Merative Social Program Management 8.1

Working with the Cúram Model in Rational® Software Architect Designer

Note

Before using this information and the product it supports, read the information in Notices on page 27

Edition

This edition applies to Merative™ Social Program Management 8.0.0, 8.0.1, 8.0.2, 8.0.3, and 8.1.

Contents

1 Using the Rational® Software Architect Designer to modify the Cúram Model

Use IBM® Rational® Software Architect Designer to modify the Cúram Model.
Related information

1.1 Integrating the Cúram model into Rational® Software Architect Designer

The Cúram model uses a subset of the functionality that is provided by Rational® Software Architect Designer. To simplify the use of Rational® Software Architect Designer for the Cúram model, some techniques are used to customize the tool to the Cúram model.

Rational® Software Architect Designer

The Rational® Software Architect Designer perspective is a particular layout of views, an editor, and toolbars where you can view, create, and maintain elements of the Cúram model.

The most common tasks that you use Rational® Software Architect Designer for include:

- Creating, opening, and closing a model
- Basic maintenance tasks common to all model elements
- Working with model fragments
- Searching in the model

Note: Rational® Software Architect Designer can be used as a modeling tool or as a plug-in for Eclipse. For the purposes of this information, the focus is on using the Rational® Software Architect Designer standalone tool.

The Cúram profile and the Cúram model

A Cúram profile is provided for working with the Cúram model. The Cúram profile defines what UML stereotyped elements and values can be defined within the Cúram model.

When you create a model in Rational® Software Architect Designer, the Cúram model template is used. This template combines the Cúram profile with a filtering capability to remove unnecessary or unsupported functionality from menus and options in the Rational® Software Architect Designer workbench. The Cúram model template is automatically used when you open an existing Cúram model. Always select the template when you create a new model.

Note: Some options that are not supported in the Cúram model cannot be hidden from the user. If the user tries an unsupported action, a validation message is displayed and the action is reversed. For more information, see Modifying an element on page 13.

The modeling views

The Rational® Software Architect Designer Modeling Perspective is used for UML modeling and consists of four views: the Project Explorer view, the Properties view, the Diagram Editor view, and the Model Editor view.

Table 1: Modeling views

View	Description
Project Explorer	See all the related parts of the Cúram model in a navigable tree structure
Properties	View and maintain information about a selected model element.
Diagram Editor	Create, view, and edit model diagrams using the custom Cúram palette.
Model Editor	View and edit a model's configuration in a tabbed view.

Project Explorer view

Use the Project Explorer view to view all the related model elements, diagrams, and children of a model element.

Right-click the context menu in the **Project Explorer** view to see the options that you can performed for the selected element.

Properties view

Use the Properties view to view and edit the properties for the selected model element. This view allows you to configure general and stereotype properties for an element, set element relationships, and manage element documentation.

The tabs in the Properties view that are used for Cúram model development are the **General** tab, and the **Documentation** tab, and the **Cúram** tab.

Table 2: Tabs in the Properties view used in Cúram modeling

Tab	Description
General	Maintain the base UML configuration of an element in a model, e.g., name, visibility, etc.
Documentation	Create, view, and edit documentation relevant to a specific element.
Cúram	Holds the properties that are relevant to a stereotyped element in the Cúram domain. These properties are specific to the Cúram model and are used to enhance the configuration of an element.

Diagram editor

Use the Diagram editor to create, view, and edit diagrams.

The Diagram editor is split into two areas, a **Diagram** view and a **Palette**. The **Palette** contains a Cúram **drawer** that contains the most commonly-used Cúram model elements that you can drag and drop into the Diagram editor. Use the **Diagram** view to view and modify a model element in relation to other elements. A right-click context menu is also available for the addition of model classes in the Diagram editor.

Model editor

Use the mdel editor to view and edit information related to a model or sub-unit fragment.

To open a model in the model editor right-click on the model in the project explorer, select **Open** > **With** > **Mdel Editor**.

View the Cúram profile in the **Details** tab of the **Model Editor**. The profiles and model libraries that are used for working with the Cúram model are preconfigured for the Cúram model. You should not need to alter the profiles or model libraries that are supplied with the Cúram model.

The five tabs in the **Model Editor** view as described in table 1.

*Table 3: Tabs in the **Model Editor***

Tab	Description
Overview	Contains general information related to the selected model. You can also edit documentary information related to the model in the **Overview** tab.
Details	Maintains the profiles and model libraries that are applied to the model that you are currently viewing.
Diagrams	Lists the available diagrams for the selected model.
References	Provides a list of other models and profiles from the workspace referenced by the selected model. It also provides a list of other models from the workspace that reference the selected model.
Fragments	The physical resources associated with the model elements are called fragments and are essentially separate files. Dividing a model into fragments is useful in large development projects. This is done by extracting packages into physical sub-units, or fragments. The physical location of the model elements are transparent, and the fragments remain a logical part of the original model. The **Fragments** tab lists the fragments that are included in the model. You can search for and absorb fragments into the containing model from this list. For more information on absorbing fragments, see Absorbing fragments on page 18.

Maintaining the model

Create, open, close, and navigate a model in Rational® Software Architect Designer.

Creating the model

Use the Rational® Software Architect Designer **Create Model** wizard to create models from stored templates.

A Cúram template is provided in the Cúram plugin for Rational® Software Architect Designer. When using the template to create a new model, the model capabilities are set to what is appropriate for that template. Using the Cúram model template to create your model ensures that the menus and options are available while modeling are those supported by the Cúram model.

There are two ways of opening the **Create Model** wizard in Rational® Software Architect Designer:

- Right-click on the model directory and select **Create Model**.
- Select **File** from the topmost menu bar, then select **New** > **UML Model**.

To create a Cúram model in the **Create Model** wizard:

1. Name the model and specify a location.
2. Select the **Standard Template** option.
3. Select **Cúram** in the **Categories** pane (check the **Show All Templates** option to see the Cúram category).
4. Select **Cúram model** in the templates pane.
5. Select the model capabilities. By default, **Cúram capabilities** are selected. It is recommended that you use the default capabilities.
6. Select the referenced models, if there are any.

Use the **Back** and **Next** buttons to step forward and backward through the **Create Model** wizard steps. You can exit the **Create Model** wizard by clicking **Finish**. The model is saved when you exit the wizard.

Opening and closing the model

Open the model in the **Modeling** perspective in Rational® Software Architect Designer.

Opening the model

The default settings for the **Modeling** perspective display the **Project Explorer** view. When a model exists in your project, the **Project Explorer** contains two more folders (**Diagrams** and **Models**). Expand these folders, and the model underneath, to open the model at that point.

Rational® Software Architect Designer uses a form of lazy loading whereby the full model is not opened initially, instead each portion is opened as it is navigated to. Alternatively, to force a load of the full model, right-click on the top-level model file and select **Open All Sub-Fragments**.

Closing the model

Right-click on the model and select **Close** or **Close All** from the context menu.

Navigating the model

Navigate through the packages and elements of a model.

To move through the packages and elements of a model, select the expand + option for the package or element where children exist in the **Project Explorer**. If a package is not currently loaded, it is loaded and the icon changes. The package is expanded to display the child elements. Modeling elements can be added to the model in the **Diagram Editor** using the **Model Palette** and the **Cúram Drawer**.

Maintaining UML model elements

The UML model elements used by Cúram include **Packages**, **Classes**, **Attributes**, **Operations**, **Parameters**, and **Relationships**.

- **Packages** are containers for classes.
- **Classes** define the business processes, value and rules objects, or database schema, for example: Facade, WebService, RDO, Entity, and so on.
- **Attributes** define fields on the value or rules objects or database entities.
- **Operations** represent the business or SQL functions on relevant to a parent class.
- **Parameters** are the input or return arguments to a parent operation.

- **Relationships** define bonds between the classes that make up the application, for example: aggregation of structs and foreign keys between entities.

Viewing an element

To view an element, ensure that the **Properties** view is opened in the foreground and select the element in the **Project Explorer**. The element is opened in the **Properties** view and a number of tabs relevant to that element are available to maintain that element.

The right-click menu in the **Project Explorer** allows you to access functions for that element as well as view information related to the element.

Selecting to expand a modeling element in **Project Explorer** loads that element's subfragments. The modeling element's icon changes in the **Project Explorer** to mark that its subfragments have loaded.

Adding an element to the model

Use the **Project Explorer** right-click context menu, or the **Diagram View and Palette** to add an element to a model in Rational® Software Architect Designer.

Project Explorer

Use the **Project Explorer** to add a child element type.

1. Right-click on the existing parent element in **Project Explorer**.
2. Select the relevant element type from the list of child element types of your selected element.

Diagram view

Use the **Diagram** view to add an element.

1. Open the diagram you wish to add the new element to in the **Diagram** view.
2. Double click on the required element in the **Cúram Palette**, or right-click in the **Diagram** view and select the element you wish to add from the right-click context menu.

Modifying an element

Modify a model element's name from the right-click context menu in the **Project Explorer** or the **Diagram** view. More extensive modification, including an element's documentation, can be performed through the tabs in the **Properties** view.

Element properties that are specific to Cúram are managed on the **Cúram** tab in the **Properties** view. Depending on what is being modified some modifications require two separate changes to be made, one on the **Cúram** tab and one on another tab in the **Properties** view. For example, changing the return type of an operation from an entity shadow class to a handcrafted struct requires you to change both the return type on the **General** tab and the **Shadow Type** on the **Cúram** tab. Similarly changing the primitive type for a domain definition from *SVR_INT16* to *SVR_STRING* requires you to change both the primitive type on the **Attributes** tab and add a *Maximum_Size* entry on the **Cúram** tab.

Important: When you attempt to perform an action that is not supported by the Cúram model, a validation is displayed and the action is reversed. Due to a bug in Rational® Software Architect Designer, the view may not be updated until refreshed for example; navigating to another element and back to the element being updated. The use of the operations, attributes and parameters tabs to add elements is not currently supported for use with the Cúram model as these tabs do not provide the correct Cúram stereotypes.

Creating a relationship between elements

Create a relationship between elements by using either the **Project Explorer** or the **Diagram** view.

Project explorer

To create a relationship between elements in the **Project Explorer**, take the following steps:

1. Select the element that you want to create a relationship with in the **Project Explorer**.
2. In the **Properties** view, browse to the **Relationships** tab.
3. Search for the other element for the relationship and select the source or target.
4. Select the type of relationship you want to create between the two elements.
5. The **General** tab of the **Relationship** can then be used to specify a name, multiplicity, and so on.

Diagram view

You can also use the **Diagram** to create relationships by using connector handles as follows:

1. Hover over an element in a diagram.
2. Drag one of the available arrowheads onto another element in the diagram to create a relationship. The two different arrowheads signify whether the element that is being dragged from is the source or target of the relationship.
3. Use the **General** tab of the **Relationship** to specify a name, multiplicity, and so on.

Removing an element from a model

Model elements can be deleted in the **Project Explorer** or **Diagram Editor**. To delete an element, right-click the element and select **Delete from Model**.

In the **Diagram** view, you can delete an element just from the diagram, or from the complete model. Deleting an element from the model means that the element becomes unavailable to other parts of the model and is removed from the **Project Explorer** tree.

Important: References to other elements in Rational® Software Architect Designer are maintained by an internal identifier system. Each element is given a unique identifier on creation and references are made to this unique identifier. If you remove a class, recreating the class with the same name is not sufficient to correct any broken references. You must follow the broken reference resolution process to reconnect broken references.

Copying and pasting classes, operations, and attributes

To save time and effort, use the Rational® Software Architect Designer **Project Explorer** to copy and paste **Classes**, **Operations**, and **Attributes**. Operations and attributes can only be copied in the same class or between classes of the same type.

An example of how to copy and paste between classes is as follows:

- Select the attributes in the tree control of the Rational® Software Architect Designer **Project Explorer**.
- Right-click on the selected attributes to be copied and choose **Copy** from the context menu.
- Right-click on the class (of the same type) to receive the new attribute and choose **Paste**.

You can use a similar technique for moving attributes by using **Cut** and **Paste**.

Note: If you try to copy and paste across different class types, you receive an error dialog indicating:

```
The requested action violates the integrity of the model.
```

Changing attribute order

By default Rational® Software Architect Designer displays the order of attributes alphabetically.

Attribute ordering is significant for entity and struct classes when they are used to define indexes as the DDL that's generated for index creation relies on this ordering. You can view the attribute ordering using the **Attribute** tab of the class. You can also change the default behavior of Rational® Software Architect Designer from its default ordering of **Stereotyped Type then Alphabetically** to **Storage Order** by selecting the **Windows** menu and **Preferences** submenu. From the resulting dialog navigate to **Views**, **Modeling**, and **Project Explorer** where you can use the **Project Explorer** settings, Sort By drop-down to change the ordering; click **OK** to save your changes.

If you need to change the order of the attributes, the **Attributes** tab provides **Move up** and **Move down** buttons as appropriate.

Searching in Rational® Software Architect Designer

Search in a model, search for element references in a model, and search for elements using the type browser.

Searching the model

Use the **Search** option to search the model using a broad range of criteria. The **Model Search** functionality can be used to see how an element is related to the rest of the model.

To search for an element in the model:

1. Select the **Search** option from the main menu bar and click on the **Model Search** tab.
2. Specify your search criteria. There are a range of search criteria that can be specified that allow you to narrow your search.
3. Search results are displayed in a tab beside the **Properties** view. Double clicking on a search result listing causes the project explorer to jump to that element.

Searching for references to an element

You can search for references to an element.

To search for references to an element, take the following steps:

1. Select the element you wish to search for references to.
2. From the right-click context menu, select the **Modeling References** option. You can choose to search for references in the enclosing model, the enclosing package, the workspace, or you can define a custom working set.
3. The results of your search are displayed in the **Search Results** tab. Double clicking on a search result listing causes the project explorer to jump to that element.

Searching for elements using the type browser

The **Type Browser** is used during the creation of an element that requires the specification of a type or element reference.

The **Type Browser** is used to search for the type of model element you want to create, for example, the parent of a domain, the return type of an operation, and so on. When you open the **Type Browser**, you can enter the name of the element to search for, or you can browse for the element directly in the model. The **Modify Search Scope** option controls the scope of the search. Searching is based on an index that Rational® Software Architect Designer maintains across sessions and does not require the complete model to be opened. The first search takes longer due to the creation of this index.

Specialized tabs and wizards

Use specialized tabs and wizards to support frequent tasks or manage complex content.

Foreign key tab

Use the **Foreign Key** tab to define and maintain a foreign key's name and mappings. This tab is visible on the **Properties** view of a foreign key relationship.

The **Name** field manages the foreign key name and manipulates the label entry on the **General** tab.

The table contains two columns; child and parent and these columns indicate the direction of the foreign key and name the entities on either end of the relationship. The table serves as a widget to edit the foreign key mappings which are stored on the appropriate role fields on the **General** tab.

Rows on the table relate to the mappings in the foreign key where a entry in the child column are mapped to an entry in the parent column. The entry in the row can be selected by using a drop-down on the row which lists the applicable attributes for that entity. Rows can be removed by setting the drop-down to blank and can be re-ordered using the **Move up** and **Move down** buttons on the table.

Secure field tab

Use the **Secure Field** tab to define and maintain a **Facade** class operation's **Secure** fields. This tab is on the **Properties** view of a facade-owned operation.

The tab contains two columns; **Field Name** and **Security Identifier (SID) Name**. The field name entries are computed from walking the available fields for the return type of the operation. SID Names can be entered, edited or deleted from the right column as required. This table serves as a widget to edit the *Secure_Fields* property on the **Cúram** tab.

Manage operation parameters wizard

Use the **Manage Operation Parameters** wizard to create and maintain the parameters and return type of an operation. The **Parameters** and **Return Type** frames are visible where the operation allows their addition.

Use the **Parameters Frame** to see a tabular listing of the parameters where parameters can be added, deleted or re-ordering using the buttons to the right of the table. The table offers direct in-place editing for the name, type and shadow type of the parameter.

Use the **Return Type** to select the type and manipulate the shadow type of the return value.

Operation wizard

Use the **Operation** wizard to create operations. The parameters and return type frames are visible where the operation allows addition of such.

The wizard utilizes the same layout and functionality as the **Manage Operation Parameters** wizard, with additionally providing a field to enter the name of the operation.

Entity operation wizard

Use the **Entity Operation** wizard to create standard and non-standard database operations where the input/output structures can be determined from the entity. The wizard contains a list of input and/or output attributes which is used to specify the attributes that form a struct class which is generated by the wizard.

The generation firstly checks whether a struct exists in the same package with the same attributes and prompts whether to use this struct or generate a new struct to promote re-use of existing structs.

The naming pattern for this generated struct class is:

```
<Entity Name><Key (Input)/Dtls (Output)>Struct<Unique Number>
```

For example, `PersonKeyStruct1`.

Domain definition wizard

Use the **Domain Definition** wizard to create a domain definition, you can also create a domain definition class.

The wizard allows you to set the name of the **Domain Definition** and browse for the type. Optionally, the **Max Size** field is editable when you choose a domain that is based on a *SVR_STRING* or *SVR_BLOB* to allow for the regular size value to be set.

Creating class diagrams

Create class diagrams in a package.

To create a class diagram, take the following steps:

1. Right-click on the package in which you want to create a class diagram in the **Project Explorer**.
2. Select **Add Diagram**, then **Class Diagram** from the right-click context menu. The new diagram is then created and opened in the **Diagram Editor**.
3. Drag and drop elements from the **Project Explorer** onto the **Diagram Editor**.

Modeling elements can be added to the model in the **Diagram Editor** using the **Model Palette** and **Cúram drawer**. For more information on using the Diagram Editor, see Diagram editor on page 10.

Creating and absorbing fragments

The Cúram model is a collection of elements that are logically related but physically separated.

When you open a model that contains fragments, the fragments do not load automatically, they load when you open them or when you access functionality that requires artifacts from the fragments. When you load a fragment, the parent resource is also loaded.

Creating fragments

Create fragments.

To create a fragment take the following steps:

1. Right-click on the package that you wish to create a controlled fragment from.
2. Select **Refactor**, and then **Create Fragment**.
3. Enter the name fragment and select the location where you wish to save the fragment when prompted.
4. When the fragment is saved, another dialog appears in which you must ensure that you have the **Update references to elements in new fragment** option set. If not set you risk breaking references to child elements contained in this fragment.

Once the fragment has been created, the icon for the package changes to signify that it is a controlled fragment.

Absorbing fragments

You might want to remove a fragment by absorbing it back into its containing fragment or model.

To absorb a fragment, take the following steps:

1. Right-click on the fragmented package and select **Refactor**.
2. Choose the **Absorb Fragment** option.
3. Ensure that a tick is placed in the **Update references to elements in the fragment** box. This ensures that existing references in the fragment are not broken in the process of absorbing the fragment.

You can also absorb all the fragments in a model at the same time:

1. Right-click on a model and select **Refactor**.
2. Select the **Absorb All Sub-Fragments** option. All the fragments in the model are absorbed. Ensure that you update element references when absorbing all the fragments in a model.

Validating a model

Validate a model by right-clicking on the model and selecting **Validate**.

The validation reports a summary in the console panel and describes any warnings or errors found in the **Problems** view. The problem description should indicate the issue and link to the location found in the model.

1.2 Modeling Cúram elements using Rational® Software Architect Designer

You can model Cúram elements by following instructions for the different model element types.

Development tasks for each model element with some of the example model element types that make up the application are as follows:

Managing domain definitions

The data types of attributes in Merative™ Social Program Management are modeled as domains.

Domains are defined in terms of a fundamental datatype such as a string or an integer, or in terms of another already existing application domain. Domains have application-specific type names such as *SOCIAL_SECURITY_NUMBER*, *PAYMENT_AMOUNT*, and so on. Domains can have associated validations defined for them such as uppercase, range checks, code tables, pattern matches, or custom validations.

Creating a domain definition

New domain definitions can be added to the model using the right-click context menu in the Project Explorer. With Rational® Software Architect Designer, you are not restricted in terms of the package structure. Domains can be added to any existing named package or combined with other elements in the same package.

For consistency, you must preserve the standard structure. This allows domains to be easily managed and re-used across the application.

You can create a domain definition using the following steps:

1. In the **Project Explorer**, navigate to the package where you want to create the new domain definition.
2. Right-click on the package and navigate to the **Add Class** menu and select **Domain Definition**.
3. In the **Create Domain Definition** wizard, enter the name of the domain and select a domain definition type. If the type you select is *SVR_STRING* you must also specify the maximum size.

   Note: When searching for the base domain types, for example *SVR_STRING*, you must modify the **Search Scope** and select the **Search non-imported UML libraries**. The base types exist in a plugin delivered with the SDEJ and can only be searched for and cannot be browsed to.
4. Choose the domain type. This can be done in two ways: through the type browser or by searching the model. Once you have selected the domain type, click **Finish**.

Renaming a domain definition

You can rename a domain definition using one of two methods.

1. Right-click on the domain definition in the **Project Explorer** and rename it.
2. Select the domain definition in the **Project Explorer** and then edit it in the **General** tab of the **Properties** view.

Important: When you rename a domain definition, you must also rename its single attribute to the same name. Use the **Attributes** tab in the **Properties** view for the domain definition to rename the attribute.

Renaming a Domain Definition maintains any references to that Domain.

Modifying a domain definition

A domain definition contains a single attribute whose type represents the domain it inherits from.

To modify a domain definition do the following:

1. Navigate to the **Attributes** tab in the **Properties** view and double-click the **Type** cell for the single attribute.
2. Search for and select the domain type in the **Type** browser.

Managing entities

Entities are objects that represent the persistent storage of the application, they have attributes which are defined as domains. Entities also have primary keys and index and foreign key relationships.

Create, read, update, and delete style operations are defined on entities as stereotyped methods. The signatures of these operations are implied by the stereotype. Other operations can be defined on entities by defining their signatures in the model. Operations requiring complex database queries can be specified in SQL.

Creating an entity

To create an entity, select the package where you want to create it and from the right-click context menu, choose **Class** > **Entity**.

As an example, consider the Person entity in the Cúram model. Once it is added to the Person package, the required attributes are created for it. Entity operations are also added which handle the data passing to and from the database tables.

Adding an attribute to an entity

Attributes store information related to an entity. For example, in the *Person* entity, the *CountryOfBirth* attribute stores the country of birth for a person. The domain definition for this attribute is *COUNTRY_CODE*.

An entity has at least one attribute that contains a unique identifier. This is identified by the key attribute. The *Person* entity contains a key attribute *concernRoleID*.

To add an attribute to an entity:

1. Select **Add Attribute** from the right-click context menu for the entity.
2. Select **Key** or **Details** as required. This opens the **Create Attribute** Wizard where you can name the attribute and select its type.

Adding an operation to an entity

Add operations to entity classes using the right-click context menu.

To add an operation to an entity, take the following steps:

1. Select **Add Operation** from the right-click context menu and choose the stereotype for the operation that you want to create.
2. Accept the default name of the operation that matches the stereotype you selected.

Most of the operation stereotypes do not require you to model the arguments or return types. If the stereotype you choose requires a return type to be modeled it must be a struct. To model a return type, take the following steps:

1. Select the return type in the wizard using the **Select Type** button.
2. If the return type you select is an entity, you must also select the **Shadow Type** from the drop down to identify the actual struct that is used. If the return type you select is a struct, do not select a **Shadow Type**.

Adding a return type to an entity operation

You can set the return type on an entity operation when you create the operation. However,. some of the entity operation stereotypes do not require you to model the return type because it is implied by the stereotype.

If you want to add a return type later or change the return type, take the following steps:

1. Select the operation in the **Project Explorer**.
2. Select the **General** tab in the **Properties** page and select **Set return type**.

If the return type you select is an entity, you must also select the **Shadow Type**. To select the **Shadow Type**:

1. Open the **Cúram** tab, select the required **Shadow Type** value for the *Shadow_Type* property.
2. If you change the return type on an operation and the new return type does not require a *Shadow_Type*, make sure that the *Shadow_Type* on the **Cúram** page is set to **unspecified**.

Adding an 'ns' operation to an entity

Complex database operations are modeled as *'ns'* type operations.

To add an *'ns'* operation to an entity, take the following steps:

1. Right click on the entity and select **Operation**.
2. Choose the stereotype of the operation from the list of available stereotypes.
3. You are presented with the **Create 'ns' Operation** wizard where you can name the operation, its parameters and select the return type. If the parameter, return type you select is an entity, you must also select a **Shadow Type**.

To add the SQL for the operation, navigate to the **Cúram** tab of the **Properties** view, and edit the SQL property string value.

Managing structs

A struct is a value object. Method arguments and return types on operations and entity classes are modeled as structs. Attributes of structs are specified as domain definitions.

Creating a struct

To create a struct, take the following steps:

1. Right-click on the package you wish to create a struct in, and select **Struct** from the right-click menu option for the package.

2. Provide a name for the struct in the properties view of the **General** tab.

Adding an attribute to a struct

Attributes describe the data that is contained in the struct.

To add an attribute to a struct, take the following steps:

1. Select the struct you wish to add an attribute to in the project explorer.
2. Select **Add Attribute**, and select **Default** from the right-click context menu.
3. In the **Create Default Attribute** Wizard, name the attribute and choose its type from the list of available types.

Creating an aggregate relationship

Relationships are bonds between classes. The following relationship types can be modeled: aggregation (one class contains another), assignable (attribute values of one class can be copied to the other), foreign key (for referential constraints), index, and unique index (to define database indexes on entity classes).

An aggregation relationship model the relationship between objects where one object contains another. In Merative™ Social Program Management, this relationship is always between two structs.

In the Project Explorer, take the following steps:

1. Select the struct that will be the containing struct in the relationship.
2. In the **Properties** view, select the **Relationships** tab for the struct. Choose to add a relationship that originates from this element.
3. Select the object to be contained as the target of the relationship and select **Aggregation** as the relationship type.

If the contained object is an entity, select the **Shadow Type** to identify the actual struct to be contained. This selection can be done in the **Cúram** tab of the **Properties** view.

On the **Relationships** page, right-click the aggregation that you created and select **Navigate** from the menu. This action opens the **Properties** view of the newly created aggregation. Verify that your aggregation is correct by viewing the diagram on the **General** tab.

In the diagram, the **Diamond** icon appears beside the containing struct. In the **Cúram** tab, type a role name for the contained struct and set the multiplicity of the relationship. The multiplicity of the container struct must be 1. The multiplicity of the contained struct can be **1..*** (for a 1-to-many relationship) or **1** (for a 1-to-1 relationship).

Managing process classes

Business functions are represented in the Cúram model as methods of process classes. The arguments and return type for methods are modeled as structs or domain types. The model defines

the interface for process class methods, but not their implementation. Process classes can call on entity classes to perform database operations as required.

Adding a process class to a package

Add a process class to a package.

To add a business process class to a package, select **Add Class**, and then **Process** from the right-click context menu and name the class.

Adding operations to a process class

Add an operation to process classes using the right-click context menu.

To add an operation to a process class take the following steps:

1. Select **Operation** from the right-click context menu and choose **Default**.
2. This opens the **Create 'default' Operation** wizard where you can name the operation, add its parameters and select its return type.

Adding an argument to a process operation

Arguments for process operations are defined as structs or domain types.

To add an argument to a process operation take the following steps:

1. Right-click on the process and select **Manage Parameters**.
2. In the **Manage Operation Parameters** wizard, name the parameter and select the parameter type. If the type you select is an entity, you must also select the **Shadow Type**.

Adding a return type to a process operation

The return type from a process class operation is a struct or domain type. You can set the return type on an process class operation when you create the operation.

If you want to add a return type later or change the return type take the following steps:

1. Select the operation in the **Project Explorer**.
2. In the **General** tab in the **Properties** page for the operation, select **Set return type**.

If the return type you select is an entity, you must also select the **Shadow Type**:

1. Open the **Cúram** tab and the select the required **Shadow Type** value for the *Shadow_Type* property.
2. If you change the return type on an operation and the new return type does not require a **Shadow Type**, make sure that the *Shadow_Type* on the **Cúram** page is set to unspecified.

Managing facade classes

Some business process functions are invoked from the client application while others provide utility functions not directly available to the client. A facade class is a process class whose interface is visible to the client.

Creating a facade class

To add a facade class to a package, select **Add Class**, then select **Facade** from the right-click context menu and name the class.

Adding operations to a facade class

Operations are added to facade classes using the right-click context menu.

To add an operation to a facade class:

1. Select **Operation** from the right-click context menu and choose **Default**.
2. In the **Create 'default' Operation** wizard, name the operation, parameters and select its return type.

Adding arguments and a return type to facade operations

Arguments and return types are added to facade operations in the same manner as they are added to process classes.

Please refer to Adding an argument to a process operation on page 23 and Adding a return type to a process operation on page 23 respectively.

1.3 Context menu options in the project explorer view

Use the right-click context menu in the Rational® Software Architect Designer project explorer window to perform various tasks for each model class.

Table 1 describes the specific attributes and operations that you can add to each class from the right-click context menu in the project explorer.

Table 4: Right click context menu options for classes

Class	Available attributes	Available operations
audit_mappings	audit_mappings	n/a
facade	n/a	default, wmdpactivity, qconnector, batch
webservice	n/a	default, wmdpactivity, qconnector, batch
wsinbound	n/a	default, wmdpactivity, qconnector, batch
process	n/a	default, wmdpactivity, qconnector, batch
struct	default	n/a

Class	Available attributes	Available operations
entity	key, details	batchinsert, batchmodify, insert, modify, nkmodify, nkread, nkreadmulti, nkremove, ns, nsinsert, nsmodify, nsmulti, nsread, nsreadmulti, nsremove, read, readmulti, remove, default
rdo	dataitem	n/a
listrdo	dataitem	n/a
loader	n/a	n/a
domain_definition	n/a	n/a
extension	default, dataitem, key, details	batchinsert, batchmodify, insert, modify, nkmodify, nkread, nkreadmulti, nkremove, ns, nsinsert, nsmodify, nsmulti, nsread, nsreadmulti, nsremove, read, readmulti, remove, default, wmdpactivity, qconnector, batch
Package	Package, Model	Any class type
Manage Parameters	Any applicable operation	n/a

1.4 Resolving broken references

You can resolve a broken reference in Rational® Software Architect Designer using only the id of the broken element reference. This requires a map of the previous models IDs to be extracted and then used to resolve references in the current model.

A resource reference is where an element refers to another element either in the same file, or in another file in a different model.

In Rational® Software Architect Designer, the element id is used to resolve a reference.

Rational® Software Architect Designer changes in references

In Rational® Software Architect Designer, only the id of the element is used to resolve a reference which has led to the possibility that there will be more instances of broken references requiring manual intervention.

The possibility of broken references is due to support of previous product versions. When an element is created in a model, it gets a unique id (adding an element across multiple product lines, which is sometimes the case required to introduce a new feature into the product) can subsequently introduce multiple unique IDs for the same added element. If a customer refers to this added element in their model and then later jumps product stream, the reference will then be broken from the customer's model to the new Social Program Management model.

A broken reference can be reported during two phases:

1. Opening your model in Rational® Software Architect Designer initiates the IBM® Rational® automated resource reference resolution process which might not find a resolution. Any failures are reported in the Problems View.
2. Extracting the model using the command line build tooling. Errors are reported in relation to the type of a attribute, parameter or relationship not being found.

Extension to broken reference resolution

To account for the possibility of references being broken during a product upgrade and avoid the requirement for manual intervention, an extension to the Rational-provided resource resolution process is provided.

The extension requires a map of the previous model's IDs is extracted and then is used to resolve references in the current model. This map is processed to look up the broken ID and determine the qualified name of what it was previously referring to and from this resolve the breakage through discovery of the id in the new model for the qualified name found.

As the map needs to be extracted from the previous model an export option has been introduced into Rational® Software Architect Designer which should be run against the previous model and it should be called as follows:

1. Navigate to **File** > **Export** > **Curam** > **Qualified Name Map**.
2. Select the project to export, for example, *EJBServer*.
3. **Browse** to a location to save the file.
4. Click **Finish** to start the export process.

The output of this task is a model map that you reference when opening a new upgraded model.

To reference the map, a **Preference** page is used within Rational® Software Architect Designer as follows:

1. Navigate to **Window** > **Preferences** > **Curam** > **Qualified Name Map**
2. Browse to the map that you created earlier.

Resource reference resolution process

If an error is found indicating a broken reference, open the model containing the broken reference and a dialog pops up that indicates a broken reference.

The repair process should then resolve and correct the reference.

If the process fails and the reference remains broken, it generates an error in the **Problems** view. Here there is a right-click option offering an additional **Search or browse for a valid reference** which can be used as a last resort.

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

Any references in this information to non-Merative websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this Merative product and use of those websites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Merative, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at https://www.merative.com/privacy.

Trademarks

Merative ™ and the Merative ™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.