

Importing MIF to UML and Generating XSD

Veteran's Administration Research Report

Dave Carlson

David Carlson & Associates, Inc.
Email: dcarlson@xmlmodeling.com
<http://www.XMLmodeling.com>

Ioana Singureanu

Eversolve
Email: ioana.Singureanu@va.gov
<http://www.eversolve.com>

Introduction

The Model Interchange Format (MIF) was created by members of the HL7 Modeling and Methodology committee as a solution for exchanging message information models between tools and with a repository. In addition, current XML Schema generation tools used for HL7 messages are written as XSLT transformations from the MIF XML documents to XSD syntax. The Unified Modeling Language (UML) was considered and rejected by HL7 members because it was believed to be too limited for representing the static and dynamic models used by HL7.

We believe that the UML is sufficient for designing message models and producing XML Schema implementations, and have started a proof-of-concept development tool to show how this can be done. This work was conceived and supported by the Veteran's Administration and, if successful, may be integrated into their message development tools and processes. This paper describes our approach and preliminary results of a work-in-progress.

A primary motivation for this research is to enable use of mature modeling standards that are widely deployed in other application domains. In particular, we are using the latest adopted versions of UML 2.1 and its serialization format XMI 2.1. These are the same specification versions implemented by current modeling tools (as of January 2007), such as IBM Rational Software Modeler (RSM) and Architect (RSA), Sparx Enterprise Architect, Magic Draw, Genteware Apollo, *hyperModel*, and others.

The first section of this paper provides a brief introduction to the Eclipse IDE platform that is used as the basis for integrating MIF transformation with free and commercial UML modeling tools. Within Eclipse, these independently developed tools provide a seamless experience for users. The next section describes details of our proof-of-concept implementation for importing MIF static model definitions into standard UML2 models. We use an HL7 HDF profile in UML to extend the model with MIF-specific metadata; this is done in a standard way that should be accepted by any UML 2.1 compliant tool. We then describe how XML Schemas are generated from the UML model, using the *hyperModel* plug-ins in Eclipse, and show results of a test where we use these generated schemas to validate sample XML documents that are distributed with the HL7 September 2006 ballot.

Eclipse IDE Platform

Most IT professionals are familiar with the Eclipse IDE and its rapid rise to prominence. Eclipse started as a state-of-the-art Java development environment, but its scope has been expanded by many other Eclipse sub-projects and vendor tools built on this platform. These tools include Web development, business intelligence and reporting, software development for embedded devices, and information modeling using UML, OWL, XSD, and metamodel code generation.

It is less commonly known that Eclipse is a generic tool integration platform using a plug-in framework for achieving seamless connectivity between tools created by different open-source projects and commercial vendors. Through the use of these plug-ins, we are able to add new support for transformation between MIF and UML, and integrate that capability with several existing tools for UML modeling and XML design.

We have developed a proof-of-concept plug-in implementation for importing MIF static model definitions into UML. This will be described in detail in this article. However, before introducing the new tool, we provide a quick overview of other Eclipse-based modeling tools that enable a complete solution for HL7 design without implementing a entire application from scratch.

hyperModel Designer

The *hyperModel* design tool has been under development and in active use for five years. Until now, its focus has been on model-driven design and visualization of XML Schemas, and bi-directional transformation between XML Schema and UML models. *hyperModel* is now being extended for model-driven development of Web Services and messaging applications, creation and use of OWL ontologies, semantic annotations of UML models and XML schemas, and semantic classification of model elements [SWESE 2006].

The design objectives of *hyperModel* are to use:

- Eclipse Modeling Framework (EMF) for model-driven design
- Metamodels defined by current industry standards: UML 2.1, OWL, and XML Schema
- UML profile for XML Schema that enables importing and generating XML Schemas using structures that cannot be defined in UML alone (e.g. differentiating between elements and attributes in a complex type)
- Wizard-driven transformations for importing XSD into UML models and generating XSD from UML, including support for semantic annotations in OWL
- Visualization of large models using an interactive browsing metaphor with class diagrams
- Common UML style interface for viewing and editing UML and OWL models

These design objectives are well aligned with HL7's model-driven development methodology, so it is relatively easy to add the MIF transformation plug-ins into *hyperModel* for a seamless user experience. *hyperModel* has been available as a free download for the past three years (although, it is not open-source), so the integrated tool with MIF import and XSD generation is available for use by HL7 members.

Eclipse Web Tools Platform

The Eclipse WebTools platform (www.eclipse.org/webtools) provides high-quality, open-source tools for editing and validating XML Schema documents, XML instance documents, WSDL interface definitions, and more. For our modeling solution, we use these plug-ins for editing and validating XSD and XML documents, and for generating sample XML documents from new schemas.

Other Commercial Modeling Tools

A growing number of commercial modeling tools are also built on the Eclipse platform using the same project libraries that are used by *hyperModel*, i.e. EMF 2.2 and UML2 2.0. It may be possible to implement one set of plug-ins that can extend all of these tools with support for MIF transformation to UML and XSD generation. We have done this successfully with *hyperModel* and IBM's Rational Software Modeler and Architect (RSM and RSA) version 7.0.

Other modeling tools are not based on the Eclipse platform and its open-source libraries for UML, but they are (or soon will be) compatible with the same versions of UML 2.1 and XMI 2.1 specifications that are used by Eclipse UML2. A popular example in this category is Sparx Enterprise Architect. It supports model import and export using these same UML and XMI specification versions, and recent testing shows that XMI import is 80% complete as of January 2007.

Importing MIF to UML2

The HL7's Model Interchange Format (MIF) contains definitions that are similar to the Unified Modeling Language (UML), but the details of model element structure and names in the MIF metamodel are different and require mapping to their counterparts in UML. The MIF also includes additional attributes and references to the RIM concepts from which the model elements are derived. In this proof-of-concept implementation we show how the MIF structures are mapped to UML and how a UML profile is used to extend the model with attributes that are unique to MIF.

We demonstrate how these features are supported in an Eclipse user interface, in addition to how the model elements are mapped logically from MIF to UML and UML to XSD. In Figure 1 the Project Explorer view presents the complete set of HL7 September 2006 ballot MIF documents and shows a simple menu we added that allows users to import a selection of MIF definitions into a new UML model.

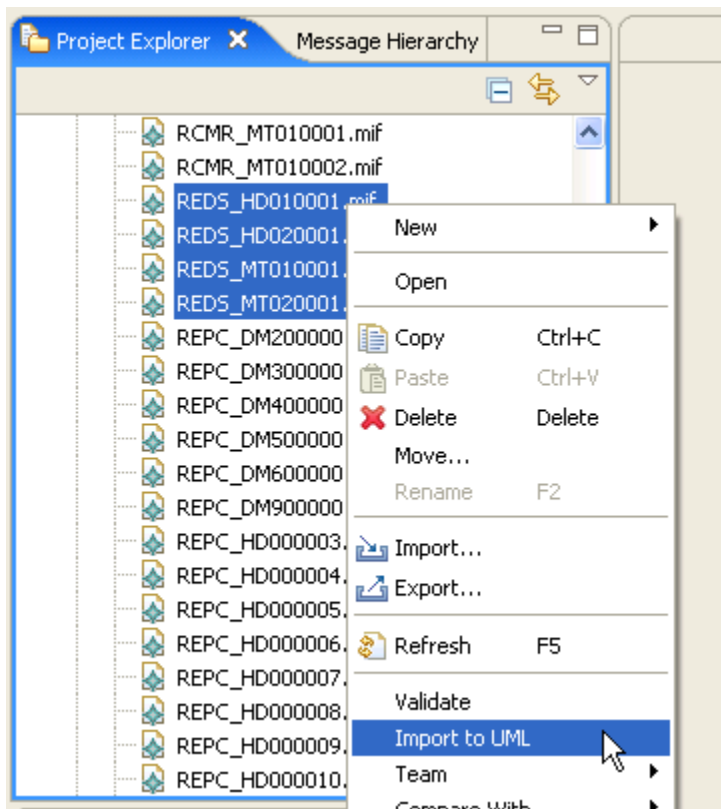


Figure 1: MIF import menu added to the Eclipse project explorer

When a MIF document is imported, the serialized static model definition is transformed to UML using a Java implementation that maps from a MIF metamodel API to the UML2 metamodel

API. To support this transformation, we used the Eclipse Modeling Framework (EMF) to generate a complete Java implementation from the MIF 1.0 schema definitions. The UML2 metamodel Java library is provided by an Eclipse open-source project (www.eclipse.org/modeling/mdt).

Many MIF documents include references to CMET definitions contained in other MIF documents. While importing these definitions into UML, we load the `cmetinfo.coremif` document that is distributed with the MIF files and use it to resolve the CMET references. Requesting one MIF document to be imported into UML may result in several documents being imported, but a MIF document will not be imported more than once if it contains a CMET definition used by several models.

Figure 2 shows a full-screen view of the *hyperModel* tool (with MIF import plug-ins installed) presenting the UML model that was imported from four selected MIF documents. A tree view of the model's packages, classes, and attributes is displayed in the Project Explorer, and a table-style editor is available for viewing and modifying the model.

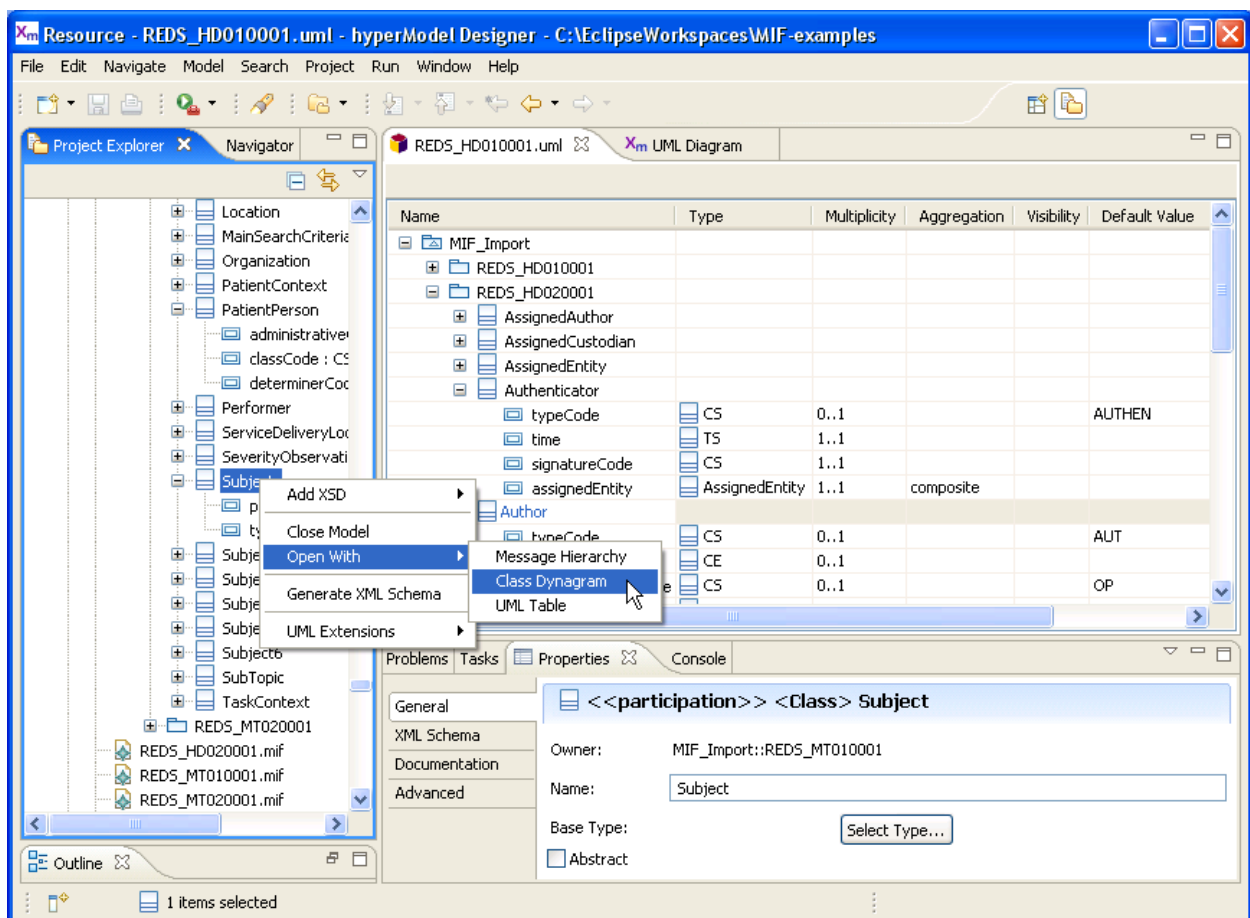


Figure 2: *hyperModel* presentation of UML model imported from MIF documents

The right-click context menu on a model element includes options to open other views and editors. In this example, the Class Dynagram menu is selected and the editor shown in Figure 3 is opened. A dynagram is a feature provided by *hyperModel* that creates a UML class diagram

“dynamically” from any selected class or package. Unlike most UML tools where a class diagram must be constructed by adding and arranging selected classes onto the diagram, a dynagram is created on-the-fly with a presentation similar to a Web browser. You can select any class in the project explorer, the table editor, or a class diagram and drill-down into a new diagram showing that class and its relationships. This is especially helpful when you are exploring a new model imported from another source (e.g. from MIF) where there are no diagrams and you may not be familiar with the model’s design.

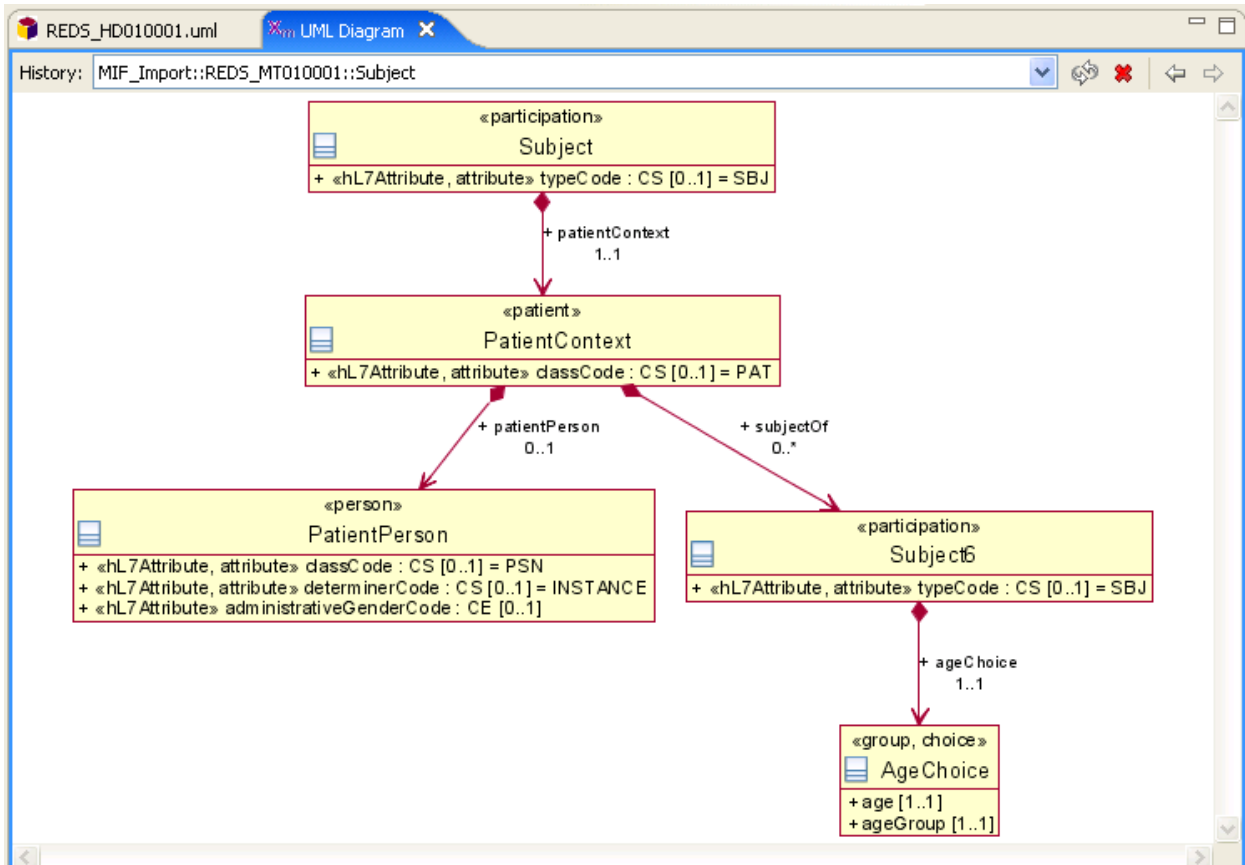


Figure 3: Dynamic diagram (“Dynagram”) created for the Subject class.

The plug-in extensibility of Eclipse allows these capabilities for MIF import to UML and XML Schema generation to be installed into other commercial modeling tools, without any change to the plug-in implementation. Figure 4 shows an identical UML model imported in IBM’s Rational Software Modeler (RSM).

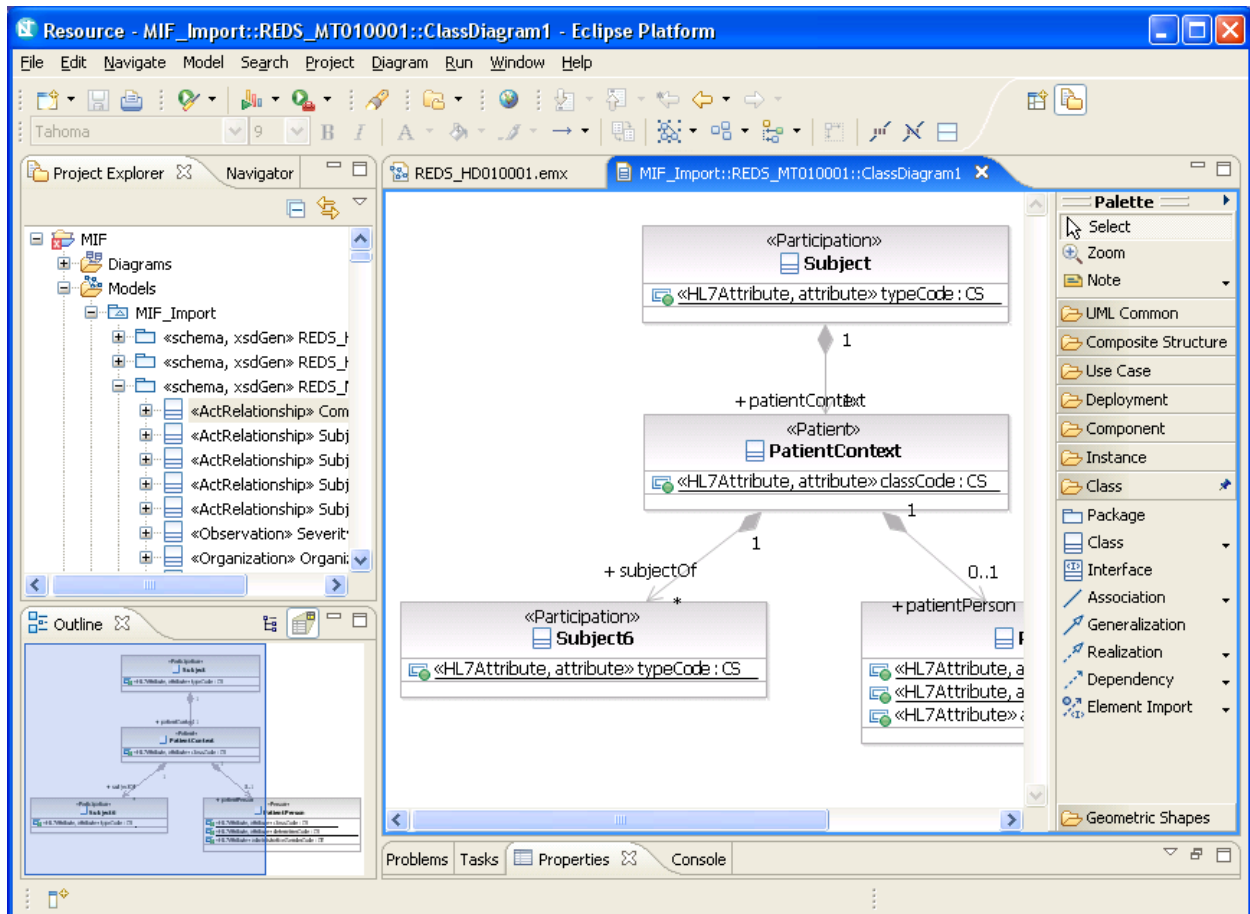


Figure 4: Screenshot from IBM’s Rational Software Modeler (RSM) with MIF plug-ins installed

Mapping MIF to UML

There is no MIF definition of the HL7 core data types, so we used *hyperModel* to import the five core schemas that are included with the HL7 messages. This “HL7 Core Datatypes” UML model is bundled into an Eclipse plug-in and is available for shared use across all HL7 message models, without duplicating the data type definitions in each model. While importing the attribute definitions in a MIF model, data type references are assigned to members of this library model.

The implementation of MIF mapping into UML is not yet complete, but the work done so far is summarized as follows:

- Each MIF file (containing one serialized static model) is mapped to one UML package.
- All nested class definitions in the MIF (multiple levels of nesting depth) are imported such that all are owned directly by the UML package.
- The serialized entry point in the MIF model is used to create global element. This is done by assigning an XSD profile stereotype and elementNameMapping property (as defined by the *hyperModel* UML to schema mapping).
- Structural attributes in the MIF model are represented in UML as static class attributes. If the MIF attribute has a fixed value, then the UML attribute is set to read-only. We are

examining other possible mappings for MIF structural attributes, such as omitting them as UML class attributes and including their values as HDF profile stereotype properties.

A UML profile for HL7 extensions was developed by a previous project [HDF Profile]. In general, a UML profile defines stereotypes and properties of those stereotypes that extend the UML metamodel with additional information not supported directly by the base modeling language. The use of profiles, and their visual presentation in diagrams, is described in the UML specification, and compliant implementations should be interchangeable between UML tools produced by different vendors.

While importing the MIF to UML, the HDF profile stereotypes and properties are assigned to represent the following information:

- Supplier derivation values in a MIF model refer to RIM, D-MIM, or R-MIM classes and attributes from which a model element was derived. As a start toward capturing this information in UML, we added all of the class names from the RIM foundational model as stereotypes in the HDF profile. When a matching class supplier derivation is found in the MIF, the corresponding stereotype is assigned in UML.
- Each attribute in the UML model is assigned the <<hl7Attribute>> stereotype, and stereotype properties are then set for isMandatory, isStructural, and sortKey. Others will be assigned as we complete the mapping.

A UML profile for XML Schema extensions was described in [Carlson 2001] and later enhanced and updated for UML 2.1. The following XML Schema profile stereotypes and properties are assigned to the model (to assist schema generation):

- The <<schema>> stereotype is assigned to each UML package, and the `targetNamespace` property is set to “urn:hl7-org:v3”.
- Each UML package is assigned the <<xsdGen>> stereotype and the `elementNameMapping` property is set to “omitElement”. This setting instructs the XML Schema generator to not generate a global element for each complexType.
- For each MIF class that represents a message entry point, the corresponding UML class is assigned the <<xsdGen>> stereotype and the `elementNameMapping` property is set to “lowerCamelCase”. This instructs the XML Schema generator to produce a global element that may be used as the root of a message document.
- <<attribute>> stereotype assigned to UML attributes that are imported from HL7 structural attributes. This is done to cause their XML Schema generation as XML attributes, but a different solution may be used in future implementation.
- Nested choice groups in a MIF class are modeled in UML the same way that similar structures are represented by *hyperModel*'s XML Schema importer. We may change this in future implementations, but for now it enable generation of XML Schemas with the same structure as produced by current HL7 tools. A nested UML class is added with the <<choice>> stereotype, and a property is added to the owning class in the position where the choice group is located in the content.

The following items are known gaps in the current MIF import:

- Documentation on MIF model elements is not yet imported into UML comments. This is complicated by the use of presentation markup in the MIF text (roughly, a subset of basic XHTML). We are considering alternative approaches for representing structured markup in UML comments.

- Some association reference types are not resolved to other classes in the model. MIF has several ways of representing references and we are still working out the details.
- Additional class and attribute metadata will be mapped into HDF stereotype properties.
- Some supplier derivation references are not contained in the foundation RIM classes and thus not included in the HDF profile stereotypes. We are working on a complete solution for modeling supplier derivations. One possible alternative is to represent all RIM references through use of an OWL ontology and semantic annotations [SWESE].

Generating XML Schemas from UML

We can take advantage of robust, pre-existing tools for generating XML Schemas from our UML models that are imported from MIF. Of course, the same approach can be used when a new UML model is created to define a new HL7 R-MIM, where references are added to CMET models that were previously imported from their MIF definitions.

The most flexible and complete tool available for generating XML Schemas from UML (and importing XSD into UML) is *hyperModel*. It is available for free download from www.xmlmodeling.com. We can install the MIF import plug-ins into *hyperModel* to obtain an integrated environment for message design and implementation. Using the UML model described in the previous section, we can simply open the XML Schema generation wizard as shown in Figure 5, select four model packages (representing the four MIF files that were initially imported), and choose preferred options for schema generation.

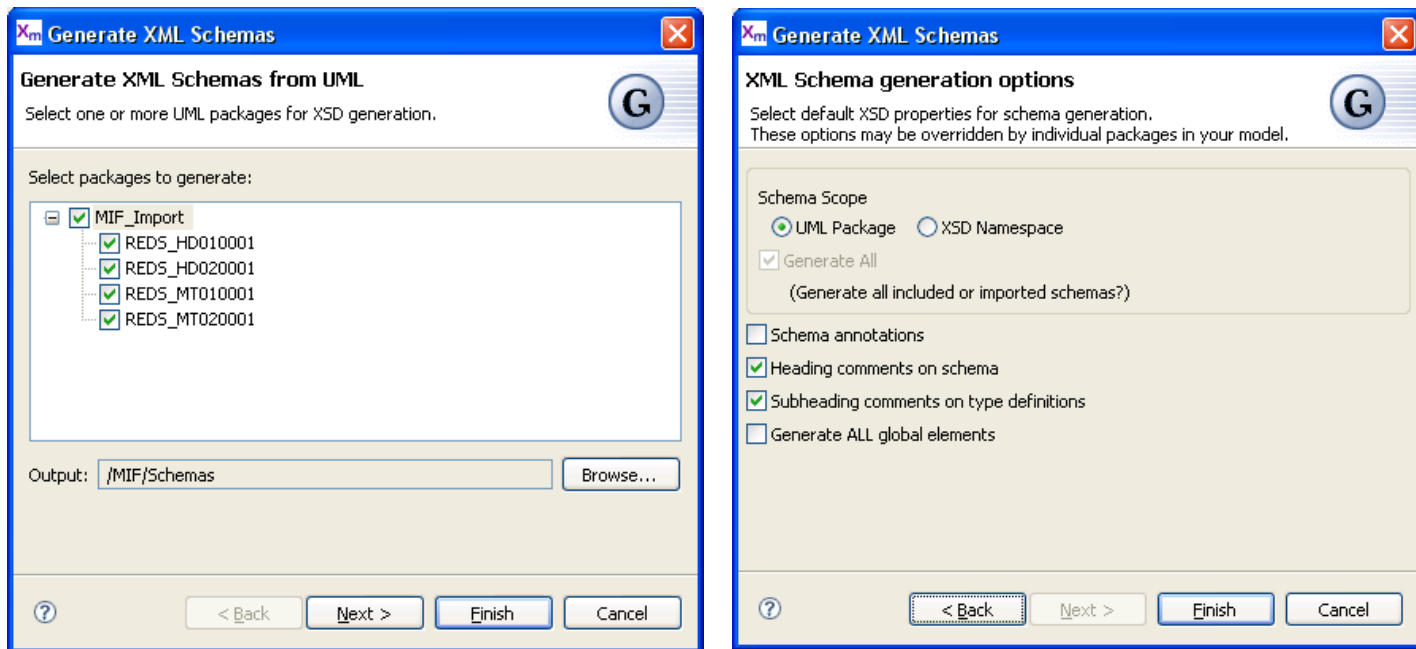


Figure 5: XSD generation wizard

An excerpt of the schema generated for REDS_MT010001.xsd is shown here:

```
<xsd:schema xmlns="urn:hl7-org:v3" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="urn:hl7-org:v3">
  <xsd:include schemaLocation="../../../HL7 Core Datatypes/datatypes-base.xsd"/>
```

```

<!-- ===== -->
<!-- ===== Element Declarations -->
<!-- ===== -->
<xsd:element name="eventNotification" type="REDS_MT010001.EventNotification"/>

<!-- ===== -->
<!-- ===== Complex Type Definitions -->
<!-- ===== -->
<!-- ===== -->
<!-- AssignedEntity -->
<!-- ===== -->
<xsd:complexType name="REDS_MT010001.AssignedEntity">
  <xsd:sequence>
    <xsd:element minOccurs="0" name="name" type="ST"/>
    <xsd:element minOccurs="0" name="certificateText" type="ED"/>
    <xsd:element minOccurs="0" name="assignedAuthorizedPerson"
      type="REDS_MT010001.AuthorizedPerson"/>
    <xsd:element minOccurs="0" name="representedOrganization"
      type="REDS_MT010001.Organization"/>
  </xsd:sequence>
  <xsd:attribute default="ASSIGNED" name="classCode"/>
</xsd:complexType>
<!-- ===== -->
<!-- AuthorizedPerson <<person>> -->
<!-- ===== -->
<xsd:complexType name="REDS_MT010001.AuthorizedPerson">
  <xsd:sequence>
    <xsd:element minOccurs="0" name="name" type="PN"/>
  </xsd:sequence>
  <xsd:attribute fixed="PSN" name="classCode"/>
  <xsd:attribute fixed="INSTANCE" name="determinerCode"/>
</xsd:complexType>

```

Validating Generated XSD

Our schemas generated from UML were produced using a completely different process than those produced by existing HL7 tools for the balloted artifacts. So we need to test whether our generated schemas can successfully validate existing XML documents. For our test, we use five example XML documents that are included with the HL7 September 2006 ballot:

- REDS_EX010001.xml
- REDS_EX010002.xml
- REDS_EX010003.xml
- REDS_EX010004.xml
- REDS_EX020001.xml

The Eclipse WebTools project provides high-quality editors and validators for XML Schema and XML instance documents. We used these tools to validate both our generated schemas and to verify that the five sample documents are successfully validated against their corresponding message schemas.

All five sample documents are validated with no errors using our generated schemas! We know that some schema constructs required by HL7 are missing from our generated schemas (e.g. the common element group and attribute group that are added to every complexType by the HL7 tools), and we need to perform more complete tests on the schemas, but this initial evaluation provides convincing evidence that we are on the right track.

The XML instance editor provided by Eclipse WebTools includes helpful tips and guidance while editing. Figure 6 shows how you can review the defined content model for any element in the document by hovering the mouse cursor over an element. This XML document is being edited with reference to our schema that was generated from the UML model.

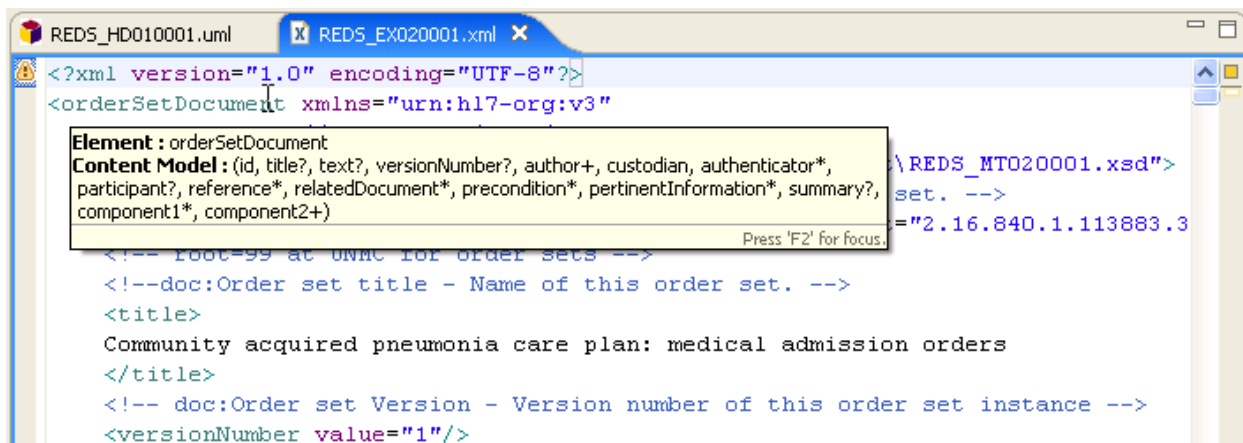


Figure 6: Content assist in the XML document editor (hover over an XML element)

Most Eclipse text editors include a feature generically referred to as “content assist” that is invoked in a consistent way in each editor. Press the “Ctrl and Space” keys simultaneously to open a pop-up list showing content that may be inserted at that point. Figure 7 shows possible elements that may be inserted into a new document we are creating. Elements in the list that are shown with bold font are valid at this point, whereas elements shown non-bold are part of this element’s content model, but not valid at this insertion point. When you select an element from the list it is added to your document.

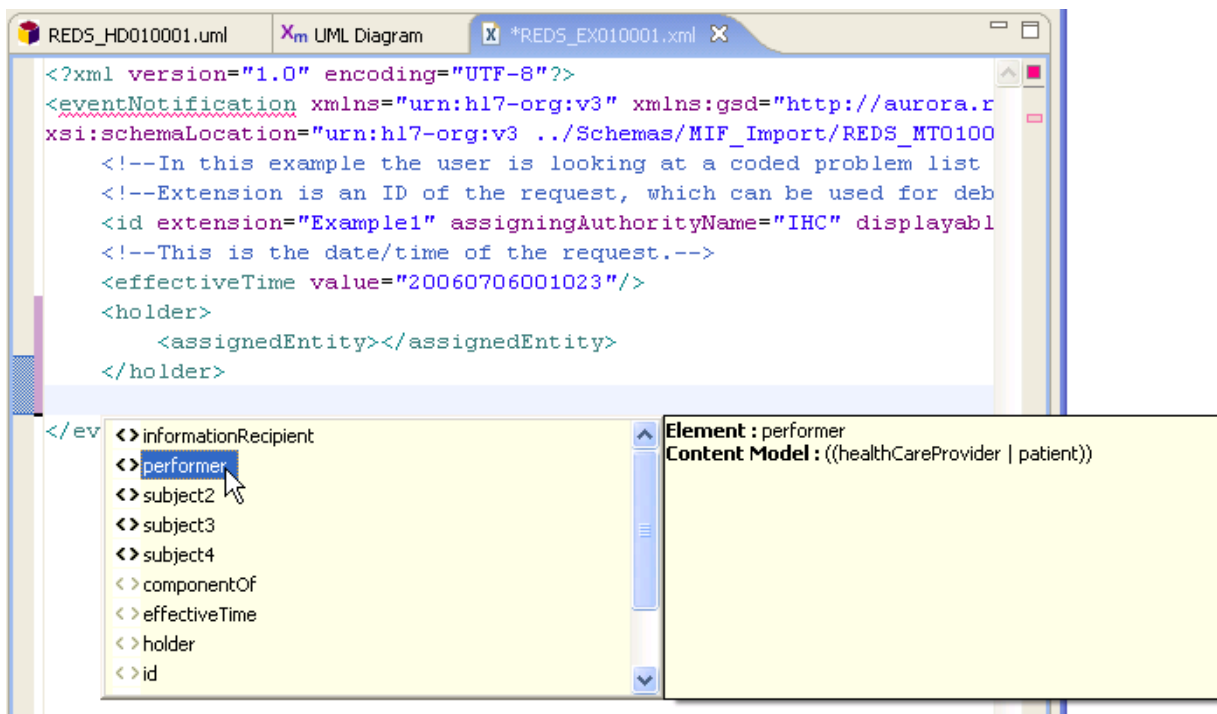


Figure 7: Content assist when inserting a new element (press Ctrl+Space)

Alternatively, the XML editor has a table view for editing documents. Figure 8 shows this view with a context menu used to insert a new child element under <author>. As in the previous figure, this editor is guided by our schema generated from the UML model, but the insertion menu only includes valid elements, so only <functionCode> is offered as a child.

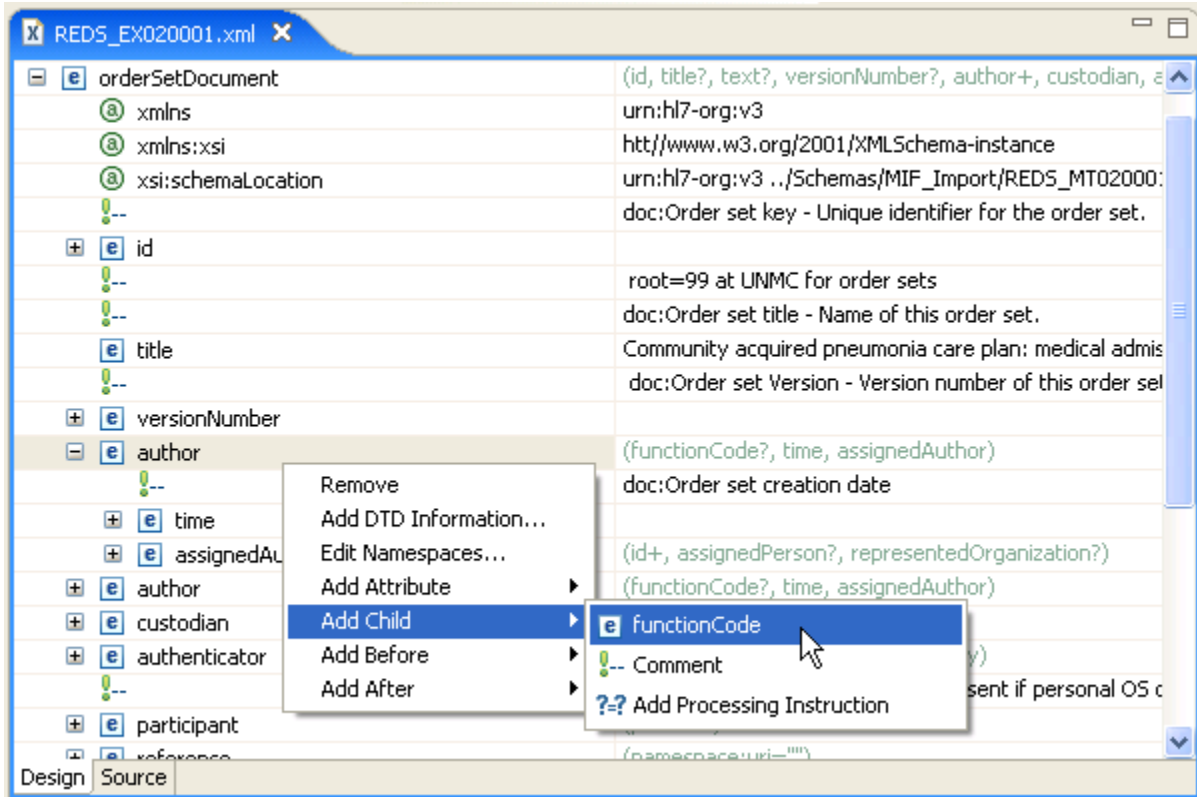


Figure 8: XML editor table view

Next Steps

This work was conceived and supported by the Veteran's Administration to investigate use of the UML modeling language and tools for creating message models. Although planned as a proof-of-concept design and implementation, the results are sufficiently complete and robust that we consider this to be the first iteration of a process and tooling that can yield a production quality implementation.

Some deficiencies of the current MIF import were described in previous sections. Other new features or extensions to this work are:

- The XSD generation is currently based on a generic implementation provided by *hyperModel* and has no logic specific to HL7 design rules. This is a benefit in that general-purpose tooling can be used that is maintained and supported by users across many industries. To support message schemas that are fully compliant with the HL7 requirements, we will customize schema generation, where necessary, through use of an extension point that will be added to *hyperModel*.

- We will create a proof-of-concept implementation of a MIF exporter from UML. The motivation is to allow UML to be used as a message modeling tool, but still use MIF as a model interchange format with existing HL7 repositories and tools.
- We see significant opportunities for leveraging OWL ontologies to represent RIM semantics, model integrity constraints, and derivation of UML model elements from the RIM concepts. Related ideas were described in [SWESE 2006] with application the financial services domain, but most concepts apply equally well to HL7 models. Indeed, other researchers have already created a RIM ontology using OWL [RIM-OWL].

Tool Download

TODO: Provide URLs and instructions for tool download:

- MIF import plug-ins alone (transform MIF files into a UML 2.1 model, but no other functionality). Can install in any Eclipse 3.2 IDE.
- MIF import bundled with *hyperModel* 3.0 and ready to use
- Instructions for adding MIF import and XSD generation plug-ins into Rational RSM/A
- When available, tips and guidelines for using other UML tools and XML interchange with the models imported from MIF.

References

[SWESE 2006] David Carlson, *Semantic Models for XML Schema with UML Tooling*, In Proceedings of the 2nd International Workshop on Semantic Web Enabled Software Engineering, November 2006. <http://www.xmlmodeling.com/articles/SWESE-SemanticModelsForXSD.pdf>

[Carlson 2001] David Carlson, *Modeling XML Applications with UML: Practical e-Business Applications*, Addison-Wesley, 2001.

[EMF] Eclipse Modeling Project. <http://www.eclipse.org/modeling>

[WTP] Eclipse WebTools Platform. <http://www.eclipse.org/webtools>

[HDF Profile] TODO: citation and URL.

[RIM-OWL] HL7 RIM Ontology, created by the W3C HCLS/ACPP Task Force. <http://esw.w3.org/topic/HCLS/ACPPTaskForce>