EXTRACT from: Documentation of HL7 Internal Tooling Structure, Metadata, and Semantics, Release 1

George W. Beeler, Jr.
Co-Chair, Methodology & Modeling Committee

1 Introduction

2 HL7 tooling structure

2.1 Current Tools architecture

2.2 Tool overview

2.3 XML Model Interchange Format

Note that several of the critical tooling steps involved in publishing the specifications are dependent upon the generation and exchange of XML files. The initial definition of XML DTD specifications for these files was based upon a set of publishing document definitions created by W3C for use in publishing its own XML specifications. HL7 volunteers expanded those DTDs to serve the requirements of publishing HL7 Version 3. This required the addition of XML element hierarchies to represent such things as the Reference Information Model, vocabulary, RMIMs, and the core domain content for publication. The currently used DTDs are published in Annex B of this document.

Although each of these specifications built upon the core foundation of the W3C DTDs, there was no attempt to design and maintain an overall top-down design for these documents. Over time, the lack of a fixed, carefully-defined format for these documents began to impede the development of additional tools. For example, the DTD that controlled the specification of an RMIM this was being influenced both by the requirement for Schema Generation, as well as by the requirements for comprehensible and comprehensive publication of the contained HMDs.

Changes made to the DTD to address publication issues ran the risk of "breaking" the tools used for schema generation, and vice versa.

Accordingly, in the fall of 2002, the Methodology and Modeling Committee undertook a project to develop the fully defined set of XML schema for a Model Interchange Format (MIF) to serve as the foundation for exchanging HL7-defined artifacts between tools within the HL7 development permit. This project was spurred by the efforts of developers and the United Kingdom and Canada to participate in tools development, as well is to extend those tools to meet new needs.

The resulting Modeled Interchange Format is reviewed in detail in a separate section this document. Despite the fact that the MIF is still incomplete, it has already had a significant impact on the direction and development of HL7 tools. The stated objective within the Methodology and Modeling Committee is that each tool in the future should be capable of using the MIF file as its input or source, and of emitting a MIF file as its output. In this way, developers can undertake to develop new tools independent of the particular architectural nuances of the associated tools. With XML as the lingua franca of HL7 tooling, and with a robust set of schema definitions in the MIF, it is possible to establish a relatively open software environment to support of HL7.

3 Current HL7-specific applications

4 Model Interchange Format

4.1 Introduction

The concept of the Model Interchange Format (MIF) was introduced in section 2.3 above. This portion of the tooling documentation provides an overview of the MIF, and a specific guide to delving into the details of the MIF.

The MIF embodies much of the meta-data and semantics HL7 Version 3 design artifacts and is expanding to encompass all such artifacts. Fortunately, the MIF has been designed to be self-documenting, particularly when used with "tools" that provide graphic representation of the schema constructs, such as XML Spy. This document seeks to be the reader's-guide to the MIF, leading the reader to the details of the meta-data. To this end, exemplary excerpts from the MIF schemas will be included here. The full content of the MIF schemas is published as Annex C of this document.

4.2 Overview & Management

MIF stands for 'Model Interchange Format'. It is a set of inter-related schemas that define the set of primary artifacts that may be developed or exchanged as part of HL7 Version 3 standards development and implementation.

4.2.1 Purpose

The initial construction of the MIF started with two objectives:

- Provide a common exchange format for use between tools, standard developers and repositories. The intention was for a single definition to encompass all artifacts.

- To provide the most complete view to date of 'what artifacts HL7 produces,' and 'how those artifacts inter-relate' and to use that information as a primary input to the HDF (HL7 Development Framework) project.

The second objective grew to a broader purpose, namely:

- To document the meta-data for the artifacts specified as part of HL7's development frameworks – both the older MDF and the evolving HDF. In short, to serve as a meta-model for the HL7 Version 3 development methodology, including the "rules" that determine when an artifact is "well formed."

4.2.2 MIF Development

Lloyd McKenzie of HL7-Canada undertook the initial development of the MIF for the Methodology & Modeling Committee. Over the course of the last 12 months, he has delivered for review three revisions of these schemas.

As of March 2004, the specification for the "static" models -- RIM, RMIM, HMD, data types, and CMETs -- is considered to be complete. These schemas, all along with the foundation schemas for enumerations, patterns, and publication markup have been "released for use" by tool developers.

Work continues on the representation of the remaining critical artifacts -- vocabulary specifications, behavioral specifications (application roles, trigger events and interactions), and ballot packages.

4.2.3 Principles for Adopting Changes to MIF

In the course of reviewing the MIF versions, the Methodology and Modeling Committee began to evolve "principles" for excepting changes to the meta-model and MIF representation. The following set of principles has been proposed, but not finally accepted by M&M for this purpose.

All proposed changes to the HL7 meta-model should be approved by the Modeling and Methodology committee. Meta-model changes include:

- creation of new types of artifacts;
- changing the characteristics of an artifact (adding new elements or relationships); and
- deprecating artifact types

Changes will typically originate from standards developers, tool-smiths or the M&M committee. In rare cases they may arise out of the effort to resolve a negative ballot.

Because these changes can impact both the tooling supporting HL7 development and the process used to publish HL7 artifacts, both the Tooling Committee and the Publishing Committee must have the opportunity to participate in deliberations on proposed changes. As a general practice, the International Committee should also be consulted because several of the affiliates have local publication and tooling efforts in place.

Criteria to be considered when considering a proposal include:

- Can the desired functionality be achieved in some other manner within the existing meta-model?
- How essential is the desired change to the approval and implementation of HL7 Version 3 standards?
- What is the necessary timeframe in which the proposed change must be rolled out to achieve the desired impact on the approval and implementation of HL7 Version 3 standards?
- How much impact will the proposed change have on existing artifacts, publication processes or tooling?
- If existing artifacts are affected, is it possible to automate the process of updating those artifacts to reflect the change.
- Does the proposed change align with the existing model architecture?
- Is it possible to express the proposed change as part of the HDF UML profile?

In evaluating these criteria, M&M must come to three decisions:

1. Whether the proposal should be accepted at all
2. Whether the change should be incorporated as submitted, or whether modifications are needed.
3. What is the timeframe for inclusion? Possibilities include:
    - immediate (changes to tooling implemented prior to the next ballot cycle)
    - next ballot cycle (changes to tooling made in time for the following ballot cycle)
    - next release (changes to tooling not made until the next major 'release' of version 3.)

4.3 MIF Design process

4.3.1 Sources

As noted above, the purpose of the MIF used to support two primary functions -- consistent representation of HL7 artifacts to support tooling, and formal representation of the meta-data defined in the HL7 Version 3 development methodology. These are mutually compatible requirements, because both the tooling and publication of HL7 Version 3 specifications are governed by the underlying development methodology.

Accordingly, the initial releases of the MIF drew their material from several sources:

- **Message Development Framework:** The Version 3 Message Development Framework formed the basis for early development of the standards. It was documented with a meta-model developed and published most recently in 2000. Additions to this meta-model were assembled and documented within the Methodology and Modeling Committee as an initial resource for the HL7 Development Framework (HDF). These documents were a critical starting point for MIF development.

- **Existing HL7 DTDs:** The current publishing and tooling process depends upon XML DTDs that define such elements as the HL7 RIM, vocabulary, HMDs, etc. Although these sources are secondary to the MDF, they provided additional documentation to the metadata particularly as it represents the document structure and markup used by the Publishing Committee.

- **HL7 Development Framework:** the evolution of the HL7 HDF has been contemporary to the development of the MIF. As one of the objectives are the MIF is to serve as the meta--model for the HDF, each step along the path of HDF development, has been noted, and included in the MIF representations.

- **HDF UML Profile:** part of the HDF project has been to develop a formal UML profile of the HL7 Development Framework. This effort involves mapping each of the HL7 artifacts to UML, and incorporating in the extensions in a properly defined profile. This mapping as significantly influenced the structures and representations selected for use in the MIF. Thus, element names and relationships within the schemas of the MIF, mirror, in many cases, equivalent structures in the HL7 UML profile.

- **Identified Requirements:** Requirements for desired enhancements to the HL7 metamodel were also gathered from various tool users, as well as from documents prepared by the International Committee, Template Committee and Conformance Committee.

Finally, and it should be noted that the MIF development has had its own positive feedback effect upon the HDF. Design of the MIF to meet particular tooling purposes, frequently lead to the need for additions, such as graphic representation of designs, to the UML profile for the HDF.

4.3.2 Schema structures

The specification of XML schemas affords a design environment very similar to that used in building object models. Elements have properties that may be simple attributes, or other elements. Elements may be built in hierarchies where the derived element "extends" the referenced element, or "restricts" its source. Inclusion of elements within a parent element may be done as single elements, sets of elements, sequences of elements, or choices of the preceding.

Further, each attribute and each element within the schema may be assigned a type that determines or constrains the representation that may be used within that attribute or element. These types may include patterns and enumerated lists among their constraints. The art of designing the MIF resides in the appropriate selection of meta-data to include in the each type specification, and in the assembly of the element hierarchy. Each of the structural capabilities has been used in defining the MIF schemas.

4.3.3 Schematron rules

The structures inherent in XML schemas provide for the representation of most of the critical relationships and rules that need to be embodied in the meta-model. There are, however, rules that go beyond the ability to express them within the XML schema constructs. In order to be able to capture and process such rules, Schematron rules have been added to the MIF schemas. Schematron is "a language for making assertions about patterns found in XML documents". It is not designed to replace XML schemas, but rather is designed to be embedded within XML schemas in order to represent rules and patterns that are beyond the capability of the XML schema specification. When an XML instance that claims to be conformant with a particular schema is presented, XML schema validators can check that instance against the schema specification, and a further validation check can be run with a Schematron validator to verify that rules embedded in the schema in the Schematron language are also being adhered to.

4.3.4 Internal documentation

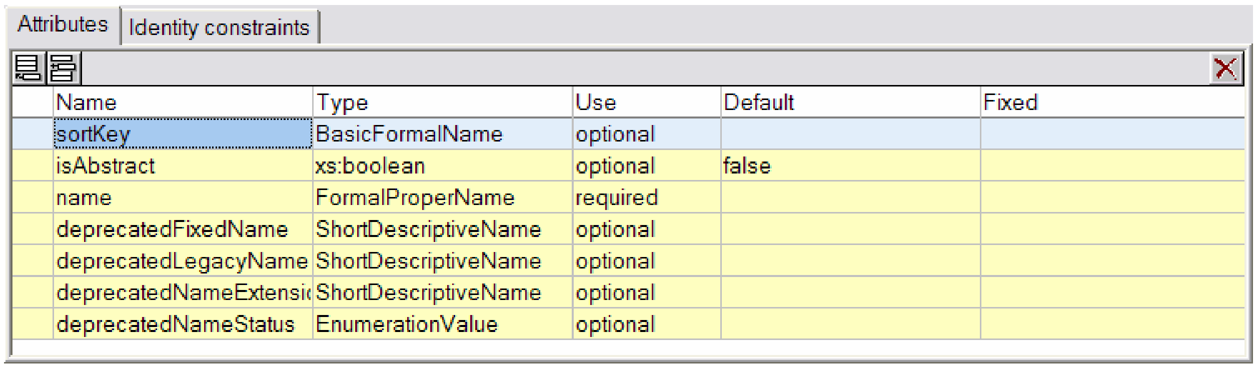
Finally, the MIF schema design has taken extensive advantage of the schema elements `<xs:annotation/>` and `<xs:documentation/>` to provide detailed descriptions of the design elements that can be viewed in the diagrams, and which will ultimately be extracted to provide an overview of the MIF schemas in HTML.

4.4 MIF & Documentation example

As an example of how the various facets of MIF design contribute to a specification, and of how these can be used by a "student" of the MIF, this section looks closely at the MIF content of a particular element – a "class" in a 'flat' static model. This would represent a "cloned class" in an RMIM clone. Graphics and text markup were extracted from a review of this schema in XML Spy.

4.4.1 Graphic representation

The figure below shows a "screen shot" from XML Spy of a portion of the graphic representation of the type definition for type "Class" in MIF schema "mifStaticModelFlat." This is the type that describes a class "clone" in an HL7 RMIM



Name	Type	Use	Default	Fixed
sortKey	BasicFormalName	optional		
isAbstract	xs:boolean	optional	false	
name	FormalProperName	required		
deprecatedFixedName	ShortDescriptiveName	optional		
deprecatedLegacyName	ShortDescriptiveName	optional		
deprecatedNameExtensi	ShortDescriptiveName	optional		
deprecatedNameStatus	EnumerationValue	optional		

Figure 1 Attribute panel from XML Spy graphic depiction of "class" element in mifStaticModelFlat.

This panel displays all of the attributes of the element, the attribute name, its type, whether it is required and whether it has either default or fixed values. There is one required attribute, the name of the class. Although they do not appear in this XML Spy view, attributes in the schema also have descriptions of their function. Further, XML Spy contains "helper windows" (not shown here) that show additional characteristics such as maximum length, patterns and enumeration values.

Note that each of the types, with the exception of "xs:Boolean" references an HL7-specific type definition defined elsewhere in the schmas. For the most part, these types represent constraints around the pattern of allowed values, the length of the value, or the allowed values (enumerations). Enumerated types have names that end with "Kind."

The order of display in this panel has no semantic import. Attributes appear in the order in which they were brought into the element definition. Most of the element definitions of the MIF have been built by extending more basic types. Thus the order of extension tends to determine the order of appearance in Spy.

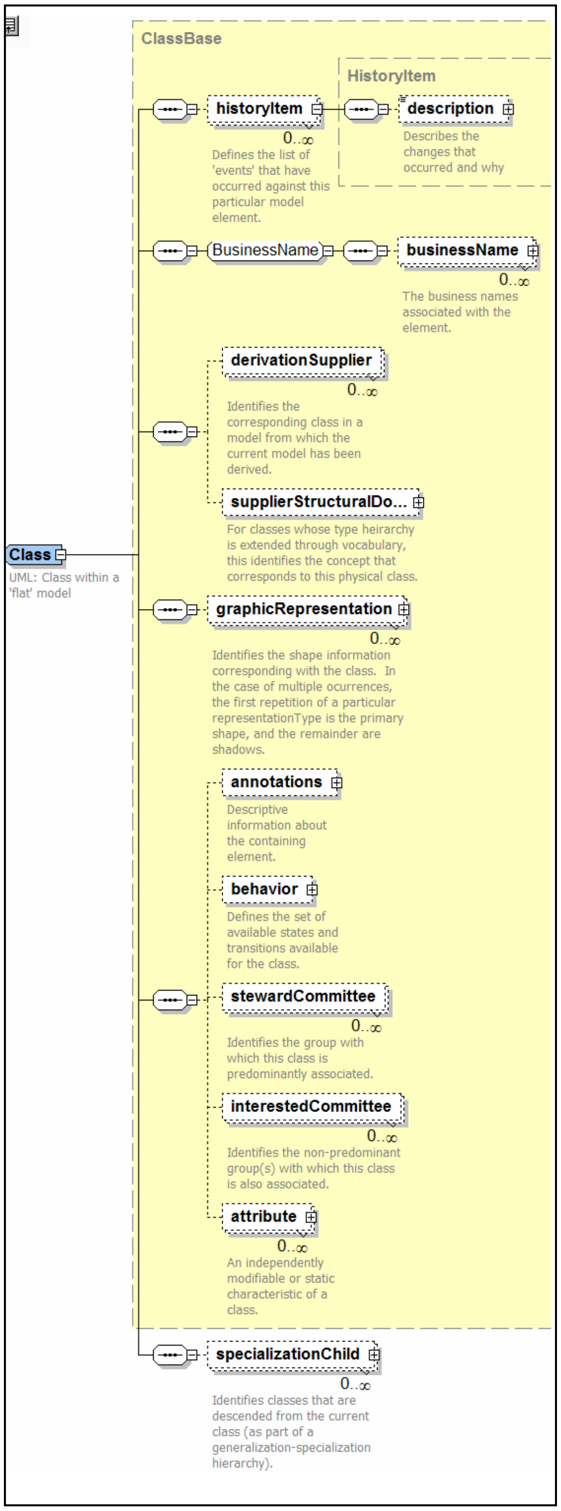
The figure at the right is another "screen shot" from XML Spy showing the element content of the type "Class" in schema "mifStaticModelFlat." This is the upper portion of the previous graphic.

The diagram shows the elements that comprise the "Class" type as rectangular boxes. It shows further that the type has been defined by extending the "ClassBase" type definition (yellow box) by adding an optional, repeating (dotted line and multiplicity 0..∞) "specializationChild" at the bottom of the figure. Note that the documentation for each of the elements appears directly below the element.

The element model of this type contains:

- a set of historyItems to track the development history of the class
- a set of businessNames (familiar or language-specific) names. (The clone name is captured in an attribute.)
- the derivation from DMIMs or the RIM
- data to enable graphic reconstruction of the RMMIM
- descriptive annotations
- a state model (behaviour)
- lists of steward and interested committees
- a set of attributes, and
- a set of specializations.

In Spy, clicking on any of these elements will expose its attributes in a separate panel. If the element has a "+" symbol on its right hand end, clicking this will expose the elements in the type-structure for that element.



Defines the list of 'events' that have occurred against this particular model element.

Describes the changes that occurred and why

The business names associated with the element.

Identifies the corresponding class in a model from which the current model has been derived.

For classes whose type heirarchy is extended through vocabulary, this identifies the concept that corresponds to this physical class.

UML: Class within a 'flat' model

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.

Descriptive information about the containing element.

Defines the set of available states and transitions available for the class.

Identifies the group with which this class is predominantly associated.

Identifies the non-predominant group(s) with which this class is also associated.

An independently modifiable or static characteristic of a class.

Identifies classes that are descended from the current class (as part of a generalization-specialization hierarchy).

Figure 2 Graphic representation of the XML type "Class" in schema "mifStaticModelBase" as shown in XML Spy.

4.4.2 Textual documentation

A series of excerpts from individual scheams in the MIF heierarchy follows. The snippets in this series were selected to illustrate the form of the documentation, and where to find HL7-specific documentation.

The first snippet (below) comes from the beginning of schema "mifStaticModelFlat". The opening annotation of the schema (and of all others in the MIF series) identifies its purpose, and lists the schema(s) that it "includes." In this case, if one wishes to "drill down" the schema "mifStaticModelBase" provides the foundation for "mifStaticModelFlat."

```
<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 a flat (non-serialized) view of a static data model
********************************************************************************
    </xs:documentation>
   </xs:annotation>
   <xs:include schemaLocation="mifStaticModelBase.xsd"/>
```

4.4.2.1 Drilling down from type "Class"

The second snippet (below) shows the schema definition for the same type, "Class," that is displayed above in graphic form. Note that the documentation that appears on the SPY diagram comes from the `<xs:annotation/>` and `<xs:domumentation/>` elements of the schema specification. Most such element notations include two `<xs:domumentation/>` elements, one of which begins "UML: " These elements carry information to facilitate mapping the MIF concepts to the HL7 HDF UML profile. In this snippet, the sixth line identifies that "Class" extends "ClassBase" and the following eight lines provide the extension

```
   <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>
```

Drilling down into schema "mifStaticModelBase", we find that it is based on (includes) "mifStaticBase". However the type specification for "ClassBase" in "mifStaticModelBase." Portions of that definition follow.

4.4.2.2 to type "ClassBase"

The first snippet from the "ClassBase" type definition exhibits an instance of the declaration of Schematron rules. The Schematron declarations are captured in element in the namespace "sch" (elements whose names begin "sch:".) Here, we see defined a pattern of rules named " Check class attributes and elements" which contains one rule that may include the execution of one or more of four reports. The intent of these reports is readily discerned from the error message that the report contains. In this case, the reports are testing that an abstract class will have at least one descendant; that the primary subject area is declared if any subject areas exist; and will issue warnings if the class lacks attributes and associations or has an unusual name. Note that these are not rules that could have been embedded in the schema directly. For example descendants must be optional since most classes do not have specializations, but if a class is "abstract" (an attribute value) then there must be at least one descendant. Finally, this snippet shows that the "ClassBase" type is an extension of the "ClassRoot" type that happens to also be defined in "mifStaticModelBase."

```
  <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">
```

```
The second snippet from "ClassBase" (below) outlines the extensions of "ClassRoot"
that are involved.  Both elements and attributes are being added.  At the top, a
sequence of several elements including a set of annotations, a complete state machine
and a set of attributes is being added.  (Note that the occurrence of four dots at the
start of a line indicate that one or, more probably, several lines have been dropped
from this snippet.) In the second half of the snippet attributes are being added to
the element, including the "name" of the class.  Note, again, the binding to
representation of the same concept in the HDF UML profile.
```

```
        <xs:sequence>
          <xs:element name="annotations" type="ClassAnnotations" minOccurs="0">
....
          <xs:element name="behavior" type="StateMachine" minOccurs="0">
....
          <xs:element name="attribute" type="Attribute" minOccurs="0"
maxOccurs="unbounded">
....
          </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:extension>
    </xs:complexContent>
  </xs:complexType>
```

4.4.2.3 to type "ClassRoot"

The next snippet also comes from "mifStaticModelBase" and shows the type "ClassRoot" which was extended above to create "ClassBase". In this case, the elements for graphic representation of the class are being added as an extension to type "ClassifierBase." Note the influence of the UML profile in the naming of the MIF terms "ClassifierBase" and "ClassNodeSemanticModelBridge."

```
  <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:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

4.4.2.4 to type "ClassifierBase"

Continuing the process of drilling down, we find that "ClassifierBase" is itself an extension of "Classifier" As shownb in the following snippet, "ClassifierBase" adds the concepts of "derivation" – the element or class (in this case) from which the clone is derived – and "supplierStructu8ralDomain" – the binding of a class to its classCode vocabulary domain.

```
  <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:annotation>
    <xs:complexContent>
      <xs:extension base="Classifier">
        <xs:sequence>
          <xs:element name="derivationSupplier" type="ClassDerivation" minOccurs="0"
maxOccurs="unbounded">
....
          <xs:element name="supplierStructuralDomain" type="DomainSpecification"
minOccurs="0">
....
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

4.4.2.5 to type "Classifier"

The next step in the drill down takes us to type "Classifier" in "mifStaticBase." ("mifStaticBase" is built on or includes "mifBase.") At this stage, the schema is extending type "ModelElement" by adding a sequence of business names as elements and by adding attributes that allow the classifier to be generalized.

```
  <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>
```

4.4.2.6 and (finally) to type "ModelElement"

The final step in the drill-down process takes us to the last snippet (from "mifBase") that contains the definition of the "ModelElement" type. As its name suggests, it is based on the UML ModelElement, and has but one contained element, a set of history items expressing an instances revision history, and a single attribute, a sortKey.

```
  <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:element>
    </xs:sequence>
    <xs:attribute name="sortKey" type="BasicFormalName" use="optional">
....
    </xs:attribute>
  </xs:complexType>
```

4.4.3 The point of this exercise

At this point, the impatient reader might well ask: "Was it really necessary to show six schema fragments in order to represent a single type?" Or, phrased another way, "Is it not possible to collapse the hierarchy of types?" The answer to these questions is to be found in the multiple uses to which individual type definitions in the MIF specifications are put. For example, as the documentation for "ModelElement" says, it is the basis for all "semantic" elements in the MIF. Thus this type definition is used throughout the schemas.

Further, one must recognize that there are several "flavors" of representations for a class within the MIF. They may be represented by a complete structure as illustrated above, they may be represented by stubs that replace them with CMETs, and they may be simple references to a class in another environment. Each of these examples requires a variant of the type that represents a class. Most of these variants will be represented as extensions of one of the element types in the hierarchy illustrated above.

Of course, the simple answer to the question is that "reading" the schema specification is definitely **not** the best way to understand them. Rather, reviewing the structure and content with a graphic tool such as illustrated in the previous section with XML Spy, provides an integrated view of the "class" type fully assembled from the extension hierarchy illustrated in the textual documentation.

Finally, it is worth noting the relationship that exists between these type definitions and the corresponding representation of the HL7 artifacts in UML as will be reflected in the HDF UML profile. (Diagrams of similar relationships appear later in this document.) Working from bottom to top we have a hierarchy with the following types:

- **ModelElement:** is the most basic level of both the MIF representation, and of the elements in UML.
- **Classifier:** is an extension of ModelElement in the UML (through Namespace) and directly maps the MIF to UML.
- **ClassifierBase:** this type adds to the Classifier type the concept of derivation from another class, as well as the notion of restriction by the structural vocabulary domain.

- **ClassRoot:** this type further extends the ClassifierBase concept by adding graphic representations in order that the element may be rendered in graphic form in Visio, UML, or other representations. Not all classes within the HL7 MIF hierarchy are so rendered.
- **ClassBase:** this type is directly bound to the UML concept of Class.
- **Class:** this type corresponds to an "HL7 class" as represented in a "flat" model. (A flat model, is the more general representation of an object model, as would normally be seen in UML diagrams and in the HL7 RMIMs.)

4.5 Overall design structure

4.5.1 Schema Hierarchy

As noted previously, the MIF schemas are defined in a refinement hierarchy that develops each new, more complex schema element or element type from a more basic definition contained in a foundation schema that is "included" in the more complex schema. The following figure shows the hierarchy of MIF schemas, as they existed in January 2004.

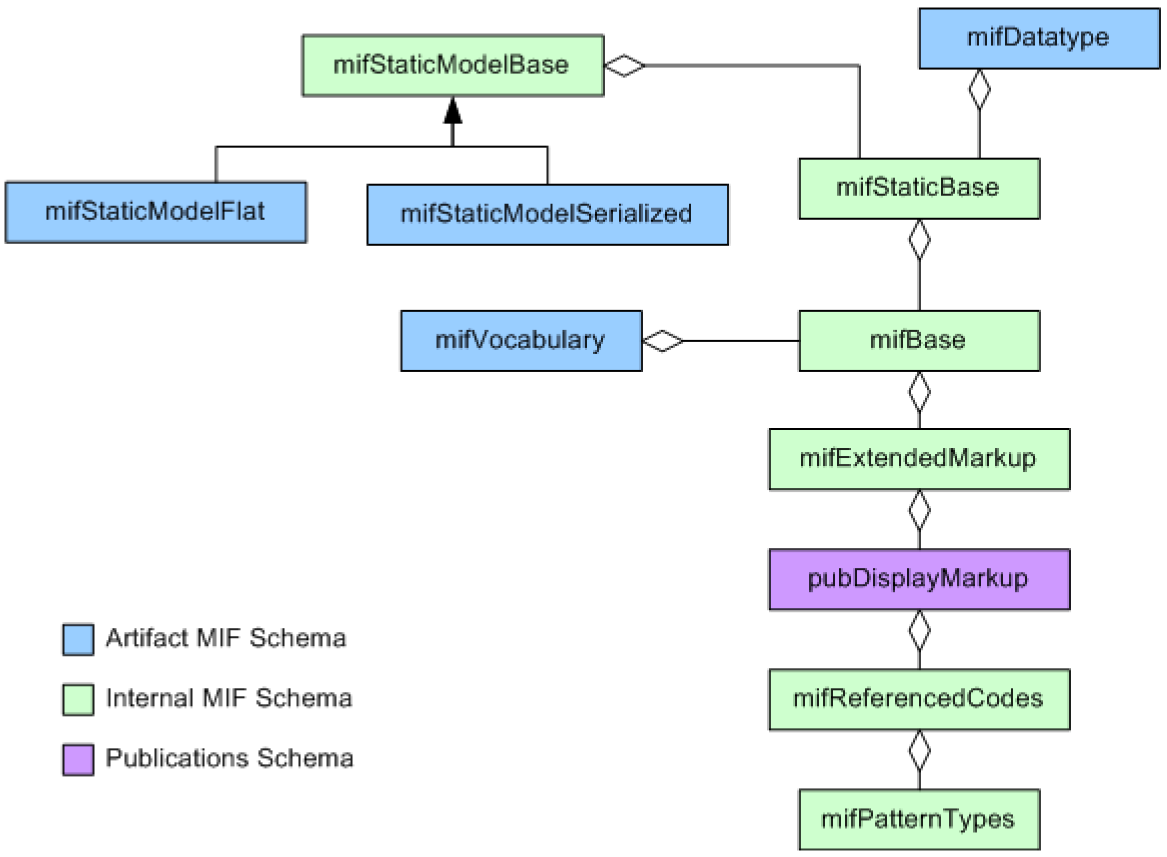


Figure 3 Hierarchy of Schemas in MIF Design

The key in the lower left of the diagram shows the color code used to classify the function of each schema. The blue schemas are the only ones that are actually instantiated to represent HL7 artifacts. The others all provide supporting definitions.

4.5.1.1 Pattern, Enumeration and Markup Schemas

Working from bottom to top, the bottom four schemas define the fundamental "simple" types along with the publication markup that is used for descriptions in the more complex schemas. Specifically:

- **mifPatternTypes:** represents a series of over 30 type definitions used in the MIF. As with the more complex element types, these simple types also build upon themselves to assemble richer patterns. Examples include patterns for OIDs, class names, UpperCamelCase names, etc.
- **mifReferencedCodes:** contains the definitions for all of the enumerated simple types used in the MIF. It contains over 40 different enumeration types. In a few cases, the content may be drawn from a defined HL7 terminology.
- **pubDisplayMarkup:** this schema is primarily controlled by the Publishing Committee and contains text markup for use within descriptive text and annotations in the MIF instances. It includes XML elements to express simple markup (bold, underlined, italics, etc.); reference markup that binds one element to another, or hyperlinks between elements; and complex markup that includes such constructs as lists, tables, etc.
- **mifExtendedMarkup:** for the present, this schema is included in the hierarchy, but has no content. It functions to pass through the content from "pubDisplayMarkup" to schemas above that. In the future, markup specifications contained in the higher-level schemas will be moved to this schema. Such a move is dependent upon critical tools such as XML Spy being able to recognize the redefinition of XML "groups."

4.5.1.2 Abstract Semantic Schemas

The upper seven schemas in the diagram build upon this base and begin to incorporate the semantics that are critical to representing artifacts from the HDF and MDF. Three of these schemas are abstract. That is, they provide semantic definitions for the other schemas, but are not instantiable themselves. These abstract schemas include:

- **mifBase:** defines a set of element types that are used in the higher-level schemas. The primary criterion for including an element at this level is the fact that it is used in more than one context in the higher-level schemas.
- **mifStaticBase:** defines elements that are common to both static models and to data types. Examples include "Classifier" "Interface", etc.
- **mifStaticModelBase:** this schema provides 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. This is an abstract schema. Instances of static models are built from either "mifStaticModelFlat," or "mifStaticModelSerialized."

4.5.1.3 Schemas Instantiated for Artifact Representation

Finally, the four schemas colored blue include elements that can be instantiated to hold HL7 Version 3 artifacts and/or packages thereof. They include:

- **mifDatatypes:** provides elements for defining data types and their properties. It includes an element "datatypeModelLibrary", which captures a complete HL7 data type specification.
- **mifStaticModelFlat:** provides a flat (non-serialized) view of a static data model. Examples of such models are at the HL7 RIM, DMIMs, and some of the RMIMs.
- **mifStaticModelSerialized:** provides a serialized (hierarchical) view of a static data model. Examples of such models include most of the RMIMs in the current HL7 Version 3 ballots, and all HMD's and message types.
- **mifVocabulary:** iss an extension of "mifBase", that defines concepts related to vocabulary. Elements in this schema allow representation of vocabulary elements as multiple separate packages each of which contains one or more: value sets, vocabulary domains, and/or code systems. Each of these subelements has a package location, which allows the individual componments to be repackaged and referenced as needed. This schema has not received as complete review as have the others in this series. However, because of immediate tooling requirements, it was pressed into service in December 2003, with the recognition that it may need to be refined in the future.

4.5.2 MIF-to-UML Profile Relationships

The period of time during which the MIF schemas were developed coincided with further development of the HDF, and the concomitant UML profile to represent HL7 artifacts within UML. In order to track these efforts, and to assure a comprehensible mapping between the MIF schemas and the UML profile, the developer maintained a UML model that represents both core UML constructs and known HL7 constraints or extensions which will be part of an HL7 UML profile.

These UML models have not been reviewed formally in the Methodology and Modeling Committee, but they have proven quite useful as aides to interpreting the MIF schemas, particularly for people who have a strong background in UML, but only a modest exposure to XML schemas.

Therefore, the available diagrams have been included as part of this documentation. In the figures that follow there are four UML diagrams that represent the content of the following MIF schemas: mifBase, mifStaticBase, mifStaticModelBase, and mifDatatypes.

In these diagrams, the following conventions are used to distinguish between elements that stem from UML, as opposed to those that represent HL7 restrictions or extensions. In particular, each class within the model carries a namespace prefix and has color-coded lines. Classes with a "uml" name prefix and blue lines represent elements in UML 1.5; classes with a "di" name prefix and green lines are drawn from the UML specifications for diagram interchange to be released as part of UML 2.0; and classes with an "HL7" name prefix and black lines are part of the HL7 profile.

A apecific binding to MIF schemas is not shown on these figures, but, as noted earlier, each MIF element includes a UML binding notation, and frequently, the MIF element name corresponds to the UML name.

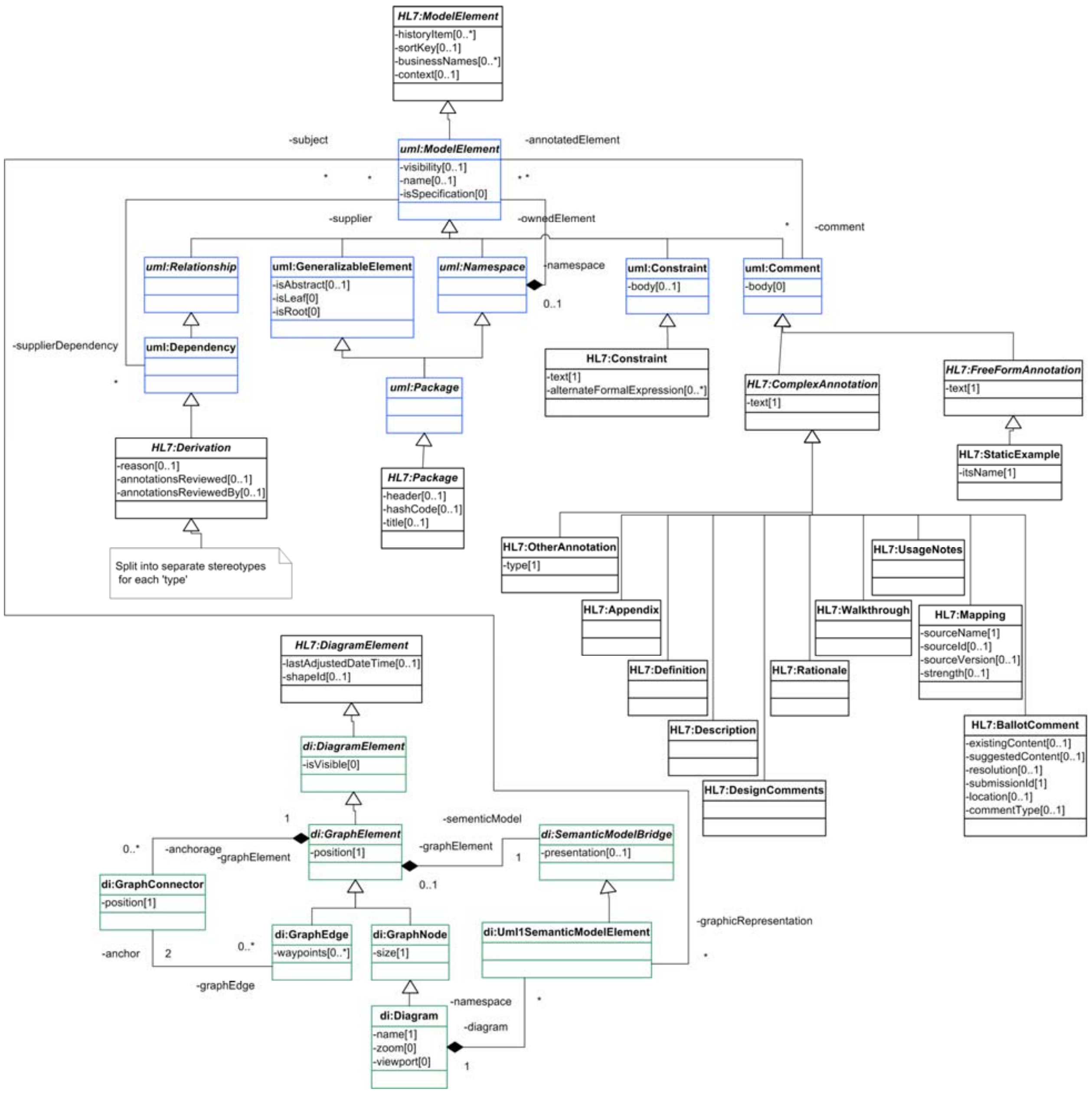


Figure 4 Schema mifBase *vis-a-vis* UML

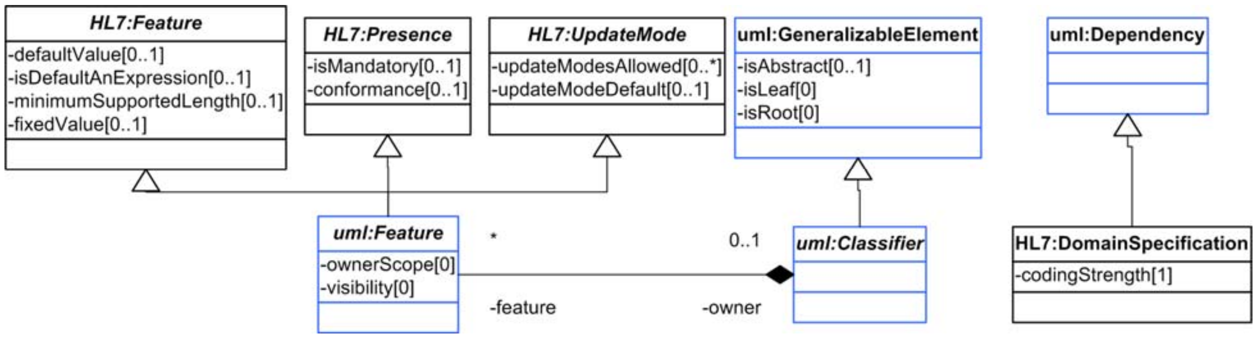


Figure 5 Schema mifStaticBase *vis-a-vis* UML

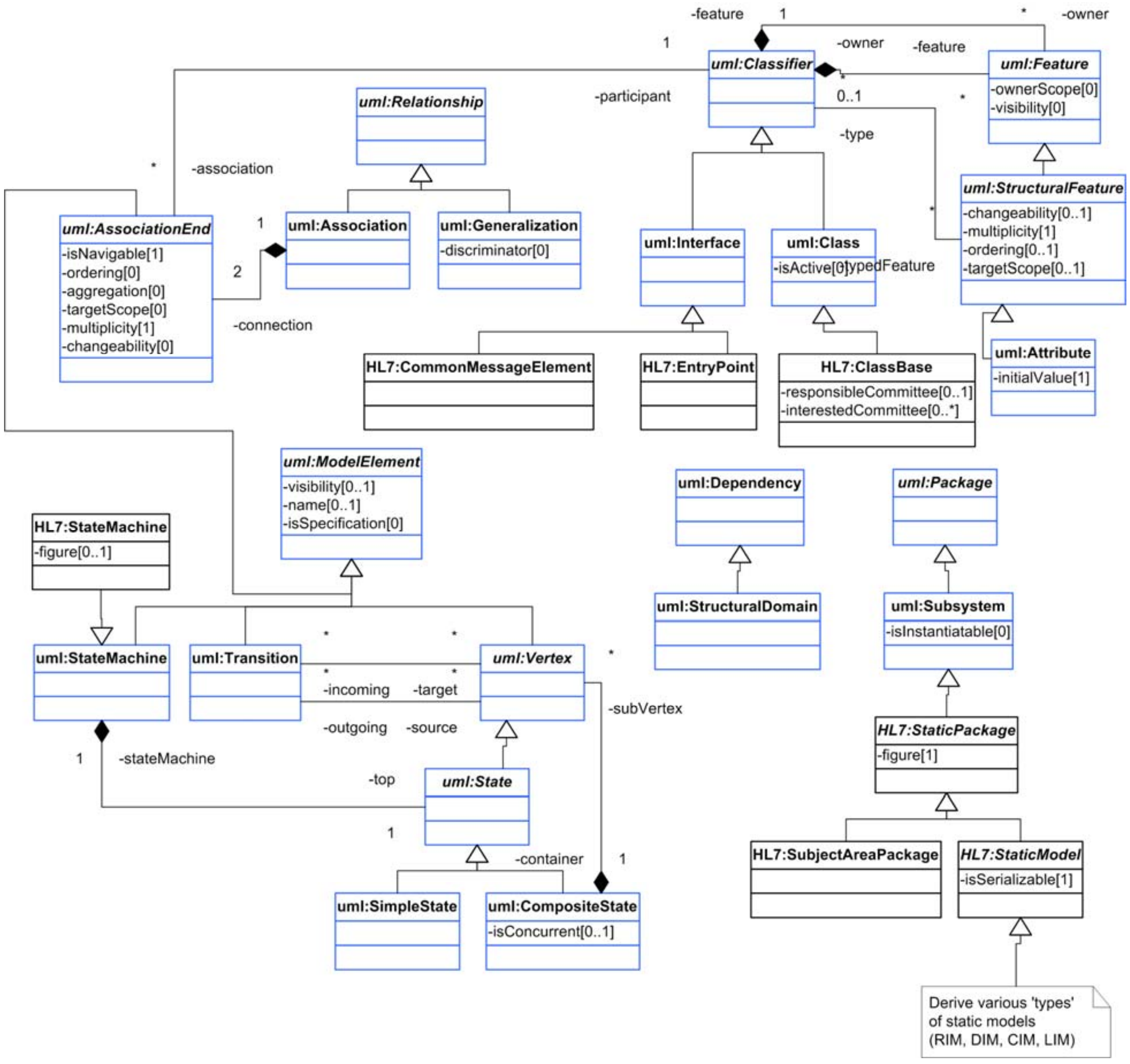


Figure 6 Schema mifStaticModelBase *vis-a-vis* UML

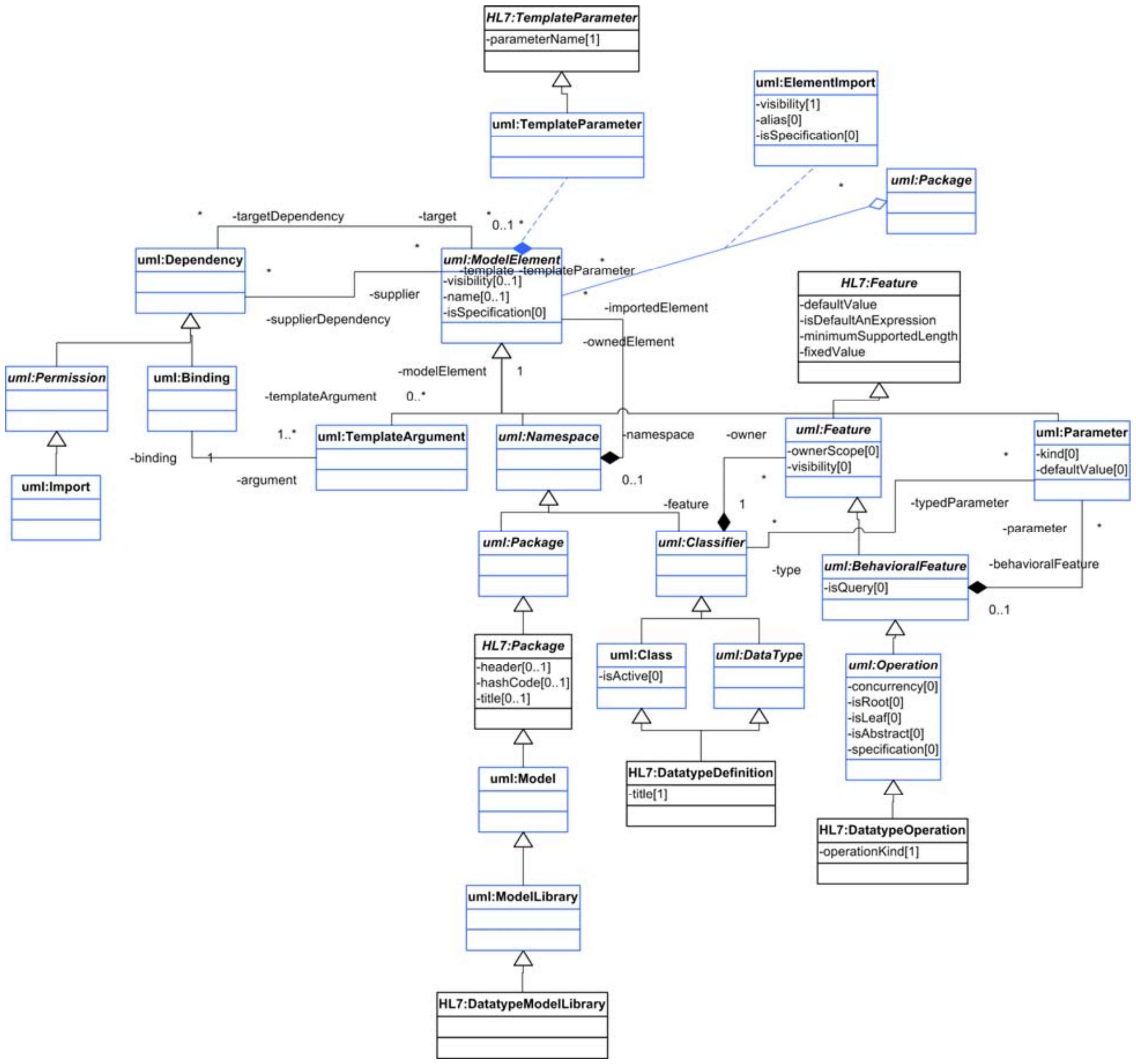


Figure 7 Schema mifDatatype *vis-a-vis* UML