XMI Reader Diagram Interchange Plugin Component 1.0 Specification

1. Design

The XMI Reader Diagram Interchange component is a plugin for XMI Reader component. It implements the `ContentHandler` (from org.xml.sax) interface and is able to parse all the elements for the classes in *Diagram Interchange* component.
The main purpose is to be able to create an object model for the xmi, which describes (or declares) the instances of classes that make up the *Diagram Interchange* component.

1.1 General approach

1.1.1 Anatomy of the proposed design

The basic tasks of this design can be broken up as follows:

1. We will need to have the ability to parse out the XMI data that defines Diagram. We will mostly deal with parser events such as startElement and endElement, which will hold the bulk of our processing.
2. Actually we will even go further, and create something of a factory, which will generate the appropriate object based on the type of Element that is being read in XMI. Thus we will have the ability to plug-in new classes of model items (such as GraphNode, GraphEdge, or DiagramElement) into our factory. This factory is fully configurable and works on the assumption that each such class acts as a java bean and has a default constructor as well as well defined setters.

1.1.2 Basic Parsing approach

The basic aspect of this component is of course the parsing of the xmi. The approach that this design takes is quite straightforward:

- As we read the xmi (xml) the parser will fire off `startElement` and `endElement` events, which our handler will process.
- Each `startEvent` will tell us of a (possibly) new element of the graph which we need to translate into an object instance.
- For each <UML:XXX element we try to match it to a specific class and we instantiate through reflection.
  For example:
  ```
  <UML:Diagram xmi.id = 'I10ad419m10729d8da08mm7f54' isVisible = 'true'
  name = 'Class Diagram_1' zoom = '1.0'>
  ```
  Would result in creation of a new `DiagramImpl` instance which would have its `setVisible()`, `setName()` and `setZoom()` invoked (more on method invocation later)
- Once an element has been instantiated it will be placed in the XMIReader's foundElements map for future reference with the key being xmi.id This way when we later need to find this instance we simply look for an element with xmi.idref equal to the xmi.id just placed (since xmi uses references when an element is shared across the file)
- This process is basically repeated for each subsequent note/tag which will eventually build all the pieces of the Diagram.

Here is a fragment xml (xmi) for it in UML 1.5, which describes the Diagram Interchange aspects (the full xmi is provided in \docs\xmi_example.xmi):

Listing 1. Example of Diagram Interchange xmi (for diagram 1)

```
<UML:Diagram xmi.id = 'I10ad419m10729d8da08mm7f4e' isVisible = 'true'
      name = 'State diagram_1' zoom = '1.0'>
  <UML:GraphElement.position>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:GraphElement.position>
  <UML:GraphNode.size>
    <XMI.field>394.3838</XMI.field>
    <XMI.field>230.0</XMI.field>
  </UML:GraphNode.size>
  <UML:Diagram.viewport>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:Diagram.viewport>
  <UML:GraphElement.semanticModel>
    <UML:SimpleSemanticModelElement xmi.id = 'I10ad419m10729d8da08mm7f4d' presentation = ''
      typeInfo = 'StateDiagram'/>
  </UML:GraphElement.semanticModel>
  <UML:GraphElement.contained>
    <UML:GraphNode xmi.id = 'I10ad419m10729d8da08mm7f44' isVisible = 'true'>
      <UML:GraphElement.position>
        <XMI.field>80.0</XMI.field>
        <XMI.field>100.0</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>100.0</XMI.field>
        <XMI.field>50.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:Uml1SemanticModelBridge xmi.id = 'I10ad419m10729d8da08mm7f43' presentation = ''>
          <UML:Uml1SemanticModelBridge.element>
            <UML:SimpleState xmi.idref = 'I10ad419m10729d8da08mm7f45'/>
          </UML:Uml1SemanticModelBridge.element>
        </UML:Uml1SemanticModelBridge>
      </UML:GraphElement.semanticModel>
      <UML:GraphElement.contained>
        <UML:GraphNode xmi.id = 'I10ad419m10729d8da08mm7f42' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>0.0</XMI.field>
            <XMI.field>0.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>100.0</XMI.field>
            <XMI.field>19.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I10ad419m10729d8da08mm7f41'
                 presentation = '' typeInfo = 'NameCompartment'/>
          </UML:GraphElement.semanticModel>
          <UML:GraphElement.contained>
            <UML:GraphNode xmi.id = 'I10ad419m10729d8da08mm7f40' isVisible = 'true'>
              <UML:GraphElement.position>
                <XMI.field>11.1697</XMI.field>
                <XMI.field>2.0</XMI.field>
              </UML:GraphElement.position>
              <UML:GraphNode.size>
                <XMI.field>77.6606</XMI.field>
                <XMI.field>15.0</XMI.field>
              </UML:GraphNode.size>
              <UML:GraphElement.semanticModel>
                <UML:SimpleSemanticModelElement xmi.id = 'I10ad419m10729d8da08mm7f3f'
                     presentation = '' typeInfo = 'Name'/>
              </UML:GraphElement.semanticModel>
            </UML:GraphNode>
          </UML:GraphElement.contained>
        </UML:GraphNode>
        <UML:GraphNode xmi.id = 'I10ad419m10729d8da08mm7f3e' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>0.0</XMI.field>
            <XMI.field>19.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>100.0</XMI.field>
            <XMI.field>1.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I10ad419m10729d8da08mm7f3d'
                 presentation = '' typeInfo = 'CompartmentSeparator'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
        <UML:GraphNode xmi.id = 'I10ad419m10729d8da08mm7f3c' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>0.0</XMI.field>
            <XMI.field>20.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>100.0</XMI.field>
            <XMI.field>10.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I10ad419m10729d8da08mm7f3b'
                 presentation = '' typeInfo = 'InternalTransitionCompartment'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
      <UML:GraphElement.anchorage>
        <UML:GraphConnector xmi.id = 'I10ad419m10729d8da08mm7f20'>
          <UML:GraphConnector.position>
            <XMI.field>50.0</XMI.field>
            <XMI.field>50.0</XMI.field>
          </UML:GraphConnector.position>
          <UML:GraphConnector.graphEdge>
            <UML:GraphEdge xmi.idref = 'I10ad419m10729d8da08mm7f1e'/>
          </UML:GraphConnector.graphEdge>
        </UML:GraphConnector>
        <UML:GraphConnector xmi.id = 'I10ad419m10729d8da08mm7f17'>
          <UML:GraphConnector.position>
            <XMI.field>100.0</XMI.field>
            <XMI.field>25.0</XMI.field>
          </UML:GraphConnector.position>
          <UML:GraphConnector.graphEdge>
            <UML:GraphEdge xmi.idref = 'I10ad419m10729d8da08mm7f15'/>
          </UML:GraphConnector.graphEdge>
        </UML:GraphConnector>
      </UML:GraphElement.anchorage>
    </UML:GraphNode>
    [...]
```

The one thing to note is that this is just a simple xml file with a very specific (xsd is provided in \docs\di-schema.xsd) format. The way that we will have this handled is by simply walking down the DOM of the xmi and by pushing specific elements in form of SAX parsing events.
For example we could envision that parsing the above information would produce the following order of events (a small subset of the above xmi), where each event is depicted as an entry into an xml element:

Example 1. Elements that will be translated into actual objects

```
<UML:Diagram>                           // We build a Diagram instance
  <UML:GraphElement.position>           // We build a Point instance
    <XMI.field>                         // setX for the point
    <XMI.field>                         // setY for the point
  <UML:GraphNode.size>                  // We build a Dimension instance
```

```
      <XMI.field>                         // setWidth for the size
      <XMI.field>                         // setHeight for the size
    <UML:Diagram.viewport>                // We build a Point instance
      <XMI.field>                         // setX for the point
      <XMI.field>                         // setY for the point
    <UML:GraphElement.semanticModel>      // We build a SimpleSemanticModelElement instance
      <UML:SimpleSemanticModelElement>    // set properties of the element
    <UML:GraphElement.contained>          // We build a DiagramElement instance
```

One very important aspect to note is that we are dealing with a pointer based data structure in the sense that some elements are defined using ids, which then get resolved into proper objects. Thus we need to be aware that in XMI we will have references and objects (classes)

Consider how graph connectors are actually defined. We have an actual physical definition of a GraphConnector as in:

Example 2. GraphConnector definition in xmi

```
<UML:GraphConnector xmi.id = 'I10ad419m10729d8da08mm7f17'>
  <UML:GraphConnector.position>
    <XMI.field>100.0</XMI.field>
    <XMI.field>25.0</XMI.field>
  </UML:GraphConnector.position>
  <UML:GraphConnector.graphEdge>
    <UML:GraphEdge xmi.idref = 'I10ad419m10729d8da08mm7f15'/>
  </UML:GraphConnector.graphEdge>
</UML:GraphConnector>
```

And then we have a reference to this as in:

Example 3. GraphNode reference example

```
<UML:GraphEdge.anchor>
  <UML:GraphConnector xmi.idref = 'I10ad419m10729d8da08mm7f17'/>
  <UML:GraphConnector xmi.idref = 'I10ad419m10729d8da08mm7f16'/>
</UML:GraphEdge.anchor>
```

This means that our algorithm will have to basically build a mapping of ids to entities and then fill in the complete model with instances of classes that are substituted for the ids.

1.1.3 Mapping from xml tags elements to actual UML model

The first thing is to know how to create proper model objects based on parsed elements. This actually rather uncomplicated and works on the premise that a number of tags will map directly to implementation classes. For example the `<UML:DiagramLink …/>` tag maps directly into our `DiagramLinkImpl` class, and thus when we encounter this tag we will be most probably creating (unless it is a reference) a new `DiagramLinkImpl`.
Here is a table that maps the main classes used here to their xmi tags:

Table 1. Diagram Interchange mapping from xmi to Object Model

Xmi element	Object Model element	Description
UML:Property	Property	Class
UML:Reference	Reference	Class
UML:GraphElement.position UML:GraphElement.semanticModel UML:GraphElement.contained UML:GraphElement.anchorage UML:GraphElement.link	Point SimpleSemanticModelElement List<DiagramElement> GraphConnector Collection<DiagramLink>	Class Class List of DiagramElement(s) Class Collection of DiagramLink(s)
UML:DiagramLink UML:DiagramLink.viewport	DiagramLink Collection<Point>	Class Collection of Point(s)
UML:GraphConnector	GraphConnector	Class

UML:GraphConnector.position	Point[]	Class
UML:GraphEdge UML:GraphEdge.waypoints	GraphEdge List<Point>	Class List of Point(s)
UML:Point	Point	Class
UML:BezierPoint UML:BezierPoint.controls	BezierPoint List<Point>	Class List of Point(s)
UML:Dimension	Dimension	Class
UML:Polyline	Polyline	Class
UML:Ellipse UML:Ellipse.point	Ellipse Point	Class Class
UML:TextElement	TextElement	Class
UML:Image	Image	Class
UML:CoreSemanticModelBridge	CoreSemanticModelBridge	Class
UML:Uml1SemanticModelBridge	Uml1SemanticModelBridge	Class
UML:SimpleSemanticModelElement	SimpleSemanticModelElement	Class
UML:GraphNode UML:GraphNode.size	GraphNode Dimension	Class Class
UML:Diagram UML:Diagram.vieport UML:Diagram. diagramLink UML:Diagram.owner	Diagram Collection<Point> DiagramLink SemanticModelBridge	Class Collection of Point(s) Class Class

But we have a special case here as well:

1. If the element has a attribute named `xmi.id`, we first check if a reference object was created for this (due to a forward `xmi.idref` declaration) and create a new create a new object if it wasn't.
2. If this attribute doesn't exist we then look for an `xmi.idref` attribute.
    a. If we have the attribute then we use its value to lookup to see if this object was already created.
    b. If it has been created we simply fetch it. And if it has not been created, we create it and place the empty object instance into a table of forward references.

1.1.4 Using reflection to map setters to their xmi counterparts

The object instances that we create are empty, and they need to be filled up with the data from xmi.
The most challenging aspect of this task is to parse out enough information to match the data with the proper setter in the class. The idea is that we will use the fact that the XSD definition for the xmi UML is exactly matched in the setters and getters of all the models in the supporting components. This means that when we encounter something like `<UML:GraphEdge.anchor …/>` we can very easily 'guess' the correct setter of the `GraphConnectorImpl` class (which maps to `UML:GraphConnector`) will be `setAnchor`.
This means that most of the time we will easily be able to construct the correct setter call. But there are special cases that deal with Collections, Lists and boolean variables. We can follow this simple algorithm as follows (assume some elementName):

1. We check if we have a setElementName method in the instance.
2. [If failed] we next check if this was a Collection and we look for an addElementName method

Another case is if the element name is in the format of isElementName.

3. We check if we have a setElementName (we drop the 'is')

These are currently the only three possibilities (based on the XSD) for setters.

1.1.5 How do we know when we have a setter?

1. Anytime we have an attribute we have a setter (except that we do not treat the xmi.id and xmi.idref) as setters.

Any time we have a tag in the format of <UML.inner.outer … /> the outer is treated as a setter. So for example if we had <UML:GraphEdge.anchor . . .> anchor would be the outer element and we would have a setter in the form of setAnchor(…)

1.1.6 Element ids and reference ids

We also need to make a distinction when we are using xmi.id vs. xmi.idref. The difference is crucial to parsing. The xmi.id refers to the actual entity (i.e. we will be creating an instance for it) whereas xmi.idref is a reference to such an entity. In effect it is a pointer to the definition of an entity.
Please note that the xmi.id xmi.idref is what the XMIReader component uses when its XMIReader.foundElements and XMIReader.forwardReferences are being used. xmi.id is used as a key in foundElements and xmi.idref is usually used in forwardReferences.

1.2 Design Patterns

We can say that the **Strategy** pattern is used since the handler that we are implementing plugs-in into the reader.
Also **Factory** pattern is used to create instances of different model classes based on some key.

1.3 Industry Standards

UML 1.5
XMI 1.2

1.4 Required Algorithms

The main algorithm is of course the actual parsing of the document for the specific elements.

1.4.1 Parsing the document's elements

The implemented SAX handler is really doing things through a call back. The main methods that we care about are startElement and endElement, which tell us that we have entered or existed an element.
The general aspect will be as follows (this is an example that the developer can generalize):

<u>StartElement:</u>
For each element that we get through start element we do the following:
We get the elements name:

Based on the element name we do the following:

```
if(name == “UML:Property”){
        // We create a new Property instance (property)
        property = getModelElementFactory.createModel(“UML:Property”);
        // We set all the provided data in properties according to the xsd
        property.setKey( attribute key );
        property.setValue( attribute value );
        // We also get the xmi.id and we use this id as follows:
        getXMIReader().putElement(xmi.id, property);
        // we also set the last object to be Property
        getXMIReader().setLastObject(property);

}
if(name == “UML:GraphConnector”){
        // We create a new GraphConnector instance (graphConnector)
        graphConnector = getModelElementFactory.createModel(“UML:GraphConnector”);
        // We set all the provided data in properties according to the xsd.
        // We also get the xmi.id and we use this id as follows:
        getXMIReader().putElement(xmi.id, graphConnector);
        // we also set the last object to be graph connector
        getXMIReader().setLastObject(graphConnector);

}
if(name == “UML:GraphNode.size”){
        // Create a new instance. . . call it size
        graphConnector = getModelElementFactory.createModel(“UML:GraphNode.size”);
        // We also get the xmi.id and we use this id as follows:
        // getXMIReader().putElement(xmi.id, size);
        // This is an intermediate node, we will just remember it
        getXMIReader().setLastProperty(size);
}
if(name == “UML:GraphElement.semanticModel”){
        // Create a new instance. . . call it semantic_model
        model =
                getModelElementFactory.createModel(“UML:GraphNode. semanticModel”);
        // We also get the xmi.id and we use this id as follows:
        // getXMIReader().putElement(xmi.id, size);
        // This is an intermediate node, we will just remember it
        getXMIReader().setLastProperty(size);
}
if(name == “XMI.field”){
        // This is used to populate points
        // We need to get the last object used
        last = getXMIReader().getLastProperty();
        // test what we have
        if(last == instanceOf Point){
        // We set the x value of this point with the XMI.field data
        // if x value is already set we the y value instead
        }
}
```

Note that similar ideas will apply to all the UML elements. The developer should have no issues expanding the above simple exposition.

endElement:

```
        if(name  == “UML:Diagram”){
                // use the xmi.id to get the actual object
                Diagram diagram =
                                (Diagram)(getXMIReader().getElement(xmi.id));
                // Add this to the uml model manager
                getUMLModelManager().addDiagram(diagram);
        }
```

1.4.2 Managing Properties

During parsing, "forward references" will likely be encountered. XMIHandler implementations can still capture properties about elements that have not been fully defined. Specifically, the XMIReader can keep track of which defined elements are "waiting on" forward references. In this XMI fragment, the referenced Stereotype might not have been defined yet, but the XMIHandler for Stereotypes can capture the forward reference:

```
<UML:Operation xmi.id = 'I3d057933m10e2fd324d0mm7242'
             name = 'XMIReaderConfigException'
             visibility = 'public'
             isSpecification = 'false'
             ownerScope = 'instance'
             isQuery = 'false'
             concurrency = 'sequential'
             isRoot = 'false'
             isLeaf = 'false'
             isAbstract = 'false'>

        <UML:ModelElement.stereotype>
             <UML:Stereotype xmi.idref = 'I3fc39a51m10e2acac631mm7e57'/>
        </UML:ModelElement.stereotype>
```

...

To capture this, the XMIHandler for Stereotypes would execute:
```
                reader.putElementProperty("I3fc39a51m10e2acac631mm7e57",
                                 " UML:ModelElement.stereotype", operationObject);
```

This connects the `operationObject` with the as-yet-undefined `UML:Stereotype` object whose id is "`I3fc39a51m10e2acac631mm7e57`", indicating that the `UML:Stereotype` property is the property that should be used to connect the two.

Later, when the Stereotype is actually found, its XMI fragment might look like this:
```
<UML:Stereotype xmi.id = 'I3fc39a51m10e2acac631mm7e57' name = 'create'
       isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
       isAbstract = 'false'>
     <UML:Stereotype.baseClass>BehavioralFeature</UML:Stereotype.baseClass>
</UML:Stereotype>
```

The Stereotype handler could then instantiate a `StereotypeObject` with all its properties, and then tell all the "waiting" objects about the newly created `StereotypeObject`.
```
       Map<String, List> waiting;
       waiting = reader.getElementProperties("I3fc39a51m10e2acac631mm7e57");
```

Then for each property, and each object on the list of that property, they can be told what the Stereotype object is. Finally, the `StereotypeHandler` would actually define the Stereotype object in the reader:

```
       reader.putElement("I3fc39a51m10e2acac631mm7e57", stereotypeObject);
```

which will remove the corresponding entry in the `forwardReferences` map.

Lets take a different example:

```
            <UML:FinalState xmi.id = 'I10ad419m10729d8da08mm7f2f'
                                name = 'Final_State_1'
                                visibility = 'public'
                                isSpecification = 'false'>
                  <UML:StateVertex.incoming>
                      <UML:Transition xmi.idref = 'I10ad419m10729d8da08mm7f2a'/>
                     <UML:Transition xmi.idref = 'I10ad419m10729d8da08mm7f21'/>
                  </UML:StateVertex.incoming>
            </UML:FinalState>
```

step 1 -- XMIReader will call the handler for UML:FinalState which is handler1
(a) ---- handler 1 will create the object (object1) for this element and set the attribute properties to the object.

step 2 -- XMIReader will call the handler for UML:StateVertex.incoming which is also done by handler 1
(a) ---- handler 1will do nothing at this point

step 3 -- XMIReader will call the handler for UML:Transition which lets assume is done by handler2 (for illustrative purposes)
(a) ---- handler2 does the following:
reader.putElementProperty("I10ad419m10729d8da08mm7f2a",

" UML:StateVertex.incoming ", operationObject);
where operation object is the object1 created in step 1(a)

step 4 --XMIReader will call the handler for UML:Transition which lets assume is done by handler2 (for illustrative purposes)
(a) ---- handler2 does the following:
reader.putElementProperty("I10ad419m10729d8da08mm7f21",
" UML:StateVertex.incoming ", operationObject);
where operation object is the object1 created in step 1(a)

So the first question is how does handler2 get the reference to object1 and know about "UML:StateVertex.incoming"?

How do we that?

```
handlers = reader.getActiveHandlers().
lastIndex = handlers.size()-1;
String property = handlers.get(lastIndex-1).getLastProperty();
Object refObject = handlers.get(lastIndex-2).getLastRefObject();
```

Note that for property lastIndex-1 was used, and for refObject lastIndex-2 was used. The handler on lastIndex position should be the currently active handler.

1.5 Component Class Overview

1.5.1 com.topcoder.xmi.reader.handlers.diagraminterchange

DiagramInterchangeXMIHandler <<concrete class>>
This is a concrete (but indirect) implementation of the `ContentHandler` interface. This is done by extending the `DefaultXMIHandler` abstract class, which implements some convenience methods that deal with XMIReader. This is important since we need to utilize XMIReader instance in the actual implementation.
This class is responsible for parsing out Diagram Interchange model elements. This handler uses the ModelElementFactory to instantiate the proper model class instance. Basically as we parse the xml elements we use the element designation name (like `UML:GraphConnector` for example) as a key to create an instance of a class in the Object Model which represents the `UML:GraphConnector` which in our case (depending on configuration of course) would be `GraphConnector`.

Thread Safety : This class is mutable and so is not thread safe. As this class is only intended to be used inside the XMIReader component, the thread safety should be handled by XMIReader component.

1.5.2 com.topcoder.xmi.reader.handlers.modelfactory

ModelElementFactory <<concrete class>>
This is a configurable factory, which creates UML Model class instances based on a mapping from xml element name to a class name of the model element that currently represents it.

This implementation is thread-safe. Even though it is not anticipated that many thread will be hitting the handler, it was decided that it might be worthwhile to have a single copy of the factory (to save memory and configuration read time) Synchronization should be done in a block to minimize performance impact.

1.6 Component Exception Definitions

1.6.1 Custom Exceptions

ElementCreationException:
Thrown when the ModelElementFactory cannot create the requested instance. This could be due to reflection issues, sandbox security issues, or something else.

ConfigurationException:
It will be thrown by the two ModelElementFactory constructors if there are issues with configuration (i.e. wrong format or missing variables) or if Configuration Manager thrown ConfigManagerException .

1.6.2 System and general exceptions

In general this component will check for empty string input or null reference parameter input. It will also check of arrays have any null references or empty strings.

IllegalArgumentException:
This exception is used for invalid arguments. Invalid arguments in this design are used on some occasions with null input elements. The exact details are documented for each method in its documentation.

SAXException:
This exception is used to signal to the invoking SAX parser that the handler is unable to continue parsing the given data.

1.7 Thread Safety

Implementation is not thread-safe, because the SAX parser doesn't require a handler to be thread safe and the thread safety issue should be handled in the XMI Reader component, this component is only a plugin for the XMI Reader component.

2. Environment Requirements

2.1 Environment

- Development language: Java1.5
- Compile target: Java1.5

2.2 TopCoder Software Components

- **UML Model – Diagram Interchange 1.0**

Used to model diagram interchange elements.

- **Config Manager 2.1.5**

Used to implement to read configuration detail for mapping xml element names (q names) with class names, which will be used (via reflection) to create model elements for the xml element.

- **XMI Reader 1.0**

This is the base component for which we are writing this plugin. We not only will plug-in the handler but we also use the XMIReader itself in this component to manager ids and reference ids for xml elements.

- **UML Model Manager 1.0**

This is used to store the parsed out and created Activity Graph diagram

- **Base Exception 1.0**

This is used as a common exception-handling base, which allows for chaining exceptions.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Name

com.topcoder.xmi.reader.handlers.diagraminterchange
com.topcoder.xmi.reader.handlers.modelfactory

3.2 Configuration Parameters

3.2.1 ModelElementFactory configuration for Config manager

Parameter	Description	Values
xml_to_element_mapping	These are 1 or more xml element name mapping to a class name used to create an object, which can store information about the xml element in the UML Model. These are comma delimited values, where the first element is the xml element name and the second element is a class name". ***Required*** Note that sometimes the xml element name might be in the format of <UML:outer.inner> where outer is the main encompassing class, and inner is the included instance	"UML:Diagram , com.topcoder.diagraminterchange.Diagram" "UML:StereoType.definedTag, com.topcoder.uml.model.extensionmechanisms.TagDefinitionImpl"

The actual table of all these associations is shown below. Please note that `<UML:outer.inner>` which is shown with a yellow highlight in there is shown as a concatenation of `<UML:outer` followed by all the instances of `.inner>` that are affiliated with `<UML:outer`. This is done to save space.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

To support the current State Machine and Activity Graph models the user should configure the following mappings:

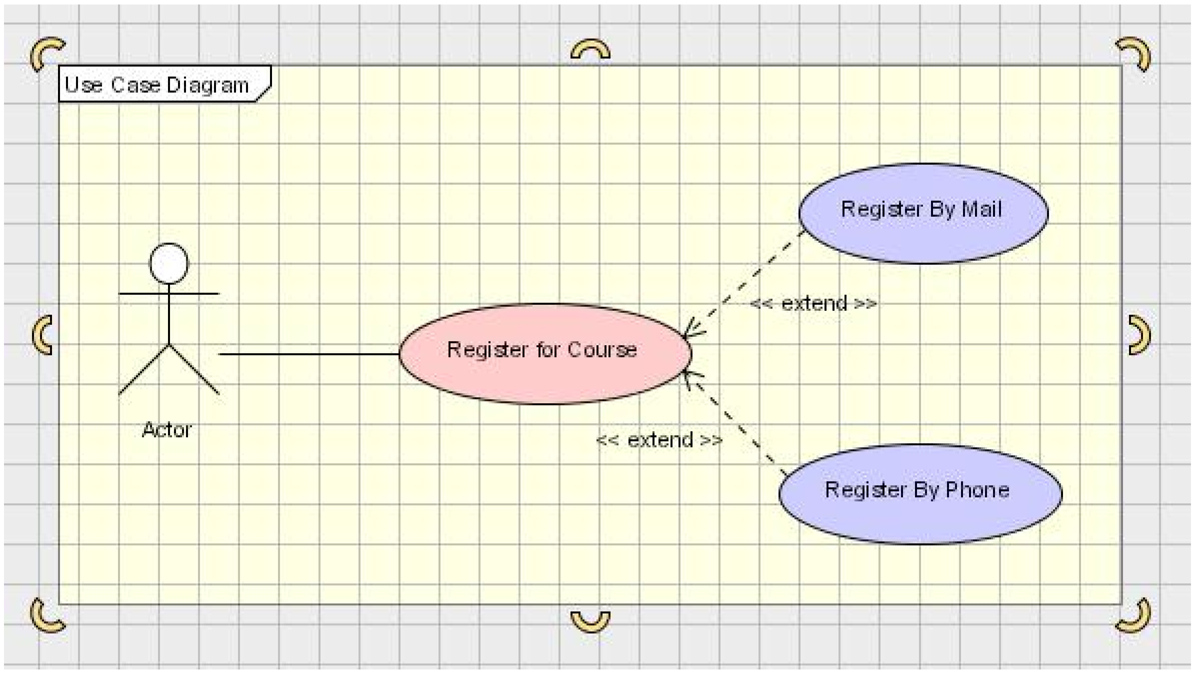
Xmi element	Corresponding class
UML:Property	com.topcoder.diagraminterchange.Property
UML:Reference	com.topcoder.diagraminterchange.Reference
UML:GraphElement.position UML:GraphElement.semanticModel UML:GraphElement.contained UML:GraphElement.anchorage UML:GraphElement.link	com.topcoder.diagraminterchange.Point com.topcoder.diagraminterchange.SimpleSemanticModelElement com.topcoder.diagraminterchange.DiagramElement com.topcoder.diagraminterchange.GraphConnector com.topcoder.diagraminterchange.DiagramLink
UML:DiagramLink UML:DiagramLink.viewport	com.topcoder.diagraminterchange.DiagramLink com.topcoder.diagraminterchange.Point
UML:GraphConnector UML:GraphConnector.position	com.topcoder.diagraminterchange.GraphConnector com.topcoder.diagraminterchange.Property
UML:GraphEdge UML:GraphEdge.waypoints	com.topcoder.diagraminterchange.GraphEdge com.topcoder.diagraminterchange.Point
UML:Point	com.topcoder.diagraminterchange.Point
UML:BezierPoint UML:BezierPoint.controls	com.topcoder.diagraminterchange.BezierPoint com.topcoder.diagraminterchange.Point
UML:Dimension	com.topcoder.diagraminterchange.Dimension
UML:Polyline	com.topcoder.diagraminterchange.Polyline
UML:Ellipse UML:Ellipse.point	com.topcoder.diagraminterchange.Ellipse com.topcoder.diagraminterchange.POint
UML:TextElement	com.topcoder.diagraminterchange.TextElement
UML:Image	com.topcoder.diagraminterchange.Image
UML:CoreSemanticModelBridge	com.topcoder.diagraminterchange.CoreSemanticModelBridge
UML:Uml1SemanticModelBridge	com.topcoder.diagraminterchange.Uml1SemanticModelBridge
UML:SimpleSemanticModelElement	com.topcoder.diagraminterchange.SimpleSemanticModelElement
UML:GraphNode UML:GraphNode.size	com.topcoder.diagraminterchange.GraphNode com.topcoder.diagraminterchange.Dimension
UML:Diagram UML:Diagram.vieport UML:Diagram.diagramLink UML:Diagram.owner	com.topcoder.diagraminterchange.Diagram com.topcoder.diagraminterchange.Point com.topcoder.diagraminterchange.DiagramLink com.topcoder.diagraminterchange.SemanticModelBridge

4.3 Demo

4.3.1 Demonstrates the usage of XMIReader and DiagramInterchangeXMIHandler to parse a xmi file which has diagram interchange information

It is quite difficult to demo a handler, which is called internally by a parser.

Lets take this example as our example:



Here is the xmi for it:

```
<UML:Diagram xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f50' isVisible = 'true'
             name = 'Example Class Diagram' zoom = '1.0'>
      <UML:GraphElement.position>
        <XMI.field>0.0</XMI.field>
        <XMI.field>0.0</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>0.0</XMI.field>
        <XMI.field>0.0</XMI.field>
      </UML:GraphNode.size>
      <UML:Diagram.viewport>
        <XMI.field>0.0</XMI.field>
        <XMI.field>0.0</XMI.field>
      </UML:Diagram.viewport>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f4f' presentation = ''
                                        typeInfo = 'ClassDiagram'/>
      </UML:GraphElement.semanticModel>
      <UML:GraphElement.contained>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f48' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>80.0</XMI.field>
            <XMI.field>140.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>50.0</XMI.field>
            <XMI.field>80.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f47' presentation = ''>
              <UML:Uml1SemanticModelBridge.element>
                <UML:Actor xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f49'/>
              </UML:Uml1SemanticModelBridge.element>
            </UML:Uml1SemanticModelBridge>
          </UML:GraphElement.semanticModel>
          <UML:GraphElement.contained>
            <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f44' isVisible = 'true'>
              <UML:GraphElement.position>
            <XMI.field>12.1631</XMI.field>
            <XMI.field>85.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>25.6738</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f43'
                                            presentation = '' typeInfo = 'Name'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
    </UML:GraphNode>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f4d' isVisible = 'true'>
      <UML:GraphElement.position>
        <XMI.field>50.0</XMI.field>
        <XMI.field>91.0</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>131.9814</XMI.field>
        <XMI.field>19.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f4c'
                                        presentation = '' typeInfo = 'NameCompartment'/>
      </UML:GraphElement.semanticModel>
      <UML:GraphElement.contained>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f4b' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>4.0</XMI.field>
            <XMI.field>2.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>84.3638</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f4a'
                                            presentation = '' typeInfo = 'Name'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
    </UML:GraphNode>
  </UML:GraphElement.contained>
  <UML:Diagram.owner>
    <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f4e' presentation = ''>
      <UML:Uml1SemanticModelBridge.element>
        <UML:Model xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f51'/>
      </UML:Uml1SemanticModelBridge.element>
    </UML:Uml1SemanticModelBridge>
  </UML:Diagram.owner>
</UML:Diagram>
<UML:Diagram xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f42' isVisible = 'true'
             name = 'Use Case Diagram' zoom = '1.0'>
  <UML:GraphElement.position>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:GraphElement.position>
  <UML:GraphNode.size>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:GraphNode.size>
  <UML:Diagram.viewport>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:Diagram.viewport>
  <UML:GraphElement.semanticModel>
    <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f41' presentation = ''
                                    typeInfo = 'UseCaseDiagram'/>
  </UML:GraphElement.semanticModel>
```

```
<UML:GraphElement.contained>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef3' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>390.0</XMI.field>
      <XMI.field>250.0</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>141.0208</XMI.field>
      <XMI.field>50.0</XMI.field>
    </UML:GraphNode.size>
    <UML:DiagramElement.property>
      <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7edd' key = 'fill' value = '#ccccff'/>
    </UML:DiagramElement.property>
    <UML:GraphElement.semanticModel>
      <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef2' presentation = ''>
        <UML:Uml1SemanticModelBridge.element>
          <UML:UseCase xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7ef4'/>
        </UML:Uml1SemanticModelBridge.element>
      </UML:Uml1SemanticModelBridge>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef1' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>22.652</XMI.field>
          <XMI.field>13.5</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>95.7168</XMI.field>
          <XMI.field>19.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef0'
                                          presentation = '' typeInfo = 'NameCompartment'/>
        </UML:GraphElement.semanticModel>
        <UML:GraphElement.contained>
          <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7eef' isVisible = 'true'>
            <UML:GraphElement.position>
              <XMI.field>2.0</XMI.field>
              <XMI.field>2.0</XMI.field>
            </UML:GraphElement.position>
            <UML:GraphNode.size>
              <XMI.field>91.7168</XMI.field>
              <XMI.field>15.0</XMI.field>
            </UML:GraphNode.size>
            <UML:GraphElement.semanticModel>
              <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7eee'
                                              presentation = '' typeInfo = 'Name'/>
            </UML:GraphElement.semanticModel>
          </UML:GraphNode>
        </UML:GraphElement.contained>
      </UML:GraphNode>
    </UML:GraphElement.contained>
    <UML:GraphElement.anchorage>
      <UML:GraphConnector xmi.id = 'I1aa8eb7m10f6dbd4de0mm7eec'>
        <UML:GraphConnector.position>
          <XMI.field>4.2144</XMI.field>
          <XMI.field>16.4865</XMI.field>
        </UML:GraphConnector.position>
        <UML:GraphConnector.graphEdge>
          <UML:GraphEdge xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7eea'/>
        </UML:GraphConnector.graphEdge>
      </UML:GraphConnector>
    </UML:GraphElement.anchorage>
  </UML:GraphNode>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f2c' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>400.0</XMI.field>
      <XMI.field>110.0</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>124.5606</XMI.field>
      <XMI.field>50.0</XMI.field>
    </UML:GraphNode.size>
    <UML:DiagramElement.property>
      <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7edc' key = 'fill' value = '#ccccff'/>
    </UML:DiagramElement.property>
    <UML:GraphElement.semanticModel>
      <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f2b' presentation = ''>
        <UML:Uml1SemanticModelBridge.element>
          <UML:UseCase xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f2d'/>
        </UML:Uml1SemanticModelBridge.element>
      </UML:Uml1SemanticModelBridge>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f2a' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>20.2415</XMI.field>
          <XMI.field>13.5</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>84.0776</XMI.field>
          <XMI.field>19.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f29'
                                          presentation = '' typeInfo = 'NameCompartment'/>
        </UML:GraphElement.semanticModel>
        <UML:GraphElement.contained>
          <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f28' isVisible = 'true'>
            <UML:GraphElement.position>
              <XMI.field>2.0</XMI.field>
              <XMI.field>2.0</XMI.field>
            </UML:GraphElement.position>
            <UML:GraphNode.size>
              <XMI.field>80.0776</XMI.field>
              <XMI.field>15.0</XMI.field>
            </UML:GraphNode.size>
            <UML:GraphElement.semanticModel>
              <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f27'
                                              presentation = '' typeInfo = 'Name'/>
            </UML:GraphElement.semanticModel>
          </UML:GraphNode>
        </UML:GraphElement.contained>
      </UML:GraphNode>
    </UML:GraphElement.contained>
    <UML:GraphElement.anchorage>
      <UML:GraphConnector xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f02'>
        <UML:GraphConnector.position>
          <XMI.field>3.6609</XMI.field>
          <XMI.field>33.445</XMI.field>
        </UML:GraphConnector.position>
        <UML:GraphConnector.graphEdge>
          <UML:GraphEdge xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f00'/>
        </UML:GraphConnector.graphEdge>
      </UML:GraphConnector>
    </UML:GraphElement.anchorage>
  </UML:GraphNode>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f33' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>200.0</XMI.field>
      <XMI.field>180.0</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>146.186</XMI.field>
      <XMI.field>50.0</XMI.field>
    </UML:GraphNode.size>
    <UML:DiagramElement.property>
      <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ede' key = 'fill' value = '#ffcccc'/>
    </UML:DiagramElement.property>
    <UML:GraphElement.semanticModel>
      <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f32' presentation = ''>
        <UML:Uml1SemanticModelBridge.element>
        <UML:UseCase xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f34'/>
      </UML:Uml1SemanticModelBridge.element>
    </UML:Uml1SemanticModelBridge>
  </UML:GraphElement.semanticModel>
  <UML:GraphElement.contained>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f31' isVisible = 'true'>
      <UML:GraphElement.position>
        <XMI.field>23.4084</XMI.field>
        <XMI.field>13.5</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>99.3691</XMI.field>
        <XMI.field>19.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f30'
                                        presentation = '' typeInfo = 'NameCompartment'/>
      </UML:GraphElement.semanticModel>
      <UML:GraphElement.contained>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f2f' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>2.0</XMI.field>
            <XMI.field>2.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>95.3691</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f2e'
                                            presentation = '' typeInfo = 'Name'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
    </UML:GraphNode>
  </UML:GraphElement.contained>
  <UML:GraphElement.anchorage>
    <UML:GraphConnector xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f1e'>
      <UML:GraphConnector.position>
        <XMI.field>0.0</XMI.field>
        <XMI.field>25.0</XMI.field>
      </UML:GraphConnector.position>
      <UML:GraphConnector.graphEdge>
        <UML:GraphEdge xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f1d'/>
      </UML:GraphConnector.graphEdge>
    </UML:GraphConnector>
    <UML:GraphConnector xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f01'>
      <UML:GraphConnector.position>
        <XMI.field>142.9908</XMI.field>
        <XMI.field>17.6891</XMI.field>
      </UML:GraphConnector.position>
      <UML:GraphConnector.graphEdge>
        <UML:GraphEdge xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f00'/>
      </UML:GraphConnector.graphEdge>
    </UML:GraphConnector>
    <UML:GraphConnector xmi.id = 'I1aa8eb7m10f6dbd4de0mm7eeb'>
      <UML:GraphConnector.position>
        <XMI.field>142.0955</XMI.field>
        <XMI.field>33.246</XMI.field>
      </UML:GraphConnector.position>
      <UML:GraphConnector.graphEdge>
        <UML:GraphEdge xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7eea'/>
      </UML:GraphConnector.graphEdge>
    </UML:GraphConnector>
  </UML:GraphElement.anchorage>
</UML:GraphNode>
<UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f3a' isVisible = 'true'>
  <UML:GraphElement.position>
    <XMI.field>60.0</XMI.field>
    <XMI.field>150.0</XMI.field>
  </UML:GraphElement.position>
```

```
    <UML:GraphNode.size>
      <XMI.field>50.0</XMI.field>
      <XMI.field>80.0</XMI.field>
    </UML:GraphNode.size>
    <UML:GraphElement.semanticModel>
      <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f39' presentation = ''>
        <UML:Uml1SemanticModelBridge.element>
          <UML:Actor xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f3b'/>
        </UML:Uml1SemanticModelBridge.element>
      </UML:Uml1SemanticModelBridge>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f36' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>12.1631</XMI.field>
          <XMI.field>85.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>25.6738</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f35'
                                          presentation = '' typeInfo = 'Name'/>
        </UML:GraphElement.semanticModel>
      </UML:GraphNode>
    </UML:GraphElement.contained>
    <UML:GraphElement.anchorage>
      <UML:GraphConnector xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f1f'>
        <UML:GraphConnector.position>
          <XMI.field>50.0</XMI.field>
          <XMI.field>55.0</XMI.field>
        </UML:GraphConnector.position>
        <UML:GraphConnector.graphEdge>
          <UML:GraphEdge xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f1d'/>
        </UML:GraphConnector.graphEdge>
      </UML:GraphConnector>
    </UML:GraphElement.anchorage>
  </UML:GraphNode>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f3f' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>30.0</XMI.field>
      <XMI.field>61.0</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>106.9199</XMI.field>
      <XMI.field>19.0</XMI.field>
    </UML:GraphNode.size>
    <UML:GraphElement.semanticModel>
      <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f3e'
                                      presentation = '' typeInfo = 'NameCompartment'/>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f3d' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>4.0</XMI.field>
          <XMI.field>2.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>92.9199</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f3c'
                                          presentation = '' typeInfo = 'Name'/>
        </UML:GraphElement.semanticModel>
      </UML:GraphNode>
    </UML:GraphElement.contained>
  </UML:GraphNode>
  <UML:GraphEdge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f1d' isVisible = 'true'>
    <UML:GraphElement.position>
  <XMI.field>0.0</XMI.field>
  <XMI.field>0.0</XMI.field>
</UML:GraphElement.position>
<UML:GraphEdge.waypoints>
  <XMI.field>
    <XMI.field>110.0</XMI.field>
    <XMI.field>205.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>200.0</XMI.field>
    <XMI.field>205.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
</UML:GraphEdge.waypoints>
<UML:GraphElement.semanticModel>
  <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f1c' presentation = ''>
    <UML:Uml1SemanticModelBridge.element>
      <UML:Association xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f20'/>
    </UML:Uml1SemanticModelBridge.element>
  </UML:Uml1SemanticModelBridge>
</UML:GraphElement.semanticModel>
<UML:GraphElement.contained>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f1b' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>110.0</XMI.field>
      <XMI.field>205.0</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>0.0</XMI.field>
      <XMI.field>0.0</XMI.field>
    </UML:GraphNode.size>
    <UML:DiagramElement.property>
      <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f19'
                    key = 'gentleware-custom-width' value = '0.0'/>
      <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f18'
                    key = 'gentleware-custom-height' value = '0.0'/>
    </UML:DiagramElement.property>
    <UML:GraphElement.semanticModel>
      <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f1a'
                                   presentation = ''>
        <UML:Uml1SemanticModelBridge.element>
          <UML:AssociationEnd xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f26'/>
        </UML:Uml1SemanticModelBridge.element>
      </UML:Uml1SemanticModelBridge>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f17' isVisible = 'false'>
        <UML:GraphElement.position>
          <XMI.field>12.9904</XMI.field>
          <XMI.field>7.0814</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>24.4546</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f16'
                                        presentation = '' typeInfo = 'Name'/>
      </UML:GraphElement.semanticModel>
    </UML:GraphNode>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f15' isVisible = 'false'>
      <UML:GraphElement.position>
        <XMI.field>4.5666</XMI.field>
        <XMI.field>7.0814</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>6.4238</XMI.field>
        <XMI.field>15.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f14'
                                        presentation = '' typeInfo = 'Visibility'/>
      </UML:GraphElement.semanticModel>
    </UML:GraphNode>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f13' isVisible = 'false'>
      <UML:GraphElement.position>
        <XMI.field>12.9904</XMI.field>
        <XMI.field>-22.5</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>6.1177</XMI.field>
        <XMI.field>15.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f12'
                                        presentation = '' typeInfo = 'Multiplicity'/>
      </UML:GraphElement.semanticModel>
    </UML:GraphNode>
  </UML:GraphElement.contained>
</UML:GraphNode>
<UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f11' isVisible = 'true'>
  <UML:GraphElement.position>
    <XMI.field>200.0</XMI.field>
    <XMI.field>205.0</XMI.field>
  </UML:GraphElement.position>
  <UML:GraphNode.size>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:GraphNode.size>
  <UML:DiagramElement.property>
    <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f0f'
                  key = 'gentleware-custom-width' value = '0.0'/>
    <UML:Property xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f0e'
                  key = 'gentleware-custom-height' value = '0.0'/>
  </UML:DiagramElement.property>
  <UML:GraphElement.semanticModel>
    <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f10'
                                 presentation = ''>
      <UML:Uml1SemanticModelBridge.element>
        <UML:AssociationEnd xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f23'/>
      </UML:Uml1SemanticModelBridge.element>
    </UML:Uml1SemanticModelBridge>
  </UML:GraphElement.semanticModel>
  <UML:GraphElement.contained>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f0d' isVisible = 'false'>
      <UML:GraphElement.position>
        <XMI.field>-74.758</XMI.field>
        <XMI.field>-17.9454</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>61.7676</XMI.field>
        <XMI.field>15.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f0c'
                                        presentation = '' typeInfo = 'Name'/>
      </UML:GraphElement.semanticModel>
```

```
        </UML:GraphNode>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f0b' isVisible = 'false'>
          <UML:GraphElement.position>
            <XMI.field>-83.1818</XMI.field>
            <XMI.field>-17.9454</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>6.4238</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f0a'
                                            presentation = '' typeInfo = 'Visibility'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f09' isVisible = 'false'>
          <UML:GraphElement.position>
            <XMI.field>-19.1081</XMI.field>
            <XMI.field>7.5</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>6.1177</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f08'
                                            presentation = '' typeInfo = 'Multiplicity'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
    </UML:GraphNode>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f07' isVisible = 'false'>
      <UML:GraphElement.position>
        <XMI.field>99.0547</XMI.field>
        <XMI.field>215.0</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>111.8906</XMI.field>
        <XMI.field>15.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f06'
                                        presentation = '' typeInfo = 'DirectedName'/>
      </UML:GraphElement.semanticModel>
      <UML:GraphElement.contained>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f05' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>0.0</XMI.field>
            <XMI.field>0.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>111.8906</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f04'
                                            presentation = '' typeInfo = 'Name'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
    </UML:GraphNode>
  </UML:GraphElement.contained>
  <UML:GraphEdge.anchor>
    <UML:GraphConnector xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f1f'/>
    <UML:GraphConnector xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f1e'/>
  </UML:GraphEdge.anchor>
</UML:GraphEdge>
<UML:GraphEdge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f00' isVisible = 'true'>
  <UML:GraphElement.position>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
```

```
</UML:GraphElement.position>
<UML:GraphEdge.waypoints>
  <XMI.field>
    <XMI.field>403.6609</XMI.field>
    <XMI.field>143.445</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>342.9908</XMI.field>
    <XMI.field>197.6891</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
</UML:GraphEdge.waypoints>
<UML:GraphElement.semanticModel>
  <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7eff' presentation = ''>
    <UML:Uml1SemanticModelBridge.element>
      <UML:Extend xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f03'/>
    </UML:Uml1SemanticModelBridge.element>
  </UML:Uml1SemanticModelBridge>
</UML:GraphElement.semanticModel>
<UML:GraphElement.contained>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7efe' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>376.6584</XMI.field>
      <XMI.field>172.1172</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>64.7222</XMI.field>
      <XMI.field>15.0</XMI.field>
    </UML:GraphNode.size>
    <UML:GraphElement.semanticModel>
      <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7efd'
                                      presentation = ''
                                      typeInfo = 'StereotypeCompartment'/>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7efc' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>0.0</XMI.field>
          <XMI.field>0.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>12.8477</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7efb'
                                          presentation = ''
                                          typeInfo = 'StereotypeStart'/>
        </UML:GraphElement.semanticModel>
      </UML:GraphNode>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7efa' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>15.8477</XMI.field>
          <XMI.field>0.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
```

```
            <XMI.field>33.0269</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef9'
                                            presentation = ''
                                            typeInfo = 'KeywordMetaclass'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
        <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef8' isVisible = 'true'>
          <UML:GraphElement.position>
            <XMI.field>51.8745</XMI.field>
            <XMI.field>0.0</XMI.field>
          </UML:GraphElement.position>
          <UML:GraphNode.size>
            <XMI.field>12.8477</XMI.field>
            <XMI.field>15.0</XMI.field>
          </UML:GraphNode.size>
          <UML:GraphElement.semanticModel>
            <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef7'
                                            presentation = ''
                                            typeInfo = 'StereotypeEnd'/>
          </UML:GraphElement.semanticModel>
        </UML:GraphNode>
      </UML:GraphElement.contained>
    </UML:GraphNode>
    <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef6' isVisible = 'false'>
      <UML:GraphElement.position>
        <XMI.field>0.0</XMI.field>
        <XMI.field>0.0</XMI.field>
      </UML:GraphElement.position>
      <UML:GraphNode.size>
        <XMI.field>0.0</XMI.field>
        <XMI.field>0.0</XMI.field>
      </UML:GraphNode.size>
      <UML:GraphElement.semanticModel>
        <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ef5'
                                        presentation = '' typeInfo = 'Name'/>
      </UML:GraphElement.semanticModel>
    </UML:GraphNode>
  </UML:GraphElement.contained>
  <UML:GraphEdge.anchor>
    <UML:GraphConnector xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f02'/>
    <UML:GraphConnector xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f01'/>
  </UML:GraphEdge.anchor>
</UML:GraphEdge>
<UML:GraphEdge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7eea' isVisible = 'true'>
  <UML:GraphElement.position>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </UML:GraphElement.position>
  <UML:GraphEdge.waypoints>
    <XMI.field>
      <XMI.field>394.2144</XMI.field>
      <XMI.field>266.4865</XMI.field>
    </XMI.field>
    <XMI.field>
      <XMI.field>0.0</XMI.field>
      <XMI.field>0.0</XMI.field>
    </XMI.field>
    <XMI.field>
      <XMI.field>0.0</XMI.field>
      <XMI.field>0.0</XMI.field>
    </XMI.field>
    <XMI.field>
      <XMI.field>342.0955</XMI.field>
      <XMI.field>213.246</XMI.field>
    </XMI.field>
    <XMI.field>
      <XMI.field>0.0</XMI.field>
      <XMI.field>0.0</XMI.field>
  </XMI.field>
  <XMI.field>
    <XMI.field>0.0</XMI.field>
    <XMI.field>0.0</XMI.field>
  </XMI.field>
</UML:GraphEdge.waypoints>
<UML:GraphElement.semanticModel>
  <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee9' presentation = ''>
    <UML:Uml1SemanticModelBridge.element>
      <UML:Extend xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7eed'/>
    </UML:Uml1SemanticModelBridge.element>
  </UML:Uml1SemanticModelBridge>
</UML:GraphElement.semanticModel>
<UML:GraphElement.contained>
  <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee8' isVisible = 'true'>
    <UML:GraphElement.position>
      <XMI.field>299.86</XMI.field>
      <XMI.field>240.5572</XMI.field>
    </UML:GraphElement.position>
    <UML:GraphNode.size>
      <XMI.field>64.7222</XMI.field>
      <XMI.field>15.0</XMI.field>
    </UML:GraphNode.size>
    <UML:GraphElement.semanticModel>
      <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee7'
                                      presentation = ''
                                      typeInfo = 'StereotypeCompartment'/>
    </UML:GraphElement.semanticModel>
    <UML:GraphElement.contained>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee6' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>0.0</XMI.field>
          <XMI.field>0.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>12.8477</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee5'
                                          presentation = ''
                                          typeInfo = 'StereotypeStart'/>
        </UML:GraphElement.semanticModel>
      </UML:GraphNode>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee4' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>15.8477</XMI.field>
          <XMI.field>0.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>33.0269</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee3'
                                          presentation = ''
                                          typeInfo = 'KeywordMetaclass'/>
        </UML:GraphElement.semanticModel>
      </UML:GraphNode>
      <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee2' isVisible = 'true'>
        <UML:GraphElement.position>
          <XMI.field>51.8745</XMI.field>
          <XMI.field>0.0</XMI.field>
        </UML:GraphElement.position>
        <UML:GraphNode.size>
          <XMI.field>12.8477</XMI.field>
          <XMI.field>15.0</XMI.field>
        </UML:GraphNode.size>
        <UML:GraphElement.semanticModel>
          <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee1'
                                          presentation = ''
```

```
                                        typeInfo = 'StereotypeEnd'/>
                </UML:GraphElement.semanticModel>
              </UML:GraphNode>
            </UML:GraphElement.contained>
          </UML:GraphNode>
          <UML:GraphNode xmi.id = 'I1aa8eb7m10f6dbd4de0mm7ee0' isVisible = 'false'>
            <UML:GraphElement.position>
              <XMI.field>0.0</XMI.field>
              <XMI.field>0.0</XMI.field>
            </UML:GraphElement.position>
            <UML:GraphNode.size>
              <XMI.field>0.0</XMI.field>
              <XMI.field>0.0</XMI.field>
            </UML:GraphNode.size>
            <UML:GraphElement.semanticModel>
              <UML:SimpleSemanticModelElement xmi.id = 'I1aa8eb7m10f6dbd4de0mm7edf'
                                              presentation = ''
                                              typeInfo = 'Name'/>
            </UML:GraphElement.semanticModel>
          </UML:GraphNode>
        </UML:GraphElement.contained>
        <UML:GraphEdge.anchor>
          <UML:GraphConnector xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7eec'/>
          <UML:GraphConnector xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7eeb'/>
        </UML:GraphEdge.anchor>
      </UML:GraphEdge>
    </UML:GraphElement.contained>
    <UML:Diagram.owner>
      <UML:Uml1SemanticModelBridge xmi.id = 'I1aa8eb7m10f6dbd4de0mm7f40' presentation = ''>
        <UML:Uml1SemanticModelBridge.element>
          <UML:Model xmi.idref = 'I1aa8eb7m10f6dbd4de0mm7f51'/>
        </UML:Uml1SemanticModelBridge.element>
      </UML:Uml1SemanticModelBridge>
    </UML:Diagram.owner>
  </UML:Diagram>
```

To build the model to this we would run this:

```
        // get the associated uml model manager
        DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler)
reader.getHandler("UML:Diagram");
        UMLModelManager manager = handler.getUmlModelManager();

        manager.clearDiagrams();

        reader.parse(new File("test_files" + File.separator + "demo.xmi"));

        // now we can access the diagrams using xmi id values
        // class diagram - diagram definitions
        Diagram diagram = (Diagram) reader.getElement("I1aa8eb7m10f6dbd4de0mm7f50");
        System.out.println(diagram.getName());

        // use case diagram - diagram definitions
        diagram = (Diagram) reader.getElement("I1aa8eb7m10f6dbd4de0mm7f42");
        System.out.println(diagram.getName());

        // we can use the uml model manager to access the diagrams as well
        List<Diagram> diagrams = manager.getDiagrams();
        for (Diagram dg : diagrams) {
            System.out.println(dg.getName());
          }
```

4.3.2 It demonstrates the functionality of ModelElementFactory

```
        // Create a default instance
        ModelElementFactory modelElementFactory = new ModelElementFactory();

        // Create an instance with configuration data
        modelElementFactory = new ModelElementFactory(ModelElementFactory.class.getName());
```

```
// Adds a new mapping to the class
modelElementFactory.addMapping("UML:Diagram", "com.topcoder.diagraminterchange.Diagram");

// Creates an actual instance maps to the specific xml element
modelElementFactory.createModelElement("UML:Diagram");

// Gets the class name for the given xml element
modelElementFactory.getMapping("UML:Diagram");

// Return the complete mapping
modelElementFactory.getAllMappings();

// Remove the xml element mapping
modelElementFactory.removeMapping("UML:Diagram");
```

5. Future Enhancements

- None at this time