Acme Lab 2:
Defining Styles in AcmeStudio

By now, you should be familiar with the basic features of AcmeStudio to describe C&C architectures. This lab will introduce you to some of the advanced features of AcmeStudio and in particular how to modify and create a family, edit visualizations, edit rules and properties, and check rules.

Example: A Three-Tiered Family

For this lab, you will augment an existing three-tiered family. To do this, we will add a couple of new types, edit visualizations, and add several properties and rules.

Importing a Project from Zip archive

First we need to create a new project and place in it the files supplied to you.

Create a new Project called AcmeLab2

1. From the file menu, choose File → New → Project. A wizard dialog appears.
2. Select AcmeStudio New Wizards → Acme Project and click [Next].
3. Enter AcmeLab2 as project name and click [Finish].

Import a Zip archive into the Project

4. Download AcmeLab2.zip
5. In the Navigator, select the project AcmeLab2
6. From the menu bar, choose File→ Import…
7. Select Archive file from the Import dialog box and click [Next >].
8. Browse to the location of the Zip file distributed with this tutorial and click [OK].
9. Make sure AcmeLab2 is the project folder selected for Into folder.
10. Click [Finish].

Modifying an existing Family

From the project AcmeLab2, open the Tiered family under families/TieredFam.acme. The editor opens the family in the Family editor. We will work within this family editor for most of the lab.

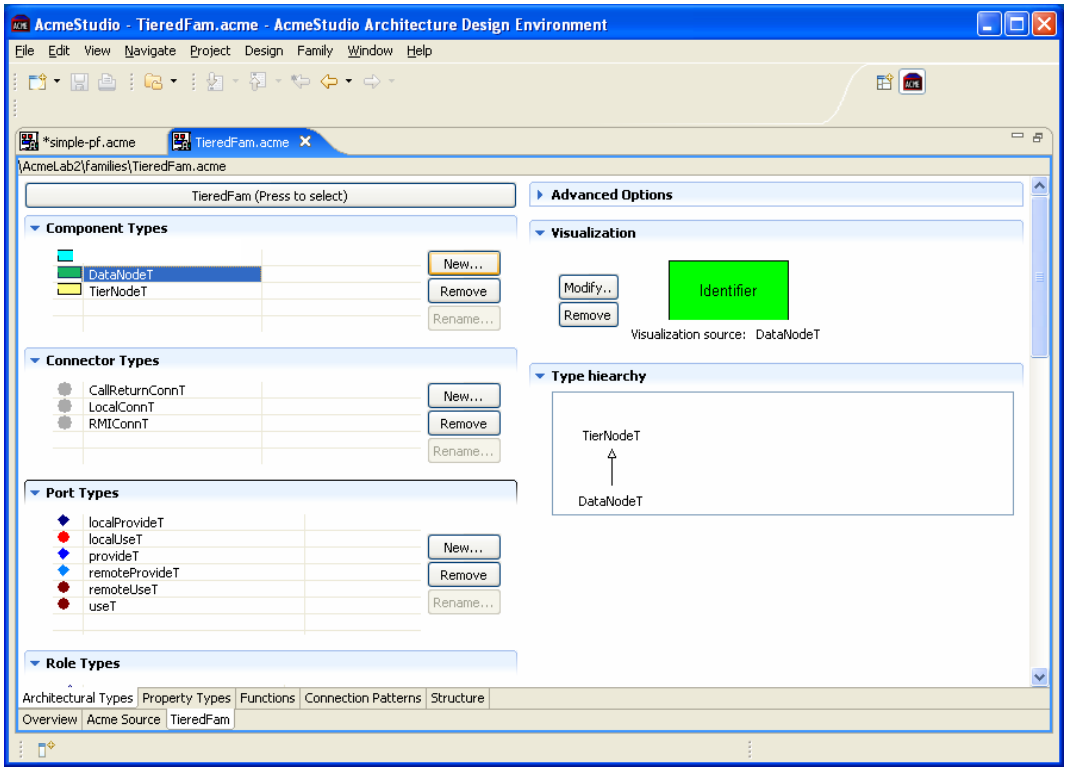


Figure 1: AcmeStudio showing the TieredFam opened in the Family editor.

Browse types

Before we begin making changes, let's inspect the types defined in this family. One way to do this is to via the Properties View. In the lower-right quadrant of AcmeStudio, make sure that the Properties View is activated. (If not, activate it by clicking on the bottom Properties View tab.) To browse a type, click on the type in the Family editor (e.g., DataNodeT) and check its properties, rules, structure, types, and other attributes available in Element View. Another way to browse types is to use the Inspect type function, useful in cases when you cannot edit the family, such as a global family (i.e., families distributed with AcmeStudio).

1. From the menu bar, Choose Family → Inspect Type…. The dialog box that opens shows a list of component, connector, port, role, and property types defined in the family.
2. Double-click on DataNodeT. Another dialog box appears showing five tabs on five different kinds of information about this component type, its properties, rules (constraints), sub-structure, inherited types, and Acme source.
3. Explore some of the other types in this family.
4. Click [OK] to exit this dialog.

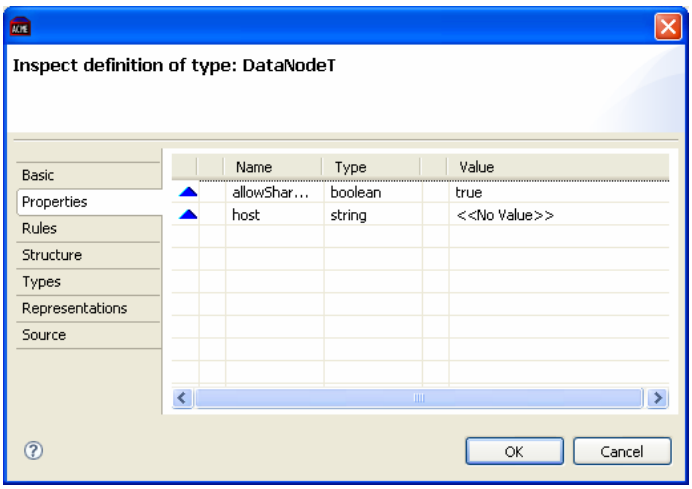


Figure 2: Dialog box for inspecting element types, in this case, DataNodeT.

Change visualizations

Presentation is an important aspect of architecture description. In AcmeStudio, we have considerable flexibility in specifying the way a family should look. For example, we can choose geometric shapes, colors, icon decorations, label fonts and alignments, and port alignment policy for components and role alignment policy for connectors.

For this tutorial, the first thing we will do is to change the shape of the DataNodeT component type to a repository shape and its color to dark blue fill, green outline, and white label.

1. Select DataNodeT; in the right column of the editor, the visualization preview will change to be the current visualization of the type. Select the Modify… button, which will open the Visualization Editor dialog. Note that there are four tabs to this dialog—Shape, Label, Variants, and Port Policy. The first three tabs are standard tabs that appear in the visualization dialog of every visualized element type (components, connectors, ports, roles). The fourth is specific to the element type and defines layout policies.
2. Let's change the shape. Back under the Shape tab, look for the Basic shape section and find the Stock image dropdown menu. Pull down the selection and choose Repository. Notice that the Preview shape has changed.
3. Next we will change the fill color to dark blue. Under the Color/Line Properties section, click on the button next to Fill Color. A color palette dialog box appears.
4. Select a dark blue color and click [OK]. Notice the change in Preview.
5. Next we will change the outline color to green. Click on the button next to Outline Color under the Properties box. A color palette dialog box appears.
6. Select a green color and click [OK]. Notice the change in Preview.
7. Lastly we will change the color of the label. Click on the Label tab and in the Font Settings section, click the button next to Font Color in the Font Settings section. Choose the white color on the palette and press [OK]. The figure below shows what the result looks like if you return to the Shape tab.
8. Press [OK] on the visualization dialog box. The visualization for DataNodeT is now changed.

At this point, you might try experimenting with other forms of customizations for this and other element types.

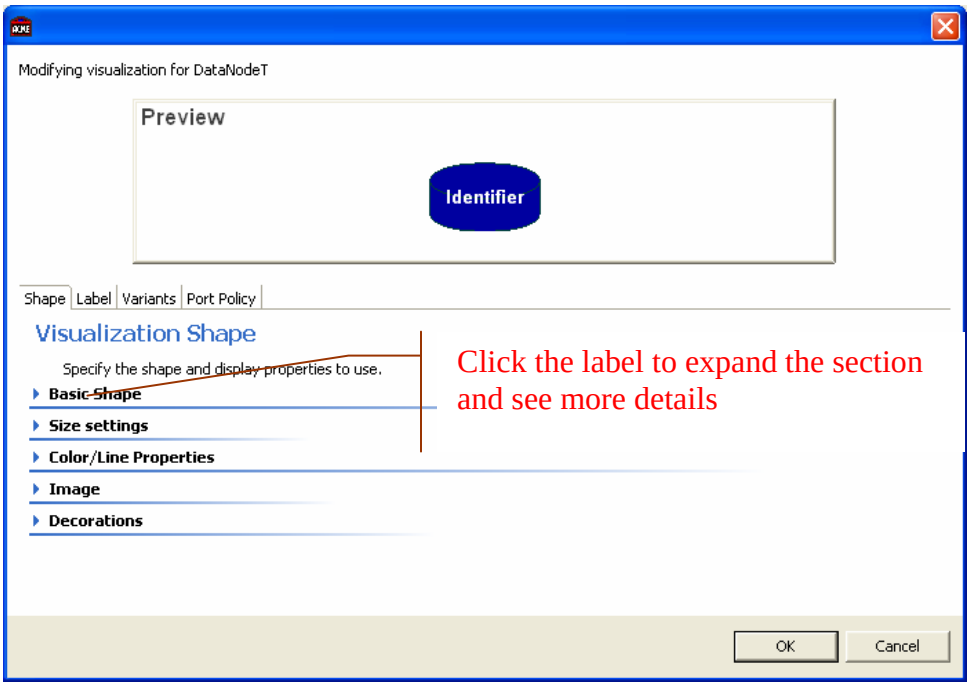


Figure 3: Visualization options for DataNodeT to customize the geometric shape.

Add a new element type

You might notice that although this family is a three-tiered family, it is actually missing the component type definition for one of the tiers: the "Logic" tier. Let's add a LogicNodeT component type.

1. In the Family editor, select the New… button in the Component Types section. A type entry dialog box appears.
2. Enter `LogicNodeT` for the Type Name.
3. Select `TierNodeT` as the Super type.
4. Notice that you can click [Next] a few times to enter ports, properties, and rules. We'll just click [Finish].
5. Notice that the newly added component type now appears in the Component type table.
6. Change the visualization of the LogicNodeT so that it is light-green filled with a black outline of size 2, everything else remaining the same. The outcome is shown in Figure 5.

Add a new property type

Since this is a three-tiered family, we want to make it explicit that a node belongs to a tier via a property for each node. This property can be defined using a property type that specifies an Enum with three values. The node of each tier can then instantiate a property of this type assigning the appropriate tier value. Let's start by creating a property type.

1. In the Family editor, select the Property Types editor (at the bottom of the family editor), and open the Property Types section by selecting on the label.
2. Select the New… button in this section. A type entry dialog box appears.
3. Enter `TierPropT` for the Type Name.
4. Select enum as the Type. The Values field becomes enabled.
5. For the values, enter `client, logic, data`.
6. Figure 4 below shows what the dialog box looks like at this point. Click [OK].

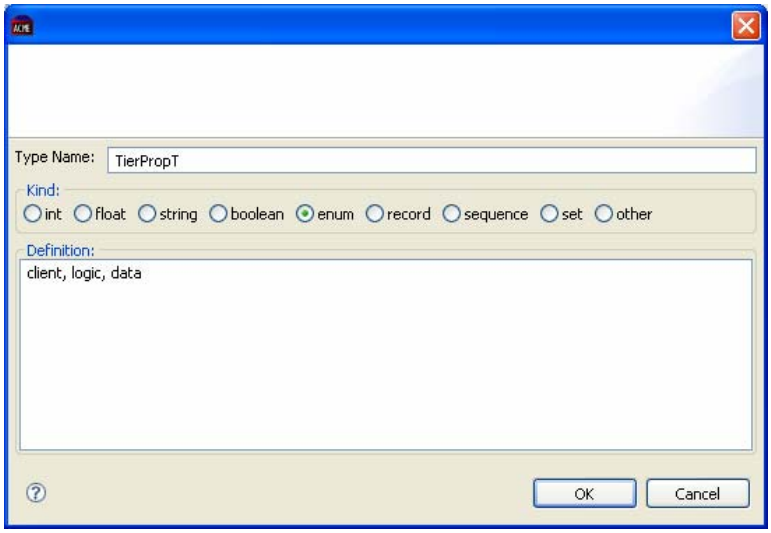


Figure 4: New Property Type dialog box.

7. Notice that the newly added property type appears in the Property Types section. To brows the details, you should open the Property Type Details section.

Editing with the Properties View

Switch back to the Architectural Types editor. To continue the task of adding a tier property to each type of node, we will now add a property and a rule.

Add a property

We begin by adding a property called tier to each of the three tier node types. To do this, we simply add a property to the parent node type TierNodeT, since the other types are subtypes.

1. Activate the Properties View tab.
2. In the Family editor, choose TierNodeT. The details of TierNodeT are shown in the Properties View. We will call this process "focusing on" an element.
3. Select the Properties tab if it's not already selected.
4. Right-click in the white region and select New Property… or click the  icon on the Element View toolbar at the right-hand corner of the Element View title bar.
5. For the Name, enter `tier`.
6. For the Type, choose TierPropT, which we defined in the previous section.
7. Figure 6 below shows what the dialog box looks like at this point.

8. Note, that because TierNodeT is a root type, we want to have sub-types and instances of it specify the actual value for the tier property. Therefore, we will not assign a value to the property here. Click [OK].

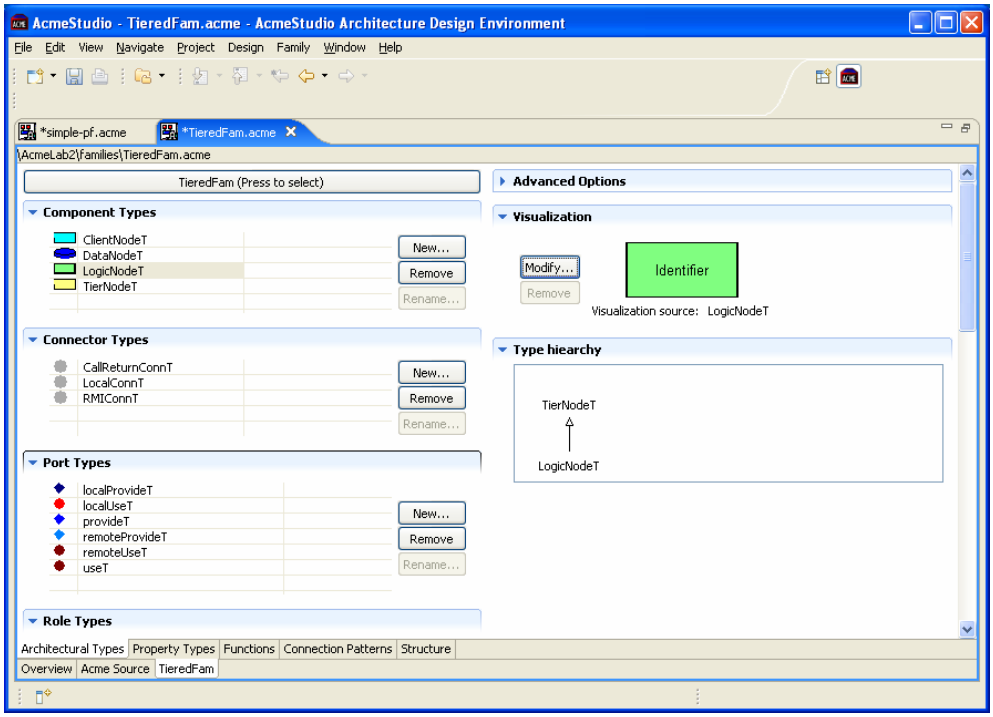


Figure 5: The resulting TieredFam description after adding some types and changing visualizations.

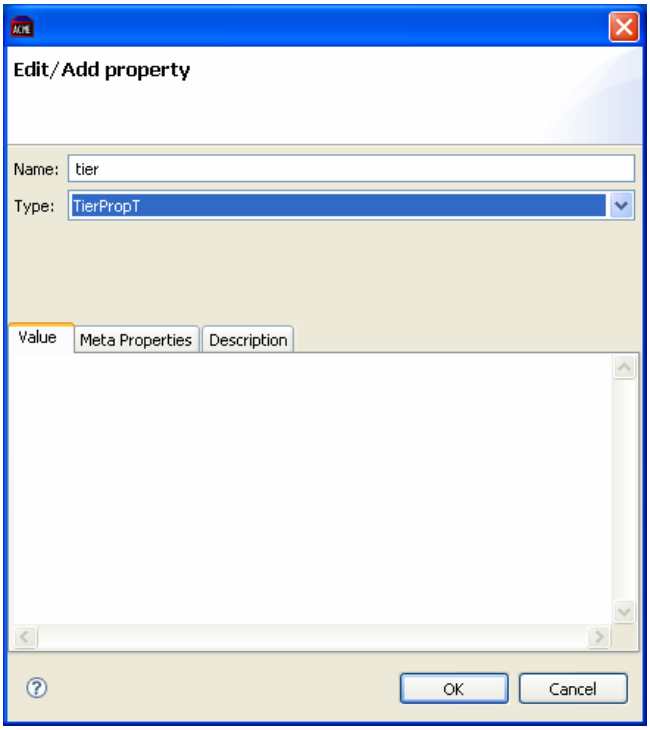


Figure 6: Property editing dialog for the "tier" property.

Add a rule

You might have noticed that the TierNodeT component type already defines two properties called host and allowShareHost. The first property is a string that indicates the host on which an instance of the TierNodeT component type runs. The second property is a Boolean that indicates whether some other nodes can run on the same host. These properties are meant to indicate that a node that doesn't allow sharing a host should not be on the same host as another node. We would like AcmeStudio to flag a warning if this condition is violated. To do this, we will add a rule that captures this constraint. In order to apply to all elements in the system, this rule needs to reside at the family level.

1. In the Family editor, focus on TieredFam by pressing the button above the Component Types section labeled TieredFam (Press to select).
2. Select the Rules tab in the Properties View.
3. Click the  icon on the Properties View toolbar.
4. Name the rule `hostCheck` and ensure that the invariant button is selected.
5. Type the following in the Design Rule textbox:

```
// Two nodes that cannot share the same host must not reside on the same host

Forall t1 : TierNodeT in self.Components |
  !t1.allowShareHost -> (Forall t2 : TierNodeT in self.Components |
    t1 != t2 -> t1.host != t2.host)
```

6. Click [Parse Rule] to make sure there's no error. You must execute this step in order to complete the rule definition. If there is a parse error, you must fix it (or click [Cancel]) before you can continue.
7. You may type a descriptive label for this rule, which will appear in the Rules list when the rule is satisfied. Let's label this one, "`Tier nodes respect host assignment.`"
8. You may also type a descriptive error label, which will appear in the Rules list when the rule is violated. Let's label this one, "`Two nodes that cannot share a host must not reside on the same host.`"
9. Click [OK] to complete.
   Note: if [OK] doesn't respond, you may have to click [Parse] again.
10. You should make sure that you save the family.

Optional challenge problem: See if you can come up with a heuristic that flags a warning if a particular logic node has more than three clients connected to it. Hint: Assume that each client node connects to a logic node on separate ports. Then define a heuristic associated with LogicNodeT that limits the number of ports. To count the number of ports, use the set constructor `select` to select the subset of ports in the component that declares the port type of provideT (e.g., `{select p : Port in SomeSet | declaresType(p, t) }`). A component can refer to itself as `self`. The set of ports defined on a component can be referenced via `self.Ports`. Then apply the `size()` function to that set.

Acme Source editing

Sometimes it's easier to make certain changes by directly editing the Acme source, such as changes that involve significant redundancy where copy and paste becomes handy. This can be accomplished using the Source editor. Note that ordinarily, using the Source editor requires sufficient understanding of the Acme syntax. However, we will walk through a couple of changes without relying on you to generate Acme.

Recall that we previously added a tier property to the componet supertype TierNodeT. However, we haven't made use of that property in each of the component subtypes. We will now apply the tier property to the three subtypes and assign the appropriate property value—`client` for ClientNodeT, `logic` for LogicNodeT, and `data` for DataNodeT.

1. Notice three tabs at the bottom of the Editor region— Overview, Acme Source and Family – TieredFam. To switch to the Source editor, click on Acme Source and the editor shows you the source of the family's Acme description.
2. Locate the component type TierNodeT, which can be done in a number of ways.
3. Highlight the line containing the property tier and copy it to clipboard (this could be done by pressing Ctrl-C or Ctrl-Insert or choosing from the menu bar Edit → Copy).
4. Locate ClientNodeT component type and paste the property text inside the component type definition (between the two curly braces).
5. Before the semicolon of the tier-property text, add `= client`. The entire component type definition should look like:

```
Component Type ClientNodeT extends TierNodeT with {
      Property tier : TierPropT = client;
    }
```

6. Do the same pasting and editing for LogicNodeT (assigning `logic`) and DataNodeT (assigning `data`).
7. After making these changes, the Source editor content must be synchronized with the Family editor content. Clicking on the Family – TieredFam editor tab will initiate the synchronization. Note that if there were major errors from the source change, synchronization will fail, and you will be taken back to the Source editor. Otherwise, the Family editor appears again.

Error indicators: When you change something that causes a parse error or type-check error in the Source editor, small red icons appear at the appropriate error locations. Parse errors must be removed. Certain type-checking and constraint-checking errors in the Family may be ignored, specifically, "does not satisfy constraint" or "does not satisfy type". These will often occur in Family descriptions and can only be satisfied in the instantiated system. For example, if a port type *P* declares a rule that at least one role must be attached, and one port instance *p* of *P* is defined in a component type *C*, the rule will not be satisfied for *p* until an instantiated system where instances of *C* are defined and roles of connectors attached to the components' ports.

Visualization variants

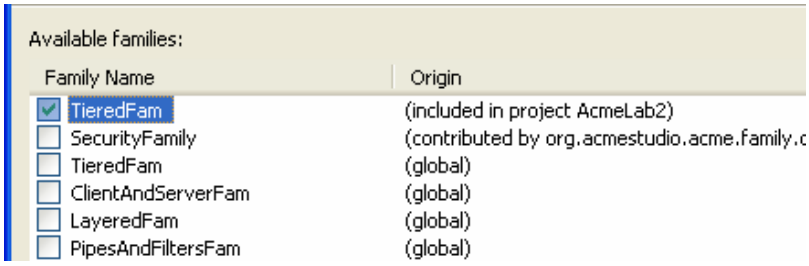
The visualization of an element can be set to vary depending on the values of its properties. This form of visualization is called a Visualization Variant. Let us create a Visualization variant for the LogicNodeT component type that would change the outline to a red line if the tier property doesn't indicate `logic`.

1. In the Family editor, edit the visualization of the LogicNodeT component.
2. Click on the Variants tab.
3. Click [New…] to create a new variant.
4. Name the variant `Not logic tier`.
5. Make sure the radio button is selected for Condition is based on property.
6. Pull the Property dropdown menu and select allowSharedHost, then choose == for the Condition, and enter `false` for the Value.
7. Choose dark green for the fill-color.
8. Click [OK] to create the new variant and return to the Visualization dialog box.
9. Select the newly added variant and check the Preview to make sure it shows a red outline.
10. Click [OK] to finish.

Testing the redefined Family

To test the family you modified, let us now create a new system based on the family.

1. In the Navigator, right-click anywhere in the current AcmeLab2 project. In the pop-up menu, choose New → Acme System. A progress indicator dialog appears while AcmeStudio searches for and parses the families available to this project.
2. When the System Editor File dialog appears, make sure that the Container says /AcmeLab2. If not, click [Browse] and choose AcmeLab2.
3. For the System name, type `TestSystem`.
4. Click [Next].
5. Notice a list of Available families. The TieredFam family is subdivided into a builtin family and a local family. This is because there is actually a global family with called TieredFam that is released with AcmeStudio. Choose your local version of TieredFam. A structured list of the types defined in this family appear in the Details box.



6. Click [Finish], and you have instantiated an empty system of the family TieredFam. The default system editor is the Diagram editor. Notice the palette of types on the left from which you can drag and drop an element to create an instance.
7. Compose a three-tiered system much like the one in Assignment 2. Change a logic node's tier property to a value other than logic and see if the visualization variant works. Test the global constraint you defined that limits the number of

clients and see if it works. You can do this by instantiating four client nodes, creating four ports on a logic node, creating four connectors to attach the client nodes to the logic node. Don't forget to use the Element View to check whether rules are satisfied. Recall that if an invariant fails, the  icon is placed next to the rule; if a heuristic fails, the  icon appears.[1]

Other advanced features to play with

There are many more features of AcmeStudio not covered in this lab that you may wish to experiment with. Try different things out. Some of the more complex features are listed below:

- Adding descriptions to the types
- Defining Port policy and alignment for Component type
- Defining Role policy and alignment for Connector type
- Defining Connection Patterns
- Generating an Acme Legend
- Exporting to XML or graphics
- Applying external analysis tools, such as performance analysis

[1]  indicates that the rule is running,  indicates that there is an evaluation error in the rule, indicates that constraint evaluation has been turned off.