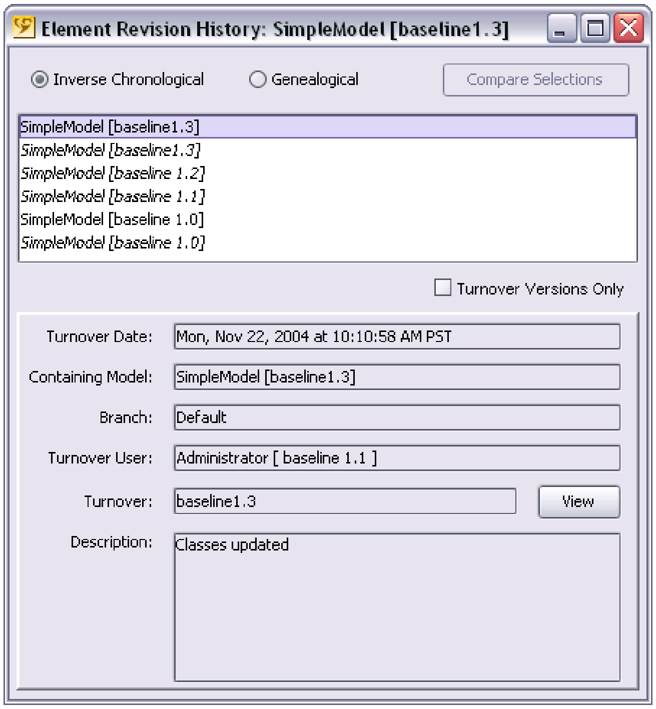
□ Latest Version Only

As with versioning, discussed above, when you log out of a modified model, that modified version is assigned to the configuration map under which you last logged on. If you imported a model into the current context, and the repository was empty when you logged on, when you log out without turning over the changed model, then log in again, the modified version appears in the default map. In the picture above, the modified version was saved under the SimpleModel configuration map because it was under that map that the user last logged on.

Version history

For any artifact or any element in your model, you can view the version history. Both the model's turn-over versions and model versions are available for comparison through the Element Revision History window.

The list of versions includes both working model versions in italics and turn-over versions not italicized. By default, the list is shown in inverse chronological order, whereby the latest version is listed at the top. You can view the list in genealogical order which presents the versions in terms of their derivations. This may be useful if the different versions' timestamp information results from different users creating different versions in different time zones.



Element Revision History: SimpleModel [baseline1.3]

● Inverse Chronological ○ Genealogical

SimpleModel [baseline1.3]
SimpleModel [baseline1.3]
SimpleModel [baseline 1.2]
SimpleModel [baseline 1.1]
SimpleModel [baseline 1.0]
SimpleModel [baseline 1.0]

□ Turnover Versions Only

Turnover Date: Mon, Nov 22, 2004 at 10:10:58 AM PST
Containing Model: SimpleModel [baseline1.3]
Branch: Default
Turnover User: Administrator [baseline 1.1]
Turnover: baseline1.3
Description: Classes updated

Figure 3-3: Element revision history

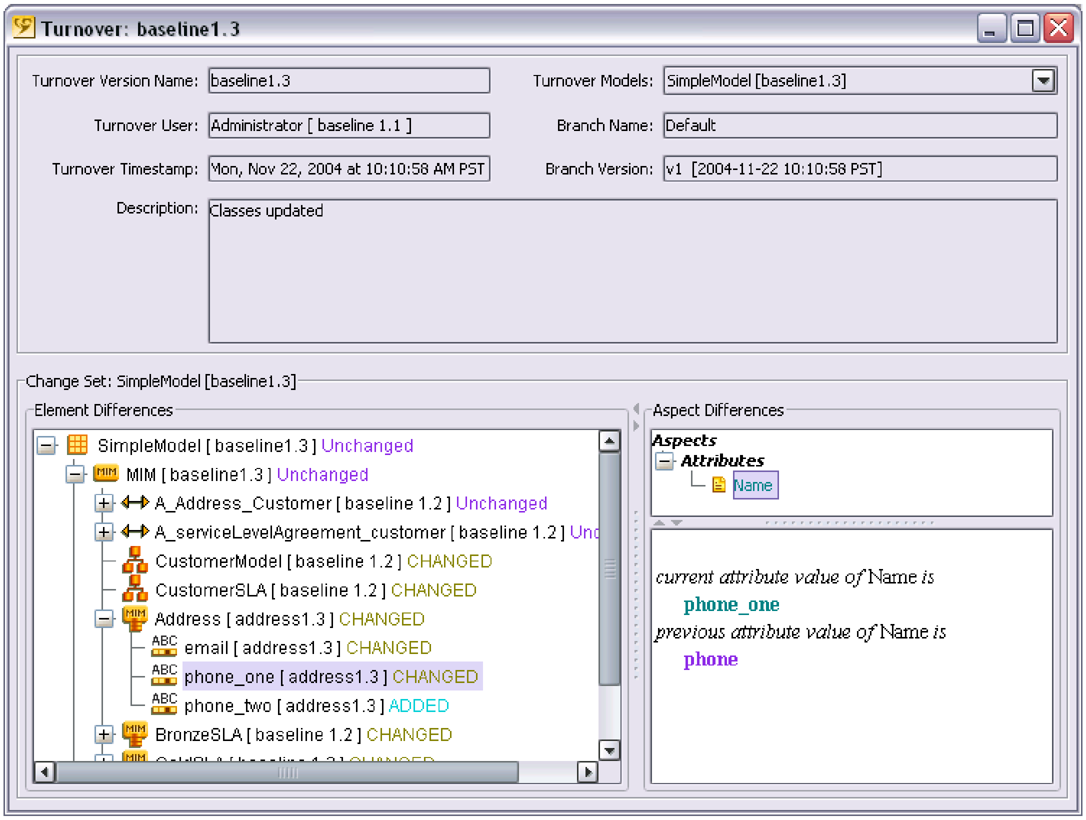
In the picture above, the SimpleModel [baseline1.3] is a turn-over version. Only turn-over versions have a Turnover name and a Description, the last two fields. Working model versions do not have this information. You can further view the element changes to a turn-over version by clicking the View button. See "Viewing turn-over version element history" on page 23.

You can compare any two versions in the list, be they model versions or turn-over versions. In the comparison window, you can view element changes aspects for the elements of of both turn-over versions and working model versions. See "Working with model version history" on page 22.

In the element revision history window, you can view the details (aspects) of changes to turn-over version elements. See "Viewing turn-over version element history" on page 23. This is akin to viewing user modifications of the model in the current context (see "User modifications" on page 9). Note that the changes to working model versions are not available through the element revision history window. To work with these aspects, see "Version comparison" on page 32.

In the Aspect Differences pane you may see any of four types of element aspects: associations, attributes, contents and superclass. For a changed element, the aspects of that change describe how the model artifact was changed in the current context. For more information on aspect types, see

"Aspect differences" on page 13. In the example below, the name attribute of the phone_one element has changed from phone to phone_one.



Turnover Version Name: baseline1.3

Turnover Models: SimpleModel [baseline1.3]

Turnover User: Administrator [baseline 1.1]

Branch Name: Default

Turnover Timestamp: Mon, Nov 22, 2004 at 10:10:58 AM PST

Branch Version: v1 [2004-11-22 10:10:58 PST]

Description: Classes updated

Change Set: SimpleModel [baseline1.3]

Element Differences

```
SimpleModel [ baseline1.3 ] Unchanged
  MIM [ baseline1.3 ] Unchanged
    A_Address_Customer [ baseline 1.2 ] Unchanged
    A_serviceLevelAgreement_customer [ baseline 1.2 ] Unc
    CustomerModel [ baseline 1.2 ] CHANGED
    CustomerSLA [ baseline 1.2 ] CHANGED
    Address [ address1.3 ] CHANGED
      email [ address1.3 ] CHANGED
      phone_one [ address1.3 ] CHANGED
      phone_two [ address1.3 ] ADDED
    BronzeSLA [ baseline 1.2 ] CHANGED
```

Aspect Differences

```
Aspects
  Attributes
    Name
```

current attribute value of Name *is*
phone_one
previous attribute value of Name *is*
phone

Figure 3-4: Turn-over version detail window

Viewing element revision history

To view element revision history:

1. In Cogility Modeler's tree view, select the model artifact for which you want to view the version history.
2. From the **Selection** menu, choose **CM Operation** > **Show Element Revisions**, or click the element revisions button .
3. To see all versions, not just turn-over versions, uncheck the **Turnover Versions Only** checkbox.
4. Click the **Genealogical** radio button, if desired, or leave the default **Inverse Chronological** radio button checked.
5. Select the version to see its context information.

Working with model version history

To compare versions through the element history screen:

1. Open the element revision history window, as described in "Version history" on page 21.
2. Select any two versions in the list and click the Compare button.
3. See "Version comparison" on page 32.

Viewing turn-over version element history

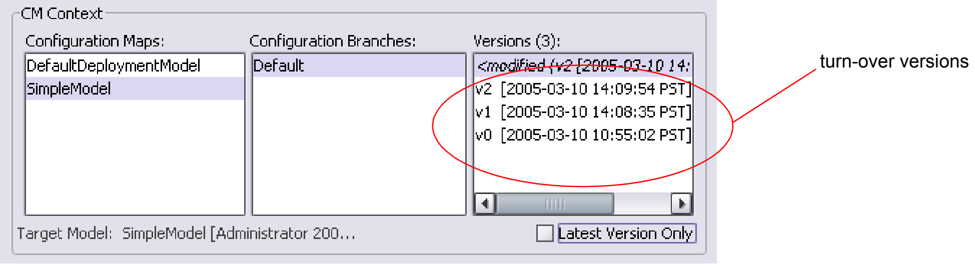
To view turn-over version element change aspects:

1. Open the element revision history window, as described in "Version history" on page 21.
2. Select a turn-over version from the list.

   The turn-over version details window appears. The turn-over version you selected is compared to the model in the current context.
3. Expand the elements in the tree view and select an element.
4. In the Aspect Differences pane, expand the aspect type and select an aspect.

Turn-over version

Once you have finished making changes to a base-line model, you should turn over that model, thus creating a new turn-over version in the authoring repository. When you turn over a model, all of the pending changes to it are frozen and the artifacts previously mutable become immutable. Each turn-over version you create becomes available for loading into the current context upon log-in. In the following picture, v0, v1, v2 and v3 are all turn-over versions that were created during the model's lifecycle.



CM Context

Configuration Maps:
DefaultDeploymentModel
SimpleModel

Configuration Branches:
Default

Versions (3):
<modified {v2 [2005-03-10 14:
v2 [2005-03-10 14:09:54 PST]
v1 [2005-03-10 14:08:35 PST]
v0 [2005-03-10 10:55:02 PST]

Target Model: SimpleModel [Administrator 200...

□ Latest Version Only

turn-over versions

Figure 3-5: Turn-over versions

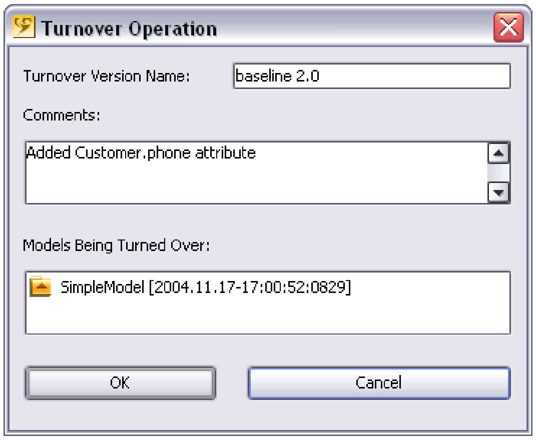
Turning over a model creates a new turn-over version of the model under change management. A model you have created from scratch is placed under change management the first time you turn it over, and it become the *head revision*. Also, the working model versions you have created, either of the entire model or of specific model artifacts, are saved under the turn-over version.

When you import a model from a file (as described in "Importing a base-line for updating" on page 6), before you can turn it over you must explicitly place that model under change management. For a model created from scratch, the model container, and thus the configuration map and version, were created within the current context. But for an imported model, the model container was created in another context and the current context does not have a map and version for it. You must explicitly create these for the model in the current context by closing Cogility Modeler after you import the model and turn it over, and then logging in again with that model in the current context. Doing so establishes the model within map and version of the current context. See "Importing a base-line for updating" on page 6.

If the common parent between a new turn-over version of a model and the head revision of the model cannot be determined, the latest head revision becomes the common parent version by default.

A model is ready to be turned over immediately following a change to any of its artifacts. You may turn over a model with inconsistencies, though you should not push that model into execution.

When you turn over a model, the turn-over dialog lists all of the models in the current context with pending changes that require a turn over. This list includes both integration models and deployment models. These are shown in the Models Being Turned Over field in the figure below.



Turnover Operation

Turnover Version Name: baseline 2.0

Comments:

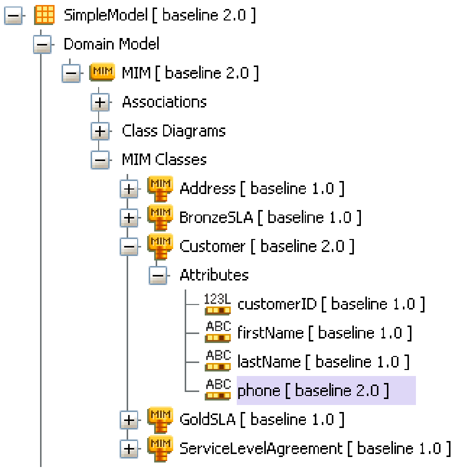
Added Customer.phone attribute

Models Being Turned Over:

SimpleModel [2004.11.17-17:00:52:0829]

Figure 3-6: Models being turned over

Notice that for each artifact added, subtracted or changed, its container receives the same version label. In the picture below, the phone attribute was added to the SimpleModel which was then turned over. The new label, "baseline 2.0" applies to the phone attribute and each of its containers: the Customer class, the MIM and the SimpleModel artifacts. All of the other artifacts and retain their original version label (in this case "baseline 1.0").



```
SimpleModel [ baseline 2.0 ]
  Domain Model
    MIM [ baseline 2.0 ]
      Associations
      Class Diagrams
      MIM Classes
        Address [ baseline 1.0 ]
        BronzeSLA [ baseline 1.0 ]
        Customer [ baseline 2.0 ]
          Attributes
            customerID [ baseline 1.0 ]
            firstName [ baseline 1.0 ]
            lastName [ baseline 1.0 ]
            phone [ baseline 2.0 ]
        GoldSLA [ baseline 1.0 ]
        ServiceLevelAgreement [ baseline 1.0 ]
```

Figure 3-7: Container and turned-over artifacts

Turning over a model or model artifacts

To turn over a model or model artifacts:

1. With any model artifact selected, click the turn over button .
2. In the **Turnover Operation** dialog, for **Turnover Version Name**, enter a meaningful version name.
3. In the **Comments** field, describe the changes that distinguish the turn-over version and click **OK**.

4

Model comparison

Either the individual model elements or the entire model may be compared against other versions of the same model elements or model. To compare the version in the current context with another version, one that other developers have worked on, that other version must be imported into the repository. For an object or model to become available for import, it must be first exported.

You can configure the Cogility Modeler interface to reflect your preference for the format by which you export and import model files. The following configuration parameter establishes this preference:

```
com.cogility.ImportExport.PrefersSingleFileOptions
```

The default is false. If false, the icons for the file set selections appear on the menu items and in the toolbar. When true, the single file icons appear on the menu items and in the toolbar. Regardless of this this setting, the menu will display selections for both types of formats; only the display of the icons in the menu is affected. The toolbar will only display icons (buttons) for the preferred format.

Model export

You can export a model to a single file or set of files, which may subsequently be imported. Exporting and importing are the means by which a model may be shared among several installations of Cogility Studio. There are two formats into which a model may be exported: a single file or a hierarchical file set. The single file format is suitable for small models such as the example models in the %DCHOME\Examples folder. For larger, real-world models, the file set format is recommended.

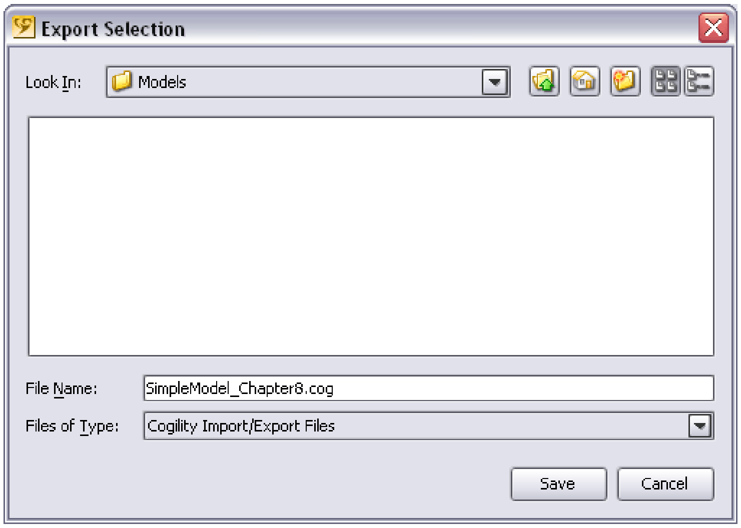
Single file format

A single file format exported model is saved in a a compressed XML file with a .cog extension. Compressed XML is a data storage format wherein the normally verbose XML file is compressed. While this saves storage space, you cannot view a compressed XML file as a normal text file until it is first uncompressed. Model files are uncompressed when they are imported into Cogility Modeler.

The single file format is suited to small example models. When larger models are exported in this format, they may be slow to import because the single file format does not take advantage of delta files the way the file set format does. Each time a model is exported as a single file, the entire model, not just the changed elements since the last export, are saved to the file. See "File set format" on page 26 for more information.

You must turn over your model or version the model before exporting it, a model with mutable artifacts cannot be exported. The default file name for the model appends the version name to the model name preceded by an underscore, as shown in the following figure. You can always save the model under a different name. Preserved with the model's metadata are the model version and the

Cogility Studio version under which it was created. You can see this information when you import the model. See “Model import” on page 27.



File Name: SimpleModel_Chapter8.cog

Files of Type: Cogility Import/Export Files

Figure 4-8: Single file export

Exporting as a single file

To export a model as a single file:

1. Turn over the model or version any mutable artifacts.

   See “Turn-over version” on page 23 and “Versions” on page 17. You cannot export a model with mutable artifacts.
2. In Cogility Modeler’s tree view, select the model container and from the **Selection** menu, choose **Import/Export** > **Export to Single File**.
3. Navigate to the location where you want to save the file, in the **File Name** field, enter a name and click **Save**.

File set format

In the file set format, the model application is saved in a hierarchical structure of XML files and subfolders that describes the model artifacts. The files are human-readable XML files, not the compressed XML format used for a single file (see “Single file format” on page 25). The files in a file set have a .cogs extension to connote that they belong to a file set. The file set format allows Cogility Modeler to isolate changed elements in a model. Before exporting the model, Cogility Modeler checks the model against the previously exported file set and creates a delta, a file set describing only the differences between the model and the existing file set. Reducing exporting and importing to only the delta greatly improves export and import speed and reduces integration problems with an enterprise configuration management system.

Configured file set

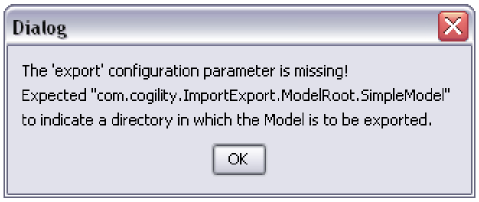
Before you can export a model file set, you must establish a location for the file set by setting the following configuration parameter:

```
com.cogility.ImportExport.ModelRoot.<ModelName>=<location>
```

In this configuration parameter, for `<ModelName>` substitute the name of the model you are exporting. Note that you may have a configuration parameter defined for each of several models, if you are working with several models. For the value of `<location>`, substitute the directory on your system where you want to save the file set. Use forward slashes, not back slashes in the path to the location as these will be mistaken for escape characters. For example, a file set for the SimpleModel will be saved to D:/temp as specified in the following parameter:

```
com.cogility.ImportExport.ModelRoot.SimpleModel=D:/temp
```

Note that there is no default defined for this parameter. You must define it in order to export a file set. If you have not defined the parameter, you will see the following waring when you try to export the model:



The 'export' configuration parameter is missing!
Expected "com.cogility.ImportExport.ModelRoot.SimpleModel"
to indicate a directory in which the Model is to be exported.

Figure 4-9: Export configuration parameter missing

The location you establish with the parameter above is intended as the location on your local system for a model that you check into a shared store in a software configuration management system (SCM). Once you have created the file set, you can check it into your organization's SCM system to serve as the base-line model. From there, other developers may import the file and merge the changes with their own versions of the model. See "Model import" on page 27.

Exporting to a configured file set

To export your model to a configured file set:

1. Set the configuration parameter for the location of the model file set.

   See "Configured file set" on page 26.
2. If Cogility Modeler is already running, reload the configuration parameters.

   See "Reloading configuration parameters into Cogility Modeler" on page 25 of the guide, *Modeling with Cogility Studio*.
3. Turn over the model or version any mutable artifacts.

   See "Turn-over version" on page 23 and "Versions" on page 17. You cannot export a model with mutable artifacts.
4. In Cogility Modeler's tree view, select the model container and, from the **Selection** menu, choose **Import/Export** > **Export to Configured File Set**.

Model import

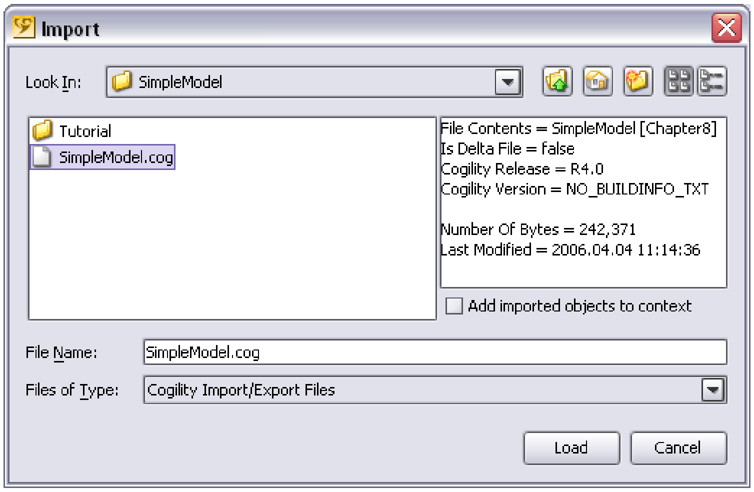
Once a model is saved as a file or file set, it may be imported into a repository for comparison against a model in the current context. As described in "Base-line model" on page 5, you create a model that other developers subsequently use as a base-line from which to continue building the model.

The base-line model or object is held in the organization's SCM system. When a developer is ready to update the base-line, he locks the file on the SCM system, imports it into his own workspace, merges the changes he has made in his own version of the model with the base-line, then checks the updated base-line back into the SCM system.

When importing a model from a nonconfigured file set or from a single file, you can see the model metadata including the following fields:

- File Contents - the model or model artifacts in the file.
- Is Delta - true only if the file contents are a delta of a changed model. See “Model export” on page 25.
- Cogility Release - the release of Cogility Studio under which the model was created.
- Cogility Version - the version of Cogility Studio under which the model was created.
- Number of Bytes - the file size.
- Last Modified - the date the file was created or modified.

These are shown in the figure below.



Look In: SimpleModel

File Contents = SimpleModel [Chapter8]
Is Delta File = false
Cogility Release = R4.0
Cogility Version = NO_BUILDINFO_TXT

Number Of Bytes = 242,371
Last Modified = 2006.04.04 11:14:36

☐ Add imported objects to context

File Name: SimpleModel.cog

Files of Type: Cogility Import/Export Files

Figure 4-10: Model import

Models may be stored either as single file or as a file set. See “Single file format” on page 25 and “File set format” on page 26. For each of these formats, there are two options for importing.

Note: It is possible to import a model from a current release into a previous release (for example, importing a R6.0 model into a R5.0 repository. Objects created in the newer release, that are not supported in the previous release, are not imported. The import reports objects that cannot be imported.

Single file format import

When you import a model from a single file format, you may either import the model into the current context - thereby replacing the extant model - or you may import the model into the repository for comparison against the model in the current context.

There are two options for importing single file format models:

- Import from single file - for importing models saved as single files in Cogility Studio version 3.1 or later. See “Importing from a single file” on page 29.
- Import from R2.x formatted file - for importing models saved as single files in Cogility Studio version 2.1 or earlier. If you have created a model under this older version of Cogility Studio, use this option to import the model into the latest version and bring it up to date. See “Importing from an R2.x formatted file” on page 29.

Importing from a single file

To import a model from a single file:

1. From the **Selection** menu, choose **Import/Export > Import from Single File**.
2. To import a model into the current context, follow the steps in "Importing a single file model into the current context" on page 29.
3. To import a model for comparison against the model in the current context, follow the steps in "Importing a single file model for comparison" on page 30.

Importing from an R2.x formatted file

To import a model from an R2.x formatted file:

1. From the **Selection** menu, choose **Import/Export > Import R2.x Formatted File**.
2. To import a model into the current context, follow the steps in "Importing a single file model into the current context" on page 29.
3. To import a model for comparison against the model in the current context, follow the steps in "Importing a single file model for comparison" on page 30.

Importing a single file model into the current context

These steps describe the basic procedure for importing a model into Cogility Modeler. Use these steps for working with the example models described throughout the documentation. If you want to import a base-line model from a shared store on an enterprise configuration management system, see "Importing a base-line for updating" on page 6.

Following these steps will cause any model presently in the current context to be replaced with the imported model. To import a model into the repository for comparison against the model in the current context, see "Importing a single file model for comparison" on page 30.

To import a model into the current context:

1. Import the model as described in "Single file format import" on page 28.

   You have the following options for importing a model:

   a. "Importing from a single file" on page 29
   b. "Importing from an R2.x formatted file" on page 29.
2. In the dialog, check the **Add imported objects to context** checkbox.

   By default, imported objects are added to the repository without placing them in the current context. When you check this box, the objects are placed into the current context. Because you are importing a base-line model that you then edit, you want the model in the current context. When you import a model for comparison against the model in the current context, you leave the box unchecked. See "Model import" on page 27.
3. Navigate to and select the .cog or .cogs file, and click **Load**.

   Example files are located in the **%DCHOME\Examples** directory.

   The model appears in Cogility Modeler's tree view with a yellow inconsistency flag. The model is inconsistent with the rules of modeling in Cogility Studio because it lacks a configuration map and version, as described in the Consistency tab at the bottom of the window. You must explicitly place the model under change management to clear the inconsistency. See "Placing a model under change management" on page 6.

Importing a single file model for comparison

These steps describe how to import a model for comparison against the model in the current context. It assumes you have an updated and turned-over model in the current context, one that originated from a base-line model whose descendent you now import for comparison. This procedure is different from that for importing the initial base-line model for updating. If you want to import a base-line model, then make changes to it, see "Importing a base-line for updating" on page 6. If you want to import a model into the current context, replacing any extant model, see "Importing a single file model into the current context" on page 29.

To import a single file model for comparison:

1. Import the model as described in "Model import" on page 27.

   You have the following options for importing a model:

   a. "Importing from a single file" on page 29

   b. "Importing from an R2.x formatted file" on page 29.

2. Leave unchecked the **Add imported objects to context** checkbox.

   If you check this box, the imported model replaces the model in the current context (that displayed in Cogility Modeler). Whether or not the box is checked, the imported model is added to the repository and is available as a turn-over version for comparison. Cogility Modeler refers to the model displayed in the current context as the Current Model and to the model selected for comparison as the Other Model.

3. Navigate to and select the .cog or .cogs file and click **Load**.

4. In the dialog that prompts you to make the object visible, click **No**.

   You are given another chance to add the object or delta to the view here, which would replace the model in the current context.

   Cogility Modeler places the imported model in the repository where it is available for comparison against the model in the current context. See "Version comparison" on page 32.

File set format import

When you import a model from a file set, that model replaces the model in the current context automatically. You must, therefore, turn over the extant model before importing another. Once you have imported the model, you have two options for merging the changes from the previous model: you can simply turn over the imported model, at which point the changes made in the previous model will be merged automatically, or you can open the conflicts and differences window and merge the changes yourself.

There are two options for importing file set format models:

- Import from a configured file set - A configured file set is one for which the location of the file set is established in a configuration parameter as described in "Configured file set" on page 26. When you use this option, the file set will be imported automatically from the location without requiring any navigation or selection. Use this option to import a model from a shared store in a configuration management system. See "Importing from a configured file set" on page 30.
- Import from a nonconfigured file set - Models may be stored in locations not specified by a configuration parameter. This option allows you to navigate to that location and select the file set. See "Importing from a selected file set" on page 31.

Importing from a configured file set

To import a model from a configured file set:

1. Turn over the model presently in the current context.

2. From the **Selection** menu, choose **Import/Export > Import from Configured File Set**.

   The model originally in the current context is replaced by the model from the configured location. See "Configured file set" on page 26. The model previously in the current context now resides only in the current context.

3. To merge the imported model now in the current context with the previous model now in the repository, follow the steps in "Merging changes to an imported model" on page 31.

Importing from a selected file set

To import a model from a selected file set:

1. From the **Selection** menu, choose **Import/Export > Import from Selected File Set**.
2. Navigate to and select the .cogs file, and click **Load**.

   The model originally in the current context is replaced by the model from the configured location. See "Configured file set" on page 26. The model previously in the current context now resides only in the current context.

3. To merge the imported model now in the current context with the previous model now in the repository, follow the steps in "Merging changes to an imported model" on page 31.

Merging changes to an imported model

These steps assume you have imported a model from a file set as described in "Importing from a configured file set" on page 30 and "Importing from a selected file set" on page 31.

To merge changes to an imported model:

1. To merge the changes automatically, skip to step3, below.
2. To compare the imported model now in the current context with the previous model now in the repository, open the Conflicts & Differences window.

   See "Version comparison" on page 32.

3. Turn over the model in the current context.

   See "Turn-over version" on page 23. Turning over the imported model in the current context will cause any pending differences between it and the previous model to be merged automatically if the changes were not compared and merged manually as in step 2, above and if there are no conflicts. If there are conflicts, the Conflicts & Differences window will open to allow you to resolve the conflicts. See "Differences and conflicts" on page 37.

Repository model import

You can add a model container to the current context from the repository. You may have imported the model into the repository without adding it to the view originally, but now you want to work with it, or you have accidentally deleted your model from the current context.

If there is a model already in the current context when you perform this operation, the model you add is placed alongside the model in the current context. Just as it is not recommended that you include two or more schematically different models in the same repository, you should not include two different models in the current context. Instead, use a separate repository for each model with all of its versions and work with only that repository in Cogility Modeler. See "Authoring repository" on page 19 of the guide, *Modeling with Cogility Studio*.

Adding a model from the repository

This operation is for placing a new model into the current context. To replace the current model with a different version, see “Model element replacement” on page 34.

To add a model from the repository:

1. In Cogility Modeler click the add model button .

   You can also right-click and select **Import/Export > Add Existing Model from Repository**. The same option is also available from the Selection menu. Note that it makes no difference whether a model already in the current context is selected. The added model is included in the current context alongside any existing model.
2. Select the model container from the list and click **OK**.

The model is placed into the current context. To establish a map and version for the model, you must place it under change management. See “Placing a model under change management” on page 6.

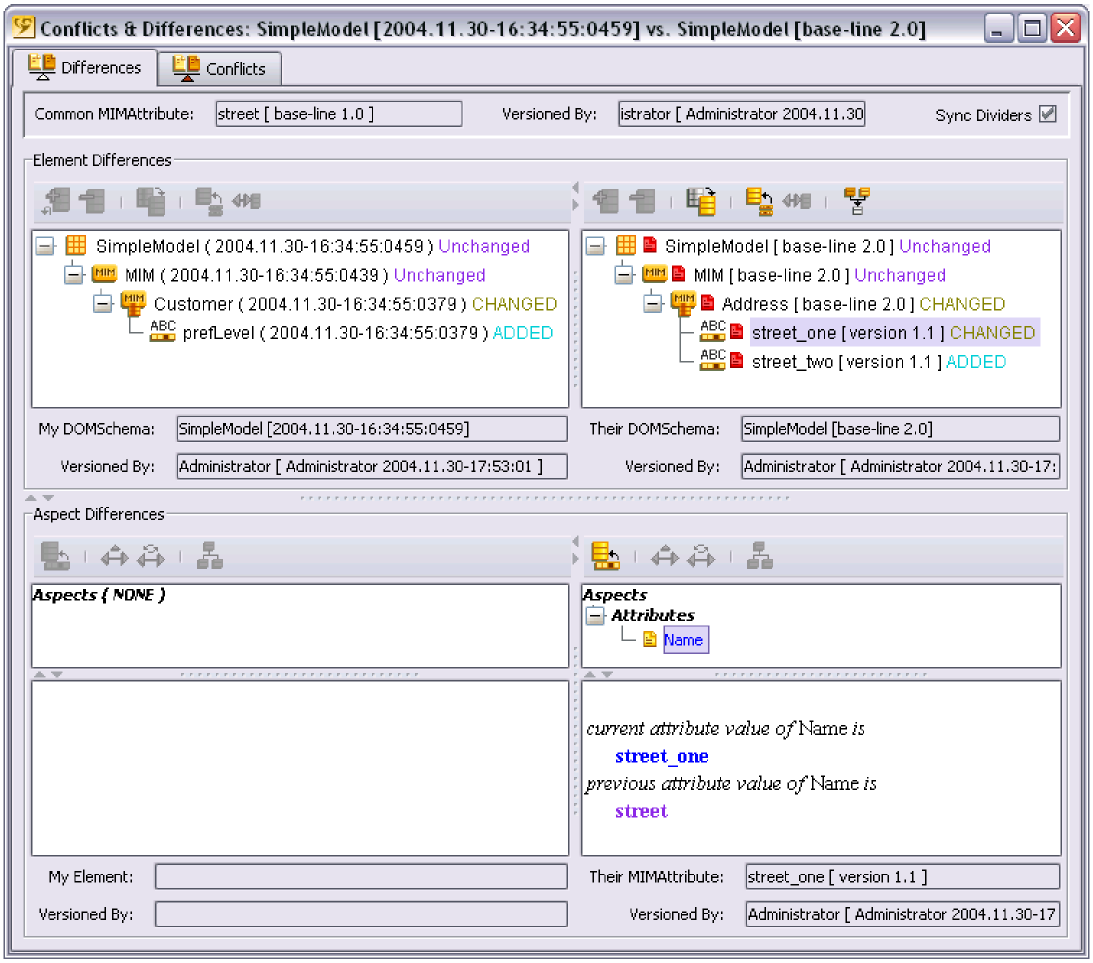
Version comparison

As you develop a model in the current context, you may create several versions, either of specific artifacts or of the entire model. This is described in “Versions” on page 17. You may want to compare the different versions against the model in the current context for the purpose of merging selected changes.

This process also applies to updating a base-line model with changes from a model in the current context. You can import a base-line model into your repository and compare it to the model in the current context. See “Model import” on page 27.

The Conflicts and Differences view displays any differences and conflicts between your working version and a version held in the repository that you select. You can select individual artifacts, such as class attributes, or you can select their containers. Selecting the model object (the top level container) lets you compare all of the differences between the model in the current context and its corresponding version.

Cogility Modeler refers to the model displayed in the current context as the Current Model and to the model selected from the repository for comparison as the Other Model.



Conflicts & Differences: SimpleModel [2004.11.30-16:34:55:0459] vs. SimpleModel [base-line 2.0]

Common MIMAttribute: street [base-line 1.0]

Versioned By: istrator [Administrator 2004.11.30

Sync Dividers ☑

SimpleModel (2004.11.30-16:34:55:0459) Unchanged
MIM (2004.11.30-16:34:55:0439) Unchanged
Customer (2004.11.30-16:34:55:0379) CHANGED
prefLevel (2004.11.30-16:34:55:0379) ADDED

SimpleModel [base-line 2.0] Unchanged
MIM [base-line 2.0] Unchanged
Address [base-line 2.0] CHANGED
street_one [version 1.1] CHANGED
street_two [version 1.1] ADDED

My DOMSchema: SimpleModel [2004.11.30-16:34:55:0459]

Versioned By: Administrator [Administrator 2004.11.30-17:53:01]

Their DOMSchema: SimpleModel [base-line 2.0]

Versioned By: Administrator [Administrator 2004.11.30-17:

Aspects { NONE }

Aspects
Attributes
Name

current attribute value of Name is
street_one
previous attribute value of Name is
street

My Element:

Versioned By:

Their MIMAttribute: street_one [version 1.1]

Versioned By: Administrator [Administrator 2004.11.30-17

Figure 4-11: Conflicts and Differences view

Compare options

The CM Preferences dialog allows you to select the options Cogility Change Management uses when it performs a compare. Click a checkbox to select that option.

The following options are available:

- Report content differences of a diagram as physical conflict
- Warn before applying other version’s changes to mutable version in context
- Separate potential (logical) and physical conflicts
- Do not report potential conflicts
- Clear all in-memory resolution tracking objects after every turnover
- Treat ‘key’ and ‘alias’ as one single attribute called ‘Name’
- Automatically execute a sync compare after each resolution
- Perform garbage collection after each compare

Note: If you deselect “auto sync”, only objects in the left hand pane are refreshed after applying a resolution. Pressing F5 forces a full refresh of the Conflicts and Differences window, allowing you to do a full refresh even when “auto sync” is disabled.

Comparing versions

To compare a version against the model in the current context:

1. In Cogility Modeler’s tree view, select an artifact that you want to compare to a previous version of that same artifact.
2. From the **Selection** menu, choose **Versioning**>**Version**>**Compare with**, or click the compare versions button .

   The list of versions held in the repository displays. The list shows all turn-over versions and all working versions of the model in the current context as well as all turn-over versions and all working versions of any objects or deltas imported into the repository.
3. Select the version you want to compare against that of the model in the current context and click **OK**.

   The Conflicts and Differences view displays any differences and conflicts between your working version in the current context and a version held in the repository that you selected.
4. In the element differences panes, click the plus signs + next to the artifacts to expand the artifacts you want to compare.

You can now use this window to resolve conflicts and differences. See “Differences and conflicts” on page 37.

Model element replacement

You can replace either an entire model container or a selected element in the current context with a version in the repository.

Replacing model elements

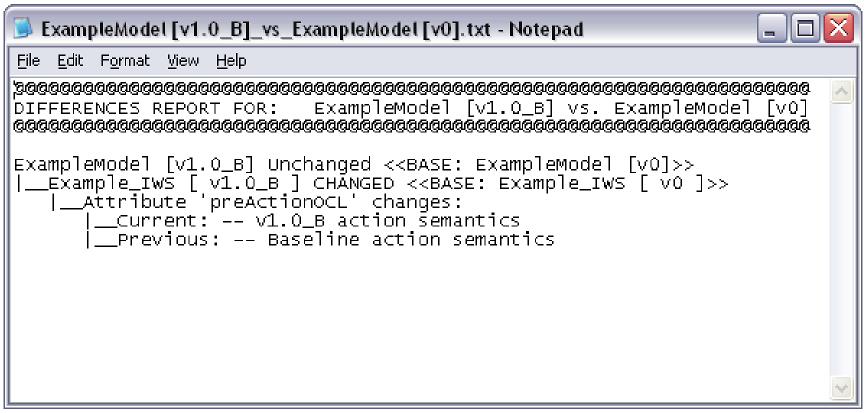
This operation is for replacing an element or model container with a different version. To replace the model container with an entirely different model container, see “Adding a model from the repository” on page 32.

To replace an object in the current context with a version in the repository:

1. In Cogility Modeler’s tree view, select the element
2. From the **Selection** menu, choose **Versioning**>**Version**>**Replace with**, or click the replace button .
3. Select the version to bring into the current context from the Version Select dialog and click OK.

Difference report

The information in the Conflicts and Differences view can be printed to a text file so that model changes can be documented outside of the model repository. You can create a file comparing the parent version of the model to either the Other Model or the Current Model version. A differences report text file is shown below.


ExampleModel [v1.0_B]_vs_ExampleModel [v0].txt - Notepad
File Edit Format View Help

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
DIFFERENCES REPORT FOR:   ExampleModel [v1.0_B] vs. ExampleModel [v0]
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

ExampleModel [v1.0_B] Unchanged <<BASE: ExampleModel [v0]>>
|__Example_IWS [ v1.0_B ] CHANGED <<BASE: Example_IWS [ v0 ]>>
   |__Attribute 'preActionOCL' changes:
      |__Current: -- v1.0_B action semantics
      |__Previous: -- Baseline action semantics
```

Figure 4-12: Differences report file

Creating a differences report file

To create a differences report file:

1. In Cogility Modeler's tree view, select the element and click **Create Diff Report**.
2. In the dialog, select from the following:
    a. **Current** to create a comparison between the Current Model version and the parent version.
    b. **Other** to create a comparison between the Other Model version and the parent version.
3. Navigate to the location where you want the file and click **Save**.

5

Differences and conflicts

"Version comparison" on page 32 describes how to bring up the Conflicts and Differences view to compare two versions. The Conflicts and Differences view separates into two tabs, one to display differences and the other to display conflicts (see "Conflicts" on page 48). The differences tab displays by default. In the left panes, the element and aspect differences for the artifact in the current context are shown. In the right pane, those of the selected version are shown. These panes refer to Current Model elements and aspects and Other Model elements and aspects, respectively. The Current Model elements are those in the current context (what is displayed in Cogility Modeler), and the Other Model elements are those selected from the authoring repository for comparison. You can change the size of the panes by moving the dividing bars between them; to move the bars independently of each other, uncheck the Sync Dividers checkbox.

Element differences

In the top panes the elements (or artifacts) of the Current Model and the Other Model are shown. The elements may be Unchanged, CHANGED, REMOVED or ADDED. Note that although a container may be shown as Unchanged, elements within it may be CHANGED, REMOVED or ADDED. Elements may also be in CONFLICT. For working with these, see "Conflicts" on page 48.

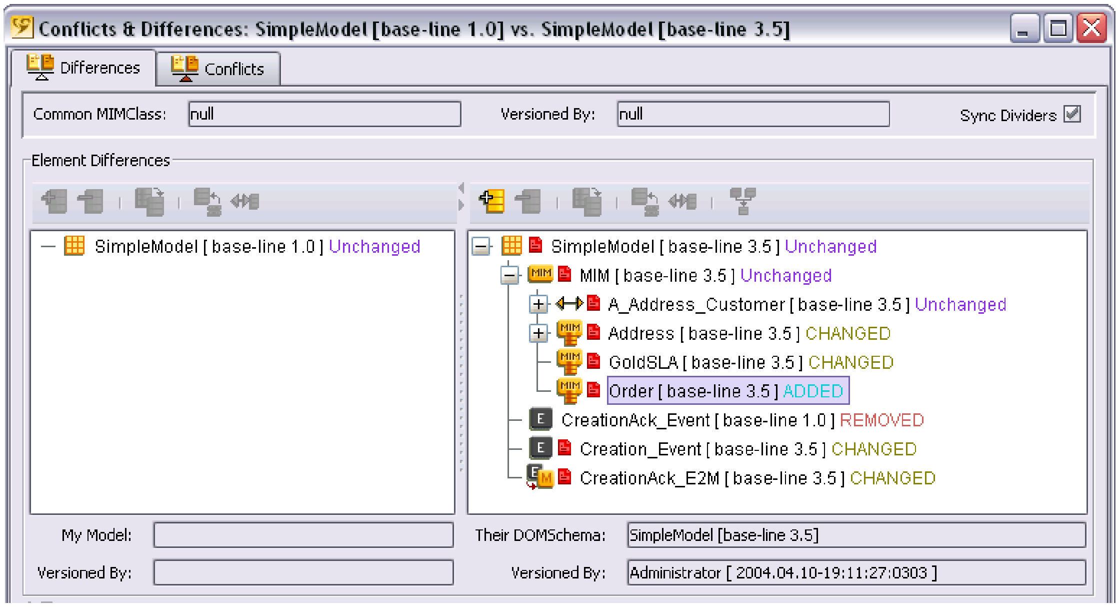
When you select an element the Common Element, Versioned By, Current Model and Other Model fields are populated. The Common Element field describes the original artifact that both versions descended from. The Versioned By field shows the login ID of the user who made the change.

For the elements of the model in the current context (the Current Model elements) in the left pane, you can revert back to the last turn-over version of that element, just as if you were working with the elements in the user modification view. See "Element differences" on page 11.

For the elements of the version you are comparing (the Other Model elements) in the right pane, the buttons above the pane let you bring the changes from the compared version into the model in the current context. Note that when you add, replace or remove an element, this impacts the entire containment tree of the element.

Added elements

Elements that have been added to the version you are comparing (Other Model elements) are shown as ADDED in the right element differences pane. In the picture below, the Order class (selected) has been added to Other Model.


Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 3.5]
Differences
Conflicts
Element Differences

Common MIMClass: null
Versioned By: null
Sync Dividers ☑

- SimpleModel [base-line 1.0] Unchanged

- SimpleModel [base-line 3.5] Unchanged
  - MIM [base-line 3.5] Unchanged
    - A_Address_Customer [base-line 3.5] Unchanged
    - Address [base-line 3.5] CHANGED
    - GoldSLA [base-line 3.5] CHANGED
    - Order [base-line 3.5] ADDED
  - CreationAck_Event [base-line 1.0] REMOVED
  - Creation_Event [base-line 3.5] CHANGED
  - CreationAck_E2M [base-line 3.5] CHANGED

My Model:
Versioned By:
Their DOMSchema: SimpleModel [base-line 3.5]
Versioned By: Administrator [2004.04.10-19:11:27:0303]

Figure 5-13: Added elements

Adding elements

Adding new class attributes individually, rather than replacing the entire class may result in conflicts, as described in "Replaced elements" on page 39.

To add elements to the model in the current context:

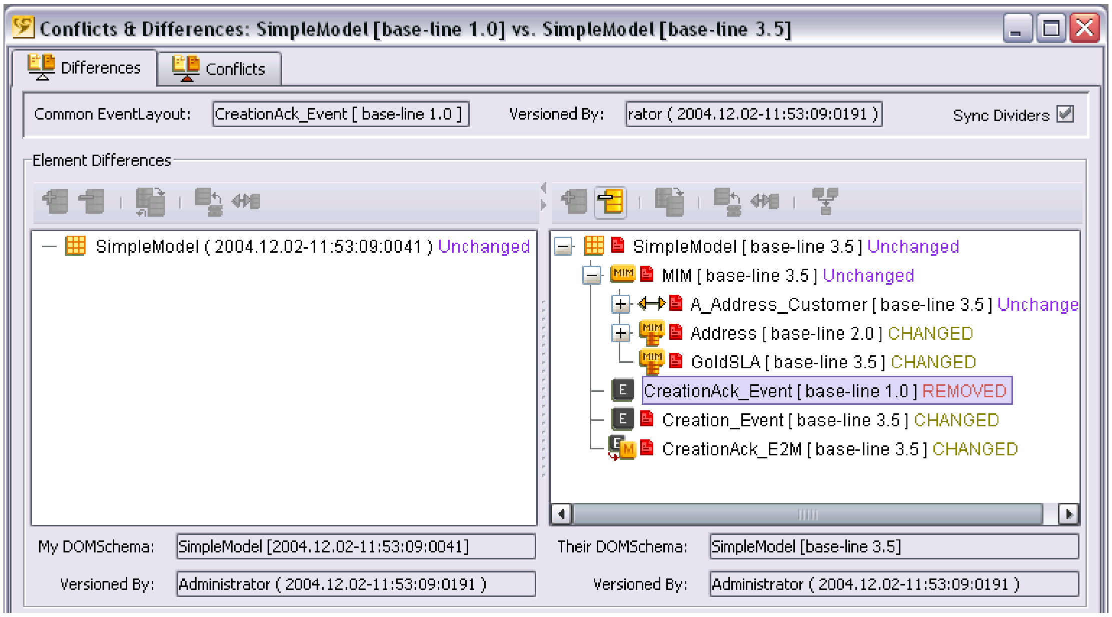
1. In the right **Element Differences** pane, select the element and click the add element button .

   You can select multiple elements by holding down the Ctrl key while selecting.

The element from the compared model is added to the model in the current context. You can verify this by checking the tree view of the Cogility Modeler window.

Removed elements

Elements that have been removed from the version you are comparing (Other Model elements) are shown as REMOVED in the right element differences pane. In the picture below, the CreationAck_Event element (selected) has been removed from the Other Model version.


Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 3.5]
Differences
Conflicts
Sync Dividers ☑
Element Differences

Common EventLayout: CreationAck_Event [base-line 1.0]
Versioned By: rator (2004.12.02-11:53:09:0191)

SimpleModel (2004.12.02-11:53:09:0041) Unchanged

SimpleModel [base-line 3.5] Unchanged
MIM [base-line 3.5] Unchanged
A_Address_Customer [base-line 3.5] Unchange
Address [base-line 2.0] CHANGED
GoldSLA [base-line 3.5] CHANGED
CreationAck_Event [base-line 1.0] REMOVED
Creation_Event [base-line 3.5] CHANGED
CreationAck_E2M [base-line 3.5] CHANGED

My DOMSchema: SimpleModel [2004.12.02-11:53:09:0041]
Versioned By: Administrator (2004.12.02-11:53:09:0191)
Their DOMSchema: SimpleModel [base-line 3.5]
Versioned By: Administrator (2004.12.02-11:53:09:0191)

Figure 5-14: Removed elements

Removing elements

To remove an element from the model in the current context:

1. In the right **Element Differences** pane, select the element and click the remove element button .

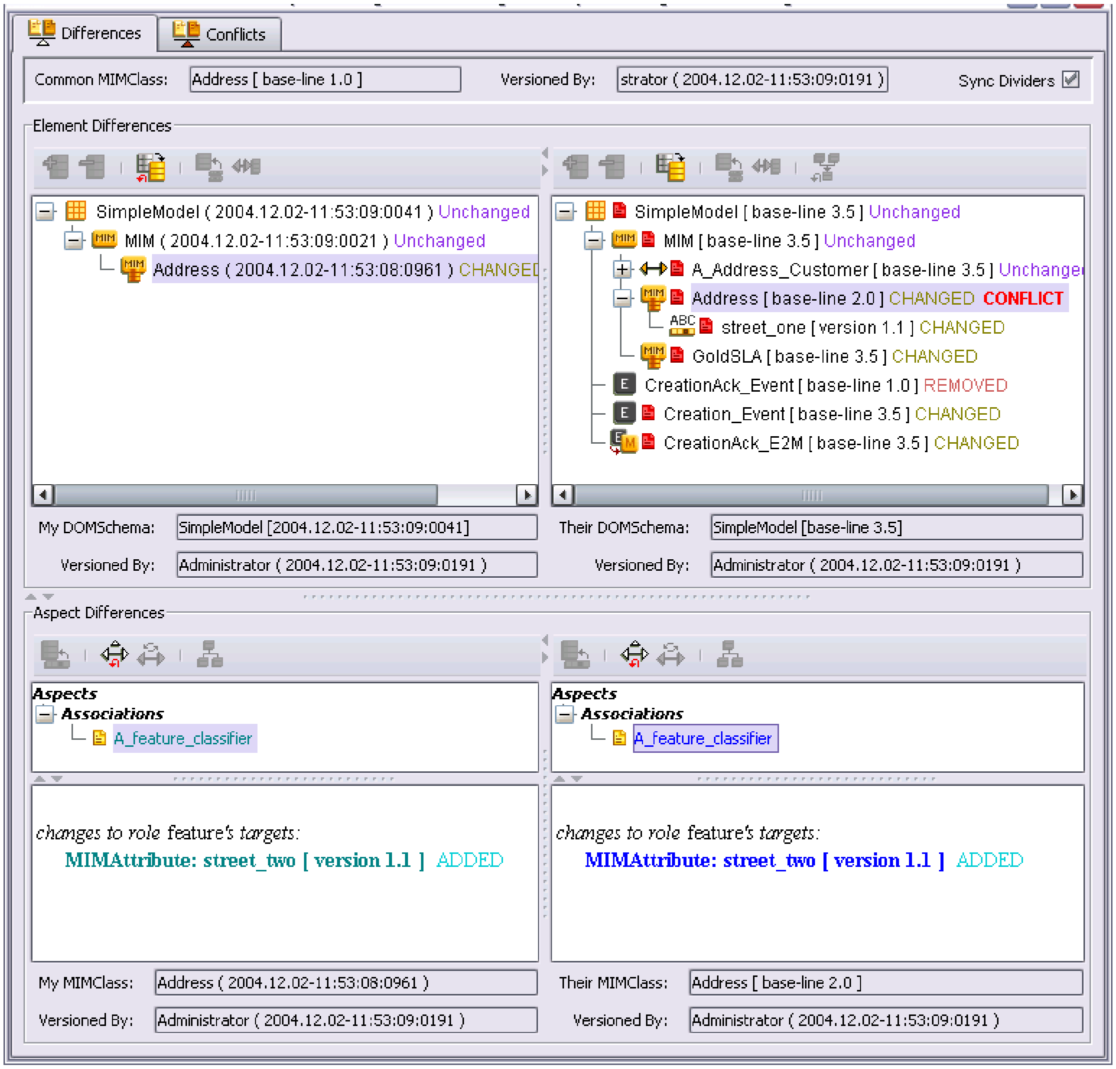
   You can select multiple elements by holding down the Ctrl key while selecting.

The element from the compared model is removed from the model in the current context. You can verify this by checking the tree view of the Cogility Modeler window.

Replaced elements

Conflicts may result if you try to add, remove or commit changed class attributes individually from the compared version (the Other Model) to the model in the current context (the Current Model). This is because classes are treated as atomic artifacts, just like events, messages, and so on, but they contain elements themselves. In the following picture, the street_two attribute was added to the Current Model independently of its container, the Address class. A conflict results because both the

Other Model and the Current Model model have an Address class that is different from the common parent version of both models.



Common MIMClass: Address [base-line 1.0]

Versioned By: strator (2004.12.02-11:53:09:0191)

Sync Dividers ☑

SimpleModel (2004.12.02-11:53:09:0041) Unchanged
MIM (2004.12.02-11:53:09:0021) Unchanged
Address (2004.12.02-11:53:08:0961) CHANGED

SimpleModel [base-line 3.5] Unchanged
MIM [base-line 3.5] Unchanged
A_Address_Customer [base-line 3.5] Unchanged
Address [base-line 2.0] CHANGED CONFLICT
street_one [version 1.1] CHANGED
GoldSLA [base-line 3.5] CHANGED
CreationAck_Event [base-line 1.0] REMOVED
Creation_Event [base-line 3.5] CHANGED
CreationAck_E2M [base-line 3.5] CHANGED

My DOMSchema: SimpleModel [2004.12.02-11:53:09:0041]

Versioned By: Administrator (2004.12.02-11:53:09:0191)

Their DOMSchema: SimpleModel [base-line 3.5]

Versioned By: Administrator (2004.12.02-11:53:09:0191)

Aspects
Associations
A_feature_classifier

changes to role feature's targets:
MIMAttribute: street_two [version 1.1] ADDED

Aspects
Associations
A_feature_classifier

changes to role feature's targets:
MIMAttribute: street_two [version 1.1] ADDED

My MIMClass: Address (2004.12.02-11:53:08:0961)

Versioned By: Administrator (2004.12.02-11:53:09:0191)

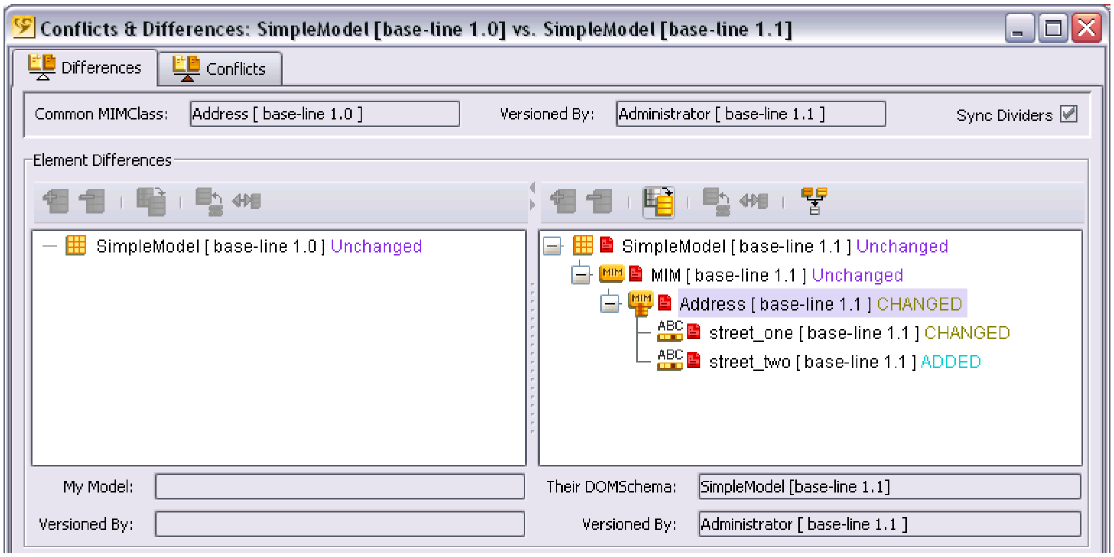
Their MIMClass: Address [base-line 2.0]

Versioned By: Administrator (2004.12.02-11:53:09:0191)

Figure 5-15: Replaced elements

You can resolve such conflicts, as described in "Conflicts" on page 48. But to avoid the conflict, replace the entire container (in this example, the Address class) of the model in the current context with the version from the compared model.

Elements of a class that have been added to, removed from or changed in the version you are comparing (Other Model elements) are shown as ADDED, REMOVED or CHANGED in the right element differences pane. In the picture below, the attributes of the Address class (selected) have been added and changed in the Other Model version. Rather than add or commit these attributes individually, replace the entire class in the current context (the Current Model Address class, in the example below) with the one in the compared version (the Other Model Address class).



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 1.1]

Common MIMClass: Address [base-line 1.0]

Versioned By: Administrator [base-line 1.1]

Sync Dividers ☑

Element Differences

- SimpleModel [base-line 1.0] Unchanged

```
SimpleModel [ base-line 1.1 ] Unchanged
  MIM [ base-line 1.1 ] Unchanged
    Address [ base-line 1.1 ] CHANGED
      street_one [ base-line 1.1 ] CHANGED
      street_two [ base-line 1.1 ] ADDED
```

My Model:

Versioned By:

Their DOMSchema: SimpleModel [base-line 1.1]

Versioned By: Administrator [base-line 1.1]

Figure 5-16: Replacing elements

Replacing elements

To replace elements in the current context:

1. In the right **Element Differences** pane, select the element and click the replace elements button ▤.

   You can select multiple elements by holding down the Ctrl key while selecting.

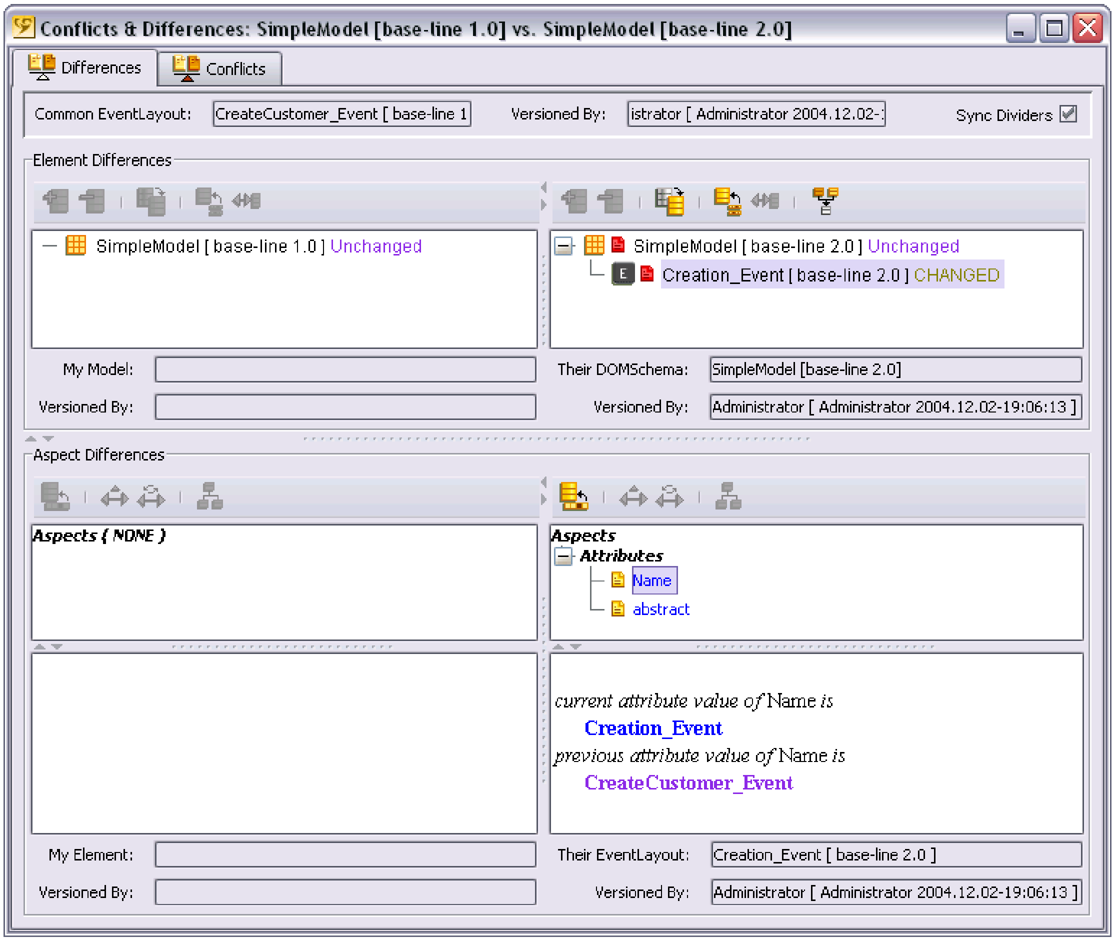
The element in the current context is replaced with that from the compared model. You can verify this by checking the tree view of the Cogility Modeler window.

CHANGED elements

Elements that have been changed in the version you are comparing (Other Model elements) are shown as CHANGED in the right element differences pane. The changes to the element may affect its attributes, associations, superclass, or content.

Attribute values

Cogility Modeler artifacts are UML objects, and as such, they have attributes. For example, an event object has a name attribute and an attribute called abstract, of type Boolean, that indicates whether the event is an abstract class or not. In the following picture, the Creation_Event object's attributes, name and abstract have been changed in the compared version. The changes to the element attributes are described in the Aspect Differences panes. When you expand the aspects and select an attribute, the lowermost pane describes the changes.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Differences | Conflicts

Common EventLayout: CreateCustomer_Event [base-line 1
Versioned By: istrator [Administrator 2004.12.02-
Sync Dividers ☑

Element Differences

SimpleModel [base-line 1.0] Unchanged

SimpleModel [base-line 2.0] Unchanged
Creation_Event [base-line 2.0] CHANGED

My Model:
Versioned By:

Their DOMSchema: SimpleModel [base-line 2.0]
Versioned By: Administrator [Administrator 2004.12.02-19:06:13]

Aspect Differences

Aspects { NONE }

Aspects
Attributes
Name
abstract

current attribute value of Name is
Creation_Event
previous attribute value of Name is
CreateCustomer_Event

My Element:
Versioned By:

Their EventLayout: Creation_Event [base-line 2.0]
Versioned By: Administrator [Administrator 2004.12.02-19:06:13]

Figure 5-17: Attribute values

You can apply the attribute values individually to the model in the current context, as described in “Attribute values” on page 44, or you can apply all attribute values for the element.

Applying all attribute values

To apply all attribute values for the element:

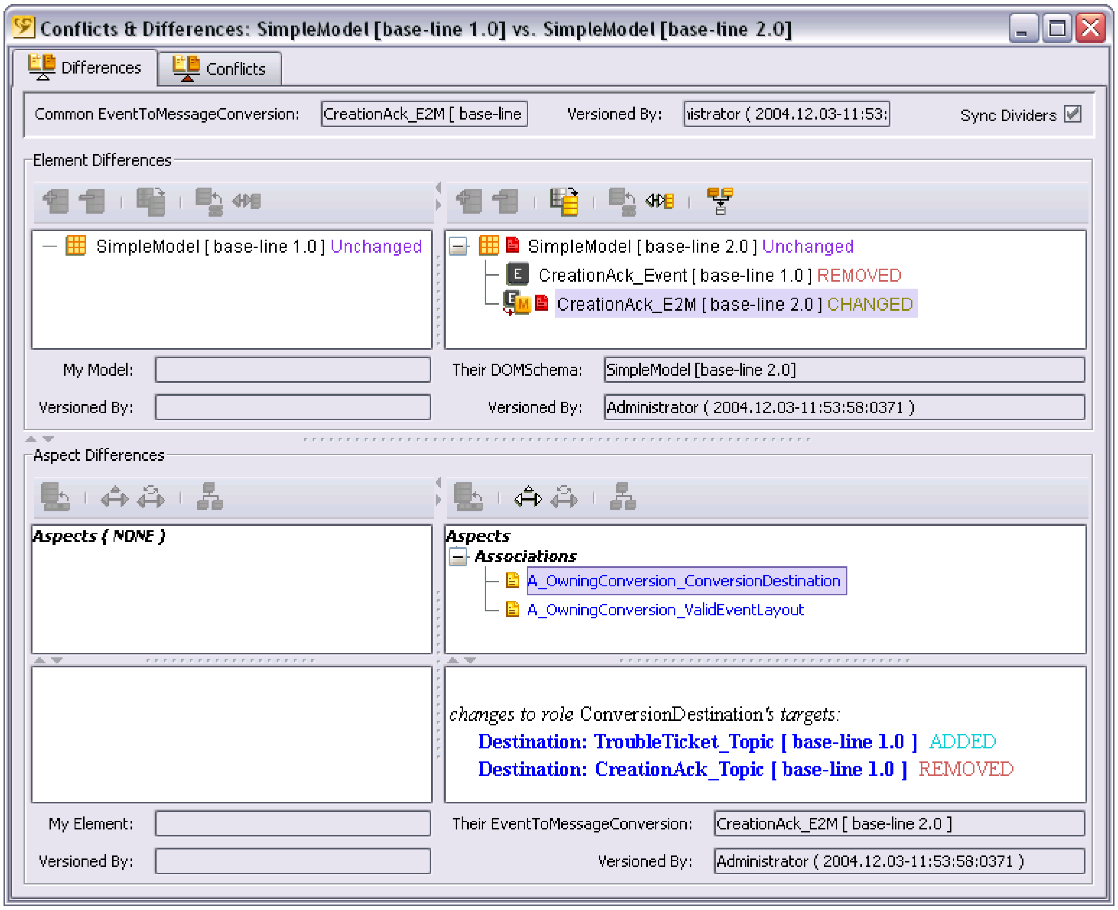
1. In the right **Element Differences** pane, select the changed element and click the apply all attributes button .

   You can select multiple elements by holding down the Ctrl key while selecting.

All of the attributes for the element in the current context are replaced with those from the element in the compared model. You can verify this by checking the element in the Cogility Modeler window.

Association targets

Cogility Modeler artifacts that interface with other artifacts, like message conversion objects, are said (in UML) to have associations with those other artifacts. For example, an event to message (E2M) conversion has an association with an event and a message destination. In the picture below, these associations have changed in the CreationAck_E2M of the model in the compared version. The aspects of these changes are shown in the lowermost Aspect Differences pane.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common EventToMessageConversion: CreationAck_E2M [base-line

Versioned By: istrator (2004.12.03-11:53:

Sync Dividers ☑

Element Differences

SimpleModel [base-line 1.0] Unchanged

SimpleModel [base-line 2.0] Unchanged
CreationAck_Event [base-line 1.0] REMOVED
CreationAck_E2M [base-line 2.0] CHANGED

My Model:

Versioned By:

Their DOMSchema: SimpleModel [base-line 2.0]

Versioned By: Administrator (2004.12.03-11:53:58:0371)

Aspect Differences

Aspects { NONE }

Aspects
Associations
A_OwningConversion_ConversionDestination
A_OwningConversion_ValidEventLayout

changes to role ConversionDestination's targets:
Destination: TroubleTicket_Topic [base-line 1.0] ADDED
Destination: CreationAck_Topic [base-line 1.0] REMOVED

My Element:

Versioned By:

Their EventToMessageConversion: CreationAck_E2M [base-line 2.0]

Versioned By: Administrator (2004.12.03-11:53:58:0371)

Figure 5-18: Association targets

You can merge the association targets individually, replacing those in the model in the current context, as described in “Associations” on page 45, or you can set all of the association targets for the element.

Setting association targets

To set all association targets for the element:

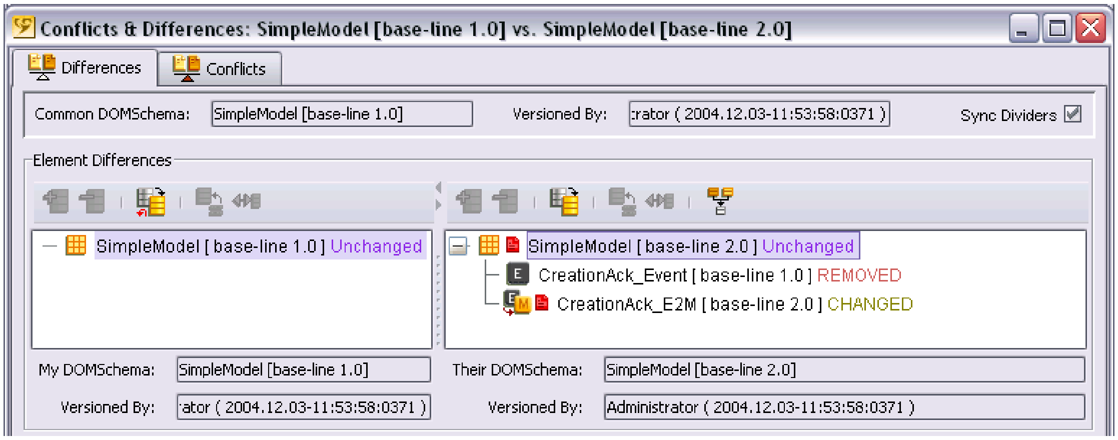
1. In the right **Element Differences** pane, select the changed element and click the set association targets button .

All of the associations for the element in the current context are replaced with those from the element in the compared model. You can verify this by checking the element in the Cogility Modeler window.

Merge all

When you have several changed elements in a container artifact, such as a model container, and you know you can bring them in from the compared model to the model in the current context without conflicts, you can merge all of the differences at once.

This operation replaces all of the elements in the model in the current context (the Current Model) with the elements from the compared model (the Other Model). This operation is available only when there are no conflicts between those elements.


Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]
Differences
Conflicts
Sync Dividers ☑
Element Differences

Common DOMSchema: SimpleModel [base-line 1.0]   Versioned By: :rator (2004.12.03-11:53:58:0371)

SimpleModel [base-line 1.0] Unchanged

SimpleModel [base-line 2.0] Unchanged
CreationAck_Event [base-line 1.0] REMOVED
CreationAck_E2M [base-line 2.0] CHANGED

My DOMSchema: SimpleModel [base-line 1.0]   Their DOMSchema: SimpleModel [base-line 2.0]
Versioned By: :ator (2004.12.03-11:53:58:0371)   Versioned By: Administrator (2004.12.03-11:53:58:0371)

Figure 5-19: Merging element differences

Merging all element differences

To merge all changes for a selected element:

1. In the right **Element Differences** pane select the element with changes and click the merge all differences button .

   You can select multiple artifacts by holding down the Ctrl key while selecting.

All of the elements in the current context are replaced with the corresponding elements in the compared model. You can verify this by checking the element in the Cogility Modeler window.

Aspect differences

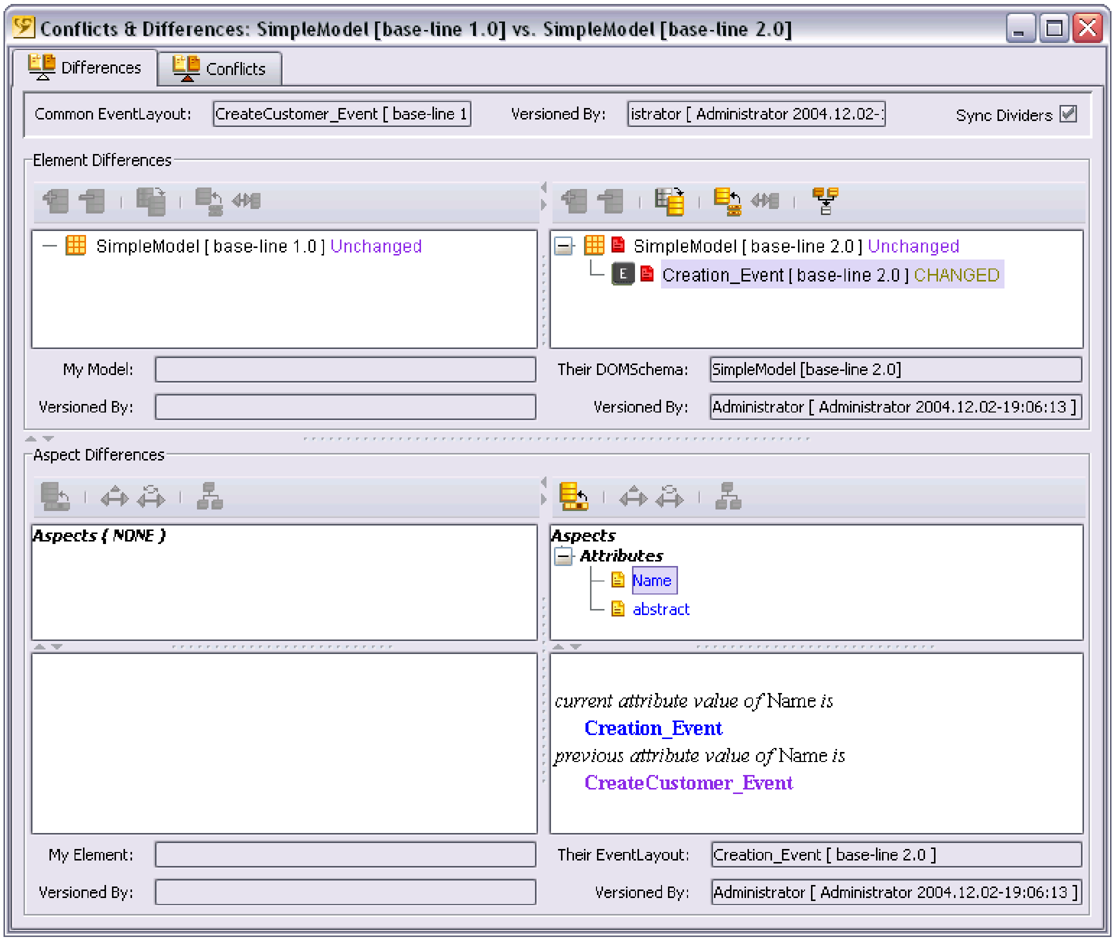
Elements that have been changed in the version you are comparing (Other Model elements) are shown as CHANGED in the right element differences pane. Cogility Modeler artifacts are UML objects, and as such, they have attributes, associations, a superclass, and content. When any of these are changed in the compared version, the *aspects* of the change - how the change affects the element's attributes, associations, superclass or content - are shown in the Aspect Differences panes.

Applying changes aspect by aspect is useful only in extreme cases, where there are conflicts being resolved in a particular element, for instance. Otherwise, it is always preferable to substitute the entire new element. Doing so reduces the chance of errors and maintains a clean version history because this approach maintains the version name provided by the person making the change in the first place.

This is true for any container. Always replace elements at the highest level possible, where there are no conflicts. It is better to replace an entire subtree and maintain versions in the replaced subtree than to create a new mutable containment tree all the way to the root.

Attribute values

All Cogility Modeler artifacts have attributes. For example, an event object has a name attribute and an attribute called abstract, of type Boolean, that indicates whether the event is an abstract or not. In the following picture, the Creation_Event object's attributes, name and abstract have been changed in the compared version. The changes to the element attributes are described in the Aspect Differences panes. When you expand the aspects and select an attribute, the lowermost pane describes the changes.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common EventLayout: CreateCustomer_Event [base-line 1

Versioned By: istrator [Administrator 2004.12.02-

Sync Dividers ☑

SimpleModel [base-line 1.0] Unchanged

SimpleModel [base-line 2.0] Unchanged

Creation_Event [base-line 2.0] CHANGED

Their DOMSchema: SimpleModel [base-line 2.0]

Versioned By: Administrator [Administrator 2004.12.02-19:06:13]

Aspects { NONE }

Aspects

Attributes

Name

abstract

current attribute value of Name is
Creation_Event
previous attribute value of Name is
CreateCustomer_Event

Their EventLayout: Creation_Event [base-line 2.0]

Versioned By: Administrator [Administrator 2004.12.02-19:06:13]

Figure 5-20: Aspect differences

You can apply all of the attribute values to the model in the current context, as described in “Attribute values” on page 41, or you can apply attribute values individually.

Applying attribute values

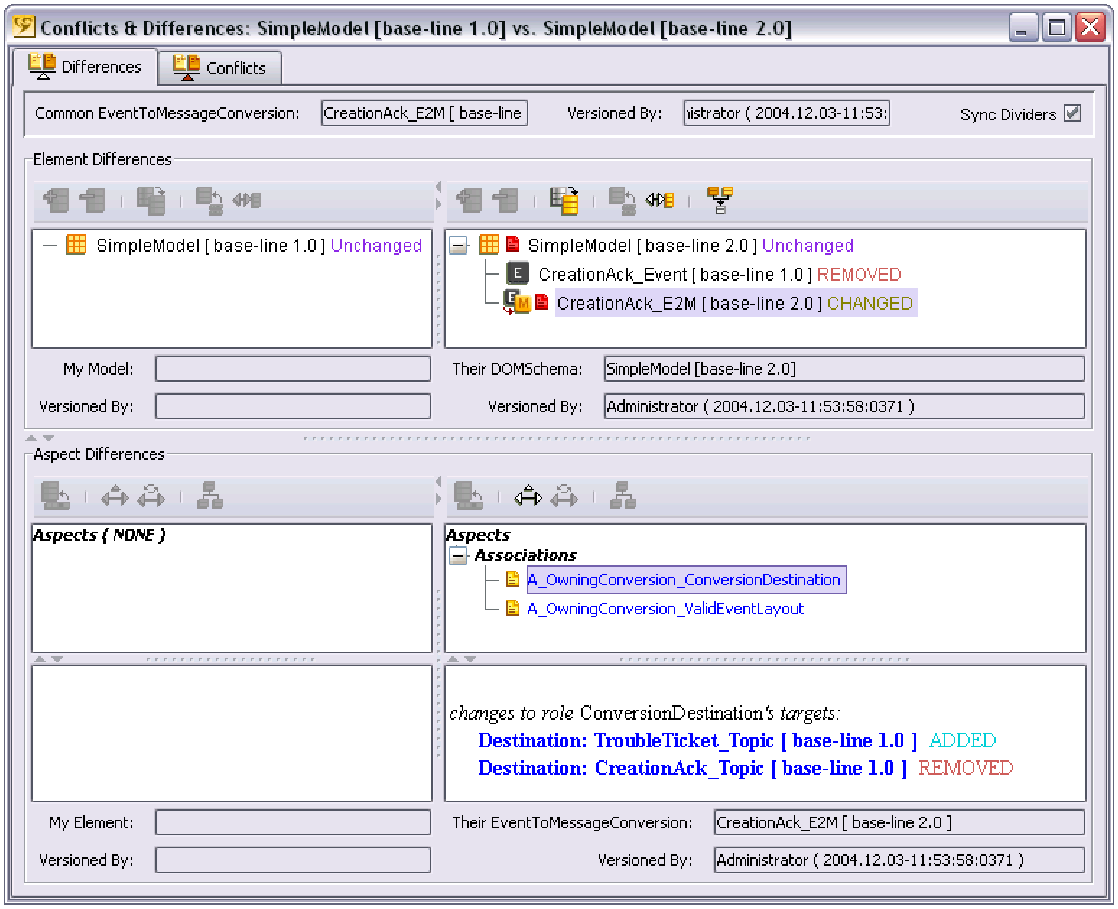
To apply attribute values:

1. In the right **Aspect Differences** pane, select the attribute and click the apply attributes button .

All of the attributes for the element in the current context are replaced with those from the element in the compared model. You can verify this by checking the element in the Cogility Modeler window.

Associations

Cogility Modeler artifacts that interface with other artifacts, like message conversion objects, are said (in UML) to have associations with those other artifacts. For example, an event to message (E2M) conversion has an association with an event and a message destination. In the picture below, these associations have changed in the CreationAck_E2M of the model in the compared version. The aspects of these changes are shown in the lowermost Aspect Differences pane.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common EventToMessageConversion: CreationAck_E2M [base-line

Versioned By: istrator (2004.12.03-11:53:

SimpleModel [base-line 1.0] Unchanged

SimpleModel [base-line 2.0] Unchanged
CreationAck_Event [base-line 1.0] REMOVED
CreationAck_E2M [base-line 2.0] CHANGED

My Model:
Versioned By:

Their DOMSchema: SimpleModel [base-line 2.0]
Versioned By: Administrator (2004.12.03-11:53:58:0371)

Aspects { NONE }

Aspects
Associations
A_OwningConversion_ConversionDestination
A_OwningConversion_ValidEventLayout

changes to role ConversionDestination's targets:
Destination: TroubleTicket_Topic [base-line 1.0] ADDED
Destination: CreationAck_Topic [base-line 1.0] REMOVED

My Element:
Versioned By:

Their EventToMessageConversion: CreationAck_E2M [base-line 2.0]
Versioned By: Administrator (2004.12.03-11:53:58:0371)

Figure 5-21: Associations

You can set all of the association targets for the element, as described in “Association targets” on page 42, or you can merge the association targets individually, replacing those in the model in the current context with those in the compared model.

Merging association targets

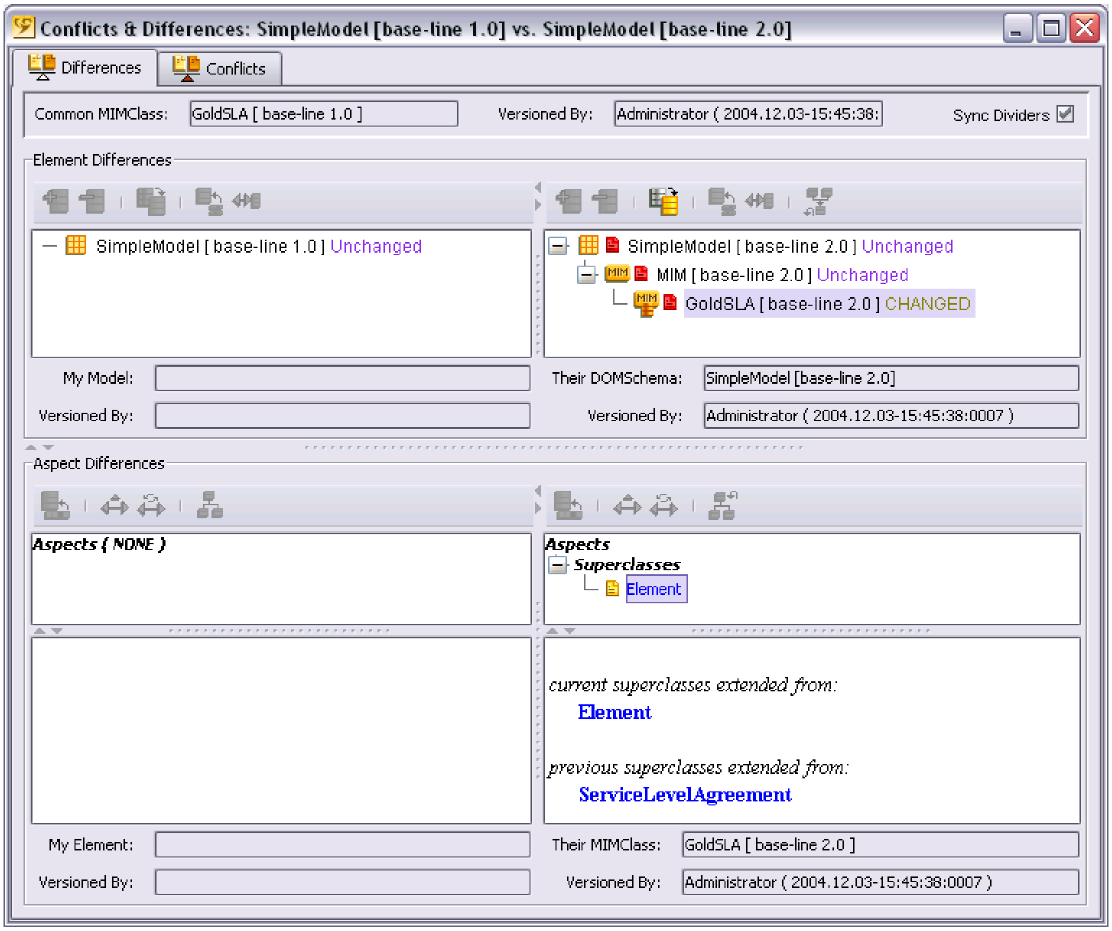
To merge an association target:

1. In the right **Aspect Differences** pane select the association and click the merge association targets button ⇔.

The selected association for the element in the current context is replaced with that from the element in the compared model. You can verify this by checking the element in the Cogility Modeler window.

Superclass aspect differences

When an element’s superclass changes in the compared version (Other Model), the element shows as CHANGED in the right element differences pane. To see the aspects of the change, in the Aspect Differences window, select the superclass, and the details of the change are displayed in the lowermost right pane. To update the element of the model in the current context (the Current Model) with the changed superclass from the compared model (the Other Model), you have only to replace the element.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common MIMClass: GoldSLA [base-line 1.0]
Versioned By: Administrator (2004.12.03-15:45:38:
Sync Dividers ☑

Element Differences

SimpleModel [base-line 1.0] Unchanged

SimpleModel [base-line 2.0] Unchanged
MIM [base-line 2.0] Unchanged
GoldSLA [base-line 2.0] CHANGED

My Model:
Versioned By:

Their DOMSchema: SimpleModel [base-line 2.0]
Versioned By: Administrator (2004.12.03-15:45:38:0007)

Aspect Differences

Aspects { NONE }

Aspects
Superclasses
Element

current superclasses extended from:
Element

previous superclasses extended from:
ServiceLevelAgreement

My Element:
Versioned By:

Their MIMClass: GoldSLA [base-line 2.0]
Versioned By: Administrator (2004.12.03-15:45:38:0007)

Figure 5-22: Superclass element

Replacing superclass elements

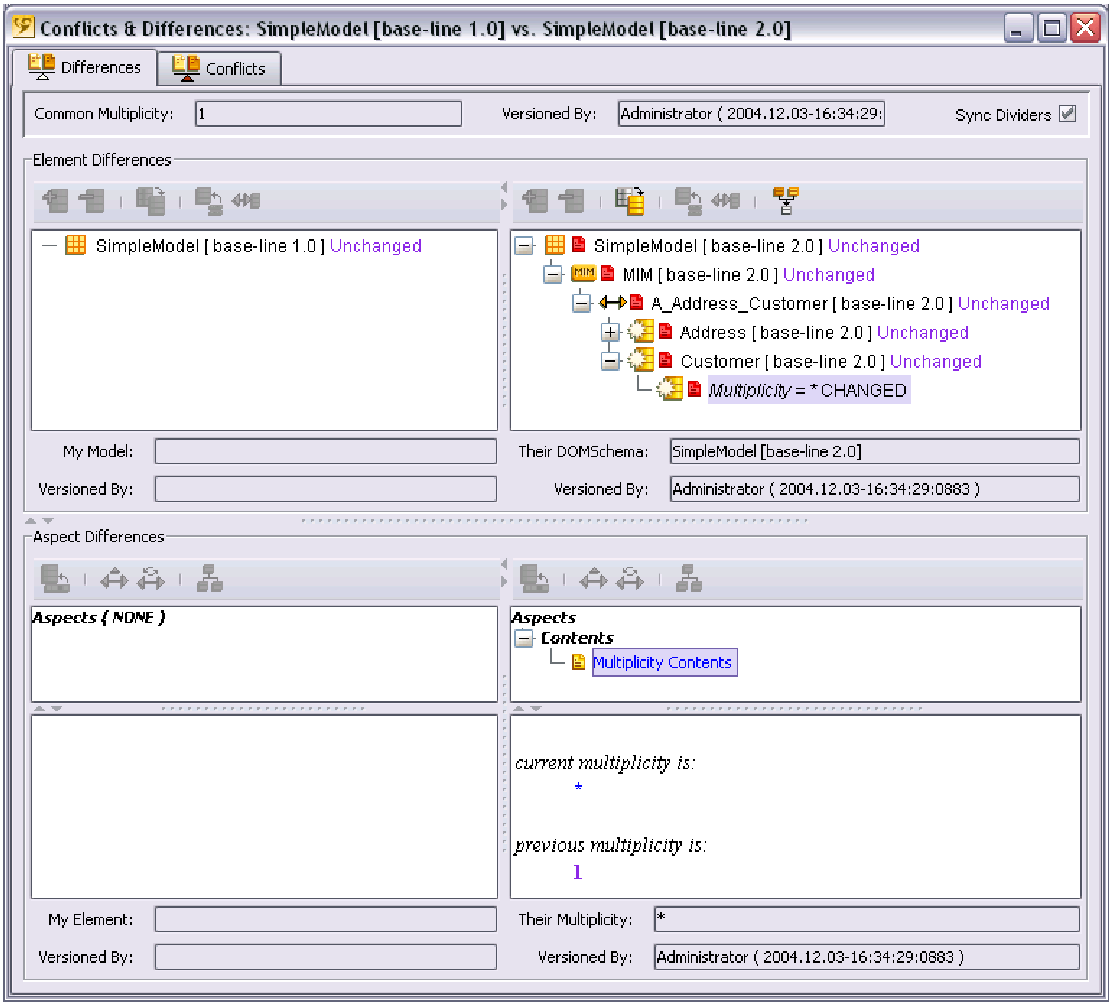
To replace superclass elements in the current context:

1. In the right **Element Differences** pane, select the element and click the replace elements button .

The element in the current context is replaced with that from the compared model. You can verify this by checking the tree view of the Cogility Modeler window.

Content aspect differences

Changes to Cogility Modeler artifacts' contents are also described in the Aspect Differences. For example in an association object, the target and source multiplicities are considered contents of that artifact.When the elements of an association in the compared model are different from those of the model in the current context, the elements appear as CHANGED in the rightmost element differences pane. In the Aspect Differences pane you can select the contents aspect to see the details. In the picture below, the multiplicity of the Customer object in the A_Address_Customer association has changed. To update the element of the model in the current context (the Current Model) with the changed element from the compared model (the Other Model), you have only to replace the element.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common Multiplicity: 1

Versioned By: Administrator (2004.12.03-16:34:29:

Sync Dividers ☑

Element Differences

- SimpleModel [base-line 1.0] Unchanged

- SimpleModel [base-line 2.0] Unchanged
  - MIM [base-line 2.0] Unchanged
    - A_Address_Customer [base-line 2.0] Unchanged
      - Address [base-line 2.0] Unchanged
      - Customer [base-line 2.0] Unchanged
        - Multiplicity = * CHANGED

My Model:

Versioned By:

Their DOMSchema: SimpleModel [base-line 2.0]

Versioned By: Administrator (2004.12.03-16:34:29:0883)

Aspect Differences

Aspects (NONE)

Aspects
- Contents
  - Multiplicity Contents

current multiplicity is:
*

previous multiplicity is:
1

My Element:

Versioned By:

Their Multiplicity: *

Versioned By: Administrator (2004.12.03-16:34:29:0883)

Figure 5-23: Element contents

Replacing element contents

To replace element contents in the current context:

1. In the right **Element Differences** pane, select the element and click the replace elements button .

The element in the current context is replaced with that from the compared model. You can verify this by checking the tree view of the Cogility Modeler window.

Conflicts

"Version comparison" on page 32 describes how to bring up the Conflicts and Differences view to compare two versions. The Conflicts and Differences view separates into two tabs, one to display differences (see "Differences and conflicts" on page 37) and the other to display conflicts. The Differences tab displays by default. Click the Conflicts tab to view the details of a conflict between two elements.

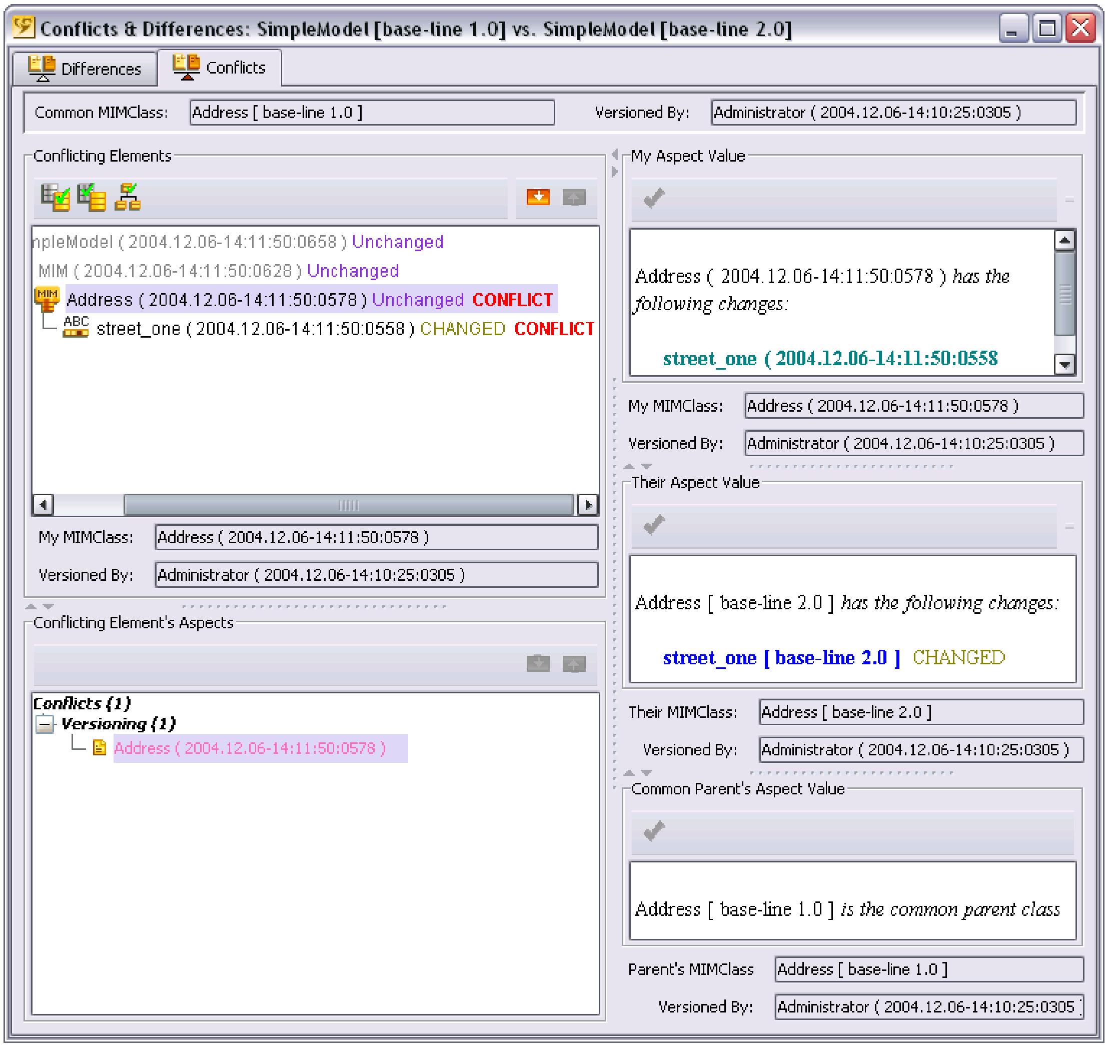
Generally, conflicts arise when two different users, working separately, attempt to modify the same object. A first step toward avoiding conflicts is to separate among the users the work of modifying model artifacts. No two users should work on the same artifacts.

Version conflicts

Two elements are in conflict when the version compared (the Other Model version) and the version in the current context (the Current Model version) both attempt to change the same parent element. The conflict view makes a three way comparison between both versions and the parent.

In the example below, the street_one attribute has been changed in the Address class of the model in the current context (Current Model). The changed attribute was added from the compared version (Other Model). Note that first it was changed from the parent version (where it was originally named street), then it was added from Other Model. This is effectively two ambiguous actions performed on the same element.

Both the Address attribute, street_one and its container, the Address class are in conflict with the model in the current context. You can resolve the conflicts at both levels.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common MIMClass: Address [base-line 1.0]   Versioned By: Administrator (2004.12.06-14:10:25:0305)

Conflicting Elements

npleModel (2004.12.06-14:11:50:0658) Unchanged
MIM (2004.12.06-14:11:50:0628) Unchanged
Address (2004.12.06-14:11:50:0578) Unchanged CONFLICT
street_one (2004.12.06-14:11:50:0558) CHANGED CONFLICT

My MIMClass: Address (2004.12.06-14:11:50:0578)
Versioned By: Administrator (2004.12.06-14:10:25:0305)

Conflicting Element's Aspects

Conflicts (1)
Versioning (1)
Address (2004.12.06-14:11:50:0578)

My Aspect Value

Address (2004.12.06-14:11:50:0578) *has the following changes:*

street_one (2004.12.06-14:11:50:0558

My MIMClass: Address (2004.12.06-14:11:50:0578)
Versioned By: Administrator (2004.12.06-14:10:25:0305)

Their Aspect Value

Address [base-line 2.0] *has the following changes:*

street_one [base-line 2.0] CHANGED

Their MIMClass: Address [base-line 2.0]
Versioned By: Administrator (2004.12.06-14:10:25:0305)

Common Parent's Aspect Value

Address [base-line 1.0] *is the common parent class*

Parent's MIMClass Address [base-line 1.0]
Versioned By: Administrator (2004.12.06-14:10:25:0305

Figure 5-24: Conflicting element versions

In Figure 5-24 on page 49 above, the Address class has a versioning conflict. Two actions were performed on the Address class element: an attribute named street_one was added and an attribute named street was changed. Cogility Modeler does not know which action takes precedence.

Resolving version conflicts

To resolve a versioning conflict:

1. In the **Conflicting Elements** pane (upper left), select the element.
2. In the **Conflicting Element's Aspects** pane (lower left), select the versioning conflict aspect.

   For each of the versions, the Current, Other and the parent version, the changes are detailed in the right panes.
3. Select one of the following options and click the corresponding button:
   - Take the version in the current context (the Current Model version) and make the version of the compared model (the Other Model version) the parent version by clicking the take-current button .

     In the example shown in Figure 5-24 on page 49 above, the result would be that the street attribute is first renamed street_one, then added to the Address class in the current context. The new street_one attribute would then include a previous version named street that has the same label (base-line 2.0) as the compared model.
   - Take the compared version (the Other Model version) by clicking the take-other button .

     In the example shown in Figure 5-24 on page 49 above, the result would be that the street_one attribute is added to the Address class which then has both a street attribute and the street_one attribute.
   - Take the parent version by clicking the take-parent button .

     In the example shown in Figure 5-24 on page 49 above, the result would be that the Address class in the current context reverts to the parent version with a street attribute and no street_one attribute.
4. Move up or down to the next element in the list by clicking the up ↑ or down ↓ arrows.

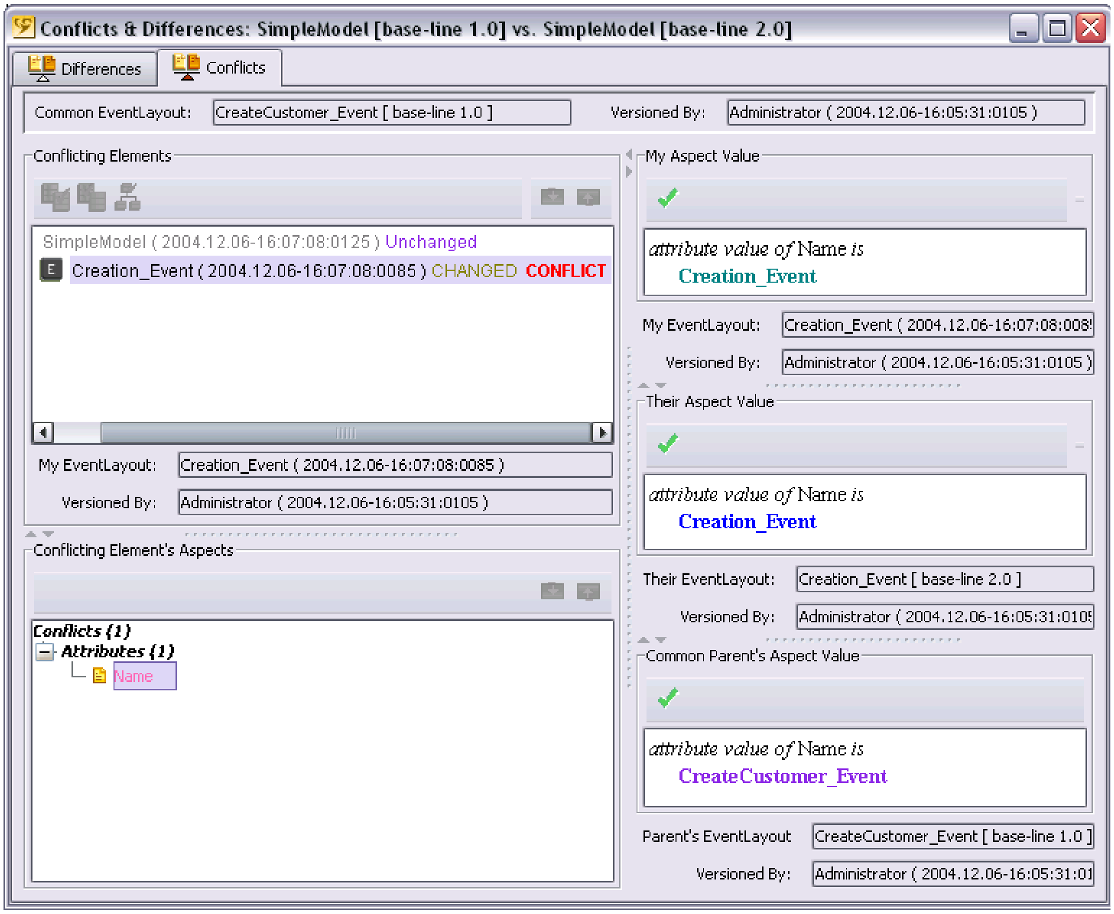
Aspect conflicts

Cogility Modeler artifacts are UML objects, and as such, they have attributes, associations, a superclass, and content. When any of these *aspects* are in conflict, the aspect value panes on the right side of the conflicts view displays the details of the conflict.

In the example shown in Figure 5-25 on page 51 below, the name attribute of an event element has been changed in the model in the current context (Current Model). In the parent version, the event was named CreateCustomer_Event; in the Other Model version, it is named Creation_Event. Note that first it was changed from the parent version, then it was added from Other Model. This is effectively two ambiguous actions performed on the same element.

In the conflicting element's aspects pane (the bottom left pane), the Name attribute aspect is selected, telling you that the element's name has a change that is in conflict. The value of this name attribute for each of three versions is shown in the right panes: that for the model in the current context

(Current Model) at the top, for the compared model (Other Model) in the middle and for the common parent in the pane at the bottom.



Conflicts & Differences: SimpleModel [base-line 1.0] vs. SimpleModel [base-line 2.0]

Common EventLayout: CreateCustomer_Event [base-line 1.0] Versioned By: Administrator (2004.12.06-16:05:31:0105)

Conflicting Elements

SimpleModel (2004.12.06-16:07:08:0125) Unchanged
E Creation_Event (2004.12.06-16:07:08:0085) CHANGED CONFLICT

My EventLayout: Creation_Event (2004.12.06-16:07:08:0085)
Versioned By: Administrator (2004.12.06-16:05:31:0105)

Conflicting Element's Aspects

Conflicts (1)
Attributes (1)
Name

My Aspect Value

attribute value of Name *is*
Creation_Event

My EventLayout: Creation_Event (2004.12.06-16:07:08:008
Versioned By: Administrator (2004.12.06-16:05:31:0105)

Their Aspect Value

attribute value of Name *is*
Creation_Event

Their EventLayout: Creation_Event [base-line 2.0]
Versioned By: Administrator (2004.12.06-16:05:31:010

Common Parent's Aspect Value

attribute value of Name *is*
CreateCustomer_Event

Parent's EventLayout CreateCustomer_Event [base-line 1.0]
Versioned By: Administrator (2004.12.06-16:05:31:01

Figure 5-25: Aspect conflicts

Resolving aspect conflicts

To resolve aspect conflicts:

1. In the **Conflicting Elements** pane (upper left), select the element.
2. In the **Conflicting Element's Aspects** pane (lower left), select the conflict aspect.

   For each of the versions, the Current Model, Other Model and the parent version, the changes are detailed in the right panes.
3. For the version of the element attribute you want to keep, click the replace button ✓ for that version.
4. If prompted to version the element, click **No** to cancel the action; then version the element and start over.

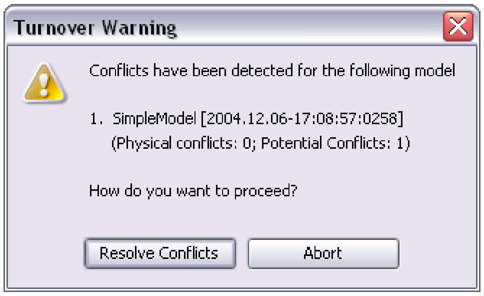
   A prompt may display reminding you to version the element before replacing it. You should version any mutable artifacts before performing this action. See "Versions" on page 17.

   Following step 3, the element is listed in the conflicting elements pane (upper left) with the RESOLVED label. You can also revert the replacement by clicking the restore button ✓.
5. Move up or down to the next element in the list by clicking the up ⬆ or down ⬇ arrows.

Containment conflicts

Containment conflicts occur when a user reverts to an older element version in an attempt to make it the latest baseline. For example, the user starts work on model S[v1] that contains element E[v1]. The user makes changes to E and turns the model over. The result is S[v2] containing E[v2]. The user then replaces E[v2] with E[v1] and attempts to do another turnover. The CM system flags E as having a containment conflict because E[v1], which is older than E[v2], is now in both container S[v1] and a derived version of S[v2].

For example, the Address class from base-line 1.0 replaced the updated Address (version 1.1) in the model in the current context. Upon turn over, the following warning appears:

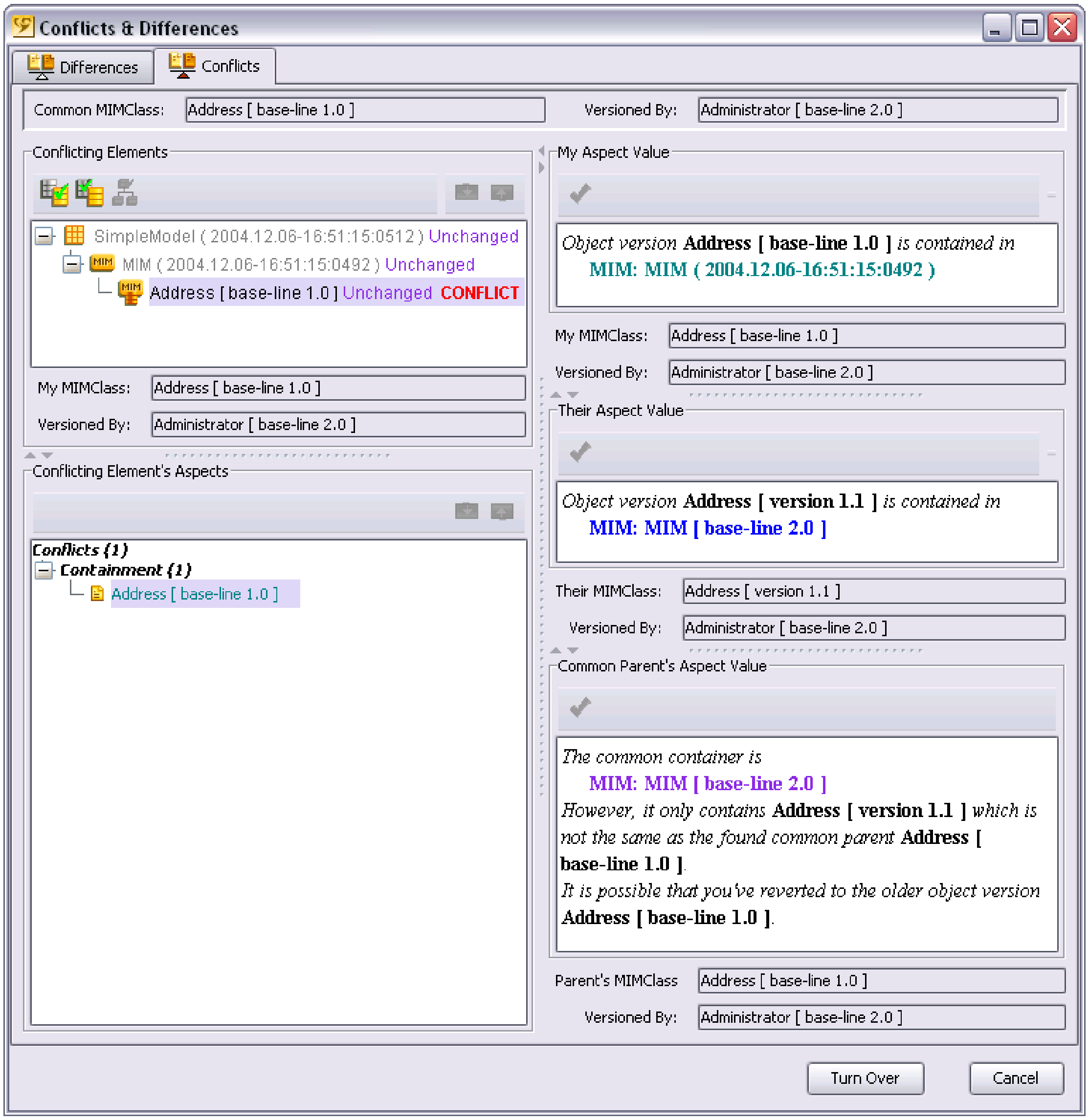


Conflicts have been detected for the following model

1. SimpleModel [2004.12.06-17:08:57:0258]
(Physical conflicts: 0; Potential Conflicts: 1)

How do you want to proceed?

When you click Resolve Conflicts, the Conflicts and Differences view displays.



Common MIMClass: Address [base-line 1.0]
Versioned By: Administrator [base-line 2.0]

SimpleModel (2004.12.06-16:51:15:0512) Unchanged
MIM (2004.12.06-16:51:15:0492) Unchanged
Address [base-line 1.0] Unchanged CONFLICT

My MIMClass: Address [base-line 1.0]
Versioned By: Administrator [base-line 2.0]

Conflicts (1)
Containment (1)
Address [base-line 1.0]

Object version **Address [base-line 1.0]** *is contained in*
MIM: MIM (2004.12.06-16:51:15:0492)

My MIMClass: Address [base-line 1.0]
Versioned By: Administrator [base-line 2.0]

Object version **Address [version 1.1]** *is contained in*
MIM: MIM [base-line 2.0]

Their MIMClass: Address [version 1.1]
Versioned By: Administrator [base-line 2.0]

The common container is
MIM: MIM [base-line 2.0]
However, it only contains **Address [version 1.1]** *which is not the same as the found common parent* **Address [base-line 1.0]**.
It is possible that you've reverted to the older object version **Address [base-line 1.0]**.

Parent's MIMClass Address [base-line 1.0]
Versioned By: Administrator [base-line 2.0]

Figure 5-26: Containment conflicts

Resolving containment conflicts

To resolve containment conflicts:

1. If you know which elements are in conflict, you can click **Abort**, then revert the changes that caused the conflicts. If you do not know, click **Resolve Conflicts**.
2. In the **Conflicts** view, select the element in conflict.
3. In the **Conflicting Element's Aspects** pane, select the containment conflict.

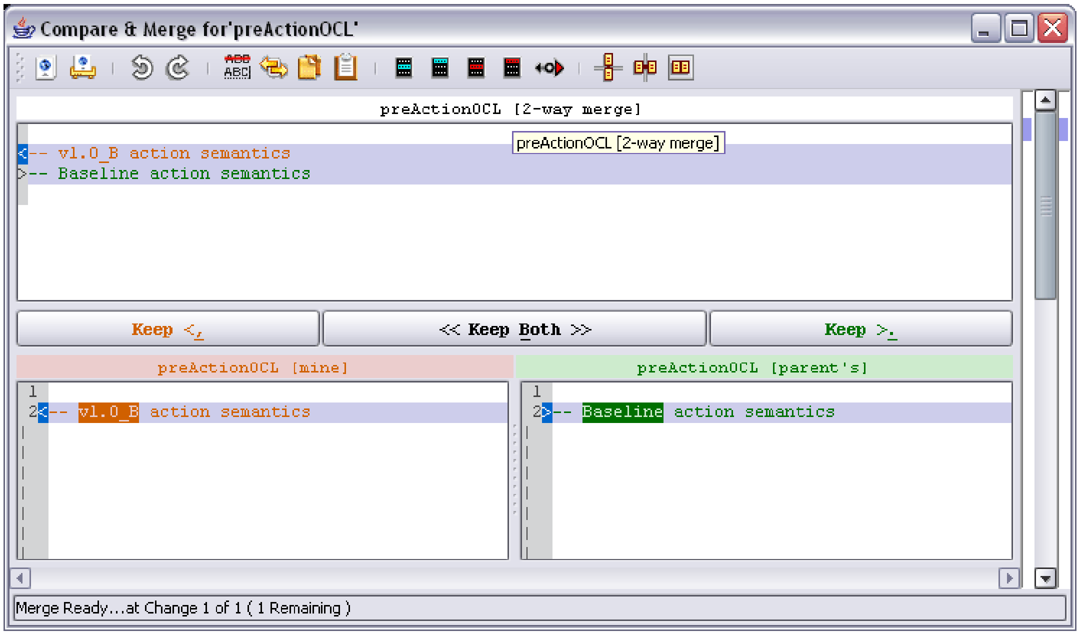
   The conflict is explained in the aspect value panes.
4. Resolve the conflict by clicking one of the following buttons under Conflicting Elements:
   - Take The Current Version As Is - this marks the conflict as resolved, but if the versioning problem persists the same conflict shows up the next time you try to turn over the model.

- Take The Other Version - this reverts the element to the previous version and resolves the conflict.

Action semantics conflicts

With conflicts between artifacts, you are forced to choose either the Other Model version or the parent version. However, with a conflict between the Current Model and the Other Model action semantics in the same artifact a two-way or three-way merge is possible. A two-way merge compares the Current Model version against the parent version; a three-way merge compares both the Current Model and the Other Model versions agains the parent. With a two-way merge, you can keep either the parent version or the Current Model version. With a three-way merge you can keep either the Current Model or the Other Model or both versions.

An example of a two-way merge is shown below. The baseline model is version v0. The model in the current context is version 1.0_B, and both are changing the same line of code. Keeping both, in this example, would insert the action semantics from the Current Model version after the action semantics in the parent version.



Compare & Merge for'preActionOCL'

preActionOCL [2-way merge]

```
<-- v1.0_B action semantics
>-- Baseline action semantics
```

preActionOCL [mine]

```
1
2<-- v1.0_B action semantics
```

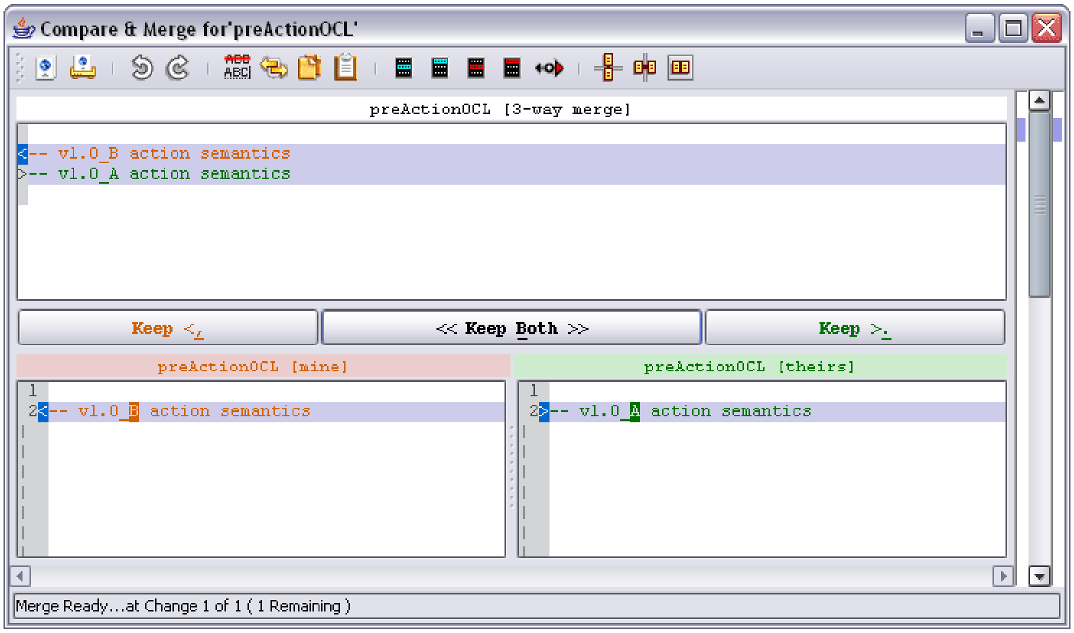
preActionOCL [parent's]

```
1
2>-- Baseline action semantics
```

Merge Ready...at Change 1 of 1 (1 Remaining)

Figure 5-27: Two-way merge

An example of a three-way merge is shown below. The baseline model is version v0. The model in the current context is version 1.0_B, the model in the repository is version 1.0_A, and both are changing the same line of code (originally `-- v0 action semantics`, not shown). Keeping both, in this example, would insert the action semantics from the Other Model version after the action semantics in the Current Model version.



```
<-- v1.0_B action semantics
>-- v1.0_A action semantics
```

```
1
2<-- v1.0_B action semantics
```

```
1
2>-- v1.0_A action semantics
```

Merge Ready...at Change 1 of 1 (1 Remaining)

Figure 5-28: Three-way merge

You can also see this information in the difference report file. See "Difference report" on page 34.

Merging action semantics

To merge action semantics from different versions:

1. In the **Differences** view, in the **Aspect Differences** window, select the Attribute aspect.

   This might be labeled preActionOCL or something similar.
2. To perform a three-way merge, click the three-way merge button.
3. To perform a two-way merge, click the two-way merge button.
4. In the Compare and Merge window, choose from the following:
   a. Click **Keep Both** to place the action semantics from all compared versions into the current context.
   b. Above the field for the model version with the desired action semantics (current, other or parent's), click **Keep**.
5. Click **Save Merge & Exit** to commit the merge.
6. Click **Undo** to revert to the previously unmerged state.
7. Click **Exit** to close the merge window.
8. If you choose **Exit**, you must then choose from the following when prompted to discard the current merge data:
   a. **Yes** discards the merge data, reverts the elements to the previously unmerged state and exits.
   b. **No** does nothing.

Index