Annexes for Documentation of HL7 Internal Tooling Structure, Metadata, and Semantics

May, 2004

Annex A: HTML listing of repository table contents

Annex B: DTDs for current XML Interchange formats

Annex C: MIF Schemas

This Annex provides a textual listing of each of the XML schemas that make up the current definition of the Model Interchange Format (MIF). They represent the state of the MIF schemas as they existed the beginning of 2004. The only editing that has been performed on the schemas is to Format them for better display, and to eliminate large "comment" sections that were present for use by the original editor, but which did not contribute to an understanding of the material. Needless to say, these schemas are more useful when loaded into an appropriate XML tool, such as XML Spy, for review.

C.1 mifBase

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- mifBase -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3/mif"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
********************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
  This schema defines types that are re-used throughout other v3 schemas.  No specific elements
are defined by this schema

Modification History:
   2003-01-23: Added 'keyword' to header element
```

```
          - Added 'withdrawalDate' to ballotStatus
          - Added 'Draft', 'Non-standard, available for use' and 'Withdrawn' as ballot statuses
          - Updated 'usageNotes' element annotation to reflect that it also includes guidelines
on how *not* to use an element
          - Changed 'language' attribute name to be 'lang' (to be consistent w/ XML practices)
   2003-05-11: Added 'mapping' annotation in place of externalReference
          - made formalName required instead of optional
   2003-05-15: Added 'substantiveChangeIndicator' to historyItem
   2003-06-29: Added 'hashCode' to modelType
   2003-09-06: Completely revamped as part of UML alignment

Todo:
  - Enforce that 'underscores' only appear in class names if the class has descendants.
Future todo:
  - More restriction on definition of formal part of constraint (once we come up with what the
formal language will be)
  - More rigourous definition for static example content (i.e. a more formal definition of
example content, possibly using nested elements)
*****************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifExtendedMarkup.xsd"/>
   <xs:complexType name="ModelElement" abstract="true">
      <xs:annotation>
         <xs:documentation>The base type for all 'semantic' elements in the
MIF</xs:documentation>
         <xs:documentation>UML: ModelElement stereotype</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="historyItem" type="HistoryItem" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Defines the list of 'events' that have occurred against this
particular model element.</xs:documentation>
               <xs:documentation>UML: Object-typed Tag value on ModelElement
stereotype.</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate historyItem element">
                     <sch:rule context="mif:historyItem">
                        <sch:extends rule="HistoryItem"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="sortKey" type="BasicFormalName" use="optional">
         <xs:annotation>
            <xs:documentation>A name used in determining the sort order of the model element
within its siblings.</xs:documentation>
            <xs:documentation>Impl: This will usually be a sequential number, but could be
something else.</xs:documentation>
            <xs:documentation>UML: tag value within ModelElement</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:group name="Context">
      <xs:sequence>
         <xs:element name="context" type="ContextElement" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Identifies the context(s) associated with the
element.</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate context element">
                     <sch:rule context="mif:context">
                        <sch:extends rule="ContextElement"/>
                     </sch:rule>
                  </sch:pattern>
```

```
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:complexType name="ContextElement">
      <xs:annotation>
         <xs:documentation>UML: Part of a complex tag</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ContextKind type">
               <sch:rule abstract="true" id="ContextKind">
                  <sch:report test="count(preceding-
sibling::mif:*[name(.)=name(current())][@value=current()/@value])!=0">
                        ERROR: A given context may only be listed once within a particular
element.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="value" type="ContextKind" use="required">
         <xs:annotation>
            <xs:documentation>Indicates a single context value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:group name="BusinessName">
      <xs:sequence>
         <xs:element name="businessName" type="BusinessName" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>The business names associated with the
element.</xs:documentation>
               <xs:documentation>UML: Tag value inherited from ModelElement</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate businessName element">
                     <sch:rule context="mif:businessName">
                        <sch:extends rule="BusinessName"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:complexType name="BusinessName">
      <xs:annotation>
         <xs:documentation>A name for the artifact that will be meaningful to business or domain
experts.  One artifact may have different business name translations for each realm and/or
language.</xs:documentation>
         <xs:documentation>UML: tagged value on ModelElement</xs:documentation>
         <xs:documentation>DublinCore: Maps to 'Alternative'</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate BusinessName type">
               <sch:rule abstract="true" id="BusinessName">
                  <sch:report test="count(preceding-
sibling::mif:*[name(.)=name(current())][mif:context/@value=current()/mif:context/@value or
(not(mif:context) and not(current()/mif:context))])!=0">
                        ERROR: Only one business name may exist for a particular
context.</sch:report>
                  <sch:report test="count(preceding-
sibling::*[name(.)=name(current())][@name=current()/@name])!=0">
                        ERROR: Business names must be unique for a given element.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:sequence>
         <xs:group ref="Context"/>
      </xs:sequence>
```

```
      <xs:attribute name="name" type="ShortDescriptiveName" use="required">
         <xs:annotation>
            <xs:documentation>The assigned name.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="lang" type="xs:language" use="optional" default="EN">
         <xs:annotation>
            <xs:documentation>Indicates the language in which the business name is
expressed</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            <xs:documentation>DublinCore: Language</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="HistoryItem">
      <!-- Todo: Ensure that dateTime <= now -->
      <xs:annotation>
         <xs:documentation>Provides internal versioning information about an item.  Each
repetition indicates a separate version the artifact has gone through.</xs:documentation>
         <xs:documentation>UML: tagged object on ModelElement.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate HistoryItem type">
               <sch:rule abstract="true" id="HistoryItem">
                  <sch:report test="@id and preceding::mif:historyItem[@id=current/@id]">
                        ERROR: HistoryItem ids must be unique across the document</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="description" type="FullMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Describes the changes that occurred and why</xs:documentation>
               <xs:documentation>UML: part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="dateTime" type="DateOrTimestamp" use="required">
         <xs:annotation>
            <xs:documentation>Identifies the date and time on which the version was
created.</xs:documentation>
            <xs:documentation>UML: part of a complex tag value</xs:documentation>
            <xs:documentation>DublinCore: First occurrence corresponds to 'created', all others
correspond to 'Modified'</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="responsiblePersonName" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the name of the person responsible for creating the new
version.  (To see who created the initial version, look for the history item with the earliest
dateTime).</xs:documentation>
            <xs:documentation>UML: part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="id" type="Uuid" use="optional">
         <xs:annotation>
            <xs:documentation>A unique UUID assigned to this particular version of the
artifact.</xs:documentation>
            <xs:documentation>UML: part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="modifiedForPackageVersion" type="Version" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the version of the package this element is part of that
the change was first included in.</xs:documentation>
            <xs:documentation>UML: part of a complex tag value</xs:documentation>
         </xs:annotation>
```

```
      </xs:attribute>
      <xs:attribute name="isSubstantiveChange" type="xs:boolean" use="optional">
         <xs:annotation>
            <xs:documentation>If true, indicates that the changes made are considered
'substantive'.  Substantive changes have impact on balloting requirements.</xs:documentation>
            <xs:documentation>UML: part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="isBackwardCompatibleChange" type="xs:boolean" use="optional">
         <xs:annotation>
            <xs:documentation>If true, indicates that the changes made are considered backward
compatible.  I.e implementers (creators and readers) of the previous version should be able to
support the new version without adjusting their interfaces</xs:documentation>
            <xs:documentation>UML: part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="Relationship" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Relationship meta class</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ModelElement"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Dependency" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Dependency meta class</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Relationship"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Derivation" abstract="true">
      <xs:annotation>
         <xs:documentation>Used for all 'derivationSupplier' elements</xs:documentation>
         <xs:documentation>UML: Abstract stereotype on Dependency</xs:documentation>
         <xs:documentation>DubinCore: 'relation' to source model</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Derivation type">
               <sch:rule abstract="true" id="Derivation">
                  <sch:report test="@areAnnotationsReviewed='false' and @annotationsReviewedBy">
                     ERROR: AnnotationsReviewedBy may only be specified if the annotations have
been reviewed.</sch:report>
                  <sch:report test="@areAnnotationsReviewed='false' and
@relationship='unchanged'">
                     ERROR: AnnotationsReviewed must be 'true' for 'unchanged'
derivations.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Dependency">
            <xs:sequence>
               <xs:element name="reason" type="BasicMarkup" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Identifies the reason why the derived element has changed
from its source.</xs:documentation>
                     <xs:documentation>UML: Tag on Derivation stereotype</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="areAnnotationsReviewed" type="xs:boolean" use="optional"
default="true">
               <xs:annotation>
                  <xs:documentation>Identifies whether the analysis of the derivation has
included review of changed annotations.  This is false if (a) the annotations are identical; (b)
```

```
an automated process was able to take into account the changes in making the assessment of
derivation relationship; or (c) a human has manually reviewed the changes.</xs:documentation>
                  <xs:documentation>UML: Tag on Derivation stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="annotationsReviewedBy" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates the human responsible for the manual review of
annotations.</xs:documentation>
                  <xs:documentation>UML: Tag on Derivation stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="relationship" type="DerivationRelationshipKind" use="optional"
default="restriction">
               <xs:annotation>
                  <xs:documentation>Identifies the relationship between the current element and
the element it was derived from.</xs:documentation>
                  <xs:documentation>UML: Each code corresponds to a distinct non-abstract
stereotype under Derivation</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SemanticModelBridge" abstract="true">
      <xs:annotation>
         <xs:documentation>Forms the bridge between the UML semantic content and it's diagramatic
representation</xs:documentation>
         <xs:documentation>UML: Represents the UML1SemanticModelBridge class from Data
Interchange specification.  Have merged it with SemanticModelBridge for
convenience.</xs:documentation>
      </xs:annotation>
      <!--      <xs:attribute name="graphicRepresentation" type="PresentationCode"
use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the diagramming format used to display the
element.</xs:documentation>
         </xs:annotation>
      </xs:attribute>-->
   </xs:complexType>
   <xs:complexType name="DiagramSemanticModelBridge">
      <xs:annotation>
         <xs:documentation>The graphic representation for a UML concept that corresponds to a
complete diagram</xs:documentation>
         <xs:documentation>UML: Stereotype restricting SemanticModelBridge to a
Diagram</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="SemanticModelBridge">
            <xs:sequence>
               <xs:element name="graphElement" type="Diagram">
                  <xs:annotation>
                     <xs:documentation>The graphic node corresponding to the
element</xs:documentation>
                     <xs:documentation>UML: graphElement association to Diagram specialization of
ModelElement</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate graphElement element">
                           <sch:rule context="mif:graphElement[@name]">
                              <sch:extends rule="Diagram"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
```

```
   </xs:complexType>
   <xs:complexType name="NodeSemanticModelBridge">
      <xs:annotation>
         <xs:documentation>The graphic representation for a UML concept displayed as a single
node</xs:documentation>
         <xs:documentation>UML: Stereotype restricting SemanticModelBridge to a single
Node</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="SemanticModelBridge">
            <xs:sequence>
               <xs:element name="graphElement" type="ContainedGraphNode">
                  <xs:annotation>
                     <xs:documentation>The graphic node corresponding to the
element</xs:documentation>
                     <xs:documentation>UML: graphElement association to GraphNode specialization
of ModelElement</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate graphElement element">
                           <sch:rule context="mif:graphElement">
                              <sch:extends rule="ContainedGraphNode"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GraphEdgeSemanticModelBridge">
      <xs:annotation>
         <xs:documentation>The graphic representation for a UML concept displayed as a single
node</xs:documentation>
         <xs:documentation>UML: Stereotype restricting SemanticModelBridge to a single Edge with
two GraphConnectors</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="SemanticModelBridge">
            <xs:sequence>
               <xs:element name="graphElement" type="ContainedGraphConnectorWithEdge">
                  <xs:annotation>
                     <xs:documentation>The graphic node corresponding to the
element</xs:documentation>
                     <xs:documentation>UML: graphElement association to GraphConnectorWithEdge
specialization of ModelElement</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="NodeWithConnectionSemanticModelBridge">
      <xs:annotation>
         <xs:documentation>The graphic representation for a UML concept displayed as a single
node with a connection to its parent</xs:documentation>
         <xs:documentation>UML: Stereotype restricting SemanticModelBridge to a single Node with
a connection to its parent</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate graphElement element">
               <sch:rule context="mif:graphElement[mif:anchorage]">
                  <sch:extends rule="GraphNodeWithOptionalConnection"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="SemanticModelBridge">
```

```
            <xs:sequence>
               <xs:element name="graphElement" type="GraphNodeWithOptionalConnection">
                  <xs:annotation>
                     <xs:documentation>The graphic node corresponding to the
element</xs:documentation>
                     <xs:documentation>UML: graphElement association to GraphNode specialization
of ModelElement where GraphNode can have an edge</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DiagramElement" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Represents a stereotype on the DiagramElement class from Data
Interchange specification.</xs:documentation>
      </xs:annotation>
      <xs:attribute name="lastAdjustedDateTime" type="xs:dateTime" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the when the position of this shape was last
adjusted/confirmed.  (Used to identify shapes whose characteristics have been adjusted, but whose
positions have not yet been confirmed.</xs:documentation>
            <xs:documentation>UML: Tagged value on HL7GraphElement stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="shapeId" type="BasicId" use="optional">
         <xs:annotation>
            <xs:documentation>The internal identifier assigned to a particular shape.  Used for
cross-referencing.</xs:documentation>
            <xs:documentation>UML: Tag on DiagramElement stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="GraphElement" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Represents the GraphElement class from Data Interchange
specification.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DiagramElement">
            <xs:sequence>
               <xs:element name="position" type="Point">
                  <xs:annotation>
                     <xs:documentation>Identifies the top-left location of the diagram element
with respect to it's containing diagram.</xs:documentation>
                     <xs:documentation>UML: position attribute on GraphElement</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GraphNode">
      <xs:annotation>
         <xs:documentation>Corresponds to a single 'boxed' shape</xs:documentation>
         <xs:documentation>UML: A stereotype on the GraphNode class from Data Interchange
specification.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate GraphNode type">
               <sch:rule abstract="true" id="GraphNode">
                  <sch:report test="@textWrappingWidth and @textWrappingWidth &gt;
mif:size/@width">
                     WARNING: Text wrapping width is larger than width of element.</sch:report>
                  <sch:report test="@textWrappingWidth and @textWrappingWidth &gt;
mif:size/@width">
                     WARNING: Text wrapping width is larger than width of element.</sch:report>
               </sch:rule>
```

```
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="GraphElement">
            <xs:sequence>
               <xs:element name="size" type="Dimension">
                  <xs:annotation>
                     <xs:documentation>Identifies the soze of the node</xs:documentation>
                     <xs:documentation>UML: size attribute on GraphNode</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="textWrappingWidth" type="GraphicMeasurement" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates the width at which the text within the shape should
be wrapped</xs:documentation>
                  <xs:documentation>UML: tag value</xs:documentation>
               </xs:annotation>
               <!-- Todo: Ensure that this value is less than overall width.  Also ensure it only
appears on 'arrow' classes -->
            </xs:attribute>
            <xs:attribute name="isHeightAutoSize" type="xs:boolean" use="optional"
default="true">
               <xs:annotation>
                  <xs:documentation>Indicates whether this element should be automatically scaled
vertically to fit the contents of the element.</xs:documentation>
                  <xs:documentation>UML: Tagged value on GraphNode stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="isWidthAutoSize" type="xs:boolean" use="optional" default="true">
               <xs:annotation>
                  <xs:documentation>Indicates whether this element should be automatically scaled
horizontally to fit the contents of the element.</xs:documentation>
                  <xs:documentation>UML: Tagged value on GraphNode stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="nodeOrientation" type="NodeOrientation" use="optional">
               <xs:annotation>
                  <xs:documentation>Code specifying the orientation of graphic node elements,
including ChoiceBox alignment and the four, flip-orientations for a Role.</xs:documentation>
                  <xs:documentation>UML: Tagged value on GraphNode stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ContainedGraphNode">
      <xs:annotation>
         <xs:documentation>UML: A GraphNode that has a contained relationship to a
Diagram</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ContainedGraphNode type">
               <sch:rule abstract="true" id="ContainedGraphNode">
                  <sch:report
test="count(ancestor::mif:*/mif:graphElement/@name=current()/@containerDiagramName)!=1">
                     ERROR: A node that claims to be part of a diagram must be contained by an
element that has a diagram of that name.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="GraphNode">
            <xs:attribute name="containerDiagramName" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the diagram which the node is a part
of</xs:documentation>
```

```
                  <xs:documentation>UML: name property of the Diagram that is a container for
this DiagramElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="shapeTemplate" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>Deprecated: Function now handled by
SemanticModelBridge.graphicRepresentation</xs:documentation>
                  <xs:documentation>UML: deprecated</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GraphNodeWithOptionalConnection">
      <xs:annotation>
         <xs:documentation>UML: A GraphNode that has an optional connection to the shape of its
'owning' element</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate GraphNodeWithOptionalConnection type">
               <sch:rule abstract="true" id="GraphNodeWithOptionalConnection">
                  <sch:extends rule="ContainedGraphNode"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ContainedGraphNode">
            <xs:sequence>
               <xs:element name="anchorage" type="GraphConnectorWithEdge" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Shows the connection between the current element and it's
"owning" element.</xs:documentation>
                     <xs:documentation>UML: GraphNode anchorage association to
GraphConnector</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Diagram">
      <xs:annotation>
         <xs:documentation>A shape that represents the space taken up by a particular
diagram</xs:documentation>
         <xs:documentation>UML: Represents the Diagram class from Data Interchange
specification.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Diagram type">
               <sch:rule abstract="true" id="Diagram">
                  <sch:report test="count(preceding-
sibling::mif:*[name(.)=name(current())][@name=current()/@name])!=0">
                        ERROR: Only one diagram with a given name may exist within a single
element.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="GraphNode">
            <xs:attribute name="name" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The name for the diagram.</xs:documentation>
                  <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
```

```
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GraphEdge" abstract="true">
      <xs:annotation>
         <xs:documentation>Represents the line connecting two shapes</xs:documentation>
         <xs:documentation>UML: Represents the GraphEdge class from Data Interchange
specification.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DiagramElement">
            <xs:sequence>
               <xs:element name="waypoint" type="Point" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies points along the path of the graph edge between
two GraphNodes.</xs:documentation>
                     <xs:documentation>UML: waypoint attribute on GraphEdge</xs:documentation>
                     <xs:documentation>Impl: At present, these are not used in Visio diagrams,
but may be in UML</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GraphEdgeWithAnchor">
      <xs:annotation>
         <xs:documentation>Represents the line connecting two shapes</xs:documentation>
         <xs:documentation>UML: A GraphEdge class with an Anchor on its end.  (You have to have
an anchor on the end.  Just created a separate type so it would have a name.)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="GraphEdge">
            <xs:sequence>
               <xs:element name="anchor" type="GraphConnector">
                  <xs:annotation>
                     <xs:documentation>The point at which the edge connects to the other anchor
element's "owning" element.</xs:documentation>
                     <xs:documentation>UML: anchor association from GraphEdge to
GraphConnector</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GraphConnector">
      <xs:annotation>
         <xs:documentation>Represents the line connecting two shapes</xs:documentation>
         <xs:documentation>UML: Represents the GraphConnector class from Data Interchange
specification.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DiagramElement">
            <xs:sequence>
               <xs:element name="position" type="Point">
                  <xs:annotation>
                     <xs:documentation>Identifies a point of connection from a graph edge to a
graph node</xs:documentation>
                     <xs:documentation>UML: position attribute on
GraphConnector</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="connectToShapeId" type="BasicId" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates the shape</xs:documentation>
                  <xs:documentation>UML: Identifies the diagramElement to which the shape is
connected</xs:documentation>
```

```
            </xs:annotation>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="GraphConnectorWithEdge">
   <xs:annotation>
      <xs:documentation>Represents the line connecting two shapes</xs:documentation>
      <xs:documentation>UML: A GraphConnector class with an out-going line on its end.  (You
have to have an outgoing line.  Just created a separate type so it would have a
name.)</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:extension base="GraphConnector">
         <xs:sequence>
            <xs:element name="graphEdge" type="GraphEdgeWithAnchor">
               <xs:annotation>
                  <xs:documentation>The edge that connects the current element to it's
containing element.</xs:documentation>
                  <xs:documentation>UML: GraphConnector graphEdge association to
GraphEdge</xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="ContainedGraphConnectorWithEdge">
   <xs:annotation>
      <xs:documentation>UML: A GraphConnector class with an out-going line on its end
contained within a diagram.  (You have to have an outgoing line.  Just created a separate type so
it would have a name.)</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:extension base="GraphConnectorWithEdge">
         <xs:attribute name="containerDiagramName" type="BasicFormalName" use="required">
            <xs:annotation>
               <xs:documentation>The name of the diagram which the node is a part
of</xs:documentation>
               <xs:documentation>UML: name property of the Diagram that is a container for
this DiagramElement</xs:documentation>
            </xs:annotation>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="Point">
   <xs:annotation>
      <xs:documentation>Represents a single position using x and y
coordinates.</xs:documentation>
      <xs:documentation>UML: The Point datatype</xs:documentation>
   </xs:annotation>
   <xs:attribute name="x" type="GraphicMeasurement" use="required">
      <xs:annotation>
         <xs:documentation>Identifies a position along the horizontal axis in inches.
Positions are relative to other GraphElements, rather than to any particular location on a page.
(The page will consist of the bounding rectangle containing all shapes.)</xs:documentation>
         <xs:documentation>UML: (Diagram Interchange) Point.x</xs:documentation>
      </xs:annotation>
   </xs:attribute>
   <xs:attribute name="y" type="GraphicMeasurement" use="required">
      <xs:annotation>
         <xs:documentation>Identifies a position along the vertical axis in inches.  Positions
are relative to other GraphElements, rather than to any particular location on a page.  (The page
will consist of the bounding rectangle containing all shapes.)</xs:documentation>
         <xs:documentation>UML: (Diagram Interchange) Point.y</xs:documentation>
      </xs:annotation>
   </xs:attribute>
```

```
   </xs:complexType>
   <xs:complexType name="Dimension">
      <xs:annotation>
         <xs:documentation>Describes the size of an item.</xs:documentation>
         <xs:documentation>UML: The Dimension datatype</xs:documentation>
      </xs:annotation>
      <xs:attribute name="width" type="GraphicMeasurement" use="required">
         <xs:annotation>
            <xs:documentation>Identifies the horizontal extent of the rectangle bounding the
GraphElement in inches.</xs:documentation>
            <xs:documentation>UML: (Diagram Interchange) Dimension.width</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="height" type="GraphicMeasurement" use="required">
         <xs:annotation>
            <xs:documentation>Identifies the vertical extent of the rectangle bounding the
GraphElement in inches.</xs:documentation>
            <xs:documentation>UML: (Diagram Interchange) Dimension.height</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="Annotations" abstract="true">
      <xs:annotation>
         <xs:documentation>Descriptive information about the containing model
element.</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
model element.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:choice minOccurs="0">
            <xs:group ref="Definition"/>
            <xs:group ref="Description"/>
         </xs:choice>
         <xs:group ref="UsageNotes" minOccurs="0"/>
         <xs:group ref="Rationale" minOccurs="0"/>
         <xs:group ref="DesignComments" minOccurs="0"/>
         <xs:group ref="Mapping" minOccurs="0"/>
         <xs:group ref="Constraint" minOccurs="0"/>
         <xs:group ref="StaticExample" minOccurs="0"/>
         <xs:group ref="Walkthrough" minOccurs="0"/>
         <xs:group ref="BallotComment" minOccurs="0"/>
         <xs:group ref="OtherAnnotation" minOccurs="0"/>
         <xs:group ref="Appendix" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <xs:group name="Definition">
      <xs:sequence>
         <xs:element name="definition" type="ComplexAnnotation">
            <xs:annotation>
               <xs:documentation>An explanation of the meaning of the element.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="Description">
      <xs:sequence>
         <xs:element name="description" type="ComplexAnnotation">
            <xs:annotation>
               <xs:documentation>An explanation of the associated element.  This may contain
formatting markup.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="UsageNotes">
```

```
      <xs:sequence>
         <xs:element name="usageNotes" type="UsageNotes">
            <xs:annotation>
               <xs:documentation>Advice to designers on how to make use of an element and/or how
*not* to make use of an element.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="Rationale">
      <xs:sequence>
         <xs:element name="rationale" type="Rationale">
            <xs:annotation>
               <xs:documentation>An explanation of why the element is necessary or potentially
useful within this context.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="DesignComments">
      <xs:sequence>
         <xs:element name="designComments" type="DesignComments">
            <xs:annotation>
               <xs:documentation>Internal development notes about why particular design decisions
were made, outstanding issues and remaining work.  They may contain formatting markup.  Not
intended for external publication.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="Mapping">
      <xs:sequence>
         <xs:element name="mapping" type="Mapping" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>A reference to an external or internal artifact that has a
degree of similarity or equivalence with the current item.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="Constraint">
      <xs:sequence>
         <xs:element name="constraint" type="Constraint" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>A formal, testable limitation on the use, representation or
value associated with the current element.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="StaticExample">
      <xs:sequence>
         <xs:element name="staticExample" type="StaticExample" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>An example instance expressed in a particular
ITS.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="Walkthrough">
      <xs:sequence>
```

```
         <xs:element name="walkthrough" type="Walkthrough">
            <xs:annotation>
               <xs:documentation>An overview of the primary and most important contents of the
element.  Used to provide broad understanding of the content without detailed review.  It may
contain formatting markup.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
               <xs:documentation>DublinCore: Corresponds to 'abstract'</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="BallotComment">
      <xs:sequence>
         <xs:element name="ballotComment" type="BallotComment">
            <xs:annotation>
               <xs:documentation>Detailed comments on a particular aspect of the
ballot.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate ballotComment element">
                     <sch:rule context="mif:ballotComment">
                        <sch:extends rule="BallotComment"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="OtherAnnotation">
      <xs:sequence>
         <xs:element name="otherAnnotation" type="OtherAnnotation" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Additional content related to the element.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:group name="Appendix">
      <xs:sequence>
         <xs:element name="appendix" type="Appendix" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Documentation that supports or relates to the current
element.</xs:documentation>
               <xs:documentation>UML: Descendant Stereotype from Annotation</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:group>
   <xs:complexType name="ComplexAnnotation">
      <xs:annotation>
         <xs:documentation>An annotation that allows complex internal markup</xs:documentation>
         <xs:documentation>UML: Stereotype on Comment or Constraint</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ModelElement">
            <xs:sequence>
               <xs:element name="text" type="ComplexMarkupWithLanguage" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>The textual content of the annotation, with the associated
language indicated.</xs:documentation>
                     <xs:documentation>UML: Tag on the SimpleAnnotation
stereotype</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate text element">
                           <sch:rule context="mif:text">
                              <sch:extends rule="ComplexMarkupWithLanguage"/>
```

```
                        </sch:rule>
                     </sch:pattern>
                  </xs:appinfo>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="FreeFormAnnotation" abstract="true">
   <xs:annotation>
      <xs:documentation>An annotation that allows arbitrary internal markup</xs:documentation>
      <xs:documentation>UML: Stereotype on Comment or Constraint</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:extension base="ModelElement">
         <xs:sequence>
            <xs:element name="text" type="FreeFormMarkupWithLanguage" minOccurs="0"
maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>The free-form content of the annotation, with the
associated language indicated.</xs:documentation>
                  <xs:documentation>UML: Tag on the SimpleAnnotation
stereotype</xs:documentation>
                  <xs:appinfo>
                     <sch:pattern name="Validate text element">
                        <sch:rule context="mif:text">
                           <sch:extends rule="FreeFormMarkupWithLanguage"/>
                        </sch:rule>
                     </sch:pattern>
                  </xs:appinfo>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="ComplexMarkupWithLanguage" mixed="true">
   <xs:annotation>
      <xs:documentation>Allows complex markup to identify the language in which it is
expressed and when it was last translated</xs:documentation>
      <xs:documentation>UML: Handled as 'mixed text' stored as a string, but with parallel
tags identifying langage</xs:documentation>
      <xs:appinfo>
         <sch:pattern name="Validate ComplexMarkupWithLanguage type">
            <sch:rule abstract="true" id="ComplexMarkupWithLanguage">
               <sch:report test="count(preceding-
sibling::mif:*[name(.)=name(current())][@lang=current()/@lang or
                                    ((@lang='EN' or not(@lang)) and (current()/@lang='EN' or
not(current()/@lang)))])!=0">
                  ERROR: Each repetition of complex markup with language must be a different
langage (no language is equivalent to 'EN').</sch:report>
               </sch:rule>
            </sch:pattern>
      </xs:appinfo>
   </xs:annotation>
   <xs:complexContent mixed="true">
      <xs:extension base="FullMarkup">
         <xs:attribute name="lang" type="xs:language" use="optional" default="EN">
            <xs:annotation>
               <xs:documentation>Indicates the language in which the marked up text is
expressed</xs:documentation>
               <xs:documentation>UML: part of a complex stereotype tag</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:attribute name="lastTranslated" type="xs:dateTime" use="optional">
            <xs:annotation>
```

```
                  <xs:documentation>Indicates when this languages rendition was last translated
from the original.</xs:documentation>
                  <xs:documentation>UML: part of a complex stereotype tag</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="FreeFormMarkupWithLanguage" mixed="true">
      <xs:annotation>
         <xs:documentation>Allows unconstrained markup to identify the language in which it is
expressed and when it was last translated</xs:documentation>
         <xs:documentation>UML: Handled as 'mixed text' stored as a string, but with parallel
tags identifying langage</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate FreeFormMarkupWithLanguage type">
               <sch:rule abstract="true" id="FreeFormMarkupWithLanguage">
                  <sch:report test="count(preceding-
sibling::mif:*[name(.)=name(current())][@lang=current()/@lang or
                                          ((@lang='EN' or not(@lang)) and (current()/@lang='EN' or
not(current()/@lang)))])!=0">
                     ERROR: Each repetition of complex markup with language must be a different
langage (no language is equivalent to 'EN').</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent mixed="true">
         <xs:extension base="VariousMixedContent">
            <xs:attribute name="lang" type="xs:language" use="optional" default="EN">
               <xs:annotation>
                  <xs:documentation>Indicates the language in which the marked up text is
expressed</xs:documentation>
                  <xs:documentation>UML: part of a complex stereotype tag</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="lastTranslated" type="xs:dateTime" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates when this languages rendition was last translated
from the original.</xs:documentation>
                  <xs:documentation>UML: part of a complex stereotype tag</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="UsageNotes">
      <xs:annotation>
         <xs:documentation>UML: UsageNotes stereotype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:group ref="Context"/>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Rationale">
```

```
      <xs:annotation>
         <xs:documentation>UML: Rationale stereotype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DesignComments">
      <xs:annotation>
         <xs:documentation>UML: DesignComments stereotype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Mapping">
      <xs:annotation>
         <xs:documentation>UML: Mapping stereotype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="derivationSupplier" type="AnnotationDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the constraint from which the current
constraint was derived.</xs:documentation>
                     <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="sourceArtifact" type="PackageOrArtifactRef" minOccurs="0">
                  <xs:annotation>
```

```
                        <xs:documentation>The model or other identifier associated with the item
being mapped to</xs:documentation>
                        <xs:documentation>UML: sourceId tag on Mapping stereotype</xs:documentation>
                        <xs:appinfo>
                           <sch:pattern name="Validate sourceArtifact element">
                              <sch:rule context="mif:sourceArtifact">
                                 <sch:extends rule="PackageOrArtifactRef"/>
                              </sch:rule>
                           </sch:pattern>
                        </xs:appinfo>
                     </xs:annotation>
                  </xs:element>
               </xs:sequence>
               <xs:attribute name="name" type="BasicFormalName" use="optional">
                  <xs:annotation>
                     <xs:documentation>The name of the constraint</xs:documentation>
                     <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
                  </xs:annotation>
               </xs:attribute>
               <xs:attribute name="sourceName" type="MappingSourceKind" use="required">
                  <xs:annotation>
                     <xs:documentation>The name of the standard from which the reference is
taken.</xs:documentation>
                     <xs:documentation>UML: sourceName tag on Mapping stereotype</xs:documentation>
                  </xs:annotation>
               </xs:attribute>
               <xs:attribute name="sourceVersion" type="ShortDescriptiveName" use="optional">
                  <xs:annotation>
                     <xs:documentation>The version number or label associated with the item being
mapped to</xs:documentation>
                     <xs:documentation>UML: sourceVersion tag on Mapping
stereotype</xs:documentation>
                  </xs:annotation>
               </xs:attribute>
               <xs:attribute name="strength" type="MapRelationshipKind" use="optional">
                  <xs:annotation>
                     <xs:documentation>Identifies the quality of the mapping.</xs:documentation>
                     <xs:documentation>UML: strength tag on Mapping stereotype</xs:documentation>
                  </xs:annotation>
               </xs:attribute>
            </xs:extension>
         </xs:complexContent>
         <!-- We may want to use a different approach for links between analysis and other elements
-->
      </xs:complexType>
      <xs:complexType name="Constraint">
         <xs:annotation>
            <xs:documentation>UML: Constraint stereotype</xs:documentation>
         </xs:annotation>
         <xs:complexContent>
            <xs:extension base="ComplexAnnotation">
               <xs:sequence>
                  <xs:group ref="BusinessName"/>
                  <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                     <xs:annotation>
                        <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                        <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                     </xs:annotation>
                  </xs:element>
                  <xs:element name="derivationSupplier" type="AnnotationDerivation" minOccurs="0"
maxOccurs="unbounded">
                     <xs:annotation>
                        <xs:documentation>Identifies the constraint from which the current
constraint was derived.</xs:documentation>
```

```
                        <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="body" type="VariousMixedContent" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>The formal OCL expression of the
constraint</xs:documentation>
                        <xs:documentation>UML: body attribute on Constraint</xs:documentation>
                    </xs:annotation>
                    <!-- Todo: See is we can restrict the contents of this any further -->
                </xs:element>
                <xs:element name="alternateFormalExpression" type="FormalExpression" minOccurs="0"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Alternate (non-OCL) expression of the constraint.  Should
be maintained at the same time as the OCL expression</xs:documentation>
                        <xs:documentation>UML: tag on Constraint stereotype</xs:documentation>
                    </xs:annotation>
                    <!-- Todo: See is we can restrict the contents of this any further -->
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="BasicFormalName" use="optional">
                <xs:annotation>
                    <xs:documentation>The name of the constraint</xs:documentation>
                    <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="FormalExpression" mixed="true">
    <xs:annotation>
        <xs:documentation>A constraint expressed in a particular language</xs:documentation>
        <xs:documentation>UML: Type used by a complex stereotype</xs:documentation>
    </xs:annotation>
    <xs:complexContent mixed="true">
        <xs:extension base="VariousMixedContent">
            <xs:attribute name="encoding" type="ExpressionLanguageKind" use="required">
                <xs:annotation>
                    <xs:documentation>Identifies the language in which the expression is
encoded</xs:documentation>
                    <xs:documentation>UML: Type used by a complex stereotype</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="StaticExample">
    <xs:annotation>
        <xs:documentation>UML: StaticExample stereotype</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="FreeFormAnnotation">
            <xs:sequence>
                <xs:group ref="Context"/>
                <xs:group ref="BusinessName"/>
                <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                        <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                    </xs:annotation>
                </xs:element>
```

```
               <xs:element name="derivationSupplier" type="AnnotationDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the constraint from which the current
constraint was derived.</xs:documentation>
                     <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>The name of the constraint</xs:documentation>
                  <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="itsName" type="ITSKind" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the its in which the example is
expressed.</xs:documentation>
                  <xs:documentation>UML: itsName tag on StaticExample
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Walkthrough">
      <xs:annotation>
         <xs:documentation>UML: Walkthrough stereotype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:group ref="Context"/>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="BallotComment">
      <xs:annotation>
         <xs:documentation>UML: BallotResponse stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate BallotComment type">
               <sch:rule abstract="true" id="BallotComment">
                  <sch:report
test="not(ancestor::mif:*/mif:ballotInfo/mif:ballotResponse/@submissionId=current()/@submissionId
)">
                                 ERROR: Cannot have a ballot comment that is not part of an
identified ballot response associated with the package.</sch:report>
                  <sch:report
test="preceding::mif:*[name(.)=name(current())][@submissionId=current()/@submissionId][@name=curr
ent()/@name]">
                                 ERROR: Cannot have multiple ballot comments with the same name
within the same submission.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
```

```
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:element name="existingContent" type="FullMarkup" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Indicates the current content of concern in the
ballot.</xs:documentation>
                     <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="suggestedReplacement" type="FullMarkup" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Indicates the content that the balloter would prefer to
see used in place of the existing content.</xs:documentation>
                     <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="resolution" type="BallotCommentResolution" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates what step(s) the ballot group has taken in
dealing with the ballot</xs:documentation>
                     <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="submissionId" type="BasicId" use="required">
               <xs:annotation>
                  <xs:documentation>A reference to the BallotResponse defined in the BallotInfo
of the containing package.</xs:documentation>
                  <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="name" type="SmallNonNegativeInteger" use="required">
               <xs:annotation>
                  <xs:documentation>A unique identifier for the comment, within the context of a
particular submission.</xs:documentation>
                  <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="location" type="ShortDescriptiveName" use="optional">
               <xs:annotation>
                  <xs:documentation>Identifies the quality of the mapping.</xs:documentation>
                  <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="commentType" type="VoteKind" use="required">
               <xs:annotation>
                  <xs:documentation>Indicates the value of the vote</xs:documentation>
                  <xs:documentation>UML: tag on BallotComment stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="BallotCommentResolution">
      <xs:annotation>
         <xs:documentation>Defines the information needed when recording ballot comment
resolutions</xs:documentation>
         <xs:documentation>UML: part of complex tag object</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="resolutionComments" type="BasicMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Comments created by the group reviewing the ballot
comment</xs:documentation>
               <xs:documentation>UML: part of complex tag object</xs:documentation>
```

```
            </xs:annotation>
         </xs:element>
         <xs:element name="vote" type="GroupVote">
            <xs:annotation>
               <xs:documentation>Indicates the vote results of the resolution</xs:documentation>
               <xs:documentation>UML: part of complex tag object</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="resolution" type="VoteResolutionKind" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the decision the ballot group has come to in evaluating a
ballot comment</xs:documentation>
            <xs:documentation>UML: part of complex tag object</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="resolutionDate" type="xs:date" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the decision the ballot group has come to in evaluating a
ballot comment</xs:documentation>
            <xs:documentation>UML: part of complex tag object</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="GroupVote">
      <xs:annotation>
         <xs:documentation>Defines the information needed to record a group
vote</xs:documentation>
         <xs:documentation>UML: part of complex tag object</xs:documentation>
      </xs:annotation>
      <xs:attribute name="motionBy" type="ShortDescriptiveName" use="required">
         <xs:annotation>
            <xs:documentation>Name of the person who made the motion being
voted</xs:documentation>
            <xs:documentation>UML: used by a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="secondedBy" type="ShortDescriptiveName" use="required">
         <xs:annotation>
            <xs:documentation>Name of the person who seconded the motion being
voted</xs:documentation>
            <xs:documentation>UML: used by a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="inFavour" type="SmallNonNegativeInteger" use="required">
         <xs:annotation>
            <xs:documentation>The number of those present who voted in favour of the
motion</xs:documentation>
            <xs:documentation>UML: used by a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="opposed" type="SmallNonNegativeInteger" use="required">
         <xs:annotation>
            <xs:documentation>The number of those present who voted against the
motion</xs:documentation>
            <xs:documentation>UML: used by a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="abstain" type="SmallNonNegativeInteger" use="required">
         <xs:annotation>
            <xs:documentation>The number of those present who abstained from the
vote</xs:documentation>
            <xs:documentation>UML: used by a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="OtherAnnotation">
      <xs:annotation>
```

```
         <xs:documentation>UML: OtherAnnotation stereotype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:group ref="Context"/>
               <xs:group ref="BusinessName"/>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates a graphic representation associated with the
constraint on a particular diagram</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="derivationSupplier" type="AnnotationDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the constraint from which the current
constraint was derived.</xs:documentation>
                     <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>The name of the constraint</xs:documentation>
                  <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="type" type="ShortDescriptiveName" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies what kind of annotation is being used.  All new
annotation types must be approved by the publications team.</xs:documentation>
                  <xs:documentation>UML: type tag on OtherAnnotation
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Appendix">
      <xs:annotation>
         <xs:documentation>UML: Appendix</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ComplexAnnotation">
            <xs:sequence>
               <xs:group ref="Context"/>
               <xs:group ref="BusinessName"/>
               <xs:element name="derivationSupplier" type="AnnotationDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the constraint from which the current
constraint was derived.</xs:documentation>
                     <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>The short, referencable name of the
appendix</xs:documentation>
```

```
                  <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="title" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The descriptive name of the appendix</xs:documentation>
                  <xs:documentation>UML: tag on Appendix stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AnnotationDerivation">
      <xs:annotation>
         <xs:documentation>UML: a Derivation that points to another annotation.  (To make XML
handle the graph-like nature of UML)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Derivation">
            <xs:attribute name="targetAnnotationName" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>The name of the annotation from which this annotation was
derived.</xs:documentation>
                  <xs:documentation>UML: The name of the Annotation element (inherited from
ModelElement) that is being derived from</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="PackageBase" abstract="true">
      <xs:complexContent>
         <xs:extension base="ModelElement">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
            </xs:sequence>
            <xs:attribute name="name" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The unique name or id of the package (within the context of
its parents)</xs:documentation>
                  <xs:documentation>UML: name attribute inherited from
ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Package" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Package stereotype</xs:documentation>
         <xs:documentation>
             DublinCore: The 'specialization' of Package is used to identify 'type'
                         The 'format' is always "HL7 MIF"
                         The 'identifier' is the name attribute defined in specializations
                         HL7 artifacts tend to be 'original' and therefore have no 'source'
          </xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Package type">
               <sch:rule abstract="true" id="Package">
                  <sch:report test="count(ancestor-or-self::mif:*/mif:header)=0">
                     WARNING: This element or one of it's ancestors must have a header
defined.</sch:report>
                  <sch:report test="ancestor::mif:*/@name and mif:packageLocation">
                     ERROR: Can't define a package location for an element that is located
inside a package.</sch:report>
               </sch:rule>
            </sch:pattern>
```

```
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageBase">
            <xs:sequence>
               <xs:group ref="Context"/>
               <xs:element name="packageLocation" type="PackageRef" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Identifies where (within the 'repository' package
hierarchy) this package resides</xs:documentation>
                     <xs:documentation>UML: Identifies the namespace the package is part
of</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate packageLocation element">
                           <sch:rule context="mif:packageLocation">
                              <sch:extends rule="PackageRef"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
               <xs:element name="header" type="Header" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>General metadata information about the
package</xs:documentation>
                     <xs:documentation>UML: complex tag value on Package
stereotype</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="replaces" type="PackageRef" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates the package and versions (or packages) that are
intended to be superceded by this package.</xs:documentation>
                     <xs:documentation>UML: tag on Package stereotype</xs:documentation>
                     <xs:documentation>DublinCore: Corresponds to 'replaces'</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate replaces element">
                           <sch:rule context="mif:replaces">
                              <sch:extends rule="PackageRef"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="hashCode" type="HashCode" use="optional">
               <xs:annotation>
                  <xs:documentation>A base64 encoded 160 bit SHA-1 hashcode.  The hashcode will
be calculated upon the full canonicalized content (including comments) of the package,
transformed to exclude the hashcode attribute itself.  Refer to http://www.w3.org/TR/2002/REC-
xmldsig-core-20020212 for details.</xs:documentation>
                  <xs:documentation>UML: hashCode tag on Package stereotype</xs:documentation>
               </xs:annotation>
               <!-- Todo: Make required as soon as existing code supports this -->
            </xs:attribute>
            <xs:attribute name="title" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>The descriptive name for the package in circumstances where
the 'name' is more of an identifier.</xs:documentation>
                  <xs:documentation>UML: title tag on Package stereotype</xs:documentation>
                  <xs:documentation>DublinCore: Equivalent to 'title'</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="packageKind" type="PackageKind" use="required">
               <xs:annotation>
                  <xs:documentation>The 'level' or variety of package being
represented</xs:documentation>
```

```
                  <xs:documentation>UML: 'packageKind' tag on Package
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
      <!-- Todo - enforce that packageKind agrees with packageLocation -->
   </xs:complexType>
   <xs:complexType name="PackageDerivation">
      <xs:annotation>
         <xs:documentation>UML: a Derivation that points to another package.  (To make XML handle
the graph-like nature of UML)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Derivation">
            <xs:sequence>
               <xs:element name="targetPackage" type="PackageRef">
                  <xs:annotation>
                     <xs:documentation>The package from which the current package is
derived.</xs:documentation>
                     <xs:documentation>UML: The full path of Package names of the Package
(inherited from ModelElement) that is being derived from</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate targetPackage element">
                           <sch:rule context="mif:targetPackage">
                              <sch:extends rule="PackageRef"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Header">
      <xs:annotation>
         <xs:documentation>Defines common content for all major artifact types.  It should always
be present in the root element of a document.</xs:documentation>
         <xs:documentation>UML: tag object on Package stereotype</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="renderingInformation" type="RenderingInformation" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Provides information about the creation of this particular (XML,
XMI or other) representation of the package</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="legalese" type="Legalese" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Any legal restrictions or rights associated with the
package</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:documentation>DublinCore: corresponds to 'rights'</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate legalese element">
                     <sch:rule context="mif:legalese">
                        <sch:extends rule="Legalese"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="responsibleGroup" type="ResponsibleGroup" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
```

```
               <xs:documentation>Identifies organizations responsible for the content of the
package</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:documentation>DublinCore: Equivalent to 'contributor'.  (HL7 doesn't really
identify 'authors' because content tends to be mutually created.)</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate responsibleGroup element">
                     <sch:rule context="mif:responsibleGroup">
                        <sch:extends rule="ResponsibleGroup"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="contributor" type="Contributor" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Identifies individuals who contributed to the creation or
maintenance of the package</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:documentation>DublinCore: Equivalent to 'contributor'.  (HL7 doesn't really
identify 'authors' because content tends to be mutually created.)</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="subject" type="ShortDescriptiveName" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>A word to expose a key concept that might be used in searching
for the identified package.</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:documentation>DublinCore: Equivalent to 'subject'</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="ballotInfo" type="BallotInfo" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Describes the ballot status and information associated with the
package</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate ballotInfo element">
                     <sch:rule context="mif:ballotInfo">
                        <sch:extends rule="BallotInfo"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
            <!-- TODO: This should be changed to minOccurs="1" when all of our exports support
this information -->
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="RenderingInformation">
      <xs:annotation>
         <xs:documentation>Information about the rendering of this XML.  The textual content (if
any) contains additional information about the rendering process.</xs:documentation>
         <xs:documentation>UML: Part of a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="renderingNotes" type="FullMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>General comments about the rendering.  (May include tooling
'plugs' :>)</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="schemaVersion" type="Version" use="required" fixed="1">
         <xs:annotation>
```

```
            <xs:documentation>Identifies what schema version the content complies
with.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="renderingTime" type="xs:dateTime" use="optional">
         <xs:annotation>
            <xs:documentation>The date this instance was created.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="application" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the application responsible for generating the
content.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="Legalese">
      <xs:annotation>
         <xs:documentation>This element defines the legal aspects associated with the containing
element and it's descendants.</xs:documentation>
         <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Legalese type">
               <sch:rule abstract="true" id="Legalese">
                  <sch:report test="(@copyrightOwner and not(@copyrightYears)) or
(not(@copyrightOwner) and @copyrightYears)">
                                          ERROR: Either both copyrightYears and copyrightOwner
must be specified or neither should be specified.</sch:report>
                  <sch:report test="@copyrightNotation and not(@copyrightYears)">
                                          ERROR: CopyrightNotation may only be specified when
copyrightYears is present.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="notation" type="FullMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Additional details about the copyright assertion.  E.g. "All
rights reserved"</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="disclaimer" type="FullMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Provides any legal disclaimers associated with the use of the
information in the content.</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="copyrightOwner" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>The name of the individual or organization asserting
copyright.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="copyrightYears" type="Years" use="optional">
         <xs:annotation>
            <xs:documentation>The year (or years) in which copyright is
asserted.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            <xs:documentation>DublinCore: DateCopyrighted</xs:documentation>
         </xs:annotation>
```

```
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="ResponsibleGroup">
      <xs:annotation>
         <xs:documentation>Identifies the entity(s) responsible for or who have approved the
content.</xs:documentation>
         <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ResponsibleGroup type">
               <sch:rule abstract="true" id="ResponsibleGroup">
                  <sch:report test="not(@groupId or @groupName or @organizationName)">
                        ERROR: At least one of groupId, groupName or organizationName is
required.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="groupId" type="BasicId" use="optional">
         <xs:annotation>
            <xs:documentation>A reference to the 'formal' identifier assigned to the group (if
any)</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="groupName" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the technical committee, special interest or other group
responsible for the content.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="organizationName" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the organization the group responsible is a part
of.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="Contributor">
      <xs:annotation>
         <xs:documentation>Identifies one of the individuals involved in producing the
content.</xs:documentation>
         <xs:documentation>UML: Part of a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="role" type="RoleKind">
            <xs:annotation>
               <xs:documentation>Identifies the role the individual played in the development of
the content.  Enumeration values are suggestions only.  If a person played multiple rules, choose
the most significant role.</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="name" type="KeyedName">
            <xs:annotation>
               <xs:documentation>The name of the individual</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="affiliation" type="ShortDescriptiveName" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Identifies the organization or group the individual is
associated with.  (E.g. Their employer or sponsor.)</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="email" type="EMail" minOccurs="0">
```

```
            <xs:annotation>
               <xs:documentation>The e-mail address at which the individiual can be
reached.</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="notes" type="LongDescriptiveName" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Any additional information relevant to the individual's part in
the content.  (Also used for contact instructions.)</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="KeyedName">
      <xs:annotation>
         <xs:documentation>Identifies the name of the individual as it should appear when
published.</xs:documentation>
         <xs:documentation>UML: Part of a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:attribute name="name" type="ShortDescriptiveName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the person</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="key" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>Used in sorting multiple authors.  (E.g. Last name, first
name)</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="BallotInfo">
      <xs:annotation>
         <xs:documentation>Indicates the ballot status of the artifact(s).</xs:documentation>
         <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate BallotInfo type">
               <sch:rule abstract="true" id="BallotInfo">
                  <sch:report
test="count(parent::mif:*/ancestor::mif:*/mif:header/mif:*[name(.)=name(current())])!=0">
                        ERROR: Cannot have a ballotInfo when a parent package has a
ballotInfo.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="ballotSubmission" type="BallotSubmission" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>A response submitted as part of the ballot</xs:documentation>
               <xs:documentation>UML: Part of a complex tag value</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate ballotSubmission element">
                     <sch:rule context="mif:ballotSubmission">
                        <sch:extends rule="BallotSubmission"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="ballotStatus" type="BallotStatusKind" use="required">
         <xs:annotation>
```

```
            <xs:documentation>Identifies how far along in the ballot process this artifact has
progressed.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="ballotOccurrence" type="SmallPositiveInteger" use="optional"
default="1">
         <xs:annotation>
            <xs:documentation>Identifies the repetition number at the identified ballot level.
(E.g. second time at committee level ballot.)</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="ballotDate" type="xs:date" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the date of the intended ballot (for material with a
ballotStatus identifying itself as a ballot), or the date on which the material was successfully
balloted (for material with a ballotStatus identifying itself as a Standard).</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            <xs:documentation>DublinCore: for a passed ballot, maps to
'dateAccepted'</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="withdrawalDate" type="xs:date" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the the item was or is planned to be officially withdrawn
as a standard.</xs:documentation>
            <xs:documentation>UML: Part of a complex tag value</xs:documentation>
            <xs:documentation>DublinCore: indicates the end date of 'valid'</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <!-- Todo: Enforce that only one ballot status within a hierarchy.  If an element has a
ballotStatus, it's ancestors and descendants must not.  Item with a ballot status is the 'level'
at which things are balloted.  I.e. it passes or fails as a whole. -->
   </xs:complexType>
   <xs:complexType name="BallotSubmission">
      <xs:annotation>
         <xs:documentation>Defines the content necessary when submitting a
ballot</xs:documentation>
         <xs:documentation>UML: Type used in a complex tag value</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate BallotSubmission type">
               <sch:rule abstract="true" id="BallotSubmission">
                  <sch:report test="contains(@vote, 'Negative') and
not(parent::mif:*/parent::mif:header//mif:ballotComment[contains(@vote, 'Negative')] or
mif:voterComments)">
                        WARNING: You must have general comments or a negative ballotComment to
have a negative vote.</sch:report>
                  <sch:report test="@vote!='Negative' and
parent::mif:*/parent::header//mif:ballotComment[contains(@vote, 'Negative')]">
                        WARNING: Vote must be 'Negative' if there are any negative
comments.</sch:report>
                  <sch:report test="@vote!='Negative' and @resolution">
                        WARNING: Resolutions should only be specified for negative
votes.</sch:report>
                  <sch:report test="@resolution and
parent::mif:*/parent::header//mif:ballotComment[not(mif:resolution)]">
                        WARNING: Should only have a resolution for a ballot submission when all
ballotcomments have been resolved.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="voterComments" type="BasicMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>General comments explaining the reason for the ballot
response.</xs:documentation>
```

```
               <xs:documentation>UML: voterComments tag on BallotResponse
stereotype</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="submissionId" type="BasicId" use="required">
         <xs:annotation>
            <xs:documentation>A unique identifier for a specific ballot response.  All ballot
comments must refer to a submissionId.</xs:documentation>
            <xs:documentation>UML: submissionId tag on BallotResponse
stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="submitterOrganization" type="BasicFormalName" use="optional">
         <xs:annotation>
            <xs:documentation>The organization responsible for the ballot
response.</xs:documentation>
            <xs:documentation>UML: submitterOrganization tag on BallotResponse
stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="submitterName" type="BasicFormalName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the person responsible for the ballot response, and who
will be responsible for defending the ballot response and/or withdrawing it.</xs:documentation>
            <xs:documentation>UML: submitterName tag on Mapping stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="vote" type="VoteKind" use="required">
         <xs:annotation>
            <xs:documentation>Indicates whether the voter wants the content to pass the
ballot.</xs:documentation>
            <xs:documentation>UML: vote tag on Mapping stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="resolution" type="NegativeVoteResolutionKind" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates how the vote was dealt with</xs:documentation>
            <xs:documentation>UML: vote tag on Mapping stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="status" type="BallotStatusKind" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates how the ballot has been resolved (if resolution is
necessary).</xs:documentation>
            <xs:documentation>UML: status tag on Mapping stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="statusDate" type="xs:date" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates when the most recent ballot status was arrived
at.</xs:documentation>
            <xs:documentation>UML: statusDate tag on Mapping stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
</xs:schema>
```

C.2 mifStaticBase

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- mifStaticBase -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3/mif"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
*******************************************************************************
```

```
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   This schema defines elements that are common to static models and to datatypes.

Programatic rules (rules that apply but are not schema or schematron-enforced):
  - Default values, fixed values and minimumSupportedLength may only be specified for code values
and datatypes that are convertable to String
********************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifBase.xsd"/>
   <xs:complexType name="DomainSpecificationWithStrength">
      <xs:annotation>
         <xs:documentation>Identifies the vocabulary concept that is the 'root' for the content
of this attribute.
             If the concept is identified by multiple means (e.g. conceptId and domainName), it
is the responsiblity of the user to ensure that both point to the same
concept.</xs:documentation>
         <xs:documentation>UML: DomainSpecification</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate DomainSpecification type">
               <sch:rule abstract="true" id="DomainSpecificationWithStrength">
                  <sch:extends rule="DomainSpecification"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DomainSpecification">
            <xs:attribute name="codingStrength" type="CodingStrengthKind" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies the level of flexibility the constructor of a
model instance has in using coded values.</xs:documentation>
                  <xs:documentation>UML: tag on DomainSpecification</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DomainSpecification">
      <xs:annotation>
         <xs:documentation>Used to reference a domain or code associated with another
element</xs:documentation>
         <xs:documentation>UML: DomainSpecification</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate DomainSpecification type">
               <sch:rule abstract="true" id="DomainSpecification">
                  <sch:report test="not(@domainName or @mnemonic)">
                      ERROR: At least one of domainName or mnemonic must be
specified.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Dependency">
            <xs:attribute name="domainName" type="DomainName" use="optional">
               <xs:annotation>
                  <xs:documentation>The formal name for the vocabulary domain from which content
may be drawn.</xs:documentation>
                  <xs:documentation>UML: DomainSpecification supplier
association</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="mnemonic" type="ShortDescriptiveName" use="optional">
               <xs:annotation>
```

```
                  <xs:documentation>The code corresponding to the domain from which content may
be drawn.</xs:documentation>
                  <xs:documentation>UML: DomainSpecification supplier
association</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:attributeGroup name="GeneralizableElement">
      <xs:annotation>
         <xs:documentation>Common ancestor of all types that can be abstract</xs:documentation>
         <xs:documentation>UML: Maps to GeneralizableElement.  (We only include it in those types
that we want to allow to be abstract)</xs:documentation>
      </xs:annotation>
      <xs:attribute name="isAbstract" type="xs:boolean" use="optional" default="false">
         <xs:annotation>
            <xs:documentation>Indicates that the specified datatype is not intended to appear
directly in an instance.  Only derivations of the type may appear.</xs:documentation>
            <xs:documentation>UML: inherited from GeneralizableElement</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:attributeGroup>
   <xs:complexType name="Classifier">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Classifier element</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ModelElement">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
            </xs:sequence>
            <xs:attributeGroup ref="GeneralizableElement"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Interface">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Interface element</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Classifier"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Feature">
      <xs:annotation>
         <xs:documentation>Common ancestor for attributes and datatype
properties</xs:documentation>
         <xs:documentation>UML: Corresponds to Feature stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Feature type">
               <sch:rule abstract="true" id="Feature">
                  <sch:extends rule="Presence"/>
                  <sch:extends rule="UpdateMode"/>
                  <sch:report test="@fixedValue and @defaultValue">
                     ERROR: Can't have both a 'fixed' value and a 'default' value for a single
element.</sch:report>
                  <sch:report test="@defaultFrom and not(@defaultValue)">
                     ERROR: Can't have both a 'defaultFrom' value and a 'default' value for a
single element.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ModelElement">
            <xs:attribute name="defaultValue" type="ShortDescriptiveName" use="optional">
               <xs:annotation>
```

```
                  <xs:documentation>Identifies the default value for the element if it is not
present in an instance of the model.</xs:documentation>
                  <xs:documentation>UML: defaultValue tag on Feature
stereotype</xs:documentation>
               </xs:annotation>
               <!-- TODO: Figure out what 'defaults' mean for properties.  I think they are
defaults for the 'constructor'.  Should we model constructor explicitly?  Ask Gunther. -->
            </xs:attribute>
            <xs:attribute name="defaultFrom" type="DefaultDeterminerKind" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates how the default should be
inferred</xs:documentation>
                  <xs:documentation>UML: defaultFrom tag on Feature stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="fixedValue" type="ShortDescriptiveName" use="optional">
               <xs:annotation>
                  <xs:documentation>Identifies the default value for the element if it is not
present in an instance of the model.</xs:documentation>
                  <xs:documentation>UML: fixedValue tag on Feature stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="minimumSupportedLength" type="SmallPositiveInteger"
use="optional">
               <xs:annotation>
                  <xs:documentation>Identifies the minimum number of characters (in the character
set used) that an application must support to be conformant.  This does not prohibit applications
from sending more characters, and does not remove the need for applications to gracefully handle
circumstances when they handle more than the number of characters they can successfully
process.</xs:documentation>
                  <xs:documentation>UML: minimumSupportedLength tag on Feature
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attributeGroup ref="UpdateMode"/>
            <xs:attributeGroup ref="Presence"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:attributeGroup name="UpdateMode">
      <xs:annotation>
         <xs:documentation>Those types related to update mode</xs:documentation>
         <xs:documentation>UML: Corresponds to UpdateMode stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate UpdateMode type">
               <sch:rule abstract="true" id="UpdateMode">
                  <sch:report test="@updateModeDefault and not(contains(concat(' ',
@updateModesAllowed, ' '), concat(' ', @updateModeDefault, ' '))">
                     ERROR: Can't have both a 'fixed' value and a 'default' value for a single
element.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="updateModeDefault" type="UpdateModeKind" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the update mode that should be assumed if no updateMode
is specified.</xs:documentation>
            <xs:documentation>UML: tag from UpdateMode stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="updateModesAllowed" type="UpdateModeCodes" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies the list of update modes that may be used for this
element.</xs:documentation>
            <xs:documentation>UML: tag from UpdateMode stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
```

```
   </xs:attributeGroup>
   <xs:simpleType name="UpdateModeCodes">
      <xs:annotation>
         <xs:documentation>A list of update modes</xs:documentation>
         <xs:documentation>UML: Complex type used as tag value</xs:documentation>
      </xs:annotation>
      <xs:list itemType="UpdateModeKind"/>
   </xs:simpleType>
   <xs:attributeGroup name="Presence">
      <xs:annotation>
         <xs:documentation>Those attributes that define whether an element must be
present</xs:documentation>
         <xs:documentation>UML: Corresponds to Presence stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Presence type">
               <sch:rule abstract="true" id="Presence">
                  <sch:report test="@mandatory='false' and @conformance!='R'">
                      ERROR: Conformance must be 'R' for mandatory elements.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="isMandatory" type="xs:boolean" use="optional" default="false">
         <xs:annotation>
            <xs:documentation>If true, null values may not be sent for this
element.</xs:documentation>
            <xs:documentation>UML: Tag value on Feature stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="conformance" type="ConformanceKind" use="optional">
         <xs:annotation>
            <xs:documentation>Identifies whether the element must be supported by implementors or
not.  If not present, indicates that support is optional.</xs:documentation>
            <xs:documentation>UML: Tag value on Feature stereotype</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:attributeGroup>
</xs:schema>
```

C.3 mifDatatype

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- mifDatatype -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3/mif"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
******************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   Provides a format for defining datatypes and their properties.

Modification History:
   2002-12-13: Added minimumSupportedLength
   2003-01-25: Changed enumeration set for property.kind
         - Removed CME coding strength
   2003-02-07: Moved definition of property kind type to internalMarkup
   2003-02-14: Require 'kind' attribute on propertyReferences as well as properties
   2003-03-18: Changed type for domainName to 'domainNameType' to allow for x_domains

Programatic rules (rules that apply but are not schema or schematron-enforced):
  - All datatypes referenced in a DatatypeDefinition must exist in the datatype model, or in one
of the models in the inheritance hierarcy.
  - There can't be any duplicates of datatype name or formalName within a DatatypeModelLibrary or
any of the models in the inheritance hierarchy
```

```
Outstanding questions:
  - What constraints should there be on datatypeModelLibrary.modelId?
*************************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifStaticBase.xsd"/>
   <xs:element name="datatypeModelLibrary" type="DatatypeModelLibrary">
      <xs:annotation>
         <xs:documentation>A datatype model</xs:documentation>
         <xs:documentation>UML: A Package with a DatatypeModel stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate datatypeModel element">
               <sch:rule context="mif:datatypeModel">
                  <sch:extends rule="DatatypeModel"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="DatatypeModelLibrary">
      <xs:annotation>
         <xs:documentation>A container for datatype definitions</xs:documentation>
         <xs:documentation>UML: DatatypeModelLibrary stereotype.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate DatatypeModel type">
               <sch:rule abstract="true" id="DatatypeModelLibrary">
                  <sch:extends rule="Package"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Package">
            <xs:sequence>
               <xs:element name="derivationSupplier" type="PackageDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the package(s) from which the current package
was derived.</xs:documentation>
                     <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="supplierImportDatatypeModelLibrary"
type="ImportDatatypeModelLibrary" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies any datatype models that are extended by the
current model</xs:documentation>
                     <xs:documentation>UML: supplier association to Import stereotype of a
Permission dependency to another DatatypeModelLibrary</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="ownedDatatype" type="Datatype" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Lists all of the datatypes defined in the current model,
as well as those referenced from one of the inherited models.</xs:documentation>
                     <xs:documentation>UML: The ownedElement relationship to the
DatatypeDefinitions in the package</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate ownedDatatype element">
                           <sch:rule context="mif:ownedDatatype">
                              <sch:extends rule="Datatype"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
```

```
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ImportDatatypeModelLibrary">
      <xs:annotation>
         <xs:documentation>Identifies the DatatypeModelLibrary package being imported, as well as
identifying all datatypes that are to be publicly exposed from the imported model.  (Default is
'private' exposure for those datatypes not explicitly mentioned.)</xs:documentation>
         <xs:documentation>UML: Represents a combination of the Import stereotype to another
DatatypeModelLibrary (via the packageref), as well as the ElementImport association to each
imported datatype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:sequence>
               <xs:element name="importedDatatype" type="DatatypeImport" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies those datatypes that should be 'exposed' in the
current datatype model.  Exposed datatypes will be usable as attribute datatypes.  All datatypes
will be accessible for 'refinement'.</xs:documentation>
                     <xs:documentation>UML: The ElementImport association to each datatype being
imported</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DatatypeImport">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to ElementImport class association pointing to an
imported datatype</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DatatypeRef">
            <xs:attribute name="visibility" type="VisibilityKind" use="required" fixed="public">
               <xs:annotation>
                  <xs:documentation>Indicates that the imported datatype should be made 'public'
in this datatype model</xs:documentation>
                  <xs:documentation>UML: ElementImport.visibility</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Datatype">
      <xs:annotation>
         <xs:documentation>The definition of a structure for transmitting and persisting
fundamental concepts within a model.</xs:documentation>
         <xs:documentation>UML: DatatypeDefinition stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Datatype type">
               <sch:rule abstract="true" id="Datatype">
                  <sch:report
test="count(preceding::mif:*[name(.)=name(current())][@name=current()/@name][count(mif:parameter[
@name=current()/mif:parameter/@name])=count(mif:parameter)])!=0">
                     ERROR: There must not be more than one datatype definition having the same
name and parameters.</sch:report>
                  <sch:report
test="ancestor::mif:datatypeModelLibrary//*[name(.)=name(current())][@name=current()/@name]&gt;1"
>
                     ERROR: There cannot be more than one datatype with the same
name.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
```

```
      <xs:complexContent>
         <xs:extension base="Classifier">
            <xs:sequence>
               <xs:element name="derivationSupplier" type="DatatypeDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the datatype(s) from which the current datatype
was derived.</xs:documentation>
                     <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="graphicRepresentation" type="NodeSemanticModelBridge"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Indicates the display shape(s) associated with the
datatype definition</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="annotations" type="DatatypeAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a model element.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="supplierBinding" type="DatatypeBinding" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Identifies a parameterized datatype for which this
datatype is an 'instantiation'</xs:documentation>
                     <xs:documentation>UML: Indicates the supplier binding dependency for the
datatype</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="ownedDatatype" type="Datatype" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>A datatype defined to exist only within the context of
it's parent datatype.  (Generally used for protected or private types.)</xs:documentation>
                     <xs:documentation>UML: Classifiers can 'own' other classifiers because they
descend from package.  Datatype prohibits this in 1.4, so we'll use a Class underlying the
stereotype where we want to do this instead of a </xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate ownedDatatype element">
                           <sch:rule context="mif:ownedDatatype">
                              <sch:extends rule="Datatype"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
               <xs:element name="operation" type="DatatypeOperation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>A capability of a datatype.</xs:documentation>
                     <xs:documentation>UML: feature association to a DatatypeOperation
stereotype</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Validate operation element">
                           <sch:rule context="mif:operation">
                              <sch:extends rule="DatatypeOperation"/>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
```

```
                  </xs:annotation>
                  <xs:unique name="parameterNameUniqueInProperty">
                     <xs:selector xpath="parameter"/>
                     <xs:field xpath="@name"/>
                  </xs:unique>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="DatatypeName" use="required">
               <xs:annotation>
                  <xs:documentation>The formal identifier associated with the
datatype.</xs:documentation>
                  <xs:documentation>UML: inherited from modelElement.name</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="title" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The descriptive name associated with the
datatype.</xs:documentation>
                  <xs:documentation>UML: title tag on Datatype stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="visibility" type="VisibilityKind" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies the amount of exposure the datatype
has.</xs:documentation>
                  <xs:documentation>UML: Inherits from modelElement.visibility.  Restricted to
private, protected or public.  ('Package' is not allowed)</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DatatypeBinding">
      <xs:annotation>
         <xs:documentation>Identifies the parameterized datatype being 'instatntiated', as well
as the types to use for each parameter</xs:documentation>
         <xs:documentation>UML: The binding dependency that links to the datatype that the
current datatype binds to</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="targetDatatype" type="DatatypeRef">
            <xs:annotation>
               <xs:documentation>The parameterized datatype that this definition
'instantiates'</xs:documentation>
               <xs:documentation>UML: Datatype that is the target of the binding
dependency</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="argumentDatatype" type="DatatypeRef" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>The parameterized datatype that this definition
'instantiates'</xs:documentation>
               <xs:documentation>UML: Datatype that is the argument model element for the binding
dependency</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="DatatypeAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to the datatype model</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
Datatype.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
```

```
                <xs:group ref="Definition"/>
                <xs:group ref="UsageNotes"/>
                <xs:group ref="Rationale"/>
                <xs:group ref="DesignComments"/>
                <xs:group ref="Mapping"/>
                <xs:group ref="BallotComment"/>
                <xs:group ref="OtherAnnotation"/>
             </xs:sequence>
          </xs:restriction>
       </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="DatatypeDerivation">
       <xs:annotation>
          <xs:documentation>UML: A derivation dependency to another datatype</xs:documentation>
       </xs:annotation>
       <xs:complexContent>
          <xs:extension base="Derivation">
             <xs:sequence>
                <xs:element name="targetDatatype" type="DatatypeRef">
                   <xs:annotation>
                      <xs:documentation>Identifies the datatype from which the current type is
derived.</xs:documentation>
                      <xs:documentation>UML: supplier of Derivation dependency
stereotype</xs:documentation>
                   </xs:annotation>
                </xs:element>
             </xs:sequence>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="BehavioralFeature">
       <xs:annotation>
          <xs:documentation>Base class for methods and datatype operations</xs:documentation>
          <xs:documentation>UML: Corresponds to BehavioralFeature</xs:documentation>
          <xs:appinfo>
             <sch:pattern name="Validate BehavioralFeature type">
                <sch:rule abstract="true" id="BehavioralFeature">
                   <sch:extends rule="Feature"/>
                </sch:rule>
             </sch:pattern>
          </xs:appinfo>
       </xs:annotation>
       <xs:complexContent>
          <xs:extension base="Feature">
             <xs:sequence>
                <xs:element name="parameter" type="Parameter" minOccurs="0" maxOccurs="unbounded">
                   <xs:annotation>
                      <xs:documentation>A parameter passed to the property used in determining the
value for the property.</xs:documentation>
                      <xs:documentation>UML: Parameter of 'input' type</xs:documentation>
                   </xs:annotation>
                   <!-- Todo: Consider having defaults for parameter  -->
                </xs:element>
             </xs:sequence>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="Parameter">
       <xs:annotation>
          <xs:documentation>UML: Corresponds to 'Parameter' for an operation that takes a datatype
as it's argument.</xs:documentation>
       </xs:annotation>
       <xs:complexContent>
          <xs:extension base="ModelElement">
             <xs:sequence>
                <xs:element name="type" type="DatatypeRef">
                   <xs:annotation>
                      <xs:documentation>Identifies the datatype of the property</xs:documentation>
```

```
                     <xs:documentation>UML: Parameter.type association</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="FormalProperName" use="optional">
               <xs:annotation>
                  <xs:documentation>The name of the parmater as referenced by code or by formal
constraints.</xs:documentation>
                  <xs:documentation>UML: from Parameter, inherited from
modelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Operation">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to 'Operation'</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="BehavioralFeature"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DatatypeOperation">
      <xs:annotation>
         <xs:documentation>Defines the content for a datatypeOperation</xs:documentation>
         <xs:documentation>UML: The basic part of the DatatypeOperation
stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate DatatypeOperation type">
               <sch:rule abstract="true" id="DatatypeOperation">
                  <extends rule="codeValidation"/>
                  <sch:report test="@operationKind!='conversion' and count(preceding-
sibling::mif:*[name(.)=name(current())][@name=current()/@name][count(mif:parameter[@name=current(
)/mif:parameter/@name])=count(mif:parameter)]">
                     ERROR: There must not be more than one operation within a datatype
definition having the same name and parameters.</sch:report>
                  <sch:report test="@defaultFrom and count(mif:type/descendant-or-
self::mif:*[contains('CD;CE;CS;SC;BL;ST;INT;REAL;MO;PQ', @name)])!=1">
                     WARNING: DefaultFrom may only be specified for codes and simple datatypes
(CD, CE, CS, SC, BL, ST, INT, REAL, MO, PQ).</sch:report>
                  <sch:report test="count(mif:supplierDomainSpecification)=0 and
count(mif:type/descendant-or-self::mif:*[contains('CD;CE;CS;SC', @name)])!=0">
                     WARNING: Domain must be present for coded types (CD, CE, CS and
SC).</sch:report>
                  <sch:report test="count(mif:supplierDomainSpecification)!=0 and
count(mif:type/descendant-or-self::mif:*[contains('CD;CE;CS;SC', @name)])=0">
                     WARNING: Domain may only be present for coded types (CD, CE, CS and
SC).</sch:report>-->
                   <sch:report test="@default and count(mif:type/descendant-or-
self::mif:*[contains('CD;CE;CS;SC;BL;ST;INT;REAL;MO;PQ', @name)])!=1">
                     WARNING: Default may only be specified for codes and simple datatypes (CD,
CE, CS, SC, BL, ST, INT, REAL, MO, PQ).</sch:report>
                  <sch:report test="@minimumSupportedLength and count(mif:type/descendant-or-
self::mif:*[contains('CD;CE;CS;SC;BL;ST;INT;REAL;MO;PQ', @name)])!=1">
                     WARNING: Minimum supported length may only be specified for codes and
simple datatypes (CD, CE, CS, SC, BL, ST, INT, REAL, MO, PQ).</sch:report>
                  <sch:report test="count(mif:type/descendant-or-self::mif:*[@name='CS'])!=0 and
mif:supplierDomainSpecification/@codingStrength='CWE'">
                     ERROR: CodingStrength must not be CWE for CS datatype
attributes.</sch:report>
                  <sch:report test="count(mif:derivationSupplier)=0 and
count(mif:businessName)=0">
                     ERROR: Non-derived operations must have a business name.</sch:report>
                  <!-- How come? . . . -->
                  <sch:report test="count(mif:derivationSupplier)=0 and
count(mif:annotations/mif:definition)=0">
```

```
                        ERROR: Non-derived operations must have a definition
annotation.</sch:report>
                </sch:rule>
             </sch:pattern>
          </xs:appinfo>
       </xs:annotation>
       <xs:complexContent>
          <xs:extension base="Operation">
             <xs:sequence>
                <xs:element name="annotations" type="OperationAnnotations" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                      <xs:documentation>UML: Inherited from modelElement</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="derivationSupplier" type="OperationDerivation" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>Identifies the property and the datatype from which the
current property is derived.</xs:documentation>
                      <xs:documentation>UML: Property with a derivedFrom
dependency</xs:documentation>
                      <xs:appinfo>
                         <sch:pattern name="Validate derivationSupplier element">
                            <sch:rule context="mif:operation/mif:derivationSupplier">
                               <sch:extends rule="OperationDerivation"/>
                            </sch:rule>
                         </sch:pattern>
                      </xs:appinfo>
                   </xs:annotation>
                </xs:element>
                <xs:element name="type" type="DatatypeRef" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>Identifies the type of the data exposed by the
property.</xs:documentation>
                      <xs:documentation>UML: return type of 'typed'
BehavioralFeature</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="supplierDomainSpecification"
type="DomainSpecificationWithStrength" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>References the HL7 vocabulary to define the set of allowed
values that may be conveyed by this property.</xs:documentation>
                      <xs:documentation>UML: Association to the vocabularyDomain - need to figure
out what association</xs:documentation>
                      <xs:appinfo>
                         <sch:pattern name="Validate supplierDomainSpecification element">
                            <sch:rule context="mif:supplierDomainSpecification">
                               <sch:extends rule="DomainSpecificationWithStrength"/>
                            </sch:rule>
                         </sch:pattern>
                      </xs:appinfo>
                   </xs:annotation>
                </xs:element>
             </xs:sequence>
             <xs:attribute name="operationKind" type="DatatypeOperationKind" use="required">
                <xs:annotation>
                   <xs:documentation>Identifies the variety of property being represented.  This
attribute influences how the property is represented for display purposes in
documentation.</xs:documentation>
                   <xs:documentation>UML: Tagged value on datatypeProperty</xs:documentation>
                </xs:annotation>
             </xs:attribute>
             <xs:attribute name="name" type="FormalPropertyName" use="optional">
                <xs:annotation>
                   <xs:documentation>Name of the property</xs:documentation>
                   <xs:documentation>UML: Name inherited from DataElement</xs:documentation>
```

```
            </xs:annotation>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="OperationDerivation">
   <xs:annotation>
      <xs:documentation>UML: A derivation dependency on another datatype
operation</xs:documentation>
      <xs:appinfo>
         <sch:pattern name="Validate OperationDerivation type">
            <sch:rule abstract="true" id="OperationDerivation">
               <sch:report test="@operationName and count(mif:type)!=0">
                  ERROR: A property reference may only have a name or a conversion datatype,
not both.</sch:report>
               <sch:report test="not(@operationName) and count(mif:type)=0">
                  ERROR: A property reference must have either a name or a conversion
datatype.</sch:report>
            </sch:rule>
         </sch:pattern>
      </xs:appinfo>
   </xs:annotation>
   <xs:complexContent>
      <xs:extension base="Derivation">
         <xs:sequence>
            <xs:element name="type" type="DatatypeRef" minOccurs="0">
               <xs:annotation>
                  <xs:documentation>For 'conversion' properties, identifies the target
datatype for the conversion.</xs:documentation>
                  <xs:documentation>UML: result type of the operation.  Used for referencing
non-named 'conversion' operations</xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element name="targetDatatype" type="DatatypeRef">
               <xs:annotation>
                  <xs:documentation>Identifies the datatype from which the property is
derived.</xs:documentation>
                  <xs:documentation>UML: client association on the derivedFrom
dependency</xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="operationName" type="FormalPropertyName" use="optional">
            <xs:annotation>
               <xs:documentation>The name of the operation in the identified datatype on which
this property is based.</xs:documentation>
               <xs:documentation>UML: name as inherited from DataElement</xs:documentation>
            </xs:annotation>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="DatatypeTemplateParameter">
   <xs:annotation>
      <xs:documentation>UML: TemplateParameter for a datatype</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:extension base="DatatypeRef">
         <xs:attribute name="parameterName" type="FormalProperName" use="required">
            <xs:annotation>
               <xs:documentation>The name of the parameter for use in describing the datatype
or in constructing formal constraints.</xs:documentation>
               <xs:documentation>UML: Name tag on TemplateParameter
stereotype</xs:documentation>
            </xs:annotation>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
```

```
   </xs:complexType>
   <xs:complexType name="OperationAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to the operation</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
Datatype operation.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Definition"/>
               <xs:group ref="UsageNotes"/>
               <xs:group ref="Rationale"/>
               <xs:group ref="DesignComments"/>
               <xs:group ref="Mapping"/>
               <xs:group ref="Constraint"/>
               <xs:group ref="BallotComment"/>
               <xs:group ref="OtherAnnotation"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

C.4 mifStaticModelBase

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by David Markwell (The Clinical
Information Consultancy) -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3/mif"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
********************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   The purpose of this schema is to provide a common XML format for the documentation and
exchange of *all* static data models.
   For MDF based designs, this includes RIMs, D-MIMs, R-MIMs, CMETs, HMDs and Message Types
   For HDF based designs, it includes RIMs, DIMs, CIMs and LIMs (defined a little later :>)

Modification History:
   2002-12-13: Added minimumSupportedLength
   2003-01-23: Replaced 'diagramReference' attributes with 'figure' elements
   2003-01-24: Fixed the 'derivationSupplier' elements for association, state and
stateTransition to include className, and for stateTransition the endStateName
   2003-01-25: Added historyItem and annotations to 'descendant'
         - Added 'sequence' to all class types to control order of display, as well as a rule
requiring it to be unique when present and requiring it to be present everywhere when present
         - Added type and committeeName to subject area
   2003-03-18: Added 'sequence' to descendant, and rule requiring it to be unique.
         - Added 'conformance' attribute to descendant
         - Redefined 'staticModels' to make it reusable
   2003-03-21: Added 'routing points' everywhere we had 'connection' points.
   2003-05-11: Made entry point required

Outstanding Questions:
  - Should we require entry-points in the RIM too (Act, Entity, Role, Communication,
ClinicalDocument and Parameter).  (If so we can toast schematron rule and make
entryPoint.minOccurs=1)
  - Are the values for contentTypeType appropriate?  Complete?
```

```
Future:
  - Allow support for associative classes?  (On the class identify the linkClass and
linkRelationshipName for each end of the association
  - Allow constraints on datatype components? (i.e. expose the properties inside the datatype and
allow constraints on domain, cardinality, etc. within models the way we do in conformance
profiles)

Programatic rules (rules that apply but are not schema or schematron-enforced):
  - Validate contents against external file of datatypes, vocabulary and CMET values
  - Subject Area and class name must not be same as any name in parent hierarchy
  - For models that are serializable, at least one direction of every relationship should be
blocked, unless there is a loop
  - Ensure that classCode is present for classes that are descended from a vocab-extensible class
(ancestor-or-self has an attribute called classCode), and is not present otherwise
  - Ensure entire model is interconnected (all classes a reachable by a path from all other
classes
  - Ensure there is a traversal path to all classes from at least one entry-point
  - Classes should not have a subject area that is the ancestor of another subject area they are
also a part of.
  - In it's serialized form, an 'extended' model must not have any extended classes (those with
no 'derived-from' for a given modelId) have children which have a derivationSupplier of that
modelId
  - All of the derivation rules should be enforced 'as if' the model were in it's hierarchical
form.  (I.e. what derivation types are allowed inside other types)
  - Can't have a class name that is the same as any datatype name or CMET name in existence in
the datatype models and CMET models imported.
*******************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifStaticBase.xsd"/>
   <xs:complexType name="SubSystem">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to 'SubSystem'.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate SubSystem type">
               <sch:rule abstract="true" id="SubSystem">
                  <sch:extends rule="Package"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Package"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StaticModelBase">
      <xs:annotation>
         <xs:documentation>Basic content for defining static models and subject
areas</xs:documentation>
         <xs:documentation>UML: StaticModel stereotype</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate StaticModelBase type">
               <sch:rule abstract="true" id="StaticModelBase">
                  <sch:extends rule="SubSystem"/>
                  <sch:report test="@isSerializable='true' and ancestor-or-
self::mif:*/mif:packageLocation[contains(';RIM;DIM;DMIM-deprecated;', concat(';', @artifactKind,
';'))]">
                     WARNING: RIM, DMIM and DIM models are not serializable.</sch:report>
                  <sch:report test="@isSerializable='true' and ancestor-or-
self::mif:*[contains(';RIM;DIM;DMIM-deprecated;', concat(';', @packageKind, ';'))]">
                     WARNING: RIM, DMIM and DIM models are not serializable.</sch:report>
                  <sch:report test="@isSerializable='false' and ancestor-or-
self::mif:*/mif:packageLocation[contains(';RIM;DIM;DMIM-deprecated;', concat(';', @artifactKind,
';'))]">
                     WARNING: All model types except RIM, DMIM and DIM models are
serializable.</sch:report>
```

```
                    <sch:report test="@isSerializable='false' and ancestor-or-
self::mif:*[contains(';RIM;DIM;DMIM-deprecated;', concat(';', @packageKind, ';'))]">
                        WARNING: All model types except RIM, DMIM and DIM models are
serializable.</sch:report>
                    <sch:report test="@isSerializable='false' and
@representationKind='serialized'">
                        ERROR: Can't have a serialized representation of a non-serializable
model.</sch:report>
                    <sch:report test="@isSerializable='true' and @representationKind='flat' and

   count(ownedAssociation/connections[count(traversableConnection)!=1])!=0">
                        ERROR: Serializable model associations must be traverable in one and only
one direction..</sch:report>
                    <sch:report test="ancestor-or-
self::mif:*[mif:packageLocation/@artifactKind='RIM' or @packageKind='RIM'] and
count(mif:ownedSubjectArea)=0">
                        GUIDELINE: RIM models must have at least one subjectArea.</sch:report>
                    <sch:report test="@isSerializable='true' and count(mif:ownedEntryPoint)!=1">
                        ERROR: Serializable models must have exactly one entry point.</sch:report>
                    <sch:report test="ancestor-or-
self::mif:*[mif:packageLocation/@artifactKind='RIM' or @packageKind='RIM'] and
count(mif:derivationSupplier)!=0">
                        ERROR: A RIM cannot be derived from another model.</sch:report>
                    <sch:report test="not(ancestor-or-
self::mif:*[mif:packageLocation/@artifactKind='RIM' or @packageKind='RIM']) and
count(mif:derivationSupplier[mif:targetStaticModel/@artifactKind='RIM'])=0">
                        ERROR: Non-RIM models must be derived from at least one other model (one of
which must have a type of 'RIM').</sch:report>
                </sch:rule>
            </sch:pattern>
        </xs:appinfo>
        <xs:appinfo>
            <sch:pattern name="Check model annotations">
                <sch:rule context="mif:annotations/*">
                    <sch:report test="count(mif:presentation)!=0 and
not(parent::mif:annotations[parent::mif:class or parent::mif:classTypeRef or
parent::mif:classStub or parent::mif:attribute or parent::mif:association])">
                        ERROR: Annotations may only have presentation information when associated
with a class, classTypeRef, classStub, attribute or association.</sch:report>
                    <sch:report test="count(mif:presentation)!=0 and
                                    ((not(parent::mif:annotations[parent::mif:attribute or
parent::mif:association]) and

   count(parent::mif:annotations/parent::mif:*/mif:presentation[@shapeId=current()/mif:presentat
ion/@connectionShapeId])=0) or
                                    (parent::mif:annotations/parent::mif:attribute and

   count(parent::mif:annotations/parent::mif:attribute/parent::mif:*/mif:presentation[@shapeId=c
urrent()/mif:presentation/@connectionShapeId])=0) or
                                    (parent::mif:annotations/parent::mif:association and

   count(parent::mif:annotations/parent::mif:association/parent::mif:*/mif:presentation[@shapeId
=current()/mif:presentation/@connectionShapeId])=0))">
                        ERROR: Connection shapeId on annotation does not point to a shape
associated with the annotated class.</sch:report>
                </sch:rule>
            </sch:pattern>
            <sch:pattern name="Check displayInfo constraints">
                <sch:rule context="mif:staticModel">
                    <sch:report test="count(.//mif:displayInfo[@representationType='UML'])!=0 and
count(.//mif:subjectArea) !=
count(.//mif:subjectArea[count(mif:displayInfo[@representationType='UML'])!=0])">
                        ERROR: If any elements have UML displayInfo, all subjectAreas must have UML
displayInfo.</sch:report>
                    <sch:report test="count(.//mif:displayInfo[@representationType='UML'])!=0 and
count(.//mif:entryPoint) !=
count(.//mif:entryPoint[count(mif:displayInfo[@representationType='UML'])!=0])">
```

```
                    ERROR: If any elements have UML displayInfo, all entryPoints must have UML
displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='UML'])!=0 and
count(.//mif:class) !=
count(.//mif:class[count(mif:displayInfo[@representationType='UML'])!=0])">
                    ERROR: If any elements have UML displayInfo, all classes must have UML
displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='UML'])!=0 and
count(.//mif:classTypeRef) !=
count(.//mif:classTypeRef[count(mif:displayInfo[@representationType='UML'])!=0])">
                    ERROR: If any elements have UML displayInfo, all classTypeRefs must have
UML displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='UML'])!=0 and
count(.//mif:classStub) !=
count(.//mif:classStub[count(mif:displayInfo[@representationType='UML'])!=0])">
                    ERROR: If any elements have UML displayInfo, all classStubs must have UML
displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='HL7Visio'])!=0
and count(.//mif:subjectArea) !=
count(.//mif:subjectArea[count(mif:displayInfo[@representationType='HL7Visio'])!=0])">
                    ERROR: If any elements have HL7Visio displayInfo, all subjectAreas must
have HL7Visio displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='HL7Visio'])!=0
and count(.//mif:entryPoint) !=
count(.//mif:entryPoint[count(mif:displayInfo[@representationType='HL7Visio'])!=0])">
                    ERROR: If any elements have HL7Visio displayInfo, all entryPoints must have
HL7Visio displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='HL7Visio'])!=0
and count(.//mif:class) !=
count(.//mif:class[count(mif:displayInfo[@representationType='HL7Visio'])!=0])">
                    ERROR: If any elements have HL7Visio displayInfo, all classes must have
HL7Visio displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='HL7Visio'])!=0
and count(.//mif:classTypeRef) !=
count(.//mif:classTypeRef[count(mif:displayInfo[@representationType='HL7Visio'])!=0])">
                    ERROR: If any elements have HL7Visio displayInfo, all classTypeRefs must
have HL7Visio displayInfo.</sch:report>
                <sch:report test="count(.//mif:displayInfo[@representationType='HL7Visio'])!=0
and count(.//mif:classStub) !=
count(.//mif:classStub[count(mif:displayInfo[@representationType='HL7Visio'])!=0])">
                    ERROR: If any elements have HL7Visio displayInfo, all classStubs must have
HL7Visio displayInfo.</sch:report>
            </sch:rule>
            <sch:rule context="mif:displayInfo">
                <sch:report test="preceding-
sibling::mif:displayInfo[@representationType=current()/@representationType] and
not(parent::mif:class or parent::mif:classTypeRef or parent::mif:classStub)">
                    ERROR: Only classes may have more than one display info for a given
representationType.  (Classes may have 'shadow' shapes for hl7 representations).</sch:report>
                <sch:report
test="count(parent::mif:*/mif:displayInfo[@representationType='UML'])&gt;1">
                    ERROR: There may not be more than one set of UML display info for a given
element.</sch:report>
            </sch:rule>
        </sch:pattern>
        <sch:pattern name="Validate derivationSupplier constraints">
            <sch:rule
context="mif:staticModel/mif:derivationSupplier[@derivationType!='extension' and
@derivationType='incompatible']">
                <sch:report
test="count(parent::mif:staticModel//mif:*[mif:derivationSupplier[@className]/@modelId=current()/
@modelId])!=
                                        count(parent::mif:staticModel//*[self::mif:class or
self::mif:classTypeRef or self::mif:classStub])">
                            ERROR: If a derivationSupplier' element exists for a staticModel,
all classes within that model must also have a derivationSupplier for the same modelId, unless
the derivation type is 'extension'.</sch:report>
            </sch:rule>
```

```
            <sch:rule context="mif:derivationSupplier[parent::mif:class or
parent::mif:classTypeRef or parent::mif:classStub][@derivationType!='extension' and
@derivationType='incompatible']">
               <sch:report
test="count(parent::mif:*/mif:attribute[mif:derivationSupplier/@modelId=current()/@modelId])!=cou
nt(parent::mif:*/mif:attribute)">
                              ERROR: If a 'derivationSupplier' element exists for a
staticModel, all attributes within that model must also have a derivationSupplier for the same
modelId.</sch:report>
               <sch:report
test="count(parent::mif:*/mif:association[mif:derivationSupplier/@modelId=current()/@modelId])!=c
ount(parent::mif:*/mif:association)">
                              ERROR: If a 'derivationSupplier' element exists for a
staticModel, all associations within that model must also have a derivationSupplier for the same
modelId.</sch:report>
               <sch:report
test="count(parent::mif:*/mif:stateMachine/mif:state[mif:derivationSupplier/@modelId=current()/@m
odelId])!=count(parent::mif:*/mif:stateMachine/mif:state)">
                              ERROR: If a 'derivationSupplier' element exists for a
staticModel, all states within that model must also have a derivationSupplier for the same
modelId.</sch:report>
               <sch:report
test="count(parent::mif:*/mif:stateMachine/mif:transition[mif:derivationSupplier/@modelId=current
()/@modelId])!=count(parent::mif:*/mif:stateMachine/mif:transition)">
                              ERROR: If a 'derivationSupplier' element exists for a
staticModel, all transitions within that model must also have a derivationSupplier for the same
modelId.</sch:report>
            </sch:rule>
         </sch:pattern>
         <sch:pattern name="Validate additional derivationSupplier constraints">
            <sch:rule context="mif:derivationSupplier">
               <sch:report test="count(preceding-
sibling::mif:derivationSupplier[@modelId=current()/@modelId])!=0">
                              GUIDELINE: An element may not be derived from two different
elements within the same model.</sch:report>
               <sch:report test="@derivationType='unchanged' and
count(parent::mif:*//mif:derivationSupplier[@modelId=current()/@modelId][@derivationType!='unchan
ged'])">
                              ERROR: An element derivation may not be marked as 'unchanged'
unless all child elements are marked as 'unchanged'.</sch:report>
               <sch:report test="@derivationType='annotation' and
count(parent::mif:*//mif:derivationSupplier[@modelId=current()/@modelId][@derivationType!='unchan
ged' and @derivationType!='annotation'])">
                              ERROR: An element derivation may not be marked as 'annotation'
unless all child elements are marked as 'annotation' or 'unchanged'.</sch:report>
               <sch:report test="@derivationType='restriction' and
count(parent::mif:*//mif:derivationSupplier[@modelId=current()/@modelId][@derivationType!='unchan
ged' and @derivationType!='annotation' and @derivationType!='restriction'])">
                              ERROR: An element derivation may not be marked as 'restriction'
unless all child elements are marked as 'restriction', 'annotation' or 'unchanged'.</sch:report>
               <sch:report test="@derivationType='extention' and
count(parent::mif:*//mif:derivationSupplier[@modelId=current()/@modelId][@derivationType!='unchan
ged' and @derivationType!='annotation' and @derivationType!='restriction' and
@derivationType!='extention'])">
                              ERROR: An element derivation may not be marked as 'extention'
unless all child elements are marked as 'extention', 'restriction', 'annotation' or
'unchanged'.</sch:report>
            </sch:rule>
         </sch:pattern>
         <sch:pattern name="Validate still more derivationSupplier constraints">
            <sch:rule context="mif:derivationSupplier[@stateName or @stateTransitionName]">
               <sch:report
test="count(parent::mif:stateMachine/parent::mif:*/mif:derivationSupplier[@modelId=current()/@mod
elId])=0">
                              ERROR: An element may only be derived from a model that its
parent is also derived from</sch:report>
            </sch:rule>
```

```
                <sch:rule
context="mif:derivationSupplier[@attributeName|@associationName|@className]">
                    <sch:report
test="count(parent::mif:*/mif:derivationSupplier[@modelId=current()/@modelId])=0">
                                    ERROR: An element may only be derived from a model that its
parent is also derived from</sch:report>
                </sch:rule>
             </sch:pattern>
          </xs:appinfo>
       </xs:annotation>
       <xs:complexContent>
          <xs:extension base="SubSystem">
             <xs:sequence>
                <xs:element name="annotations" type="StaticModelAnnotations" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                      <xs:documentation>UML: A collector for the comments and constraints
associated with a static model package.  (Consider rendering the definition or description
annotation into ModelElement.documentation)</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="graphicRepresentation"
type="StaticPackageDiagramSemanticModelBridge" minOccurs="0" maxOccurs="2">
                   <xs:annotation>
                      <xs:documentation>Indicates the display shape(s) associated with the static
package</xs:documentation>
                      <xs:documentation>UML: association from ModelElement to SemanticModelBridge
for a diagram</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="figure" type="Image" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>A graphical representation of the classes in a static
package model.</xs:documentation>
                      <xs:documentation>UML: figure tag on StaticPackage
stereotype</xs:documentation>
                   </xs:annotation>
                   <!-- todo: consider allowing multiple occurrences and/or tying to
graphicRepresentation -->
                </xs:element>
                <xs:element name="derivationSupplier" type="StaticModelDerivation" minOccurs="0"
maxOccurs="unbounded">
                   <xs:annotation>
                      <xs:documentation>Identifies static models from which the current model is
derived</xs:documentation>
                      <xs:documentation>UML: supplier association from ModelElement to Derivation
stereotype on Dependency</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="importedDatatypeModelPackage" type="PackageRef" minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>The datatype model that is used by this
model</xs:documentation>
                      <xs:documentation>UML: A package that is imported into the current data
model</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="importedCommonModelElementPackage" type="PackageRef"
minOccurs="0">
                   <xs:annotation>
                      <xs:documentation>The CMET model that is used by this
model</xs:documentation>
                      <xs:documentation>UML: A package that is imported into the current data
model</xs:documentation>
                   </xs:annotation>
                </xs:element>
                <xs:element name="importedStubPackage" type="PackageRef" minOccurs="0">
```

```
                  <xs:annotation>
                     <xs:documentation>The stub set that is used by this model</xs:documentation>
                     <xs:documentation>UML: A package that is imported into the current data
model</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="ownedSubjectAreaPackage" type="SubjectAreaPackage" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies a 'sub-package' owned by the current static
package.  All classes within the 'sub-packages' are always imported into their parent static
package.  This means the names of all classes within a static package must be unique.
Graphically it represents a grouping of classes that may be represented on a separate
page.</xs:documentation>
                     <xs:documentation>UML: SubjectAreaPackage stereotype</xs:documentation>
                  </xs:annotation>
                  <!-- Todo: Handle varying presentations for wrap-arround boxes and 'fill-in'
shapes.  Also, enforce rules that contained elements don't appear for fill-in shapes unless they
appear in two. -->
               </xs:element>
            </xs:sequence>
            <xs:attribute name="representationKind" type="StaticModelRepresentationKind"
use="required">
               <xs:annotation>
                  <xs:documentation>Identifies whether the model is represented in its flat or
serializable form.</xs:documentation>
                  <xs:documentation>UML: Not exposed.  All models will be represented in UML as
'flat'</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="isSerializable" type="xs:boolean" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies whether this model can be represented in a
serialized form.</xs:documentation>
                  <xs:documentation>UML: isSerializable tag on staticModel
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attributeGroup ref="GeneralizableElement"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StaticPackageDiagramSemanticModelBridge">
      <xs:annotation>
         <xs:documentation>The graphic representation for a UML concept that corresponds to a
complete diagram</xs:documentation>
         <xs:documentation>UML: Stereotype restricting SemanticModelBridge to a
Diagram</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DiagramSemanticModelBridge">
            <xs:attribute name="presentation" type="StaticModelDiagramPresentationKind"
use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates the diagramming format used to display the
element.</xs:documentation>
                  <xs:documentation>UML: presentation element on
ModelElementBridge</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SubjectAreaPackage">
      <xs:annotation>
         <xs:documentation>Defines the content for subject areas</xs:documentation>
         <xs:documentation>UML: SubjectAreaPackage stereotype</xs:documentation>
         <xs:appinfo>
```

```
            <sch:pattern name="Validate SubjectAreaPackage type">
               <sch:rule abstract="true" id="SubjectAreaPackage">
                  <sch:report test="count(mif:ownedSubjectAreaPackage|mif:ownedClass)=0">
                     WARNING: SubjectAreaPackages must contain either other subject areas or
classes.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageBase">
            <xs:sequence>
               <xs:element name="graphicRepresentation"
type="StaticPackageDiagramSemanticModelBridge" minOccurs="0" maxOccurs="2">
                  <xs:annotation>
                     <xs:documentation>Indicates the display shape(s) associated with the static
package</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to SemanticModelBridge
for a diagram</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="figure" type="Image" minOccurs="0">
                  <!-- todo: consider allowing multiple occurrences and/or tying to
graphicRepresentation -->
                  <xs:annotation>
                     <xs:documentation>A graphical representation of the classes in a static
package model.</xs:documentation>
                     <xs:documentation>UML: figure tag on StaticPackage
stereotype</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="annotations" type="SubjectAreaAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the subject
area.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a subject area.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="ownedSubjectAreaPackage" type="SubjectAreaPackage" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies a 'sub-package' owned by the current static
package.  All classes within the 'sub-packages' are always imported into their parent static
package.  This means the names of all classes within a static package must be unique.
Graphically it represents a grouping of classes that may be represented on a separate
page.</xs:documentation>
                     <xs:documentation>UML: SubjectAreaPackage stereotype</xs:documentation>
                  </xs:annotation>
                  <!-- Todo: Handle varying presentations for wrap-arround boxes and 'fill-in'
shapes.  Also, enforce rules that contained elements don't appear for fill-in shapes unless they
appear in two. -->
               </xs:element>
               <xs:element name="ownedClass" type="LocalClassRef" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Classes that are part of the subject
area</xs:documentation>
                     <xs:documentation>UML: Classes accessed by the
SubjectAreaPackage</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <!-- Todo: A subject area must either own classes or other subject areas -->
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

```
   <xs:complexType name="LocalClassRef">
      <xs:annotation>
         <xs:documentation>UML: A reference to a class imported into the current
package</xs:documentation>
      </xs:annotation>
      <xs:attribute name="name" type="AllClassName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the class</xs:documentation>
            <xs:documentation>UML: The name of the referenced Class</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="SubjectAreaAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to a subject area</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
subject area.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Description" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Walkthrough" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StaticModelAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to a static model</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
static model.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Description" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Walkthrough" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
               <xs:group ref="Appendix" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StaticModelDerivation">
      <xs:annotation>
         <xs:documentation>UML: A Derivation stereotype to a StaticModel</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Derivation">
            <xs:sequence>
               <xs:element name="targetStaticModel" type="PackageRef">
                  <xs:annotation>
                     <xs:documentation>The package name of the static model from which the
current model is derived.</xs:documentation>
                     <xs:documentation>UML: Absolute package path (using package names)to the
supplying model on the derivationSupplier dependency.</xs:documentation>
```

```
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="staticModelDerivationId" type="BasicId" use="required">
               <xs:annotation>
                  <xs:documentation>A unique identifier for the static model derivation for
reference by other derivations within the model.</xs:documentation>
                  <xs:documentation>UML: No mapping.  This is internal XML shorthand to avoid
copying the package reference each time a class, association or element is
derived.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="EntryPointBase">
      <xs:annotation>
         <xs:documentation>UML: Stereotype of Interface</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate entry-point constraints">
               <sch:rule context="mif:entryPoint">
                  <sch:report test="parent::mif:staticModel/@serializable='true' and
count(mif:annotations)!=0">
                   ERROR: Annotations are only permitted on entry points for non-serializable
models.  (For serializable models, the entry point and model are 1..1)</sch:report>
                  <sch:report test="parent::mif:staticModel/@serializable='true' and
count(mif:businessName/@name)!=0">
                   ERROR: Names are only permitted on entry points for non-serializable
models.  (For serializable models, the entry point and model are 1..1)</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Interface">
            <xs:sequence>
               <xs:element name="annotations" type="EntryPointAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with an entry point.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="graphicRepresentation"
type="NodeWithConnectionSemanticModelBridge" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>This defines information about how the entry point is
represented when displayed graphically.</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to
SemanticModelBridge</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <!-- Todo: Add 'what are your root classes?' method -->
            <xs:attribute name="useKind" type="StaticModelUseKind" use="optional">
               <!-- Todo: Make this repeatable -->
               <xs:annotation>
                  <xs:documentation>Identifies the type of content represented by the model when
entered from this entry point.  The contentType determines whether the model is legitimate
content for a classStub from another model.</xs:documentation>
                  <xs:documentation>UML: Corresponds to the name (inherited from ModelElement) of
the interface being implemented</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="id" type="BasicId" use="optional">
               <xs:annotation>
```

```
                  <xs:documentation>Deprecated: The 'old'-style identifier associated with the
model that is tied to the entry-point</xs:documentation>
                  <xs:documentation>UML: Would be an attribute on the stereotype, but it's going
away soon.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="name" type="BasicFormalName" use="optional">
               <!-- Do we need this?  Can we get rid of it? -->
               <xs:annotation>
                  <xs:documentation>The descriptive name associated with the entry-
point</xs:documentation>
                  <xs:documentation>UML: Would be an attribute on the stereotype, but it's going
away soon.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="EntryPointAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to an Entry Point</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with an
Entry Point.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Description" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassifierBase" abstract="true">
      <xs:annotation>
         <xs:documentation>Common content shared by classes and class-
interfaces</xs:documentation>
         <xs:documentation>UML: A restriction on Classifier</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate common class constraints">
               <sch:rule context="mif:class|mif:classTypeRef|mif:classStub">
                  <sch:report test="@sortKey and
count(ancestor::mif:staticModel/descendant::mif:*[self::mif:class or self::mif:classTypeRef or
self::mif:classStub][@sortKey=current()/@sortKey])!=1">
                     ERROR: Sequence number must be unique across all classes in a
model.</sch:report>
                  <sch:report test="not(@sortKey) and
count(ancestor::mif:staticModel/descendant::mif:*[self::mif:class or self::mif:classTypeRef or
self::mif:classStub][@sortKey])!=0">
                     ERROR: Sequence number must present on all classes in a model if it is
present in any.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Classifier">
            <xs:sequence>
               <xs:element name="derivationSupplier" type="ClassDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the corresponding class in a model from which
the current model has been derived.</xs:documentation>
```

```
                     <xs:documentation>UML: derivationSupplier relationship</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Check class derivationSupplier constraints">
                           <sch:rule context="mif:derivationSupplier[@className]">
                              <sch:report test="@className!=parent::mif:*/@name and
count(ancestor::mif:model//mif:derivationSupplier[@className=current()/@className and
@modelId=current()/@modelId and not(@attributeName or @stateName or @stateTransitionName)])=1">
                                 GUIDELINE: A class derivation must reference the same name as
the derived-from class *unless* there are multiple classes derived from the same model
class.</sch:report>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
               <xs:element name="supplierStructuralDomain" type="DomainSpecification"
minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>For classes whose type heirarchy is extended through
vocabulary, this identifies the concept that corresponds to this physical
class.</xs:documentation>
                     <xs:documentation>UML: supplier association to a StructuralDomain dependency
to a vocabulary reference</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassDerivation">
      <xs:annotation>
         <xs:documentation>UML: A Derivation stereotype to a Class</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ClassDerivation type">
               <sch:rule abstract="true" id="ClassDerivation">
                  <sch:report
test="count(ancestor::mif:*/mif:derivationSupplier[@staticModelDerivationId=current()/@staticMode
lDerivationId])=0">
                   ERROR: Current derivation refers to a dervationId that is not found on the
parent static model.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="staticModelDerivationId" type="BasicId" use="required">
         <!-- Todo: Enforce that this must match a staticModelDerivationId on one of the ancestor
static model's derivations -->
         <xs:annotation>
            <xs:documentation>Refers to the staticModelDerivationId on the parent static model
which points to the model in which the derived class is found.</xs:documentation>
            <xs:documentation>UML: This is a shortcut to an absolute packageref for the SubSystem
in which the derived class is found.  The shortcut is resolved by looking on the current class's
parent static model, and finding a derivation with a matching
derivationModelId.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="className" type="AllClassName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the corresponding class in the 'parent'
model.</xs:documentation>
            <xs:documentation>UML: derivationSupplier supplier</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="ClassRoot">
      <xs:annotation>
         <xs:documentation>UML: Represents all classes that will be
'displayed'</xs:documentation>
```

```
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassifierBase">
            <xs:sequence>
               <xs:element name="graphicRepresentation" type="ClassNodeSemanticModelBridge"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the shape information corresponding with the
class.  In the case of multiple ocurrences, the first repetition of a particular
representationType is the primary shape, and the remainder are shadows.</xs:documentation>
                     <xs:documentation>UML: ClassPresentation</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Check class presentation constraints">
                           <sch:rule context="mif:presentation[@shapeId]">
                              <sch:report
test="count(ancestor::mif:staticModel//mif:presentation[@shapeId=current()/@shapeId])!=1">
                                 ERROR: Only one shape within a model can have a given
shapeId.</sch:report>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
                  <!-- Todo: Handle varying presentations for shadows, and enforce that there is
only one non-shadow in a given diagram -->
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassNodeSemanticModelBridge">
      <!-- Todo ensure that graphicRepresentation is not populated for UML diagrams -->
      <xs:annotation>
         <xs:documentation>The graphic representation for a UML concept displayed as a single
node</xs:documentation>
         <xs:documentation>UML: Stereotype restricting SemanticModelBridge to have
graphicRepresentations associated with classes</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="NodeSemanticModelBridge">
            <xs:attribute name="presentation" type="ClassPresentationKind" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates the diagramming format used to display the
element.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassBase" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to 'Class'</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Check class attributes and elements">
               <sch:rule context="mif:class">
                  <sch:report test="@abstract='true' and count(mif:*)=0">
                      ERROR: Abstract classes must have descendants.</sch:report>
                  <sch:report test="count(ancestor::mif:staticModel/mif:subjectArea)!=0 and
not(mif:primarySubjectArea)">
                      ERROR: PrimarySubjectArea is required if a model contains
subjectAreas.</sch:report>
                  <sch:report test="count(mif:attribute)=0 and count(mif:associations)=0 and
count(mif:*)=0">
                      WARNING: Classes should have at least one attribute, association or
descendant.</sch:report>
                  <sch:report test="contains(@name, '_') and count(mif:*)=0">
                      WARNING: Class names should only contain '_' if they have
descendants.</sch:report>
               </sch:rule>
```

```
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRoot">
            <xs:sequence>
               <xs:element name="annotations" type="ClassAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a class.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="behavior" type="StateMachine" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Defines the set of available states and transitions
available for the class.</xs:documentation>
                     <xs:documentation>UML: StateMachine that is the 'behavior' for the Class.
HL7 allows a maximum of one stateMachine per class.  Note: The 'top' state is implicit and never
modeled.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="stewardCommittee" type="CommitteeReference" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the group with which this class is
predominantly associated.</xs:documentation>
                     <xs:documentation>UML: Reference to tagged element on an ancestor package in
which the class is defined.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="interestedCommittee" type="CommitteeReference" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the non-predominant group(s) with which this
class is also associated.</xs:documentation>
                     <xs:documentation>UML: Reference to tagged element on an ancestor package in
which the class is defined.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="attribute" type="Attribute" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>An independently modifiable or static characteristic of a
class.</xs:documentation>
                     <xs:documentation>UML: Attribute that is a feature of the
class</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Check codingStrength and default attributes.">
                           <sch:rule context="mif:attribute">
                              <sch:extends rule="codeValidation"/>
                           </sch:rule>
                        </sch:pattern>
                        <sch:pattern name="Check for attribute/association duplication.">
                           <sch:rule context="mif:attribute">
                              <sch:report test="count(following-
sibling::mif:associations/mif:association[@name=current()/@name])!=0">
                                 ERROR: May not have attribute with the same name as a
association in a single class.</sch:report>
                           </sch:rule>
                        </sch:pattern>
                        <sch:pattern name="Check attribute derivations">
                           <sch:rule context="mif:attribute">
                              <sch:report
test="mif:derivationSupplier[@attributeName!=current()/@attributeName]">
                                 GUIDELINE: Attribute names must be the same as the names of
the attribute being derived from.</sch:report>
```

```
                        </sch:rule>
                     </sch:pattern>
                  </xs:appinfo>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="name" type="FormalProperName" use="required">
            <xs:annotation>
               <xs:documentation>The unique, formal name for the class within the
model.</xs:documentation>
               <xs:documentation>UML: ModelElement.name</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:attribute name="deprecatedFixedName" type="ShortDescriptiveName" use="optional">
            <xs:annotation>
               <xs:documentation>Added to provide bridge for migration of HL7 Visio-based
tools to MIF-based design files. Attribute is deprecated for ANY use, except to carry shape
property LegacyName for shapes in existing Visio files. Attribute will be dropped once tools no
longer rerquire it.</xs:documentation>
               <xs:documentation>UML: Tag on the Class stereotype.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:attribute name="deprecatedLegacyName" type="ShortDescriptiveName" use="optional">
            <xs:annotation>
               <xs:documentation>Added to provide bridge for migration of HL7 Visio-based
tools to MIF-based design files. Attribute is deprecated for ANY use, except to carry shape
property LegacyName for shapes in existing Visio files. Attribute will be dropped once tools no
longer rerquire it.</xs:documentation>
               <xs:documentation>UML: Tag on the Class stereotype.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:attribute name="deprecatedNameExtension" type="ShortDescriptiveName"
use="optional">
            <xs:annotation>
               <xs:documentation>Added to provide bridge for migration of HL7 Visio-based
tools to MIF-based design files. Attribute is deprecated for ANY use, except to carry shape
property NameExtension for shapes in existing Visio files. Attribute will be dropped once tools
no longer rerquire it.</xs:documentation>
               <xs:documentation>UML: Tag on the Class stereotype.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:attribute name="deprecatedNameStatus" type="EnumerationValue" use="optional">
            <xs:annotation>
               <xs:documentation>Added to provide bridge for migration of HL7 Visio-based
tools to MIF-based design files. Attribute is deprecated for ANY use, except to carry shape
property NameStatus for shapes in existing Visio files. Attribute will be dropped once tools no
longer rerquire it.</xs:documentation>
               <xs:documentation>UML: Tag on the Class stereotype.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="CommitteeReference">
   <xs:annotation>
      <xs:documentation>UML: part of a complex tag</xs:documentation>
   </xs:annotation>
   <xs:attribute name="id" type="BasicId" use="required">
      <xs:annotation>
         <xs:documentation>The identifier assigned to the committee</xs:documentation>
         <xs:documentation>UML: part of a complex tag</xs:documentation>
      </xs:annotation>
   </xs:attribute>
   <xs:attribute name="name" type="ShortDescriptiveName" use="optional">
      <!-- Consider removing this once we have a place to store committee definitions -->
      <xs:annotation>
         <xs:documentation>The name of the committee</xs:documentation>
         <xs:documentation>UML: part of a complex tag</xs:documentation>
```

```
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="ClassAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to a class</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
class.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Definition" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Constraint" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StaticModelClassTemplateParameter">
      <xs:annotation>
         <xs:documentation>A class 'stub' that will be bound to another model of the appropriate
type at runtime</xs:documentation>
         <xs:documentation>UML: A TemplateParameter connected to a static
model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRoot">
            <xs:attribute name="name" type="FormalProperName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the stub.</xs:documentation>
                  <xs:documentation>UML: Name tag on TemplateParameter
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CommonModelElementRefBase">
      <xs:annotation>
         <xs:documentation>A reference to a CMET (possibly binding CMET
parameters)</xs:documentation>
         <xs:documentation>UML: SubSystem referencing a particular CommonModelElement
Interface</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Check classTypeRef information">
               <sch:rule context="mif:classTypeRef">
                  <sch:report
test="count(ancestor::mif:staticModel//mif:classTypeRef[@typeId=current()/@typeId])!=1">
                              ERROR: Only one classTypeRef may exist in an staticModel with a
given typeId.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRoot">
            <xs:sequence>
               <xs:element name="annotations" type="ClassAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
```

```
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a class.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="PrefixedUpperCamelCase" use="optional">
               <!-- Todo: Make this required if there are bound parameters, not permitted
otherwise -->
               <!-- Todo: Confirm the 'type' of this attribute (may need to allow for multiple
underscores - and add to AllClassName) -->
               <xs:annotation>
                  <xs:documentation>The name of the CMET reference.</xs:documentation>
                  <xs:documentation>UML: Inherited from ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CommonModelElementRefAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to a common model element
reference</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
common model element reference.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Constraint" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CommonModelElementGeneralizationBase">
      <xs:annotation>
         <xs:documentation>UML: the Generalization for a CommonModelElement
stereotype</xs:documentation>
      </xs:annotation>
      <xs:attribute name="name" type="AllClassName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the CMET being referenced.</xs:documentation>
            <xs:documentation>UML: The name of the parent interface.  If there are binding
parameters, this is also the name of the target for the binding</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="ClassBindingArgumentBase">
      <xs:annotation>
         <xs:documentation>UML: A binding to a parameterized Static Model.</xs:documentation>
      </xs:annotation>
      <xs:attribute name="templateParameterName" type="FormalProperName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the argument being bound.</xs:documentation>
            <xs:documentation>UML: Used to identify which parameter is being bound (because
sequence is not used)</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="StateMachine">
      <xs:annotation>
```

```
         <xs:documentation>UML: StateMachine</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="annotations" type="StateMachineAnnotations" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
               <xs:documentation>UML: A collector for the comments and constraints associated
with a state machine.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="figure" type="Image" minOccurs="0">
            <xs:annotation>
               <xs:documentation>A graphical representation of the states and state transitions
in a state engine.</xs:documentation>
               <xs:documentation>UML: Tag on StateMachine stereotype</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="subState" type="State" minOccurs="2" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Identifies a 'mode' in which instantiations of the class can
exist.</xs:documentation>
               <xs:documentation>UML: The subVertexes of the 'Top' state (which we don't bother
to model)</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Check state information">
                     <sch:rule context="mif:state">
                        <sch:report test="mif:derivationSupplier[@stateName!=current()/@name]">
                                             GUIDELINE: State names must be the same as the state
names of the state being derived from.</sch:report>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="transition" type="Transition" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Defines a permitted movement between states of a class object.
States may only shift along defined states.</xs:documentation>
               <xs:documentation>UML: Transition</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Check state information">
                     <sch:rule context="mif:transition">
                        <sch:report
test="mif:derivationSupplier[@stateTransitionName!=current()/@name]">
                                             GUIDELINE: State transition names must be the same as
the state names of the state being derived from.</sch:report>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="supplierStateAttributeName" type="FormalPropertyName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the class attribute which represent's the state of the
class (and whose value set is drawn from the list of states in the class state
engine.)</xs:documentation>
            <xs:documentation>UML: A the name of the target 'StateAttribute' dependency between
the attribute and the stateMachine</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="StateMachineAnnotations">
      <xs:annotation>
         <xs:documentation>Descriptive information about the containing state
machine.</xs:documentation>
```

```
         <xs:documentation>UML: A collector for the comments and constraints associated with the
state machine.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Description" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Walkthrough" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Vertex" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to 'Vertex'</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ModelElement"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="State">
      <xs:annotation>
         <xs:documentation>UML: Represents both SimpleStates (no states point to this state as
parent) and CompositeStates (at least one state points to this state as
parent)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Vertex">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
               <xs:element name="annotations" type="StateAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a state.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="derivationSupplier" type="StateDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the corresponding state in a model from which
the current model has been derived.</xs:documentation>
                     <xs:documentation>UML: A derivationSupplier dependency associated with the
state</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The formal name for the state.</xs:documentation>
                  <xs:documentation>UML: Inherited from ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="parentStateName" type="FormalPropertyName" use="optional">
               <xs:annotation>
                  <xs:documentation>Identifies the name of the state of which this is a 'sub-
state'</xs:documentation>
                  <xs:documentation>UML: Name of the state along the 'container' association to
the CompositeState which contains this one.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
```

```
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StateAnnotations">
      <xs:annotation>
         <xs:documentation>Descriptive information about the containing State.</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
State.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Definition" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Constraint" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StateDerivation">
      <xs:annotation>
         <xs:documentation>UML: A Derivation stereotype to a State</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassDerivation">
            <xs:attribute name="stateName" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the corresponding state in the 'parent'
model.</xs:documentation>
                  <xs:documentation>UML: Name of the supplier state</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Transition">
      <xs:annotation>
         <xs:documentation>UML: Transition</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ModelElement">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
               <xs:element name="annotations" type="TransitionAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a transition.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="derivationSupplier" type="TransitionDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the corresponding state transition in a model
from which the current model has been derived.</xs:documentation>
                     <xs:documentation>UML: A derivationSupplier dependency associated with the
transition</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="FormalPropertyName" use="required">
```

```
                <xs:annotation>
                   <xs:documentation>The formal name for the transition.</xs:documentation>
                   <xs:documentation>UML: inherited from ModelElement</xs:documentation>
                </xs:annotation>
             </xs:attribute>
             <xs:attribute name="startStateName" type="FormalPropertyName" use="required">
                <xs:annotation>
                   <xs:documentation>The name of the original state which is transitioned
from.</xs:documentation>
                   <xs:documentation>UML: Name of the source StateVertex</xs:documentation>
                </xs:annotation>
             </xs:attribute>
             <xs:attribute name="endStateName" type="FormalPropertyName" use="required">
                <xs:annotation>
                   <xs:documentation>The name of the target state which is transitioned to.  (It
is possible for the start and end state to bethe same.</xs:documentation>
                   <xs:documentation>UML: Name of the target StateVertex</xs:documentation>
                </xs:annotation>
             </xs:attribute>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TransitionAnnotations">
       <xs:annotation>
          <xs:documentation>Descriptive information about the containing
transition.</xs:documentation>
          <xs:documentation>UML: A collector for the comments and constraints associated with a
transition.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
       </xs:annotation>
       <xs:complexContent>
          <xs:restriction base="Annotations">
             <xs:sequence>
                <xs:group ref="Definition" minOccurs="0"/>
                <xs:group ref="UsageNotes" minOccurs="0"/>
                <xs:group ref="Rationale" minOccurs="0"/>
                <xs:group ref="DesignComments" minOccurs="0"/>
                <xs:group ref="Mapping" minOccurs="0"/>
                <xs:group ref="Walkthrough" minOccurs="0"/>
                <xs:group ref="BallotComment" minOccurs="0"/>
                <xs:group ref="OtherAnnotation" minOccurs="0"/>
             </xs:sequence>
          </xs:restriction>
       </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TransitionDerivation">
       <xs:annotation>
          <xs:documentation>UML: A Derivation stereotype to a Transition</xs:documentation>
       </xs:annotation>
       <xs:complexContent>
          <xs:extension base="ClassDerivation">
             <xs:attribute name="startStateName" type="FormalPropertyName" use="required">
                <xs:annotation>
                   <xs:documentation>The name of the corresponding start state in the 'parent'
model.</xs:documentation>
                   <xs:documentation>UML: Name of the supplier transition's source
StateVertex</xs:documentation>
                </xs:annotation>
             </xs:attribute>
             <xs:attribute name="stateTransitionName" type="FormalPropertyName" use="required">
                <xs:annotation>
                   <xs:documentation>The name of the corresponding state transition in the
'parent' model.</xs:documentation>
                   <xs:documentation>UML: Name of the supplier transition's target
StateVertex</xs:documentation>
                </xs:annotation>
             </xs:attribute>
          </xs:extension>
```

```
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StructuralFeature" abstract="true">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to StructuralFeature</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Feature">
            <xs:attributeGroup ref="MultiplicityRange"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Attribute">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Attribute</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="StructuralFeature">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
               <xs:element name="annotations" type="AttributeAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with an attribute.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="derivationSupplier" type="AttributeDerivation" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the corresponding attribute in a model from
which the current model has been derived.</xs:documentation>
                     <xs:documentation>UML: A derivationSupplier dependency associated with the
attribute</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="type" type="DatatypeRef">
                  <xs:annotation>
                     <xs:documentation>Identifies the structure that may be used to convey the
information in an attribute.</xs:documentation>
                     <xs:documentation>UML: The 'type' association of the structural
feature.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="supplierDomainSpecification"
type="DomainSpecificationWithStrength" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>References the HL7 vocabulary to define the set of allowed
values that may be conveyed by this attribute.</xs:documentation>
                     <xs:documentation>UML: DomainSpecification dependency between to the domain
for the attribute</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The unique formal name used to identify the attribute within
the class and its ancestors.</xs:documentation>
                  <xs:documentation>UML: Inherited from ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AttributeAnnotations">
      <xs:annotation>
```

```
         <xs:documentation>Descriptive information about the containing
attribute.</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with an
Attribute.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Definition" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Constraint" minOccurs="0"/>
               <xs:group ref="StaticExample" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AttributeDerivation">
      <xs:annotation>
         <xs:documentation>UML: A Derivation stereotype to an Attribute</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassDerivation">
            <xs:attribute name="attributeName" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the attribute in the corresponding
model.</xs:documentation>
                  <xs:documentation>UML: The name of the sourcing attribute</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationBase">
      <xs:annotation>
         <xs:documentation>UML: Corresponds with Association</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Relationship">
            <xs:sequence>
               <xs:element name="graphicRepresentation" type="GraphEdgeSemanticModelBridge"
minOccurs="0" maxOccurs="2">
                  <xs:annotation>
                     <xs:documentation>Indicates the display shape(s) associated with the static
package</xs:documentation>
                     <xs:documentation>UML: association from ModelElement to SemanticModelBridge
for an association end</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="annotations" type="AssociationAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with an attribute.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationAnnotations">
```

```
      <xs:annotation>
         <xs:documentation>Descriptive information about the containing
association.</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with an
Association.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Definition" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationEndBase">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to AssociationEnd</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Check association information in a class">
               <sch:rule context="mif:association">
                  <sch:report test="ancestor::mif:staticModel/@type='RIM' and
not(@linkAssociationName)">
                     GUIDELINE: RIM models must not have blocked associations
(linkAssociationName must always be specified).</sch:report>
                  <sch:report test="@name=@linkAssociationName">
                     ERROR: The name of a association and the association linked to must not be
the same.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Relationship">
            <xs:sequence>
               <xs:group ref="BusinessName"/>
               <xs:element name="annotations" type="AssociationEndAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a association end.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="derivationSupplier" type="AssociationEndDerivation"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the corresponding association in a model from
which the current model has been derived.</xs:documentation>
                     <xs:documentation>UML: derivationSupplier dependency</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Check association derivationSupplier constraints">
                           <sch:rule context="mif:derivationSupplier[@associationName]">
                              <sch:report test="@associationName!=parent::mif:*/@name and
count(ancestor::mif:model//mif:derivationSupplier[@className=current()/@className and
@associationName=current()/@associationName and @modelId=current()/@modelId])=1">
                                 GUIDELINE: An association derivation must reference the same
name as the derived-from association *unless* there are multiple classes derived from the same
model class.</sch:report>
                           </sch:rule>
                        </sch:pattern>
```

```
                    </xs:appinfo>
                 </xs:annotation>
              </xs:element>
              <xs:element name="participantClassSpecialization"
type="AssociationEndSpecialization" minOccurs="0" maxOccurs="unbounded">
                 <xs:annotation>
                    <xs:documentation>For association ends pointing to CMETs whose root is a
choice, identifies the classes within the choice and the association names tied to those
classes.</xs:documentation>
                    <xs:documentation>UML: Identifies classes that specialize the paricipant
class for this association end</xs:documentation>
                 </xs:annotation>
              </xs:element>
           </xs:sequence>
           <xs:attribute name="name" type="FormalPropertyName" use="required">
              <xs:annotation>
                 <xs:documentation>The unique formal name for the
association.</xs:documentation>
                 <xs:documentation>UML: Inherited from ModelElement</xs:documentation>
              </xs:annotation>
           </xs:attribute>
           <!--            <xs:attribute name="isAggregate" type="xs:boolean" use="optional"
default="false">
              <xs:annotation>
                 <xs:documentation>Indicates whether this is the 'aggregate' end of an
aggregation association.  (The aggregation is a composition if maxOccurs='1'.)</xs:documentation>
                 <xs:documentation>UML: aggregation</xs:documentation>
              </xs:annotation>
           </xs:attribute>-->
           <xs:attributeGroup ref="MultiplicityRange"/>
           <xs:attributeGroup ref="Presence"/>
           <xs:attributeGroup ref="UpdateMode"/>
        </xs:extension>
     </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="AssociationEndSpecialization">
     <xs:annotation>
        <xs:documentation>UML: Identifies a class that specialize the paricipant class for this
association end</xs:documentation>
     </xs:annotation>
     <xs:sequence>
        <xs:element name="specialization" type="AssociationEndSpecialization" minOccurs="0"
maxOccurs="unbounded">
           <xs:annotation>
              <xs:documentation>For specializations that are choices or CMETs whose root is a
choice, identifies the classes within the choice and the association names tied to those
classes.</xs:documentation>
              <xs:documentation>UML: Identifies classes that specialize this generalized
class</xs:documentation>
           </xs:annotation>
        </xs:element>
     </xs:sequence>
     <xs:attribute name="className" type="AllClassName" use="required">
        <xs:annotation>
           <xs:documentation>Name of the class</xs:documentation>
           <xs:documentation>UML: Name of the specializing class</xs:documentation>
        </xs:annotation>
     </xs:attribute>
     <xs:attribute name="traversalName" type="FormalPropertyName" use="optional">
        <xs:annotation>
           <xs:documentation>Name of the element when traversing from the association end
directly to this specialized class</xs:documentation>
           <xs:documentation>UML: tagged element on the specialized class</xs:documentation>
        </xs:annotation>
     </xs:attribute>
     <!-- Todo: enforce that you either have a traversalName or children, not both -->
     <!-- Todo: enforce that specializations correspond to actual specialization classes -->
  </xs:complexType>
```

```
   <xs:complexType name="NonTraversableAssociationEnd">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to an AssociationEnd where 'isTraversable' is
false</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Relationship">
            <xs:sequence>
               <xs:element name="derivationSupplier" type="AssociationEndDerivation"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the corresponding association in a model from
which the current model has been derived.</xs:documentation>
                     <xs:documentation>UML: derivationSupplier dependency</xs:documentation>
                     <xs:appinfo>
                        <sch:pattern name="Check association derivationSupplier constraints">
                           <sch:rule context="mif:derivationSupplier[@associationName]">
                              <sch:report test="@associationName!=parent::mif:*/@name and
count(ancestor::mif:model//mif:derivationSupplier[@className=current()/@className and
@associationName=current()/@associationName and @modelId=current()/@modelId])=1">
                                 GUIDELINE: An association derivation must reference the same
name as the derived-from association *unless* there are multiple classes derived from the same
model class.</sch:report>
                           </sch:rule>
                        </sch:pattern>
                     </xs:appinfo>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="participantClassName" type="AllClassName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the class to which the association end is
attached</xs:documentation>
                  <xs:documentation>UML: The name of the participant class for the association
end</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationEndAnnotations">
      <xs:annotation>
         <xs:documentation>Descriptive information about the containing
association.</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with an
Association.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Definition" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Constraint" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationEndDerivation">
      <xs:annotation>
         <xs:documentation>UML: A Derivation stereotype to an Association</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
```

```
         <xs:extension base="ClassDerivation">
            <xs:attribute name="associationEndName" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the association in the other
model.</xs:documentation>
                  <xs:documentation>UML: Name of 'supplier' association end</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassGeneralizationBase" abstract="true">
      <!-- 19/12/2003, Charlie McCay.  Should use the uniqueness constraints; should be enforced
used schema, not schematron. -->
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Generalization</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Check descendant constraints">
               <sch:rule context="descendant">
                  <sch:report test="preceding-sibling::descendant[@sortKey =
current()/@sortKey]">
                     ERROR: Sequence number must be unique across all descendants in a
model.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Relationship">
            <xs:sequence>
               <xs:element name="annotations" type="GeneralizationAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a Generalization.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="Presence"/>
            <xs:attribute name="traversalName" type="FormalPropertyName" use="optional">
               <xs:annotation>
                  <xs:documentation>Identifies the name when walking the association to the
parent class directly to this child class</xs:documentation>
                  <xs:documentation>UML: Tag value on Generalization</xs:documentation>
               </xs:annotation>
               <!-- Todo: Make this required as soon as tools support it -->
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GeneralizationAnnotations">
      <xs:annotation>
         <xs:documentation>Descriptive information about the containing
generalization.</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
generalization.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
```

```
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CommonModelElement">
      <xs:annotation>
         <xs:documentation>Used to define CMETs.</xs:documentation>
         <xs:documentation>UML: CommonModelElement stereotype on Interface</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Check classTypeRef information">
               <sch:rule context="mif:classTypeRef">
                  <sch:report
test="count(ancestor::mif:staticModel//mif:classTypeRef[@typeId=current()/@typeId])!=1">
                                 ERROR: Only one classTypeRef may exist in an staticModel with a
given typeId.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassifierBase">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="annotations" type="CommonModelElementAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a class.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="specializationChildStaticModel" type="PackageRef">
                  <xs:annotation>
                     <xs:documentation>The modelId of the model that implements this Common Model
Element.</xs:documentation>
                     <xs:documentation>UML: The StaticModel subsystem that specializes this
interface</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="templateParameter" type="ClassTemplateParameter" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies the parameters associated with the CMET that
must be bound if the CMET is referenced.</xs:documentation>
                     <xs:documentation>UML: The TemplateParameters associated with the Interface
(if any)</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="specializationChildEntryClass" type="SpecializationClass"
minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Indicates the name of the root class of the CMET (and any
descendant classes)</xs:documentation>
                     <xs:documentation>UML: The name of the class pointed to by the entry-point
for the static model that specializes this interface</xs:documentation>
                  </xs:annotation>
                  <!-- Todo: Make required as soon as tooling supports it. -->
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="PrefixedUpperCamelCase" use="required">
               <xs:annotation>
                  <xs:documentation>The identifier of the external model.</xs:documentation>
                  <xs:documentation>UML: Inherited from modelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="attributionLevel" type="CMETAttributionKind" use="optional">
               <xs:annotation>
```

```
                  <xs:documentation>Identifies the level of detail associated with the
CMET</xs:documentation>
                  <xs:documentation>UML: Tag on CommonModelElement stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="entryKind" type="CMETEntryKind" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies the means by which the CMET can be
entered</xs:documentation>
                  <xs:documentation>UML: Tag value on CommonModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
      <!-- Todo -->
   </xs:complexType>
   <xs:complexType name="SpecializationClass">
      <xs:annotation>
         <xs:documentation>UML: A class that is a specialization of another class and which may
itself have specializations</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="specializationClass" type="SpecializationClass" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Indicates any classes that specialize the current class (if the
current class is a Choice).</xs:documentation>
               <xs:documentation>UML: Points to classes of which the current class is a
generalization.</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="AllClassName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the class, CMET or Stub</xs:documentation>
            <xs:documentation>UML: The name of the class</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="CommonModelElementAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to a Common Model Element</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
Common Model Element interface.  (Consider rendering the definition or description annotation
into ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Description" minOccurs="0"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassTemplateParameter">
      <xs:annotation>
         <xs:documentation>UML: TemplateParameter for a static model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassifierBase">
            <xs:attribute name="name" type="FormalProperName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the parameter for use in describing the datatype
or in constructing formal constraints.</xs:documentation>
```

```
                  <xs:documentation>UML: Name tag on TemplateParameter
stereotype</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="interface" type="FormalProperName" use="required">
               <xs:annotation>
                  <xs:documentation>Indicates the 'type' associated with the
parameter</xs:documentation>
                  <xs:documentation>UML: Identifies an interface implemented by the parameter
(and thereby restricts the models that can be substituted for the parameter)</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassStubInterface">
      <xs:annotation>
         <xs:documentation>Defines an allowed 'kind' for a stub within a static model.  Similar
to CMETs, but not bound to a single model</xs:documentation>
         <xs:documentation>UML: An interface that defines the allowed content types for class
TemplateParameters within static models.  Note: Inherit from ClassRoot instead of Interface for
simplicity</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassifierBase">
            <xs:sequence>
               <xs:element name="annotations" type="ClassStubInterfaceAnnotations" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive information about the containing
element.</xs:documentation>
                     <xs:documentation>UML: A collector for the comments and constraints
associated with a class.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="StaticModelUseKind" use="required">
               <xs:annotation>
                  <xs:documentation>The type of content that may be included at this point.  The
sub-model must be rooted in an entry-point having an identical contentType.</xs:documentation>
                  <xs:documentation>UML: Inherited from ModelElement</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassStubInterfaceAnnotations">
      <xs:annotation>
         <xs:documentation>Comments relating to a class stub type</xs:documentation>
         <xs:documentation>UML: A collector for the comments and constraints associated with a
Class Stub interface.  (Consider rendering the definition or description annotation into
ModelElement.documentation)</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:restriction base="Annotations">
            <xs:sequence>
               <xs:group ref="Description"/>
               <xs:group ref="UsageNotes" minOccurs="0"/>
               <xs:group ref="Rationale" minOccurs="0"/>
               <xs:group ref="DesignComments" minOccurs="0"/>
               <xs:group ref="Mapping" minOccurs="0"/>
               <xs:group ref="Constraint" minOccurs="0"/>
               <xs:group ref="BallotComment" minOccurs="0"/>
               <xs:group ref="OtherAnnotation" minOccurs="0"/>
            </xs:sequence>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
```

```
</xs:schema>
```

C.5 mifStaticModelFlat

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- mifStaticModelFlat -->
<?xml-stylesheet type="text/xsl" href="C:\Documents and Settings\Administrator\My
Documents\Eclipse\workspace\v3Schemas\uml\transforms\generateExamples.xsl"?>
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns="urn:hl7-org:v3/mif"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sch="http://www.ascc.net/xml/schematron"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
******************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   This schema provides a flat (non-serialized) view of a static data model
******************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifStaticModelBase.xsd"/>
   <xs:element name="staticModels">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="staticModel" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="staticModel" type="StaticModel">
      <xs:annotation>
         <xs:documentation>A static Model in standard (flat) notation.  Used for RIM, CIM, LIM,
etc.</xs:documentation>
         <xs:documentation>UML: An instance of a StaticModel stereotype</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="StaticModel">
      <xs:complexContent>
         <xs:extension base="StaticModelBase">
            <xs:sequence>
               <xs:element name="ownedEntryPoint" type="EntryPoint" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies a class within the model that may be used as
the initial class in a serializable representation of the model.</xs:documentation>
                     <xs:documentation>UML: An interface that is implemented by the
SubSystem</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="ownedClass" type="ClassElement" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>The classes that are part of the model</xs:documentation>
                     <xs:documentation>UML: A class owned by the package</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="ownedAssociation" type="Association" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>The associations that are part of the static
model</xs:documentation>
                     <xs:documentation>UML: The associations that are part of the static
model</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
```

```
   </xs:complexType>
   <xs:complexType name="EntryPoint">
      <xs:annotation>
         <xs:documentation>EntryPoint stereotype for a 'flat' model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="EntryPointBase">
            <xs:attribute name="className" type="AllClassName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the class that is the entry-
point</xs:documentation>
                  <xs:documentation>UML: The name of the class that specializes the EntryPoint
interface</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassElement">
      <xs:annotation>
         <xs:appinfo>
            <sch:ns uri="urn:hl7-org:v3/mif" prefix="mif"/>
            <sch:pattern name="Validate class constraints">
               <sch:rule context="mif:class|mif:classTypeRef|mif:classStub">
                  <sch:report
test="count(ancestor::mif:staticModel/descendant::mif:*[@name=current()/@name and
(self::mif:class or self::mif:classTypeRef or self::mif:classStub)])&gt;1">
                     ERROR: Only one class, classTypeRef or classStub may exist with a given
name inside an staticModel.</sch:report>
                  <sch:report
test="count(ancestor::mif:staticModel/descendant::mif:*[mif:businessName/@name=current()/mif:busi
nessName/@name and (self::mif:class or self::mif:classTypeRef or self::mif:classStub)])&gt;1">
                     ERROR: Only one class, classTypeRef or classStub may exist with a given
business name inside an staticModel.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:choice>
         <xs:element name="class" type="Class">
            <xs:annotation>
               <xs:documentation>A set of attributes and associations representing a single
instance.</xs:documentation>
               <xs:documentation>UML: Class</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="commonModelElementRef" type="CommonModelElementRef">
            <xs:annotation>
               <xs:documentation>A reference to an external model intended to be imported and re-
used at this point.</xs:documentation>
               <xs:documentation>UML: Another subsystem that is referenced via an
interface</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="templateParameter" type="StaticModelClassTemplateParameter">
            <xs:annotation>
               <xs:documentation>A point in the model at which a 'sub-model' may be placed that
corresponds to the identified contentType.  The specific model to be included may vary and is
determined at runtime.</xs:documentation>
               <xs:documentation>UML: A Class that represents one of the Template Parameters of a
template Static Model</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="Class">
      <xs:annotation>
         <xs:documentation>UML: Class within a 'flat' model</xs:documentation>
```

```
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassBase">
            <xs:sequence>
               <xs:element name="specializationChild" type="ClassGeneralization" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies classes that are descended from the current
class (as part of a generalization-specialization hierarchy).</xs:documentation>
                     <xs:documentation>UML: The children of the Generalization of this
class</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassGeneralization">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Generalization for a 'flat'
model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassGeneralizationBase">
            <xs:attribute name="childClassName" use="required">
               <xs:annotation>
                  <xs:documentation>UML: Used to reference the Class that is the child of the
generalization</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CommonModelElementRef">
      <xs:annotation>
         <xs:documentation>UML: SubSystem referencing a particular CommonModelElement Interface
in a 'flat' static model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="CommonModelElementRefBase">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="generalizationParent" type="CommonModelElementGeneralization">
                  <xs:annotation>
                     <xs:documentation>Reference the CMET definition being implemented, plus
parameters if appropriate</xs:documentation>
                     <xs:documentation>UML: The interface that is specialized (implemented) by
this subsystem</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CommonModelElementGeneralization">
      <xs:annotation>
         <xs:documentation>UML: the Generalization for a CommonModelElement stereotype in a
'flat' static model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="CommonModelElementGeneralizationBase">
            <xs:sequence>
               <xs:element name="supplierBindingArgument" type="ClassBindingArgument"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies a the classes that are to be 'bound' to a
parametericed Common Model Element</xs:documentation>
                     <xs:documentation>UML: Indicates the arguments of the supplier binding
dependency for the Common Model Element</xs:documentation>
```

```
                </xs:annotation>
             </xs:element>
          </xs:sequence>
       </xs:extension>
    </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="ClassBindingArgument">
    <xs:annotation>
       <xs:documentation>UML: A binding to a parameterized flat Static Model</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
       <xs:extension base="ClassBindingArgumentBase">
          <xs:attribute name="className" type="AllClassName" use="required">
             <xs:annotation>
                <xs:documentation>UML: A reference to the class being bound</xs:documentation>
             </xs:annotation>
          </xs:attribute>
       </xs:extension>
    </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="Association">
    <xs:annotation>
       <xs:documentation>A relationship between two classes</xs:documentation>
       <xs:documentation>UML: Association</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
       <xs:extension base="AssociationBase">
          <xs:sequence>
             <xs:element name="connections" type="AssociationEnds">
                <xs:annotation>
                   <xs:documentation>Describes each end of the association</xs:documentation>
                   <xs:documentation>UML: The AssociationEnds tied to the
Association</xs:documentation>
                </xs:annotation>
             </xs:element>
          </xs:sequence>
       </xs:extension>
    </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="AssociationEnds">
    <xs:annotation>
       <xs:documentation>UML: Handles associations that can be traversable or
not.</xs:documentation>
    </xs:annotation>
    <xs:choice minOccurs="2" maxOccurs="2">
       <xs:element name="traversableConnection" type="AssociationEnd">
          <xs:annotation>
             <xs:documentation>The ends of the association</xs:documentation>
             <xs:documentation>UML: The connection association for each association
end</xs:documentation>
          </xs:annotation>
       </xs:element>
       <xs:element name="nonTraversableConnection" type="NonTraversableAssociationEnd">
          <xs:annotation>
             <xs:documentation>The ends of the association</xs:documentation>
             <xs:documentation>UML: The connection association for each association
end</xs:documentation>
          </xs:annotation>
       </xs:element>
    </xs:choice>
 </xs:complexType>
 <xs:complexType name="AssociationEnd">
    <xs:annotation>
       <xs:documentation>UML: Corresponds to AssociationEnd</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
       <xs:extension base="AssociationEndBase">
          <xs:attribute name="participantClassName" type="AllClassName" use="required">
```

```
               <xs:annotation>
                  <xs:documentation>The name of the class to which the association end is
attached</xs:documentation>
                  <xs:documentation>UML: The name of the participant class for the association
end</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

C.6 mifStaticModelSerialized

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- mifStaticModelSerialized -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns="urn:hl7-org:v3/mif"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sch="http://www.ascc.net/xml/schematron"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
******************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   This schema provides a serialized (hierarchical) view of a static data model
******************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifStaticModelBase.xsd"/>
   <xs:element name="serializedStaticModels">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="serializedStaticModel" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="serializedStaticModel" type="SerializedStaticModel">
      <xs:annotation>
         <xs:documentation>A static Model in standard (flat) notation.  Used for CIM, LIM,
etc.</xs:documentation>
         <xs:documentation>UML: An instance of a SerializedStaticModel
stereotype</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="SerializedStaticModel">
      <xs:complexContent>
         <xs:extension base="StaticModelBase">
            <xs:sequence>
               <xs:element name="ownedEntryPoint" type="SerializedEntryPoint">
                  <xs:annotation>
                     <xs:documentation>Identifies a class within the model that may be used as
the initial class in a serializable representation of the model.</xs:documentation>
                     <xs:documentation>UML: An interface that is implemented by the
SubSystem</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SerializedEntryPoint">
      <xs:annotation>
         <xs:documentation>EntryPoint stereotype for a 'serialized' model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
```

```
         <xs:extension base="EntryPointBase">
            <xs:sequence>
               <xs:element name="specializedClass" type="SerializedClasses">
                  <xs:annotation>
                     <xs:documentation>The class pointed to by the entry-point</xs:documentation>
                     <xs:documentation>UML: The Class that specializes (implements) the Entry
Point interface</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:group name="SerializedClasses">
      <xs:choice>
         <xs:element name="class" type="SerializedClass">
            <xs:annotation>
               <xs:documentation>A set of attributes and associations representing a single
instance.</xs:documentation>
               <xs:documentation>UML: Class</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="commonModelElementRef" type="SerializedCommonModelElementRef">
            <xs:annotation>
               <xs:documentation>A reference to an external model intended to be imported and re-
used at this point.</xs:documentation>
               <xs:documentation>UML: Another subsystem that is referenced via an
interface</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="templateParameter" type="StaticModelClassTemplateParameter">
            <xs:annotation>
               <xs:documentation>A point in the model at which a 'sub-model' may be placed that
corresponds to the identified contentType.  The specific model to be included may vary and is
determined at runtime.</xs:documentation>
               <xs:documentation>UML: A Class that represents one of the Template Parameters of a
template Static Model</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:group>
   <xs:complexType name="SerializedClasses">
      <xs:annotation>
         <xs:appinfo>
            <sch:pattern name="Validate class constraints">
               <sch:rule context="mif:class|mif:classTypeRef|mif:classStub">
                  <sch:report
test="count(ancestor::mif:staticModel/descendant::mif:*[@name=current()/@name and
(self::mif:class or self::mif:classTypeRef or self::mif:classStub)])&gt;1">
                     ERROR: Only one class, classTypeRef or classStub may exist with a given
name inside an staticModel.</sch:report>
                  <sch:report
test="count(ancestor::mif:staticModel/mif:descendant::mif:*[mif:businessName/@name=current()/mif:
businessName/@name and (self::mif:class or self::mif:classTypeRef or
self::mif:classStub)])&gt;1">
                     ERROR: Only one class, classTypeRef or classStub may exist with a given
business name inside an staticModel.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:group ref="SerializedClasses"/>
   </xs:complexType>
   <xs:complexType name="SerializedClass">
      <xs:annotation>
         <xs:documentation>UML: Class within a 'serialized' model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
```

```
         <xs:extension base="ClassBase">
            <xs:sequence>
               <xs:element name="specializationChild" type="SerializedClassGeneralization"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies classes that are descended from the current
class (as part of a generalization-specialization hierarchy).</xs:documentation>
                     <xs:documentation>UML: The children of the Generalization of this
class</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="association" type="SerializedAssociationEnd" minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>The associations for the class</xs:documentation>
                     <xs:documentation>UML: Association coming from the class</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SerializedClassGeneralization">
      <xs:annotation>
         <xs:documentation>UML: Corresponds to Generalization for a 'serialized'
model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassGeneralizationBase">
            <xs:sequence>
               <xs:element name="specializedClass" type="ClassOrReference">
                  <xs:annotation>
                     <xs:documentation>A class that is a specialization</xs:documentation>
                     <xs:documentation>UML: The class that is the
specialization</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationEndWithClass">
      <xs:annotation>
         <xs:documentation>UML: an AssociationEnd that points to a Class</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="AssociationEndBase">
            <xs:sequence>
               <!-- Todo: Enforce that either element or attribute is present, not both -->
               <xs:element name="participantClass" type="ClassOrReference">
                  <xs:annotation>
                     <xs:documentation>The class to which the association end is
attached</xs:documentation>
                     <xs:documentation>UML: The name of the participant class for the association
end (used for recursion)</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationEndWithClassRef">
      <xs:annotation>
         <xs:documentation>UML: an AssociationEnd that points to a Class</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="AssociationEndBase">
            <xs:attribute name="participantClassName" type="AllClassName" use="required">
```

```
               <xs:annotation>
                  <xs:documentation>The name of the class to which the association end is
attached</xs:documentation>
                  <xs:documentation>UML: The name of the participant class for the association
end (used for recursion)</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassOrReference">
      <xs:annotation>
         <xs:documentation>UML: A full-blown class, or a reference to an already described
class.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:group ref="SerializedClasses"/>
         <xs:element name="reference" type="LocalClassReference" minOccurs="0">
            <xs:annotation>
               <xs:documentation>The class to which the association end is
attached</xs:documentation>
               <xs:documentation>UML: The name of the participant class for the association end
(used for recursion)</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="LocalClassReference">
      <xs:annotation>
         <xs:documentation>A reference (by name) to a class within the same
package</xs:documentation>
      </xs:annotation>
      <xs:attribute name="name" type="FormalProperName" use="optional">
         <xs:annotation>
            <xs:documentation>The name of the class being referenced</xs:documentation>
            <xs:documentation>UML: the name of the class being referenced within the
package</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="SerializedAssociationEnd">
      <xs:annotation>
         <xs:documentation>UML: An Association that points to an association end attached to a
class</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="AssociationBase">
            <xs:sequence>
               <xs:element name="targetConnection" type="AssociationEndWithClass">
                  <xs:annotation>
                     <xs:documentation>The end at the opposite end of the
association</xs:documentation>
                     <xs:documentation>UML: The connection association to the
Association</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="sourceConnection" type="SerializedAssociationEnds">
                  <xs:annotation>
                     <xs:documentation>The details of the association in the opposite
direction.</xs:documentation>
                     <xs:documentation>UML: The associationEnd on the near side of the
association.  Only present if the association is traversable in the opposite
direction</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
```

```
   </xs:complexType>
   <xs:complexType name="SerializedAssociationEnds">
      <xs:annotation>
         <xs:documentation>UML: Handles associations that can be traversable or
not.</xs:documentation>
      </xs:annotation>
      <xs:choice minOccurs="0">
         <xs:element name="traversableConnection" type="AssociationEndWithClassRef"
minOccurs="0">
            <xs:annotation>
               <xs:documentation>A connection that can be traversed.</xs:documentation>
               <xs:documentation>UML: An AssociationEnd that is traversable</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="nonTraversableConnection" type="NonTraversableAssociationEnd">
            <xs:annotation>
               <xs:documentation>A connection that cannot be traversed</xs:documentation>
               <xs:documentation>UML: An AssociationEnd that is not
traversable</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="SerializedCommonModelElementRef">
      <xs:annotation>
         <xs:documentation>UML: SubSystem referencing a particular CommonModelElement Interface
in a 'flat' static model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="CommonModelElementRefBase">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="generalizationParent"
type="SerializedCommonModelElementGeneralization">
                  <xs:annotation>
                     <xs:documentation>Reference the CMET definition being implemented, plus
parameters if appropriate</xs:documentation>
                     <xs:documentation>UML: The interface that is specialized (implemented) by
this subsystem</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SerializedCommonModelElementGeneralization">
      <xs:annotation>
         <xs:documentation>UML: the Generalization for a CommonModelElement stereotype in a
'serialized' static model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="CommonModelElementGeneralizationBase">
            <xs:sequence>
               <xs:element name="supplierBindingArgument" type="SerializedClassBindingArgument"
minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                     <xs:documentation>Identifies a the classes that are to be 'bound' to a
parametericed Common Model Element</xs:documentation>
                     <xs:documentation>UML: Indicates the arguments of the supplier binding
dependency for the Common Model Element</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SerializedClassBindingArgument">
      <xs:annotation>
```

```
         <xs:documentation>UML: A binding to a parameterized serialized Static
Model</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassBindingArgumentBase">
            <xs:sequence>
               <xs:element name="argumentClass" type="ClassOrReference">
                  <xs:annotation>
                     <xs:documentation>The class bound to the argument.</xs:documentation>
                     <xs:documentation>UML: The class that is bound as an
argument</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

C.7 mifPatternTypes

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- mifPatternTypes -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns="urn:hl7-org:v3/mif"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sch="http://www.ascc.net/xml/schematron"
  elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
*******************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
  Defines the various simple type patterns used by mif elements.
  This file may be overridden to 'bypass' some of the enforced rules.

Outstanding questions:
  - Are the lengths and types of various attributes and elements appropriate (particularly
  markup)?
*******************************************************************************
   </xs:documentation>
      <xs:documentation>
      UML: **ALL** types defined in this package are handled as datatypes.
      There is a mapping from some schema datatypes directly to existing UML datatypes:
         xs:boolean -> Boolean
         xs:integer -> Integer
         xs:string -> String
      Most of the types defined here will be new datatypes (some may be restrictions of the
  above).
      Where there is a mapping to an existing UML datatype, this will be documented.
   </xs:documentation>
   </xs:annotation>
   <xs:simpleType name="DateOrTimestamp">
      <xs:annotation>
         <xs:documentation>Used when we might have just a date, or a full-blown timestamp.
  Format is YYYY-MM-DD['T'HH:MM:SS[('-'|'+')ZZ:ZZ]]</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="xs:dateTime xs:date"/>
   </xs:simpleType>
   <xs:simpleType name="Years">
      <xs:annotation>
         <xs:documentation>A list of years</xs:documentation>
      </xs:annotation>
      <xs:list itemType="Year"/>
```

```
 </xs:simpleType>
 <xs:simpleType name="Year">
    <xs:annotation>
       <xs:documentation>A year (restricted to 20th and 21st century)</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:integer">
       <xs:pattern value="(19|20)\d{2}"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="SingleNonNegativeInteger">
    <xs:annotation>
       <xs:documentation>An integer that is greater than or equal to zero, with a maximum
length of 1 character</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:nonNegativeInteger">
       <xs:maxExclusive value="10"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="SmallNonNegativeInteger">
    <xs:annotation>
       <xs:documentation>An integer that is greater than or equal to zero, with a maximum
length of 6 characters</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:nonNegativeInteger">
       <xs:maxExclusive value="1000000"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="SmallPositiveInteger">
    <xs:annotation>
       <xs:documentation>An integer that is greater than or equal to 1, with a maximum length
of 6 characters</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:positiveInteger">
       <xs:maxExclusive value="1000000"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="GraphicMeasurement">
    <xs:annotation>
       <xs:documentation>A real number that is greater than or equal to 0. Used when you want a
positive number that can be fractional.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal">
       <xs:maxExclusive value="1000000"/>
       <xs:fractionDigits value="3"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="PositiveDecimal">
    <xs:annotation>
       <xs:documentation>A real number that is restricted to a maximum of 10 characters, and
may have no more than 3 decimal places.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="GraphicMeasurement">
       <xs:minExclusive value="0"/>
    </xs:restriction>
 </xs:simpleType>
 <!-- Simple types used in defining various attributes -->
 <xs:simpleType name="Uuid">
    <xs:annotation>
       <xs:documentation>Universal Unique Identifier (aka GUID).  Used for
identifiers</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
       <xs:length value="38"/>
       <xs:pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="HashCode">
    <xs:annotation>
```

```
        <xs:documentation>Base64 representation of a 160 bit RSA-1 hashcode</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:pattern value="([A-Za-z0-9/+]){27}="/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="Oid">
    <xs:annotation>
        <xs:documentation>ISO Object Identifier.  Used for identifiers</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
        <xs:maxLength value="255"/>
        <xs:pattern value="[1-9][0-9]*(\.[1-9][0-9]*)*"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="Url">
    <xs:annotation>
        <xs:documentation>Used for hypertext references.  (Must be prefixed by
"http://")</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI">
        <xs:maxLength value="255"/>
        <xs:pattern value="http://.+"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="EMail">
    <xs:annotation>
        <xs:documentation>Used for email references.  (Must be prefixed by
"mailto://")</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI">
        <xs:maxLength value="255"/>
        <xs:pattern value="mailto://.+"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="LocalFileReference">
    <xs:annotation>
        <xs:documentation>Used for file references.  (Must be prefixed by
"file://")</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI">
        <xs:maxLength value="255"/>
        <xs:pattern value="(file://)?([A-Za-z0-9_\-\.]+/)*[A-Za-z0-9_\-\.]+"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="NonEmptyString">
    <xs:annotation>
        <xs:documentation>Used as the base for most string datatypes.  (Ensures that empty
strings are not allowed.).  Should never be implemented directly because it has no length
limits.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
        <xs:minLength value="1"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="LowerCamelCase">
    <xs:annotation>
        <xs:documentation>Type for attribute and association type names.  Should never be
implemented directly because it has no length limits.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:NMTOKEN">
        <xs:pattern value="[a-z][a-z0-9]*([A-Z][a-z0-9]*)*"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="UpperCamelCase">
    <xs:annotation>
        <xs:documentation>Type for class and type names</xs:documentation>
    </xs:annotation>
```

```
      <xs:restriction base="xs:NMTOKEN">
         <xs:pattern value="([A-Z][a-z0-9_]*)+"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="PrefixedUpperCamelCase">
      <xs:annotation>
         <xs:documentation>Used for CMET type names.  (Basically class names with a prefix
separated by an underscore)</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:maxLength value="50"/>
         <xs:pattern value="[A-Z]+_([A-Z][a-z0-9_]*)+"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="BasicFormalName">
      <xs:annotation>
         <xs:documentation>Used for formal names and short descriptions</xs:documentation>
      </xs:annotation>
      <xs:restriction base="NonEmptyString">
         <xs:maxLength value="120"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="FormalProperName">
      <xs:annotation>
         <xs:documentation>Used for class and type names</xs:documentation>
      </xs:annotation>
      <xs:restriction base="UpperCamelCase">
         <xs:maxLength value="50"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="AllClassName">
      <xs:annotation>
         <xs:documentation>Choice of a class name or a CMET name</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="FormalProperName PrefixedUpperCamelCase"/>
   </xs:simpleType>
   <xs:simpleType name="FormalPropertyName">
      <xs:annotation>
         <xs:documentation>Used for attribute, association and property names</xs:documentation>
      </xs:annotation>
      <xs:restriction base="LowerCamelCase">
         <xs:maxLength value="50"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="EnumerationValue">
      <xs:annotation>
         <xs:documentation>Used as the base class for all 'codes'</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:maxLength value="50"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ShortDescriptiveName">
      <xs:annotation>
         <xs:documentation>Used when descriptions are intended to be very
short</xs:documentation>
      </xs:annotation>
      <xs:restriction base="NonEmptyString">
         <xs:maxLength value="80"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="LongDescriptiveName">
      <xs:annotation>
         <xs:documentation>Used when descriptions can be potentially quite
long</xs:documentation>
      </xs:annotation>
      <xs:restriction base="NonEmptyString">
         <xs:maxLength value="255"/>
```

```
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DomainName">
      <xs:annotation>
         <xs:documentation>Used for vocabulary domains that may include 'x-
domains'</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="BasicFormalName XDomainName"/>
   </xs:simpleType>
   <xs:simpleType name="XDomainName">
      <xs:annotation>
         <xs:documentation>Used for x-domains</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:maxLength value="50"/>
         <xs:pattern value="(x_)?([A-Z][a-z0-9_]*)+"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DatatypeName">
      <xs:annotation>
         <xs:documentation>Used for datatype names</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:maxLength value="20"/>
         <xs:pattern value="[A-Z]+|([A-Z]+.)?diff"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="Version">
      <xs:annotation>
         <xs:documentation>Used for version numbers</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:minLength value="1"/>
         <xs:maxLength value="16"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="BasicId">
      <xs:annotation>
         <xs:documentation>Used for generic identifiers</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:minLength value="1"/>
         <xs:maxLength value="40"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:attributeGroup name="MultiplicityRange">
      <xs:annotation>
         <xs:appinfo>
            <sch:pattern name="Validate MultiplicityRange type">
               <sch:rule abstract="true" id="MultiplicityRange">
                  <sch:report test="@minOccurs&gt;@maxOccurs">
                   ERROR: MinOccurs must be less than or equal to maxOccurs.</sch:report>
                  <sch:report test="@mandatory='true' and @conformance!='R'">
                   ERROR: Conformance must be 'R' when 'mandatory' is true.</sch:report>
                  <sch:report test="@mandatory='true' and @minOccurs=0">
                   ERROR: MinOccurs must be at least 1 when 'mandatory' is true.</sch:report>
                  <sch:report test="@updateModeDefault and @allowedUpdateModes and
not(contains(concat(';', translate(@allowedUpdateModes, ' ', ';'), ';'), concat(';',
@updateModeDefault, ';')))">
                   ERROR: DefaultUpdateMode must be part of allowedUpdateModes.</sch:report>
                  <sch:report test="@updateModeDefault and contains('ESA;ESD;ESC;ESAC',
@updateModeDefault) and @maxOccurs=1">
                   ERROR: Default update mode may not be one of the Edit Set values when
maxOccurs is 1.</sch:report>
                  <sch:report test="@allowedUpdateModes and contains(concat(' ',
@allowedUpdateModes), ' E') and @maxOccurs=1">
                   ERROR: Update mode set may not be one of the Edit Set values when maxOccurs is
1.</sch:report>
```

```
                </sch:rule>
             </sch:pattern>
          </xs:appinfo>
          <xs:documentation>Used to define multiplicty</xs:documentation>
          <xs:documentation>UML: Corresponds to UML type MultiplicityRange, with exception that
  upper boundary cannot be 0</xs:documentation>
       </xs:annotation>
       <xs:attribute name="minimumMultiplicity" type="SmallNonNegativeInteger" use="required">
          <xs:annotation>
             <xs:documentation>Identifies the minimum number of repetitions of this element that
  may occur within the containing element.</xs:documentation>
             <xs:documentation>UML: multiplicity (lower-bound)</xs:documentation>
          </xs:annotation>
       </xs:attribute>
       <xs:attribute name="maximumMultiplicity" type="UnlimitedInteger" use="required">
          <xs:annotation>
             <xs:documentation>Identifies the maximum number of repetitions of this element that
  may occur within the containing element.</xs:documentation>
             <xs:documentation>UML: multiplicity (upper-bound)</xs:documentation>
          </xs:annotation>
       </xs:attribute>
    </xs:attributeGroup>
    <xs:simpleType name="UnlimitedInteger">
       <xs:annotation>
          <xs:documentation>UML: Corresponds to UnlimitedInteger.  (Only distinction is that this
  type does not allow 0 while UML does)</xs:documentation>
       </xs:annotation>
       <xs:union memberTypes="SmallPositiveInteger UnlimitedMultiplicity"/>
    </xs:simpleType>
    <xs:simpleType name="UnlimitedMultiplicity">
       <xs:annotation>
          <xs:documentation>Used as a 'part' of the UnlimitedInteger type</xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:string">
          <xs:enumeration value="*">
             <xs:annotation>
                <xs:documentation>Unlimited</xs:documentation>
             </xs:annotation>
          </xs:enumeration>
       </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

C.8 mifReferencedCodes

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by George W Beeler (Beeler Consulting
LLC) -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3/mif"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
******************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   This schema provides includes all of the 'enumeration types' which are extracted from the
vocabulary database.

Todo:
   Extract this from the vocabulary database.
******************************************************************************
      </xs:documentation>
      <xs:documentation>UML: **ALL** types defined in this package are handled as datatypes that
are enumerations.</xs:documentation>
   </xs:annotation>
```

```
<xs:include schemaLocation="mifPatternTypes.xsd"/>
<xs:simpleType name="AffiliateKind">
   <xs:annotation>
      <xs:documentation>The list of HL7 affiliates.</xs:documentation>
   </xs:annotation>
   <xs:restriction base="EnumerationValue">
      <xs:enumeration value="AR">
         <xs:annotation>
            <xs:documentation>Argentina</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="AU">
         <xs:annotation>
            <xs:documentation>Australia</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="BR">
         <xs:annotation>
            <xs:documentation>Brazil</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="CA">
         <xs:annotation>
            <xs:documentation>Canada</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="CN">
         <xs:annotation>
            <xs:documentation>China</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="HR">
         <xs:annotation>
            <xs:documentation>Croatia</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="DK">
         <xs:annotation>
            <xs:documentation>Denmark</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="FI">
         <xs:annotation>
            <xs:documentation>Finland</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="DE">
         <xs:annotation>
            <xs:documentation>Germany</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="IN">
         <xs:annotation>
            <xs:documentation>India</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="JP">
         <xs:annotation>
            <xs:documentation>Japan</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="KR">
         <xs:annotation>
            <xs:documentation>Korea</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="LT">
         <xs:annotation>
```

```
                <xs:documentation>Lithuania</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="MX">
            <xs:annotation>
               <xs:documentation>Mexico</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="NE">
            <xs:annotation>
               <xs:documentation>Netherlands</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="NZ">
            <xs:annotation>
               <xs:documentation>New Zealand</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="SOA">
            <xs:annotation>
               <xs:documentation>Southern Africa</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="CH">
            <xs:annotation>
               <xs:documentation>Switzerland</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="TW">
            <xs:annotation>
               <xs:documentation>Taiwan</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="UK">
            <xs:annotation>
               <xs:documentation>United Kingdom</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="US">
            <xs:annotation>
               <xs:documentation>United States</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="UV">
            <xs:annotation>
               <xs:documentation>Applies to all realms (unless there is a realm-specific
override)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ZZ">
            <xs:annotation>
               <xs:documentation>A localized version created by a non-affiliate
entity</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="AffirmativeVoteResolutionKind">
      <xs:restriction base="ShortDescriptiveName">
         <xs:enumeration value="Affirmative-Incorporated">
            <xs:annotation>
               <xs:documentation>The recommended change has been incorporated or identified issue
has been answered.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Affirmative-Rejected">
            <xs:annotation>
```

```
                <xs:documentation>The recommended change has been refused and is not expected to
be incorporated.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Affirmative-Deferred">
            <xs:annotation>
               <xs:documentation>The recommended change has been deferred to consideration for a
future release.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="AnnotationKind">
      <xs:annotation>
         <xs:documentation>Identifies the kinds of annotations that can be
referenced</xs:documentation>
         <xs:documentation>UML: Used to reference a tagged item</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="Definition">
            <xs:annotation>
               <xs:documentation>Definition annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Description">
            <xs:annotation>
               <xs:documentation>Description annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="UsageNotes">
            <xs:annotation>
               <xs:documentation>UsageNotes annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Rationale">
            <xs:annotation>
               <xs:documentation>Rationale annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DesignComments">
            <xs:annotation>
               <xs:documentation>DesignComments annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Mapping">
            <xs:annotation>
               <xs:documentation>Mapping annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Constraint">
            <xs:annotation>
               <xs:documentation>Constraint annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="StaticExample">
            <xs:annotation>
               <xs:documentation>StaticExample annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Walkthrough">
            <xs:annotation>
               <xs:documentation>Walkthrough annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="BallotComment">
            <xs:annotation>
               <xs:documentation>BallotComment annotation</xs:documentation>
            </xs:annotation>
```

```
         </xs:enumeration>
         <xs:enumeration value="OtherAnnotation">
            <xs:annotation>
               <xs:documentation>OtherAnnotation annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Appendix">
            <xs:annotation>
               <xs:documentation>Appendix annotation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ArtifactKind">
      <xs:annotation>
         <xs:documentation>Identifies the kind of artifacts that can be
packaged</xs:documentation>
         <xs:documentation>UML: The name for a package in the package
hierarchy</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="CMET">
            <xs:annotation>
               <xs:documentation>Common Model Element Type</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="STUB">
            <xs:annotation>
               <xs:documentation>Template Parameter Definition</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RIM">
            <xs:annotation>
               <xs:documentation>Reference Information Model</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DIM">
            <xs:annotation>
               <xs:documentation>Domain Information Model (supercedes D-MIM)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="CIM">
            <xs:annotation>
               <xs:documentation>Constrained Information Model (supercedes R-MIM, HMD and Message
Type)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="LIM">
            <xs:annotation>
               <xs:documentation>Localized Information Model (a.k.a. message
profile)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DAM">
            <xs:annotation>
               <xs:documentation>Document Analysis Model</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="TP">
            <xs:annotation>
               <xs:documentation>Template</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DT">
            <xs:annotation>
               <xs:documentation>Datatype Model</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
```

```
         <xs:enumeration value="ITS">
            <xs:annotation>
               <xs:documentation>Implementation Technology Specification</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DC">
            <xs:annotation>
               <xs:documentation>Document</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="GL">
            <xs:annotation>
               <xs:documentation>Glossary</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="SB">
            <xs:annotation>
               <xs:documentation>Story Board</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="VO">
            <xs:annotation>
               <xs:documentation>Vocabulary Model</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="AR">
            <xs:annotation>
               <xs:documentation>Application Role</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="TE">
            <xs:annotation>
               <xs:documentation>Trigger Event</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="IN">
            <xs:annotation>
               <xs:documentation>Interaction</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DMIM-deprecated">
            <xs:annotation>
               <xs:documentation>Domain Message Information Model (deprecated, replaced with
DIM)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RM-deprecated">
            <xs:annotation>
               <xs:documentation>Refined Message Information Model (deprecated, replaced with
CIM)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="HD-deprecated">
            <xs:annotation>
               <xs:documentation>Hierarchical Message Descriptor (deprecated, replaced with
CIM)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="MT-deprecated">
            <xs:annotation>
               <xs:documentation>Message Type (deprecated, replaced with CIM)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ArtifactGroupKind">
      <xs:annotation>
```

```
         <xs:documentation>A list of the types of groups of artifacts that can be
referenced</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="Footnotes">
            <xs:annotation>
               <xs:documentation>A reference to all of the footnotes defined within the
artifact.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Packages">
            <xs:annotation>
               <xs:documentation>A reference to all of the packages defined within the artifact.
Only applies to packages.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Header">
            <xs:annotation>
               <xs:documentation>A reference to the header information associated with the
artifact.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Annotations">
            <xs:annotation>
               <xs:documentation>A reference to all of the annotations defined within the
artifact.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="History">
            <xs:annotation>
               <xs:documentation>A reference to all of the history associated with the
artifact.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Derivation">
            <xs:annotation>
               <xs:documentation>A reference to all of the derivations associated with the
artifact.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Properties">
            <xs:annotation>
               <xs:documentation>A reference to all of the properties within the artifact.  (Only
applies to DatatypeDefinitions.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="StaticModels">
            <xs:annotation>
               <xs:documentation>A reference to all of the static models defined within the
artifact.  (Only applies to Packages.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="SubjectAreas">
            <xs:annotation>
               <xs:documentation>A reference to all of the subject areas defined within the
artifact.  (Only applies to static models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Classes">
            <xs:annotation>
               <xs:documentation>A reference to all of the classes defined within the artifact.
(Only applies to static models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="StateEngines">
            <xs:annotation>
               <xs:documentation>A reference to all of the state engines defined within the
artifact.  (Only applies to static models.)</xs:documentation>
```

```
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="States">
            <xs:annotation>
               <xs:documentation>A reference to all of the states defined within the class.
(Only applies to classes within static models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="StateTransitions">
            <xs:annotation>
               <xs:documentation>A reference to all of the state transitions defined within the
class.  (Only applies to classes within static models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Associations">
            <xs:annotation>
               <xs:documentation>A reference to all of the associations defined within the class.
(Only applies to classes within static models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Attributes">
            <xs:annotation>
               <xs:documentation>A reference to all of the attributes defined within the class.
(Only applies to classes within static models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="TriggerEvents">
            <xs:annotation>
               <xs:documentation>A reference to all of the trigger events defined within the
artifact.  (Only applies to dynamic models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ApplicationRoles">
            <xs:annotation>
               <xs:documentation>A reference to all of the application roles defined within the
artifact.  (Only applies to dynamic models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Interactions">
            <xs:annotation>
               <xs:documentation>A reference to all of the interactions defined within the
artifact.  (Only applies to dynamic models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ReceiverResponsibilities">
            <xs:annotation>
               <xs:documentation>A reference to all of the xxx defined within the artifact.
(Only applies to interactions.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="VocabularyDomains">
            <xs:annotation>
               <xs:documentation>A reference to all of the vocabulary domains defined within the
artifact.  (Only applies to vocabulary models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="VocabularyCodes">
            <xs:annotation>
               <xs:documentation>A reference to all of the vocabulary codes defined within the
artifact.  (Only applies to vocabulary models.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Templates">
            <xs:annotation>
               <xs:documentation>A reference to all of the templates defined within the artifact.
(Only applies to packages.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
```

```
         <xs:enumeration value="CommunicationProtocols">
            <xs:annotation>
               <xs:documentation>A reference to all of the communication protocols defined within
the artifact.  (Only applies to packages.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ImplementationTechnologySpecifications">
            <xs:annotation>
               <xs:documentation>A reference to all of the implementation technology
specifications defined within the artifact.  (Only applies to implementation technology
specifications.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="BallotStatusKind">
      <xs:annotation>
         <xs:documentation>List of allowed ballot statuses</xs:documentation>
         <xs:documentation>UML: Type used in a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:restriction base="ShortDescriptiveName">
         <xs:enumeration value="Draft">
            <xs:annotation>
               <xs:documentation>Content that is under development and is not intended to be
used..</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Non-standard - Available for use">
            <xs:annotation>
               <xs:documentation>Content developed independently by an organization or individual
that is declared to be 'usable' but for which there is no present intention to submit through the
standards submission and review process.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Proposal">
            <xs:annotation>
               <xs:documentation>Content submitted to a committee for consideration for future
inclusion in the standard.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Committee Ballot - Normative">
            <xs:annotation>
               <xs:documentation>Content prepared by a committee and submitted for internal
consideration as a normative standard.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Committee Ballot - Informative">
            <xs:annotation>
               <xs:documentation>Content prepared by a committee and submitted for internal
consideration as an informative standard.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Membership Ballot - Normative">
            <xs:annotation>
               <xs:documentation>Content prepared by a committee and submitted for membership
consideration as a normative standard.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Membership Ballot - Informative">
            <xs:annotation>
               <xs:documentation>Content prepared by a committee and submitted for membership
consideration as an informative standard.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Approved Normative Standard">
            <xs:annotation>
               <xs:documentation>Content that has passed ballot as a normative
standard</xs:documentation>
```

```
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Approved Informative Standard">
            <xs:annotation>
               <xs:documentation>Content that has passed ballot as a normative
standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Affiliate Ballot - Normative">
            <xs:annotation>
               <xs:documentation>Content that is being presented to an international affiliate
for consideration as a realm-specific normative standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Affiliate Ballot - Informative">
            <xs:annotation>
               <xs:documentation>Content that is being presented to an international affiliate
for consideration as a realm-specific informative standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Reference">
            <xs:annotation>
               <xs:documentation>Content that is not intended to be balloted, but which exists to
support other balloted content</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Approved Affiliate Normative Standard">
            <xs:annotation>
               <xs:documentation>Content that has passed ballot as a realm-specific normative
standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Approved Affiliate Informative Standard">
            <xs:annotation>
               <xs:documentation>Content that has passed ballot as a realm-specific informative
standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Localized Adaptation">
            <xs:annotation>
               <xs:documentation>Content that represents an adaption of a implementable balloted
material to represent the needs or capabilities of a particular installation.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Withdrawn">
            <xs:annotation>
               <xs:documentation>Content that represents an item that was at one point a
normative or informative standard, but was subsequently withdrawn.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="CMETAttributionKind">
      <!-- Todo: Get Dale to fill in remaining definitions and/or identify whether these
'stereotypes' are useful.  Can one CMET have multiple? -->
      <xs:annotation>
         <xs:documentation>Provides the different 'types' of diagrams</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
         <xs:enumeration value="Abstract">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Basic">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
```

```
         </xs:enumeration>
         <xs:enumeration value="Clinical">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Contact">
            <xs:annotation>
               <xs:documentation>Contains the bare minimum information necessary to contact the
subject of the common model element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Identified">
            <xs:annotation>
               <xs:documentation>Contains the bare minimum information necessary to uniquely
identify the subject of the common model element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Identified/Confirmable">
            <xs:annotation>
               <xs:documentation>Contains the bare minimum information necessary to uniquely and
verifiably identify the subject of the common model element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Minimal">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="NoText">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Universal">
            <xs:annotation>
               <xs:documentation>Common model element includes maximum possible content, and
represents a union of the possible contents for the model element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="CMETEntryKind">
      <xs:annotation>
         <xs:documentation>Defines how the CMET may be entered</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="Act">
            <xs:annotation>
               <xs:documentation>The base of the CMET is a kind of Act, and may be entered via
Participation or ActRelationship</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Entity">
            <xs:annotation>
               <xs:documentation>The base of the CMET is a kind of Entity and may be entered via
Role</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RolePlayedOrPerformed">
            <xs:annotation>
               <xs:documentation>The base of the CMET is a Role whose 'Scoping' association is
already used or not intended to be used.  May be entered via participation or along plays
association from an Entity</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RoleScopedOrPerformed">
            <xs:annotation>
```

```
               <xs:documentation>The base of the CMET is a Role whose 'Playing' association is
already used or not intended to be used.  May be entered via participation or along scopes
association from an Entity</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RolePerformedOnly">
            <xs:annotation>
               <xs:documentation>The base of the CMET is a Role whose 'Playing' and 'Scoping'
associations are already used or not intended to be used.  May be entered via
participation.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="CodingStrengthKind">
      <xs:annotation>
         <xs:documentation>Defines whether the specified element is restricted to only using the
identified coding system for the element.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="CNE">
            <xs:annotation>
               <xs:documentation>If not null, the element must be coded and must be drawn from
the set of codes determined by the identified domain and the realm in which the instance is
constructed.  The set of codes is fixed to those values in place at the time this model was
successfully balloted.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="CWE">
            <xs:annotation>
               <xs:documentation>If not null, the element must be coded if there is an
appropriate value in the set of codes determined by the identified domain and the realm in which
the instance is constructed.  If no appropriate code is available, a local code may be used, or
the value may be populated with only original text.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ConformanceKind">
      <xs:annotation>
         <xs:documentation>Indicates allowed values to constrain whether vendors must implement
the specific element.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="R">
            <xs:annotation>
               <xs:documentation>Required - All implementors must support this property.  I.e.
they must be able to transmit, or to receive and usefully handle the concept.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="NP">
            <xs:annotation>
               <xs:documentation>Not Permitted - All implementors are prohibited from
transmitting this content, and may raise an error if they receive it.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ConstructedElementKind">
      <xs:annotation>
         <xs:documentation>Indicates the type of element to construct.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="DatatypeSummary">
            <xs:annotation>
               <xs:documentation>A summary table of the primary datatypes.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
```

```
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ContextKind">
      <xs:annotation>
         <xs:documentation>Allows both pre-defined contexts and free-form contexts to be
identified</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="EnumerationValue DefinedContextKind"/>
   </xs:simpleType>
   <xs:simpleType name="DatatypeOperationKind">
      <xs:annotation>
         <xs:documentation>The list of allowed datatype property kinds</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="conversion">
            <xs:annotation>
               <xs:documentation>The property exposes the datatype as converted into a more
complex or simpler datatype by defaulting properties of the new datatype and/or constraining the
properties of the original datatype.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="fixedProperty">
            <xs:annotation>
               <xs:documentation>A property whose value does not change from instance to instance
because it has been pre-constrained.  Fixed properties do not generally need to b exposed for
communication or persistance using an ITS.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="variableProperty">
            <xs:annotation>
               <xs:documentation>A property whose value may vary from instance to instance.
Variable properties will need to be comunicated or persisted by an ITS unless they can be
inferred from the value of other properties.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DefaultDeterminerKind">
      <xs:annotation>
         <xs:documentation>Indicates places defaults can be drawn from</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="ITS">
            <xs:annotation>
               <xs:documentation>The value is determined by the Implementation Technology
Specification used to encode the model referencing the datatype.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ReferencingAttribute">
            <xs:annotation>
               <xs:documentation>The value is determined by the definition of the attribute that
references the datatype.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Realm">
            <xs:annotation>
               <xs:documentation>The value is determined by the realm in which the model instance
is constructed.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DefinedContextKind">
      <xs:annotation>
         <xs:documentation>Defines pre-defined 'contexts'.  At present these consist primarily of
realms.</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="AffiliateKind"/>
```

```
   </xs:simpleType>
   <xs:simpleType name="DefinedMappingSourceKind">
      <xs:annotation>
         <xs:documentation>Lists 'pre-defined' standards that can be mapped to.  (Predefinition
helps ensure consistency of spelling, capitalization, etc.)</xs:documentation>
         <xs:documentation>UML: type for an object-typed stereotype tag</xs:documentation>
      </xs:annotation>
      <xs:restriction base="ShortDescriptiveName">
         <xs:enumeration value="HL7v2">
            <xs:annotation>
               <xs:documentation>mapping to the HL7 version 2.x messaging
standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DICOM">
            <xs:annotation>
               <xs:documentation>mapping to the DICOM standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Analysis Model">
            <xs:annotation>
               <xs:documentation>mapping to HL7 version 2.3 messaging standard</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DefinedRoleKind">
      <xs:annotation>
         <xs:documentation>Lists pre-defined roles that will commonly be used. (Ensures
consistency of spelling, capitalization, etc.)</xs:documentation>
         <xs:documentation>UML: Type used in a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:restriction base="ShortDescriptiveName">
         <xs:enumeration value="Committee co-chair">
            <xs:annotation>
               <xs:documentation>A person who was a co-chair of one of the contributing
committees at the time of ballot.  (Identify the committee in brackets following the person's
name)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Contact">
            <xs:annotation>
               <xs:documentation>A person to contact with questions about the ballot or its
contents.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Editor">
            <xs:annotation>
               <xs:documentation>A person with responsibility for the consolidation and
organization of the ballot material.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Primary Contributor">
            <xs:annotation>
               <xs:documentation>An individual who made substantial content contributions to the
ballot material.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DerivationRelationshipKind">
      <xs:annotation>
         <xs:documentation>The allowed derivation relationships between two
artifacts</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="restriction">
            <xs:annotation>
```

```
               <xs:documentation>The current element is a proper subset of the element it was
derived from, but is not identical.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="extension">
            <xs:annotation>
               <xs:documentation>The element derived from is a proper subset of this element, but
is not identical.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="conflicting">
            <xs:annotation>
               <xs:documentation>Neither the element nor the element derived from are proper
subsets of the other.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="annotated">
            <xs:annotation>
               <xs:documentation>The element is unchanged from the element it is derived from
with the exception of descriptive annotations.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="unchanged">
            <xs:annotation>
               <xs:documentation>The element is unchanged from the element it is derived
from.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ClassPresentationKind">
      <xs:annotation>
         <xs:documentation>A list of the 'styles' of graphic images that can be associated with a
class</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
         <xs:enumeration value="HL7">
            <xs:annotation>
               <xs:documentation>Standard HL7 Visio representation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="HL7Shadow">
            <xs:annotation>
               <xs:documentation>HL7 Visio shadow representation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="">
            <xs:annotation>
               <xs:documentation>Traditional UML representation</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="DomainKind">
      <xs:annotation>
         <xs:documentation>Identifies the HL7 'domain' (specific content area) to which the
package content pertains</xs:documentation>
         <xs:documentation>UML: The name for a package in the package
hierarchy</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="DD">
            <xs:annotation>
               <xs:documentation>Unknown</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="CI">
            <xs:annotation>
```

```
            <xs:documentation>Transmission Infrastructure</xs:documentation>
         </xs:annotation>
         <!-- MC -->
      </xs:enumeration>
      <xs:enumeration value="AI">
         <xs:annotation>
            <xs:documentation>Trigger Event Control Act Infrastructure</xs:documentation>
         </xs:annotation>
         <!-- MC -->
      </xs:enumeration>
      <xs:enumeration value="QI">
         <xs:annotation>
            <xs:documentation>Query Infrastructure</xs:documentation>
         </xs:annotation>
         <!-- QU -->
      </xs:enumeration>
      <xs:enumeration value="MI">
         <xs:annotation>
            <xs:documentation>Masterfile Infrastructure</xs:documentation>
         </xs:annotation>
         <!-- MF -->
      </xs:enumeration>
      <xs:enumeration value="LB">
         <xs:annotation>
            <xs:documentation>Laboratory</xs:documentation>
         </xs:annotation>
         <!-- PO -->
      </xs:enumeration>
      <xs:enumeration value="MR">
         <xs:annotation>
            <xs:documentation>Medical Records</xs:documentation>
         </xs:annotation>
         <!-- RC -->
      </xs:enumeration>
      <xs:enumeration value="PC">
         <xs:annotation>
            <xs:documentation>Patient Care</xs:documentation>
         </xs:annotation>
         <!-- RE -->
      </xs:enumeration>
      <xs:enumeration value="RX">
         <xs:annotation>
            <xs:documentation>Pharmacy</xs:documentation>
         </xs:annotation>
         <!-- PO -->
      </xs:enumeration>
      <xs:enumeration value="RR">
         <xs:annotation>
            <xs:documentation>Public Health Reporting</xs:documentation>
         </xs:annotation>
         <!-- PO -->
      </xs:enumeration>
      <xs:enumeration value="RI">
         <xs:annotation>
            <xs:documentation>Public Health Specifications</xs:documentation>
         </xs:annotation>
         <!-- PO -->
      </xs:enumeration>
      <xs:enumeration value="RT">
         <xs:annotation>
            <xs:documentation>Regulated Studies</xs:documentation>
         </xs:annotation>
         <!-- PO -->
      </xs:enumeration>
      <xs:enumeration value="AB">
         <xs:annotation>
            <xs:documentation>Account and Billing</xs:documentation>
         </xs:annotation>
```

```
            <!-- FI -->
         </xs:enumeration>
         <xs:enumeration value="CR">
            <xs:annotation>
               <xs:documentation>Claims and Reimbursement</xs:documentation>
            </xs:annotation>
            <!-- FI -->
         </xs:enumeration>
         <xs:enumeration value="PA">
            <xs:annotation>
               <xs:documentation>Patient Administration</xs:documentation>
            </xs:annotation>
            <!-- PR -->
         </xs:enumeration>
         <xs:enumeration value="PM">
            <xs:annotation>
               <xs:documentation>Personel Managment</xs:documentation>
            </xs:annotation>
            <!-- PR -->
         </xs:enumeration>
         <xs:enumeration value="SC">
            <xs:annotation>
               <xs:documentation>Scheduling</xs:documentation>
            </xs:annotation>
            <!-- PR -->
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ExpressionLanguageKind">
      <xs:annotation>
         <xs:documentation>Identifies language that constraints can be expressed in.  (Excludes
OCL, which is the default)</xs:documentation>
         <xs:documentation>UML: Type used by a complex stereotype</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="XPath">
            <xs:annotation>
               <xs:documentation>An XPath expression that assumes the use of the XML ITS, and
which evaluates to a boolean</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="HL7DatatypeDefinitionLanguage">
            <xs:annotation>
               <xs:documentation>Gunther's formal assertion language</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ImageKind">
      <xs:annotation>
         <xs:documentation>Identifies the type of image being represented.  Necessary to allow
for proper image scaling.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
         <xs:enumeration value="application/postscript">
            <xs:annotation>
               <xs:documentation>A post-script image.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ITSKind">
      <xs:annotation>
         <xs:documentation>The available v3 ITS types</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="XML">
            <xs:annotation>
```

```
                <xs:documentation>Extensible Markup Language</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="UML">
            <xs:annotation>
               <xs:documentation>Universal Modeling Language</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="MappingSourceKind">
      <xs:annotation>
         <xs:documentation>Allows for mappings to arbitrary standards, as well as 'pre-defined'
standards</xs:documentation>
         <xs:documentation>UML: type for an object-typed stereotype tag</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="DefinedMappingSourceKind ShortDescriptiveName"/>
   </xs:simpleType>
   <xs:simpleType name="MapRelationshipKind">
      <xs:annotation>
         <xs:documentation>The possible relationships between two mapped
elements</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="BT">
            <xs:annotation>
               <xs:documentation>Broader Than - The first concept is at a more abstract level
than the second concept.  For example, Hepatitis is broader than Hepatitis A, and endocrine
disease is broader than Diabetes Mellitus.  Broader than is the opposite of the narrower than
relationship.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="E">
            <xs:annotation>
               <xs:documentation>Exact - The two concepts have identical
meaning.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="NT">
            <xs:annotation>
               <xs:documentation>Narrower Than - The first concept is at a more detailed level
than the second concept.  For example, Pennicillin G is narrower than Pennicillin, and vellus
hair is narrower than hair.  Narrower than is the opposite of broader than.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="NegativeVoteResolutionKind">
      <xs:annotation>
         <xs:documentation>List of allowed resolutions to a formal ballot (based on HL7
bylaws)</xs:documentation>
         <xs:documentation>UML: Type for a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:restriction base="ShortDescriptiveName">
         <xs:enumeration value="Unresolved">
            <xs:annotation>
               <xs:documentation>Vote has not yet gone through resolution.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Non-Substantive Proposed">
            <xs:annotation>
               <xs:documentation>Responsible group has recommended that the negative vote be
considered non-substantive.  (Issue raised does not provide sufficiently convincing reason to
make changes to the item under ballot, or otherwise impede its adoption.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Non-Substantive Voted">
            <xs:annotation>
```

```
               <xs:documentation>Ballot group has voted and declared the negative vote non-
substantive.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Not Related">
            <xs:annotation>
               <xs:documentation>Responsible group has recommended that the negative vote be
considered non-substantive.  (Issue raised is not related to the current scope of the item under
ballot, or does not prevent the item under ballot for being used for its defined intent.
Recommended changes may be considered as part of future versions.)  (Perhaps after further
reading or explanation).</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Not Related">
            <xs:annotation>
               <xs:documentation>Ballot group has voted and declared the negative vote non-
related.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Previously Considered">
            <xs:annotation>
               <xs:documentation>Committee identifies that the same issue has been raised as part
of a previous ballot on the same element version and was found by the ballot group to be non-
substantive or not related.)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Retracted">
            <xs:annotation>
               <xs:documentation>Voter has formally withdrawn their vote or comment as having
been in error.  (Perhaps after further reading or explanation).</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Withdrawn">
            <xs:annotation>
               <xs:documentation>Voter has formally withdrawn their vote or comment on the basis
of agreed changes or proposed future changes.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="NodeOrientation">
      <xs:annotation>
         <xs:documentation>List of allowed values to reflect the orientation of graphic node
elements, including ChoiceBox alignment and the four, flip-orientations for a
Role.</xs:documentation>
         <xs:documentation>UML: Type for a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="Portrait">
            <xs:annotation>
               <xs:documentation>Applied to ChoiceBox, asserts that choice elements are arranged
vertically.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Landscape">
            <xs:annotation>
               <xs:documentation>Applied to ChoiceBox, asserts that choice elements are arranged
horizontally.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="UpRight">
            <xs:annotation>
               <xs:documentation>Applied to Role shape, represents orientation in which the
"player" association departs from the upper-right corner of the box.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="UpLeft">
            <xs:annotation>
```

```
                <xs:documentation>Applied to Role shape, represents orientation in which the
"player" association departs from the upper-left corner of the box.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DownRight">
            <xs:annotation>
               <xs:documentation>Applied to Role shape, represents orientation in which the
"player" association departs from the lower-right corner of the box.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="DownLeft">
            <xs:annotation>
               <xs:documentation>Applied to Role shape, represents orientation in which the
"player" association departs from the lower-left corner of the box.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="PackageRootKind">
      <xs:annotation>
         <xs:documentation>Identifies the primary 'purpose' of the packaged
content</xs:documentation>
         <xs:documentation>UML: The name for the initial set of packages within the 'HL7v3'
package</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="DEFN">
            <xs:annotation>
               <xs:documentation>The package represents the artifacts being
defined.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="BAL">
            <xs:annotation>
               <xs:documentation>The package groups a set of content defined elsewhere intended
to be submitted for ballot</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="PUB">
            <xs:annotation>
               <xs:documentation>The package groups a set of content defined elsewhere as a
formal publication</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="PROF">
            <xs:annotation>
               <xs:documentation>The package represents a particular implementation's
capabilities or requirements within a specific business area.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="PackageKind">
      <xs:annotation>
         <xs:documentation>Identifies the 'level' associated with a particular
package</xs:documentation>
         <xs:documentation>UML: stored in a tag on Package stereotypes</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="root">
            <xs:annotation>
               <xs:documentation>Indicates the fundamental categorization of the
package</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="section">
            <xs:annotation>
```

```
               <xs:documentation>Indicates a broad categorization of the business
area</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="subSection">
            <xs:annotation>
               <xs:documentation>Indicates a more specific categorization of the business
area</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="domain">
            <xs:annotation>
               <xs:documentation>Indicates a specific business area</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="realm">
            <xs:annotation>
               <xs:documentation>Indicates the geographic area covered by the
package</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="version">
            <xs:annotation>
               <xs:documentation>Indicates the iteration or revision number of a set of
content</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="artifact">
            <xs:annotation>
               <xs:documentation>Indicates the type of business-object stored in the
package</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="subArtifact">
            <xs:annotation>
               <xs:documentation>Indicates the sub-type of business-object stored in the
package</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="name">
            <xs:annotation>
               <xs:documentation>A descriptive label for a particular artifact</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="id">
            <xs:annotation>
               <xs:documentation>A numeric label for a particular artifact</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="creatorId">
            <xs:annotation>
               <xs:documentation>An OID distinguishing the namespace of a specific artifact
creator</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ParentArtifactKind">
      <xs:annotation>
         <xs:documentation>A list of the types of artifacts that can be
referenced</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="datatype">
            <xs:annotation>
               <xs:documentation>A 'datatypeDefinition' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
```

```
         <xs:enumeration value="property">
            <xs:annotation>
               <xs:documentation>A datatype 'property' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="class">
            <xs:annotation>
               <xs:documentation>A 'class' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="attribute">
            <xs:annotation>
               <xs:documentation>A class 'attribute' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="relationship">
            <xs:annotation>
               <xs:documentation>A class 'relationship' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="trigger">
            <xs:annotation>
               <xs:documentation>A 'triggerEvent' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="appRole">
            <xs:annotation>
               <xs:documentation>An 'applicationRole' element</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="RoleKind">
      <xs:annotation>
         <xs:documentation>Role names can be pre-defined or unrestricted</xs:documentation>
         <xs:documentation>UML: Type used in a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="DefinedRoleKind ShortDescriptiveName"/>
   </xs:simpleType>
   <xs:simpleType name="SectionKind">
      <xs:annotation>
         <xs:documentation>Identifies the HL7 'section' (major business category) to which the
package content pertains</xs:documentation>
         <xs:documentation>UML: The name for a package in the package
hierarchy</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="HM">
            <xs:annotation>
               <xs:documentation>Health and Clinical Management</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="AM">
            <xs:annotation>
               <xs:documentation>Administrative Management</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="IM">
            <xs:annotation>
               <xs:documentation>Infrastructure Management</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="StaticModelDiagramPresentationKind">
      <xs:annotation>
         <xs:documentation>Provides the different 'types' of diagrams</xs:documentation>
      </xs:annotation>
```

```
      <xs:restriction base="xs:string">
         <xs:enumeration value="">
            <xs:annotation>
               <xs:documentation>Standard UML format</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="HL7">
            <xs:annotation>
               <xs:documentation>HL7's "customized" diagramming format (aka R-MIM
format)</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="StaticModelRepresentationKind">
      <xs:annotation>
         <xs:documentation>The allowed mechanisms by which a static can be
serialized</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="flat">
            <xs:annotation>
               <xs:documentation>The model is represented as a list of class definitions, not
necessarily in the order they will appear when communicated, and not organized in a hierarchy for
exchange.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="serialized">
            <xs:annotation>
               <xs:documentation>The model is represented as a hierarchical list of classes and
associations ordered in the way they would appear in an instance for
exchange..</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="StaticModelUseKind">
      <xs:annotation>
         <xs:documentation>The allowed uses for a particular static model</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="CommunicationWrapper">
            <xs:annotation>
               <xs:documentation>Defines a communication structure for other
content.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ControlActWrapper">
            <xs:annotation>
               <xs:documentation>Defines common information regarding Acts upon a particular
payload.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="SemanticPayload">
            <xs:annotation>
               <xs:documentation>A set of content that is the 'focus' of a message or
document.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="PresentationStructure">
            <xs:annotation>
               <xs:documentation>Defines the organization and appearance of an underlying
model.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="QueryDefinition">
            <xs:annotation>
               <xs:documentation>Defines a set of data being solicited.</xs:documentation>
```

```
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="CMET">
            <xs:annotation>
               <xs:documentation>A re-usable static structure component.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Template">
            <xs:annotation>
               <xs:documentation>A pattern that can be applied to other
models.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="SubArtifactKind">
      <xs:annotation>
         <xs:documentation>Identifies the kind of sub artifacts that can be packaged within an
artifact</xs:documentation>
         <xs:documentation>UML: The name for a package in the package
hierarchy</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="VD">
            <xs:annotation>
               <xs:documentation>Vocabulary Domain</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="VC">
            <xs:annotation>
               <xs:documentation>Vocabulary Code System</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="VS">
            <xs:annotation>
               <xs:documentation>Value Set</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="SubSectionKind">
      <xs:annotation>
         <xs:documentation>Identifies the HL7 'subSection' (business sub-category) to which the
package content pertains</xs:documentation>
         <xs:documentation>UML: The name for a package in the package
hierarchy</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <!-- Todo: Fill in question marks -->
         <xs:enumeration value="UU">
            <xs:annotation>
               <xs:documentation>Unknown</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="MC">
            <xs:annotation>
               <xs:documentation>Message Control</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="QU">
            <xs:annotation>
               <xs:documentation>Query</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="MF">
            <xs:annotation>
               <xs:documentation>Master File</xs:documentation>
            </xs:annotation>
```

```
         </xs:enumeration>
         <xs:enumeration value="PO">
            <xs:annotation>
               <xs:documentation>Practice and Operations</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RC">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="RE">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="FI">
            <xs:annotation>
               <xs:documentation>Financial Information</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="PR">
            <xs:annotation>
               <xs:documentation>?????</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="TextStyleKind">
      <xs:annotation>
         <xs:documentation>Indicates the type of style to apply to the text.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="Requirement">
            <xs:annotation>
               <xs:documentation>Indented on a coloured background.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="UpdateModeKind">
      <xs:annotation>
         <xs:documentation>Potential update modes</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
         <!-- Note: xs:token is used because of a bug in Spy -->
         <xs:enumeration value="R">
            <xs:annotation>
               <xs:documentation>Replace</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="D">
            <xs:annotation>
               <xs:documentation>Delete</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="I">
            <xs:annotation>
               <xs:documentation>Ignore</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="V">
            <xs:annotation>
               <xs:documentation>Verify - The recipient's value must match the specified value to
process the message</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="K">
```

```
            <xs:annotation>
               <xs:documentation>Key - used as part of a unique identifier to an element within a
collection</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ESA">
            <xs:annotation>
               <xs:documentation>Edit Set Add - Add this element to the corresponding collection
in the receiving system</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ESC">
            <xs:annotation>
               <xs:documentation>Edit Set Change - Update this element in the corresponding
collection in the receiving system.  Take no action if the element does not
exist.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ESD">
            <xs:annotation>
               <xs:documentation>Edit Set Delete - Remove this element from the corresponding
collection in the receiving system</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="ESAC">
            <xs:annotation>
               <xs:documentation>Edit Set Add or Change - Add this element to the corresponding
collection in the receiving system if it does not already exist, update it
otherwise.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="VisibilityKind">
      <xs:annotation>
         <xs:documentation>The allowed levels of exposure for an element</xs:documentation>
         <xs:documentation>UML: Restriction on UML's VisibilityKind (excludes 'package'
visibility)</xs:documentation>
      </xs:annotation>
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="private">
            <xs:annotation>
               <xs:documentation>The datatype is strictly for use in deriving other datatypes or
constructing formal constraints.  It will never appear as the datatype for an attribute or as an
exposed property of another datatype.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="protected">
            <xs:annotation>
               <xs:documentation>The datatype is only for use in defining other datatypes.  It
may never be used as the datatype for an attribute, though it may appear through exposed
properties of a datatype.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="public">
            <xs:annotation>
               <xs:documentation>The datatype may be freely used, including as the datatype
associated with an attribute.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="VoteKind">
      <xs:annotation>
         <xs:documentation>Identifies the allowed types of individual responses within a ballot
response.</xs:documentation>
         <xs:documentation>UML: Type used in a complex tag value</xs:documentation>
      </xs:annotation>
```

```
      <xs:restriction base="EnumerationValue">
         <xs:enumeration value="Affirmative">
            <xs:annotation>
               <xs:documentation>The voter wants the ballot to pass</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Abstain">
            <xs:annotation>
               <xs:documentation>The voter is not expressing an opinion on whether the ballot
should pass</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Negative">
            <xs:annotation>
               <xs:documentation>The voter does *not* want the ballot to pass</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="VoteResolutionKind">
      <xs:annotation>
         <xs:documentation>List of allowed resolutions to a formal ballot (based on HL7
bylaws)</xs:documentation>
         <xs:documentation>UML: Type for a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="AffirmativeVoteResolutionKind NegativeVoteResolutionKind"/>
   </xs:simpleType>
   <xs:simpleType name="VoteCommentKind">
      <xs:annotation>
         <xs:documentation>List of allowed votes</xs:documentation>
         <xs:documentation>UML: Type for a complex tag value</xs:documentation>
      </xs:annotation>
      <xs:restriction base="ShortDescriptiveName">
         <xs:enumeration value="Affirmative - Typo">
            <xs:annotation>
               <xs:documentation>Vote is in favour of adoption of the balloted item, though
spelling or grammar error that doesn not affect semantic needs to be </xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Affirmative - Suggestion">
            <xs:annotation>
               <xs:documentation>The voter is recommending a change to the content, but they are
in favour of adoption of the balloted item regardless of whether the change is
made.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Affirmative - Question">
            <xs:annotation>
               <xs:documentation>The voter has a question with respect to the content, however
they are in favour of adoption of the balloted item regardless of whether or how the question is
answered.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Negative - Minor">
            <xs:annotation>
               <xs:documentation>Vote is opposed to the adoption of the balloted item, with a
need for minor changes in the item needed to resolve the negative.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="Negative - Major">
            <xs:annotation>
               <xs:documentation>Vote is opposed to the adoption of the balloted item, with a
need for significant changes or complete redevelopment of the item needed to resolve the
negative.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
```

```
</xs:schema>
```

C.9 pubDisplayMarkup

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- pubDisplayMarkup -->
<xs:schema targetNamespace="urn:hl7-org:v3/mif" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v3/mif"
elementFormDefault="qualified">
   <xs:annotation>
      <xs:documentation>
********************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   Defines the available text markup for use in various descriptive elements in various
artifacts.
   The contents of this schema are owned and defined by the publishing committee.  I.e. They
decide what sorts of markup are allowed

Modification History:
   2003-01-23: Revamped 'graphic' markup element to be renamed figure, and to have support for
fixed and scalable images.
   2003-01-28: Fixed graphic width to be optional
   2003-02-07: Modified property restriction to allow restriction of non-named properties
   2003-03-18: Added domainNameType as a separate type to handle x_domains
         - Changed upperCamelCaseType to support underscores to allow for proper choice names.
(We may want to revisit this.)
         - Removed the 'hide' markup element.  (If you don't want something, keep a copy or
comment it out.)
         - Added references for dynamicModel, storyboardModel, ITS and communicationProtocol
   2003-03-23: Changed domainRef to vocabularyDomainRef and added vocabularyCodeRef and
codeSystemRef
   2003-05-10: Defined references for vocabularyModels, datatypeModels and Glossaries
   2003-05-15: Moved stuff around because Spy defect has been fixed
         - Added artifactGroupRef to list of references

Outstanding questions:
  - Should we have a rule for 'indent', 'div' and/or 'listType' restricting the nesting depth of
lists?  (E.g. Can you have a list of lists of lists of lists of . . .)
  - Should we make graphic height and width mandatory?

Programatic rules (rules that apply but are not schema or schematron-enforced):
  - Ensure that datatypes are restricted to a list of defined datatypes
  - Ensure that datatypes which are supposed to have parameters do in fact have parameter types,
and those parameters are restrictions of the appropriate type as specified in the datatype
definition
  - All items that are 'referenced' must actually exist
  - All items defined as an 'artifactIdType' must be globally unique.  (I.e. there can not be
more than one item with a given number defined within a given package hierarchy.)
********************************************************************************
      </xs:documentation>
   </xs:annotation>
   <!-- Note: when this schema gets split, this include moves into the second schema -->
   <xs:include schemaLocation="mifReferencedCodes.xsd"/>
   <!-- NOTE: We could define these groups by 'ref'erencing the other groups.  However, spy has a
bug (#1779) that causes validation problems with mixed content when you do this. -->
   <xs:group name="basicMarkupGroup">
      <xs:annotation>
         <xs:documentation>Markup focusing solely on the appearance of the
text.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="p" type="BasicMarkup">
            <xs:annotation>
               <xs:documentation>Contained content is a separate paragraph.  If appearing as an
empty element, forces a line-break</xs:documentation>
```

```
            </xs:annotation>
         </xs:element>
         <xs:group ref="pubSimpleMarkupGroup"/>
         <xs:group ref="referenceMarkupGroup"/>
      </xs:choice>
   </xs:group>
   <xs:group name="formatMarkupGroup">
      <xs:annotation>
         <xs:documentation>Markup focusing solely on the appearance of the text, appropriate for
labels and titles.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="b" type="FormatMarkup">
            <xs:annotation>
               <xs:documentation>The contained content will appear as bolded
text</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="i" type="FormatMarkup">
            <xs:annotation>
               <xs:documentation>The contained content will appear as italicized
text</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="code" type="FormatMarkup">
            <xs:annotation>
               <xs:documentation>Information that is to be highlighted in non-proportional
font.</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:group>
   <xs:group name="fullMarkupGroup">
      <xs:annotation>
         <xs:documentation>A choice of all available markup types, including structural
markup.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:group ref="pubSimpleMarkupGroup"/>
         <xs:group ref="referenceMarkupGroup"/>
         <xs:group ref="pubComplexMarkupGroup"/>
      </xs:choice>
   </xs:group>
   <xs:group name="pubComplexMarkupGroup">
      <xs:annotation>
         <xs:documentation>A choice of elements that can themselves contain
markup.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="div" type="Div">
            <xs:annotation>
               <xs:documentation>Identifies a section of related content under a particular
title.</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate div element">
                     <sch:rule context="mif:div">
                        <sch:extends rule="Div"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="p" type="FullMarkup">
            <xs:annotation>
               <xs:documentation>Contained content is a separate paragraph.  If appearing as an
empty element, forces a line-break</xs:documentation>
            </xs:annotation>
         </xs:element>
```

```
         <xs:element name="ol" type="List">
            <xs:annotation>
               <xs:documentation>The content will consist of a set of list items where each item
is labeled with a number (starting with 1)</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="ul" type="List">
            <xs:annotation>
               <xs:documentation>The content will consist of a set of list items where each item
is labeled with a bullet</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="note" type="BasicMarkup">
            <xs:annotation>
               <xs:documentation>Information that is to be highlighted as a
Note.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="indent" type="FullMarkup">
            <xs:annotation>
               <xs:documentation>Identifies content that should be left-indented from the
surrounding text.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="figure" type="Figure">
            <xs:annotation>
               <xs:documentation>Inserts a pre-defined graphic image</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate figure element">
                     <sch:rule context="mif:figure">
                        <sch:extends rule="Figure"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="table" type="Table">
            <xs:annotation>
               <xs:documentation>Inserts a two-dimensional table</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate table element">
                     <sch:rule context="mif:table">
                        <sch:extends rule="Table"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="constructedElement" type="ConstructedElement">
            <xs:annotation>
               <xs:documentation>This is a cue to the publications process to auto-generate some
form of construct such as a summary table, cross reference table or artifact at a certain place
in the publication.  It may *only* be used with the pre-approval of the publications
group.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="styledText" type="StyledText">
            <xs:annotation>
               <xs:documentation>This is a cue to the publications process to auto-format the
contained text using a special style (which may include color, font, background, indentation,
etc.).  It may *only* be used with the pre-approval of the publications group.</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:group>
   <xs:group name="pubSimpleMarkupGroup">
      <xs:annotation>
```

```
         <xs:documentation>A choice of elements that may only contain simple string content (no
markup).</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="sup" type="FormatMarkup">
            <xs:annotation>
               <xs:documentation>Content that should appear in a smaller font, slightly above the
surrounding text.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="sub" type="FormatMarkup">
            <xs:annotation>
               <xs:documentation>Content that should appear in a smaller font, slightly below the
surrounding text.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="pre" type="VariousMixedContent">
            <xs:annotation>
               <xs:documentation>Indicates that the contained content should not have its spacing
adjusted.  NOTE: This element can technically contain markup elements.  However, they will be
treated as non-markup and will be displayed, not processed.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="footnote" type="FullMarkup">
            <xs:annotation>
               <xs:documentation>Supporting content that should be referenced within the
document, but displayed at the end of the page or document section.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:group ref="formatMarkupGroup"/>
      </xs:choice>
   </xs:group>
   <xs:group name="pubReferenceMarkupGroup">
      <xs:annotation>
         <xs:documentation>A choice of elements that reference other artifacts or markup
elements.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="externalRef" type="ExternalRef">
            <xs:annotation>
               <xs:documentation>A reference to an external document or artifact that allows
linking to the document or artifact.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="externalSpecRef" type="ExternalSpecRef">
            <xs:annotation>
               <xs:documentation>A reference to an externally defined
specification.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="figureRef" type="FigureRef">
            <xs:annotation>
               <xs:documentation>A reference to a graphic defined within the
document</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate figureRef element">
                     <sch:rule context="mif:figureRef">
                        <sch:extends rule="FigureRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="tableRef" type="TableRef">
            <xs:annotation>
               <xs:documentation>A reference to a table defined within the
document</xs:documentation>
               <xs:appinfo>
```

```
                  <sch:pattern name="Validate tableRef element">
                     <sch:rule context="mif:tableRef">
                        <sch:extends rule="TableRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="divRef" type="DivRef">
            <xs:annotation>
               <xs:documentation>A reference to a section defined within the
document</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate divRef element">
                     <sch:rule context="mif:divRef">
                        <sch:extends rule="DivRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:group>
   <xs:group name="referenceMarkupGroup">
      <xs:annotation>
         <xs:documentation>A choice of elements that reference other artifacts or markup
elements.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:group ref="pubReferenceMarkupGroup"/>
         <xs:group ref="artifactReferenceMarkupGroup"/>
      </xs:choice>
   </xs:group>
   <xs:complexType name="BasicMarkup" mixed="true">
      <xs:annotation>
         <xs:documentation>Basic formatting markup and paragraphs but no structural markup (lists
and subdivisions)</xs:documentation>
      </xs:annotation>
      <xs:group ref="basicMarkupGroup" minOccurs="0" maxOccurs="unbounded"/>
   </xs:complexType>
   <xs:complexType name="FormatMarkup" mixed="true">
      <xs:annotation>
         <xs:documentation>Formatting markup only.  No paragraphs or structural
markup.</xs:documentation>
      </xs:annotation>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
         <xs:group ref="formatMarkupGroup"/>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="FullMarkup" mixed="true">
      <xs:annotation>
         <xs:documentation>Full-blown markup including formating, paragraphs and structural
markup (lists and subdivisions).</xs:documentation>
      </xs:annotation>
      <xs:group ref="fullMarkupGroup" minOccurs="0" maxOccurs="unbounded"/>
   </xs:complexType>
   <xs:complexType name="PubComplexMarkup" mixed="true">
      <xs:annotation>
         <xs:documentation>Formatting markup only.  No paragraphs or structural
markup.</xs:documentation>
      </xs:annotation>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
         <xs:group ref="pubComplexMarkupGroup"/>
      </xs:choice>
   </xs:complexType>
   <!-- Sub-types that are used within the markup and subsequent schemas -->
   <xs:complexType name="VariousMixedContent" mixed="true">
      <xs:annotation>
```

```
         <xs:documentation>A type that allows unconstrained mixed-text markup</xs:documentation>
      </xs:annotation>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:any processContents="skip"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="StyledText" mixed="true">
      <xs:annotation>
         <xs:documentation>Text defined in a pre-defined named style (could affect font size,
weight, type, color, etc.)</xs:documentation>
      </xs:annotation>
      <xs:complexContent mixed="true">
         <xs:extension base="PubComplexMarkup">
            <xs:attribute name="style" type="TextStyleKind" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the style that should be applied to the
text.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ConstructedElement">
      <xs:annotation>
         <xs:documentation>Indicates that some sort of pre-defined 'auto-constructed' element
should be inserted at this location.</xs:documentation>
      </xs:annotation>
      <xs:attribute name="constructType" type="ConstructedElementKind" use="required">
         <xs:annotation>
            <xs:documentation>Indicates what type of construct should be
inserted</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="Div" mixed="true">
      <xs:annotation>
         <xs:documentation>Represents a 'section' of content.  Sections can be
nested.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Div type">
               <sch:rule abstract="true" id="Div">
                  <sch:report test="@id and
count(preceding::mif:*[name(.)=name(current())][@id=current()/@id])">
                                 ERROR: Div id must be unique within the document, if
specified.</sch:report>
                  <!-- Can't use unique because we don't know what the root node is going to be.
-->
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent mixed="true">
         <xs:extension base="FullMarkup">
            <xs:attribute name="title" type="ShortDescriptiveName" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies the heading name for the
section</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="numberSectionInd" type="xs:boolean" use="optional"
default="true">
               <xs:annotation>
                  <xs:documentation>If false, indicates that this section should not be assigned
a number or appear in the table of contents.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="id" type="BasicId" use="optional">
               <xs:annotation>
```

```
                  <xs:documentation>A unique identifier for the division within the document.
Used when referencing a table with <divRef/>.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Image">
      <xs:annotation>
         <xs:documentation>Used for all references to graphic figures</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="fixedImage" type="FixedImageContent">
            <xs:annotation>
               <xs:documentation>A fixed size image.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="resizable" type="ResizableImage">
            <xs:annotation>
               <xs:documentation>A resizable image definition that can be scaled as necessary for
display purposes.</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="Figure">
      <xs:annotation>
         <xs:documentation>A graphic image to be inserted into the document.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Figure type">
               <sch:rule abstract="true" id="Figure">
                  <sch:report test="@id and
count(preceding::mif:*[name(.)=name(current())][@id=current()/@id])">
                                 ERROR: Figure id must be unique within the document, if
specified.</sch:report>
                  <!-- Can't use unique because we don't know what the root node is going to be.
-->
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="Image">
            <xs:sequence>
               <xs:element name="caption" type="BasicMarkup" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>Descriptive text to be associated with the graphic
image.</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="BasicId" use="optional">
               <xs:annotation>
                  <xs:documentation>An identifier for the figure (unique within the document)
that can be used to reference the figure.  NOTE: The figure id will not necessarily be the
number/letter/identifier displayed for the figure in documentation.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="FixedImage">
      <xs:annotation>
         <xs:documentation>A graphic image with a fixed size</xs:documentation>
      </xs:annotation>
      <xs:attribute name="href" type="LocalFileReference" use="required">
         <xs:annotation>
```

```
            <xs:documentation>A reference to the relative file name and location of the fixed
image.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="height" type="PositiveDecimal" use="optional">
         <xs:annotation>
            <xs:documentation>The vertical size of the fixed image (in
inches).</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="width" type="PositiveDecimal" use="optional">
         <xs:annotation>
            <xs:documentation>The horizontal size of the fixed image (in
inches).</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="FixedImageContent">
      <xs:annotation>
         <xs:documentation>The actual content of a fixed size graphic image</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="pixmap" type="FixedImage">
            <xs:annotation>
               <xs:documentation>A fixed-size image.  This is the 'primary' image for the
figure.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="thumbnail" type="FixedImage" minOccurs="0">
            <xs:annotation>
               <xs:documentation>A small rendition of the pixmap image intended to be used as a
place-holder (often with a hyperlink to the larger pixmap image) when space constraints prevent
the display of the full-size image.</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="List">
      <xs:annotation>
         <xs:documentation>Used in defining all types of lists.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="li" type="FullMarkup" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>The content will appear as a separate item within a
list</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ResizableImage">
      <xs:annotation>
         <xs:documentation>A graphic image that can be scaled</xs:documentation>
      </xs:annotation>
      <xs:attribute name="href" type="LocalFileReference" use="required">
         <xs:annotation>
            <xs:documentation>A reference to the relative file name and location of the resizable
image.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="imageType" type="ImageKind" use="required">
         <xs:annotation>
            <xs:documentation>Defines what format the image is stored in.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="recommendedHeight" type="PositiveDecimal" use="optional">
         <xs:annotation>
```

```
            <xs:documentation>The suggested vertical size at which the image should be rendered
(in inches).</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="recommendedWidth" type="PositiveDecimal" use="optional">
         <xs:annotation>
            <xs:documentation>The suggested horizontal size at which the image should be rendered
(in inches).</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="Row">
      <xs:annotation>
         <xs:documentation>A horizontal row within a table</xs:documentation>
      </xs:annotation>
      <xs:choice maxOccurs="unbounded">
         <xs:element name="th" type="BasicMarkup">
            <xs:annotation>
               <xs:documentation>Represents header data within one cell of the
table</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="td" type="BasicMarkup">
            <xs:annotation>
               <xs:documentation>Represents non-header data within one cell of the
table</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="Table">
      <xs:annotation>
         <xs:documentation>Used when the contained content should be represented as a
table.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate Table type">
               <sch:rule abstract="true" id="Table">
                  <sch:report test="@id and
count(preceding::mif:*[name(.)=name(current())][@id=current()/@id])">
                              ERROR: Table id must be unique within the document, if
specified.</sch:report>
                  <!-- Can't use unique because we don't know what the root node is going to be.
-->
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="caption" type="BasicMarkup" minOccurs="0">
            <xs:annotation>
               <xs:documentation>The text of the caption that should appear above the
table</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="col" type="TableColumn" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Defines the columns for the table.  If not specified, columns
will be autosized.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="thead" type="TableHead" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Defines the contents of the table header row.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="tbody" type="TableBody">
            <xs:annotation>
               <xs:documentation>Contains the information of the table.</xs:documentation>
```

```
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="BasicId" use="optional">
         <xs:annotation>
            <xs:documentation>A unique identifier for the table within the document.  Used when
referencing a table with <tableRef/>.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="borders" type="SingleNonNegativeInteger" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the width of the border.  A width of 0 means no border
will appear.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="TableColumn">
      <xs:annotation>
         <xs:documentation>Defines a vertical row in a table</xs:documentation>
      </xs:annotation>
      <xs:attribute name="width" type="PositiveDecimal" use="required">
         <xs:annotation>
            <xs:documentation>Identifies the horizontal size of the column in
inches.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="TableHead">
      <xs:annotation>
         <xs:documentation>Defines a table row or rows that is a table heading</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="tr" type="Row" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>The row containing the table header labels</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="TableBody">
      <xs:annotation>
         <xs:documentation>Defines the set of rows that make up the 'content' of the
table</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="tr" type="Row" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Represents a row within the table</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="DivRef">
      <xs:annotation>
         <xs:documentation>A reference to a 'section' defined within the document
.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate DivRef type">
               <sch:rule abstract="true" id="DivRef">
                  <sch:report test="not(//mif:*[name(.)=name(current())][@id=current()/@id])">
                              ERROR: No div with the referenced id exists within this
document.</sch:report>
                  <!-- Can't use keyref because we don't know what the root node is going to be.
-->
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
```

```
      <xs:attribute name="id" type="BasicId" use="required">
         <xs:annotation>
            <xs:documentation>The identifier assigned to the division being
referenced.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="ExternalRef">
      <xs:annotation>
         <xs:documentation>Used to reference an external document or item by
URL.</xs:documentation>
      </xs:annotation>
      <xs:attribute name="ref" type="Url" use="required">
         <xs:annotation>
            <xs:documentation>The URL where the document or item can be found.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="alt" type="LongDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>The descriptive name for the document or item.  If not specified,
the description will be the URL itself.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="ExternalSpecRef">
      <xs:annotation>
         <xs:documentation>A reference to a defined artifact from an external
specification</xs:documentation>
      </xs:annotation>
      <xs:attribute name="spec" type="ShortDescriptiveName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the referenced specification.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="version" type="Version" use="optional">
         <xs:annotation>
            <xs:documentation>The version number of the referenced
specification.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="ref" type="Url" use="optional">
         <xs:annotation>
            <xs:documentation>The URL where the referenced artifact can be
found.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="FigureRef">
      <xs:annotation>
         <xs:documentation>A reference to a graphic image defined within the document or in an
external document.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate FigureRef type">
               <sch:rule abstract="true" id="FigureRef">
                  <sch:report test="not(//mif:*[name(.)=name(current())][@id=current()/@id])">
                              ERROR: No figure with the referenced id exists within this
document.</sch:report>
                  <!-- Can't use keyref because we don't know what the root node is going to be.
-->
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="id" type="BasicId" use="required">
         <xs:annotation>
            <xs:documentation>The identifier assigned to the figure being
referenced.</xs:documentation>
         </xs:annotation>
```

```
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="TableRef">
      <xs:annotation>
         <xs:documentation>A reference to a table defined within the document
.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate TableRef type">
               <sch:rule abstract="true" id="TableRef">
                  <sch:report test="not(//mif:*[name(.)=name(current())][@id=current()/@id])">
                              ERROR: No table with the referenced id exists within this
document.</sch:report>
                  <!-- Can't use keyref because we don't know what the root node is going to be.
-->
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:attribute name="id" type="BasicId" use="required">
         <xs:annotation>
            <xs:documentation>The identifier assigned to the table being
referenced.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <!--
*********************************************************************************************
      - The following content should be split into a separate schema, but can't until SPY fixes
a bug
      - (Spy doesn't allow valid redefines of groups, which we need to do.)
      -
********************************************************************************************* -
->
   <!-- Sub-types that are used within the markup and subsequent schemas -->
   <xs:group name="artifactReferenceMarkupGroup">
      <xs:annotation>
         <xs:documentation>A choice of elements that reference other
artifacts.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element name="itemName" type="ItemName">
            <xs:annotation>
               <xs:documentation>A reference to one of the parents of the current
element</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate itemName element">
                     <sch:rule context="mif:itemName">
                        <sch:extends rule="ItemName"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="annotationRef" type="AnnotationRef">
            <xs:annotation>
               <xs:documentation>A reference to an annotation on the current element or one of
its parents</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate annoationRef element">
                     <sch:rule context="mif:annotationRef">
                        <sch:extends rule="AnnotationRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="artifactGroupRef" type="ArtifactGroupRef">
            <xs:annotation>
```

```
               <xs:documentation>An absolute reference to a collection of artifacts of a
particular type</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate artifactGroupRef element">
                     <sch:rule context="mif:artifactGroupRef">
                        <sch:extends rule="ArtifactGroupRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="packageRef" type="PackageRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a particular package</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate packageRef element">
                     <sch:rule context="mif:packageRef">
                        <sch:extends rule="PackageRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="datatypeModelRef" type="PackageRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a datatype model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate datatypeModelRef element">
                     <sch:rule context="mif:datatypeModelRef">
                        <sch:extends rule="PackageRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="datatypeRef" type="DatatypeRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a datatype
definition</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="propertyRef" type="DatatypePropertyRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific property of a datatype
definition</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate propertyRef element">
                     <sch:rule context="mif:propertyRef">
                        <sch:extends rule="DatatypePropertyRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="staticModelRef" type="ArtifactRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific static
model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate staticModelRef element">
                     <sch:rule context="mif:staticModelRef">
                        <sch:extends rule="ArtifactRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="subjectAreaRef" type="SubjectAreaRef">
```

```
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific subject area in a static
model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate subjectAreaRef element">
                     <sch:rule context="mif:subjectAreaRef">
                        <sch:extends rule="SubjectAreaRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="classRef" type="ClassRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific class in a static
model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate classRef element">
                     <sch:rule context="mif:classRef">
                        <sch:extends rule="ClassRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="transitionRef" type="TransitionRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific state transition for a class
in a static model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate transitionRef element">
                     <sch:rule context="mif:transitionRef">
                        <sch:extends rule="TransitionRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="stateRef" type="StateRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific state for a class in a
static model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate stateRef element">
                     <sch:rule context="mif:stateRef">
                        <sch:extends rule="StateRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="attributeRef" type="AttributeRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific attribute for a class in a
static model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate attributeRef element">
                     <sch:rule context="mif:attributeRef">
                        <sch:extends rule="AttributeRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="associationEndRef" type="AssociationEndRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific association end for a class
in a static model</xs:documentation>
```

```
                <xs:appinfo>
                   <sch:pattern name="Validate associationEndRef element">
                      <sch:rule context="mif:associationEndRef">
                         <sch:extends rule="AssociationEndRef"/>
                      </sch:rule>
                   </sch:pattern>
                </xs:appinfo>
             </xs:annotation>
          </xs:element>
          <xs:element name="storyboardModelRef" type="PackageRef">
             <xs:annotation>
                <xs:documentation>An absolute reference to a specific storyboard
model</xs:documentation>
                <xs:appinfo>
                   <sch:pattern name="Validate storyboardModelRef element">
                      <sch:rule context="mif:storyboardModelRef">
                         <sch:extends rule="PackageRef"/>
                      </sch:rule>
                   </sch:pattern>
                </xs:appinfo>
             </xs:annotation>
          </xs:element>
          <xs:element name="storyboardRef" type="ArtifactRef">
             <xs:annotation>
                <xs:documentation>An absolute reference to a specific storyboard within a
storyboard model</xs:documentation>
                <xs:appinfo>
                   <sch:pattern name="Validate storyboardRef element">
                      <sch:rule context="mif:storyboardRef">
                         <sch:extends rule="ArtifactRef"/>
                      </sch:rule>
                   </sch:pattern>
                </xs:appinfo>
             </xs:annotation>
          </xs:element>
          <xs:element name="triggerEventRef" type="ArtifactRef">
             <xs:annotation>
                <xs:documentation>An absolute reference to a specific trigger
event</xs:documentation>
                <xs:appinfo>
                   <sch:pattern name="Validate triggerEventRef element">
                      <sch:rule context="mif:triggerEventRef">
                         <sch:extends rule="ArtifactRef"/>
                      </sch:rule>
                   </sch:pattern>
                </xs:appinfo>
             </xs:annotation>
          </xs:element>
          <xs:element name="applicationRoleRef" type="ArtifactRef">
             <xs:annotation>
                <xs:documentation>An absolute reference to a specific application
role</xs:documentation>
                <xs:appinfo>
                   <sch:pattern name="Validate applicationRoleRef element">
                      <sch:rule context="mif:applicationRoleRef">
                         <sch:extends rule="ArtifactRef"/>
                      </sch:rule>
                   </sch:pattern>
                </xs:appinfo>
             </xs:annotation>
          </xs:element>
          <xs:element name="interactionRef" type="ArtifactRef">
             <xs:annotation>
                <xs:documentation>An absolute reference to a specific
interaction</xs:documentation>
                <xs:appinfo>
                   <sch:pattern name="Validate interactionRef element">
                      <sch:rule context="mif:interactionRef">
```

```
                        <sch:extends rule="ArtifactRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="vocabularyModelRef" type="PackageRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific vocabulary
model</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate vocabularyModelRef element">
                     <sch:rule context="mif:vocabularyModelRef">
                        <sch:extends rule="PackageRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="vocabularyDomainRef" type="PackageRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific vocabulary
domain</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate vocabularyDomainRef element">
                     <sch:rule context="mif:vocabularyDomainRef">
                        <sch:extends rule="PackageRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="codeSystemRef" type="PackageRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific code
system</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate codeSystemRef element">
                     <sch:rule context="mif:codeSystemRef">
                        <sch:extends rule="PackageRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="vocabularyCodeRef" type="VocabularyCodeRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific code within a code
system</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate vocabularyCodeRef element">
                     <sch:rule context="mif:vocabularyCodeRef">
                        <sch:extends rule="VocabularyCodeRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="glossaryRef" type="PackageRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific glossary</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate glossaryRef element">
                     <sch:rule context="mif:glossaryRef">
                        <sch:extends rule="PackageRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
```

```
            </xs:annotation>
         </xs:element>
         <xs:element name="glossaryTermRef" type="GlossaryTermRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific term within a
glossary</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate glossaryTermRef element">
                     <sch:rule context="mif:glossaryTermRef">
                        <sch:extends rule="GlossaryTermRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="templateRef" type="ArtifactRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific template</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate templateRef element">
                     <sch:rule context="mif:templateRef">
                        <sch:extends rule="ArtifactRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="itsRef" type="ArtifactRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific implementation
technology</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate itsRef element">
                     <sch:rule context="mif:itsRef">
                        <sch:extends rule="ArtifactRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
         <xs:element name="communicationProtocolRef" type="ArtifactRef">
            <xs:annotation>
               <xs:documentation>An absolute reference to a specific communication
protocol</xs:documentation>
               <xs:appinfo>
                  <sch:pattern name="Validate communicationProtocolRef element">
                     <sch:rule context="mif:communicationProtocolRef">
                        <sch:extends rule="ArtifactRef"/>
                     </sch:rule>
                  </sch:pattern>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:group>
   <xs:complexType name="ItemName">
      <xs:annotation>
         <xs:documentation>A reference to the name of the containing element of the specified
type.  This element should be used instead of the name of the element to minimize the number of
changes necessary if a name should change.  E.g. In a definition you might say "A <itemName/> is
used when . . ."</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ItemName type">
               <sch:rule abstract="true" id="ItemName">
                  <!-- Todo: Fix names of referenced types -->
                  <sch:report test="(@item='datatype' and not(ancestor::mif:datatypeDefinition))
or
                                 (@item='property' and not(ancestor::mif:property)) or
```

```
                              (@item='class' and not(ancestor::mif:class)) or
                              (@item='attribute' and not(ancestor::mif:attribute)) or
                              (@item='relationship' and not(ancestor::mif:relationship)) or
                              (@item='trigger' and not(ancestor::mif:triggerEvent)) or
                              (@item='appRole' and not(ancestor::mif:applicationRole))">
                        ERROR: There is no ancestor of the specified type.</sch:report>
                     <sch:report test="not(@item) and (not(
                              ancestor::mif:datatypeDefinition or ancestor::mif:property or
ancestor::mif:class or ancestor::mif:attribute or
                              ancestor::mif:relationship or ancestor::mif:triggerEvent or
ancestor::mif:applicationRole))">
                        ERROR: There is no ancestor of an appropriate type having a
name.</sch:report>
                  </sch:rule>
               </sch:pattern>
            </xs:appinfo>
         </xs:annotation>
         <xs:attribute name="item" type="ParentArtifactKind" use="optional">
            <xs:annotation>
               <xs:documentation>Identifies the type of containing element the name is a reference
for.  If there are multiple containing elements of the same type, the referenced name will be the
nearest one.  If not specified, then the nearest eligible containing element will be
used.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
      </xs:complexType>
      <xs:complexType name="AnnotationRef">
         <xs:annotation>
            <xs:documentation>A reference to an annotation on the current element, or (if 'item' is
specified), one of its parents</xs:documentation>
            <xs:appinfo>
               <sch:pattern name="Validate AnnotationRef type">
                  <sch:rule abstract="true" id="AnnotationRef">
                     <sch:extends rule="ItemName"/>
                  </sch:rule>
               </sch:pattern>
            </xs:appinfo>
         </xs:annotation>
         <xs:complexContent>
            <xs:extension base="ItemName">
               <xs:attribute name="annotationKind" type="AnnotationKind" use="required">
                  <xs:annotation>
                     <xs:documentation>Identifies the kind of annotation being
referenced.</xs:documentation>
                  </xs:annotation>
               </xs:attribute>
               <xs:attribute name="annotationName" type="ShortDescriptiveName" use="optional">
                  <xs:annotation>
                     <xs:documentation>The specific name of the annotation.</xs:documentation>
                  </xs:annotation>
               </xs:attribute>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
      <xs:complexType name="PackageRef">
         <xs:annotation>
            <xs:documentation>Used to make absoloute references to other
packages.</xs:documentation>
            <xs:appinfo>
               <sch:pattern name="Validate PackageRef type">
                  <sch:rule abstract="true" id="PackageRef">
                     <sch:report test="@subSection and not(@section)">
                        ERROR: Can't have a subsection without a section.</sch:report>
                     <sch:report test="@domain and not(@subSection)">
                        ERROR: Can't have a domain without a subSection.</sch:report>
                     <sch:report test="contains(concat(';', @artifact, ';'), ';RIM;ITS;VO;DT;') and
(@section)">
```

```
                    ERROR: RIM, ITS, Vocabulary and Datatypes must be global (they may not be
within a section).</sch:report>
                <sch:report test="contains(concat(';', @artifact, ';'),
';DIM;CIM;LIM;AR;TE;IN;TP;DAM;') and not(@domain)">
                    ERROR: DIMs, DAMs, CIMs, LIMs, Application Roles, Trigger Events,
Interactions and Templates may only be defined in a domain.</sch:report>
                <sch:report test="contains(concat(';', @artifact, ';'), ';GL;DC;CMET;STUB;SB;')
and @section and not(@domain)">
                    ERROR: Glossaries, Documents, CMETs, Stubs and Storyboards must either be
global (no section) or domain-specific.</sch:report>
                <sch:report test="@subArtifact and @artifact!='VO'">
                    ERROR: Only vocabulary artifacts may have sub-artifact types.</sch:report>
                <!-- Todo: Add more constraints -->
                <!-- Todo: Can only have identifierId if realm is ZZ -->
            </sch:rule>
        </sch:pattern>
    </xs:appinfo>
  </xs:annotation>
  <xs:attribute name="root" type="PackageRootKind" use="required">
    <xs:annotation>
      <xs:documentation>Indicates the fundamental categorization of the
package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="section" type="SectionKind" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the name of the section package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="subSection" type="SubSectionKind" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the name of the sub-section package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="domain" type="DomainKind" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the name of the domain package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="realm" type="AffiliateKind" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the geographic area covered by the
package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="version" type="ShortDescriptiveName" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the name of the version package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="artifact" type="ArtifactKind" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the category of the artifacts in the
package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="subArtifact" type="SubArtifactKind" use="optional">
    <xs:annotation>
      <xs:documentation>Indicates the sub-category of the artifacts in the
package</xs:documentation>
      <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
```

```
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="ShortDescriptiveName" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the name of the package</xs:documentation>
            <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="id" type="SmallNonNegativeInteger" use="optional">
         <xs:annotation>
            <xs:documentation>Indicates the identifier number of the package</xs:documentation>
            <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="creatorId" type="Oid" use="optional">
         <xs:annotation>
            <xs:documentation>An OID distinguishing the namespace of a specific artifact
creator</xs:documentation>
            <xs:documentation>UML: ModelElement.name, inherited by package</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:simpleType name="PackageNodeName">
      <xs:annotation>
         <xs:documentation>A package node might be a name or an OID</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="Oid BasicFormalName"/>
   </xs:simpleType>
   <xs:complexType name="ArtifactRef">
      <xs:annotation>
         <xs:documentation>References to artifacts within a particular package</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ArtifactRef type">
               <sch:rule abstract="true" id="ArtifactRef">
                  <sch:extends rule="PackageRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:attribute name="artifactName" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the referenced artifact within the
package</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="PackageOrArtifactRef">
      <xs:annotation>
         <xs:documentation>References to generic elements that could be packages or
artifacts</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate PackageOrArtifactRef type">
               <sch:rule abstract="true" id="PackageOrArtifactRef">
                  <sch:extends rule="PackageRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:attribute name="artifactName" type="BasicFormalName" use="optional">
               <xs:annotation>
                  <xs:documentation>The name of the referenced artifact within the
package</xs:documentation>
```

```
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ArtifactGroupRef">
      <xs:annotation>
         <xs:documentation>A reference to a particular grouping of artifacts.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ArtifactGroupRef type">
               <sch:rule abstract="true" id="ArtifactGroupRef">
                  <sch:extends rule="PackageRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:attribute name="group" type="ArtifactGroupKind" use="required">
               <xs:annotation>
                  <xs:documentation>Identifies the type of artifact group being referenced.  Used
in circumstances where there is not a model or other artifact that contains only the specified
type of item.  NOTE: This should only be used with the approval of the publications committee.
(They will decide what 'groups' of things will actually be portrayed as
groups)</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="linkToEnd" type="xs:boolean" use="optional" default="false">
               <xs:annotation>
                  <xs:documentation>If true, indicates that the link should be to the first item
following the specified group.  If false, the link is to the begining of the
group.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
            <xs:attribute name="withinClassName" type="FormalProperName" use="optional">
               <xs:annotation>
                  <xs:documentation>Indicates the name of the class within which to link to the
group.  For example, link to the 'attributes' section in class 'A' static model 'B'.  If not
specified, the link will be to the nearest ancestor artifact that contains the specified type of
group.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="AssociationEndRef">
      <xs:annotation>
         <xs:documentation>A reference to a defined relationship within a particular model and
class.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate AssociationEndRef type">
               <sch:rule abstract="true" id="AssociationEndRef">
                  <sch:extends rule="ClassRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRef">
            <xs:attribute name="relationshipName" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the relationship being
referenced.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

```
   <xs:complexType name="AttributeRef">
      <xs:annotation>
         <xs:documentation>A reference to a defined attribute within a particular model and
class.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate AttributeRef type">
               <sch:rule abstract="true" id="AttributeRef">
                  <sch:extends rule="ClassRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRef">
            <xs:attribute name="attributeName" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the attribute being
referenced.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="ClassRef">
      <xs:annotation>
         <xs:documentation>A reference to a defined class within a particular
model.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate ClassRef type">
               <sch:rule abstract="true" id="ClassRef">
                  <sch:extends rule="PackageRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:attribute name="className" type="AllClassName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the class being referenced.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="DatatypeRef">
      <xs:annotation>
         <xs:documentation>A reference to a datatype definition.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="supplierBindingArgumentDatatype" type="DatatypeRef" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Identifies a datatype to bind to one of the referenced datatypes
templateParameters</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="DatatypeName" use="required">
         <xs:annotation>
            <xs:documentation>The name of the datatype being referenced</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:complexType name="DatatypePropertyRef">
      <xs:annotation>
         <xs:documentation>A reference to a datatype property defined within the document or in
an external document.</xs:documentation>
```

```
         <xs:appinfo>
            <sch:pattern name="Validate DatatypePropertyRef type">
               <sch:rule abstract="true" id="DatatypePropertyRef">
                  <sch:report test="@propertyName and count(mif:conversionDatatype)!=0">
                              ERROR: A property reference may only have a name or a conversion
datatype, not both.</sch:report>
                  <sch:report test="not(@propertyName) and count(mif:conversionDatatype)=0">
                              ERROR: A property reference must have either a name or a
conversion datatype.</sch:report>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="DatatypeRef">
            <xs:sequence>
               <xs:element name="conversionDatatype" type="DatatypeRef" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>For 'conversion' properties, identifies the target
datatype for the conversion.</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="propertyName" type="FormalPropertyName" use="optional">
               <xs:annotation>
                  <xs:documentation>The name of the property being referenced.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="GlossaryTermRef">
      <xs:annotation>
         <xs:documentation>A reference to a glossary term.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate GlossaryTermRef type">
               <sch:rule abstract="true" id="GlossaryTermRef">
                  <sch:extends rule="PackageRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:attribute name="termName" type="ShortDescriptiveName" use="required">
               <xs:annotation>
                  <xs:documentation>Name of the referenced term.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="StateRef">
      <xs:annotation>
         <xs:documentation>Identifies a state that is associated with the trigger
event.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate StateRef type">
               <sch:rule abstract="true" id="StateRef">
                  <sch:extends rule="ClassRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRef">
            <xs:attribute name="stateName" type="FormalPropertyName" use="required">
               <xs:annotation>
```

```
                  <xs:documentation>The name of the state within the 'focal'
class.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="TransitionRef">
      <xs:annotation>
         <xs:documentation>Identifies a state transition defined elsewhere.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate TransitionRef type">
               <sch:rule abstract="true" id="TransitionRef">
                  <sch:extends rule="ClassRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ClassRef">
            <xs:attribute name="stateTransitionName" type="FormalPropertyName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the state transition within the 'focal' class
that is the basis for the trigger event.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="SubjectAreaRef">
      <xs:annotation>
         <xs:documentation>A reference to a defined subject area within a particular
model.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate SubjectAreaRef type">
               <sch:rule abstract="true" id="SubjectAreaRef">
                  <sch:extends rule="ArtifactRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="ArtifactRef">
            <xs:attribute name="subjectAreaName" type="BasicFormalName" use="required">
               <xs:annotation>
                  <xs:documentation>The name of the subject area being
referenced.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="VocabularyCodeRef">
      <xs:annotation>
         <xs:documentation>A reference to a vocabulary code defined in another
model.</xs:documentation>
         <xs:appinfo>
            <sch:pattern name="Validate VocabularyCodeRef type">
               <sch:rule abstract="true" id="VocabularyCodeRef">
                  <sch:extends rule="PackageRef"/>
               </sch:rule>
            </sch:pattern>
         </xs:appinfo>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="PackageRef">
            <xs:attribute name="code" type="ShortDescriptiveName" use="required">
               <xs:annotation>
```

```
                  <xs:documentation>The identifier or mnemonic of the referenced
code.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

C.10 mifExtendedMarkup

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" targetNamespace="urn:hl7-org:v3/mif" xmlns="urn:hl7-
org:v3/mif" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sch="http://www.ascc.net/xml/schematron">
   <xs:annotation>
      <xs:documentation>
*******************************************************************************
Author: Initial development by Lloyd McKenzie, Dec. 2002
(c) 2002, 2003 by HL7 Inc.

Purpose:
   This schema is intended to contain the HL7 v3 model-specific markup contents.
   Those contents will be moved here once XML Spy fixes a bug with redefining groups.
*******************************************************************************
      </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="pubDisplayMarkup.xsd"/>
   <!-- Content will be moved here once a SPY bug is fixed -->
</xs:schema>
```