Veterans Health Information Model (VHIM) 3.7
Modeling Style Guide

Veterans Health Administration
Office of Information
Health Information Architecture

January 2008

REVISION HISTORY

Date	Revision	Description	Author
1/31/2008	1.0	Final Draft	VHIM Team

TABLE OF CONTENTS

REVISION HISTORY ..... **ii**
TABLE OF CONTENTS ..... **iii**
TABLE OF FIGURES ..... **iv**
1. Introduction **1**
1.1. Overview 1
1.2. Objectives 1
1.3. Required Toolset(s) 2
2. General Modeling Approach **2**
2.1. The HL7 Reference Information Model (RIM) 3
2.2. VHIM Alignment with HL7 4
3. Models **7**
4. Packages **7**
5. Diagrams **8**
6. Classes **9**
7. Attributes **9**
8. Associations **10**
9. UML Profile **11**
9.1. VHIM UML Profile Elements 12
10. Stereotypes **15**
11. Definitions **15**
12. Naming Conventions **16**
13. VistA (As-Is) Modeling Style **16**
14. Known Issues **18**
15. Acronyms **19**

TABLE OF FIGURES

Figure 1: The HL7 Reference Information Model (RIM) Foundation Classes 3
Figure 2: VHIM *PersonName* Data Type 5
Figure 3: VHIM *HL72PersonName* Data Type 5
Figure 4: Relationship between HL7 RIM and VHIM Classes 6
Figure 5: VHIM Model Transformation Artifacts 7
Figure 6: VHIM Diagram Label 8
Figure 7: Unique Diagram Names in RSA 8
Figure 8: VHIM Classes Showing HL7 Act-Role-Entity Color Coding 9
Figure 9: The VHIM UML Profile 12
Figure 10: Sample VistA (As-Is) Packages 17

Veterans Health Information Model (VHIM) 3.7
Modeling Style Guide

1. Introduction

1.1. Overview

The Veterans Health Administration (VHA) operates one of the largest healthcare systems in the United States, including over 150 hospitals and over 875 community-based clinics[1]. In order to facilitate longitudinal patient-centric care and achieve interoperability both within VHA and with external partners, there must be a consistent view and representation of information. This is the function of the VHA Health Information Model (VHIM). The VHIM is the authoritative enterprise information model for VHA. It represents the structure and content of all shared information that is exchanged across the enterprise. The VHIM will be used to define the messages and operation parameters declared by service providers, and will enable semantic mapping of the information shared between the VHA and its external partners. This document defines the elements of the style to be used in developing and maintaining the VHIM models.

1.2. Objectives

1.2.1. Harmonization Goals

The VHIM complements the VHA Standards and Terminology Services Program in that it classifies and represents data elements and their relationships, and represents data element data types and appropriate terminology.

A primary goal of the VHIM modeling project is to align with the emerging HL7 3.0 standard, both from a methodology and information modeling standpoint[2]. This ensures that the VHIM is semantically complete and can be easily mapped to and from HL7 Version 3.0 message models.

Yet unlike the HL7 Version 3 development, the VHIM development relies on standard Unified Modeling Language[3] (UML) notation and tools. VHA is committed to rigorous modeling using UML, as these artifacts will have a long life-span and may be used to generate other artifacts in multiple computer languages. To illustrate, Java is the chosen language today, yet 10 years from now, some newer language may be in vogue. By maintaining rigorous UML models in which as much of the business knowledge as practicable is built into the model (as opposed to being

---

[1] http://www1.va.gov/opa/fact/vafacts.asp

[2] HL7 is an international community of healthcare subject matter experts and information scientists collaborating to create standards for the exchange, management and integration of electronic healthcare information. See www.hl7.org.

[3] UML is a widely accepted industry notation that is well supported in the marketplace. This notation combined with commercially available, off-the-shelf tooling facilitates the forward generation of models into executable formats. UML is used to model not only application structure, behavior, and architecture, but also business process and data structure. See www.uml.org.

hidden in source code), the smooth, incremental transition from Java to that newer language will be enabled.

Therefore, a key VHIM objective is to follow the principles of the Object Management Group's (OMG's)[4] Model Driven Architecture (MDA), and develop the VHIM as a Computationally Independent Model (CIM). This will ensure that from the CIM, a standardized, repeatable transformation process can be used to create other models or information artifacts that are optimized for various purposes. In other words, the VHIM is not platform specific; instead, it concentrates on business semantics.

1.2.2. Modeling Style Objectives

Because the VHIM is prepared by multiple modelers and used by various projects throughout the enterprise, a common modeling style is needed to ensure consistency. The objectives of the VHIM modeling style are as follows:

- Define and document the additional metadata needed to enable model driven architecture.
- Create a consistent style across all models.
- Enhance readability and alleviate the learning curve for new users of the model.
- Reduce complexity for implementation.
- Identify common issues in modeling, and provide solutions.

1.3. Required Toolset(s)

All models conformant to this style guide are required to be developed with the most recent version of the toolset in use. The VHIM style leverages the functionality provided by UML 2. Therefore, a UML 2.x modeling tool must be used to develop the VHIM. The current toolset used for VHIM development is IBM® Rational® Software Architect (RSA). The style is intended to leverage the power of the tool to create code in Java™ or XML/XMI, which are the end products of the VHIM model transformation process.

2. General Modeling Approach

One of the primary goals of the VHIM is to ensure harmonization with industry standards, such as the HL7 Reference Information Model (RIM) and the HL7 3.0 Vocabulary. The VHIM uses the RIM as a pattern to create a canonical model by which data from different trading partners can be translated. This ensures that transformations increase linearly, not exponentially as the number of trading partners increases.

The HL7 development methodology uses well defined tools, processes, and a distinctive modeling style, all of which have their origins in the UML, but are not strictly consistent with UML practices. With the objective of leveraging the expertise and efforts that have gone into

[4] OMG is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. See www.omg.org

developing the HL7 standard and maintaining interoperability, the VHIM modeling style retains critical HL7 metadata, using UML conformant modeling practices.

2.1. The HL7 Reference Information Model (RIM)

The HL7 process begins with the definition of approximately 60 high-level classes, collectively called the Reference Information Model (RIM) Foundation Classes (see Figure 1). These classes serve as a pattern or *metamodel* for modeling efforts with HL7. The RIM foundation classes are further categorized into what HL7 calls the *Act-Role-Entity pattern*. In basic terms, *Entities* (people, organization, things, etc.) play a *Role* (patient, provider, etc.) as they *Participate* in *Acts* (planned or unplanned events, or actions). These classes and their subclasses are color-coded in the RIM: Entities are green, Roles are yellow, Participations are blue, and Acts are light red. The VHIM follows the same color-coding.

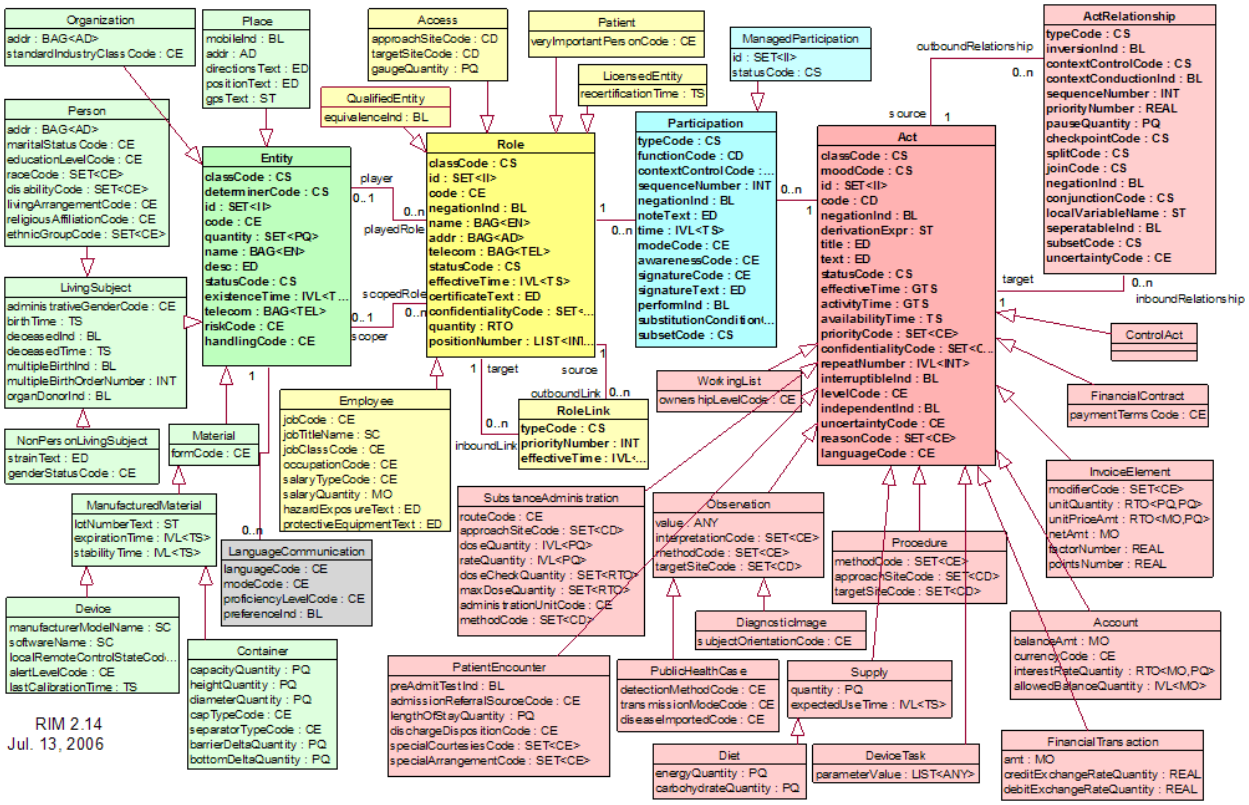

Figure 1: The HL7 Reference Information Model (RIM) Foundation Classes

2.1.1. HL7 RIM Derivations

The various groups within HL7 model their area of interest, or domain (e.g., Laboratory, Pharmacy, etc.), using specializations of the RIM foundation classes. Such a model is called a Domain Message Information Model (D-MIM).

For example, the *Act* foundation class, which represents any act or event, is specialized into various types of acts. The Pharmacy D-MIM thus contains a *SubstanceAdministration* class that

is a specialization of the Act class, used for "introducing or otherwise applying a material into or to the subject"[5]. It is important to note that when a foundation class is specialized, any RIM attributes that are not needed by the D-MIM are removed. This is a departure from standard UML specialization practice.

Once the D-MIM is completed, Refined Message Information Models (R-MIMs) may be derived. The R-MIM is a subset of the D-MIM that is used to model an individual message specification. For example, the Patient Administration domain has an *Active Inpatient Encounter Appointment* R-MIM, which is derived from the Patient Administration D-MIM, and defines the message used to report that an inpatient encounter has been activated or reactivated. As with RIM refinement, when a D-MIM is refined, class attributes that are not needed by the R-MIM are removed.

HL7 data types for class attributes may be replaced by other types, either for clarification, or to constrain the use of the attribute[6]. For example, the RIM and the D-MIM may allow an attribute to be a number, but the R-MIM may require it to be an integer. Similarly, the RIM and the D-MIM may allow an attribute to have zero to many occurrences, but the R-MIM may require it to exist only once.

2.2. VHIM Alignment with HL7

2.2.1. HL7 Data Types in the VHIM

The strict use of the HL7 Version 3 data types can cause difficulties in implementation, particularly database implementation. The HL7 data types are typically complex types which are then often "constrained" based on their usage. Thus an "effective time" may contain one time in a lab order, but five times in a pharmacy order. Additionally, as in the above model derivation process, unneeded attributes are removed from subclasses. Moreover, the data types contain segments that are apparently never used, and often have type codes to identify the data type. Even after manually simplifying many of the HL7 data types in the VHIM (e.g., reducing Character String (ST) to a simple string), considerable manual effort is still needed to create suitable database models from a model based upon these data types. The VHIM *Datatypes* package describes those HL7 data types used within the VHIM in rigorous UML.

---

[5] HL7 Version 3 Standard: Reference Information Model (RIM), Release 1, 12/17/2003

[6] HL7 Version 3 Standard: Data Types - Abstract Specification, Release 1, 11/29/2004

Figure 2 shows how the HL7 Version 3 Person Name (PN) data type is simplified for use in the VHIM.

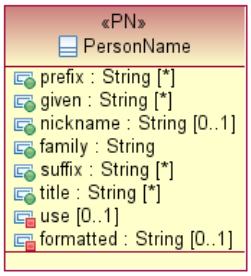

Figure 2: VHIM *PersonName* Data Type

In addition to HL7 Version 3 data types, the VHIM also maintains representations of those HL7 Version 2.4 data types used in VistA HL7 Version 2.4 messages where those data types vary from the VHIM representation of the corresponding RIM data types. Figure 3 shows how the HL7 Version 2.4 Extended Person Name (XPN) is represented in the VHIM using the *HL72PersonName* class. Note the differences between the HL7 RIM-based *PersonName* class and this class. While the former allows multiple occurrences of attributes such as given name (where a middle name would be captured), the HL7 Version 2.4 representation does not; the middle name attribute is specifically called out in this class.

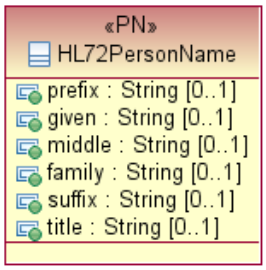

Figure 3: VHIM *HL72PersonName* Data Type

2.2.2. HL7 Classes in the VHIM

Unlike the HL7 process, wherein RIM classes are copied to create subclasses, the VHIM process uses the RIM classes as a metamodel. These classes become a UML Profile (see Section 9 below), which dictates the structure of the VHIM classes. VHIM classes are stereotyped to reference their corresponding RIM classes.

In HL7, top-level classes contain a Class Code (or, in some cases, a Type Code), which indicates what "subclass" of the foundation class is being modeled. For example, in Figure 4 the class code SBADM for *SubstanceAdministration* is a "subclass" of Act.

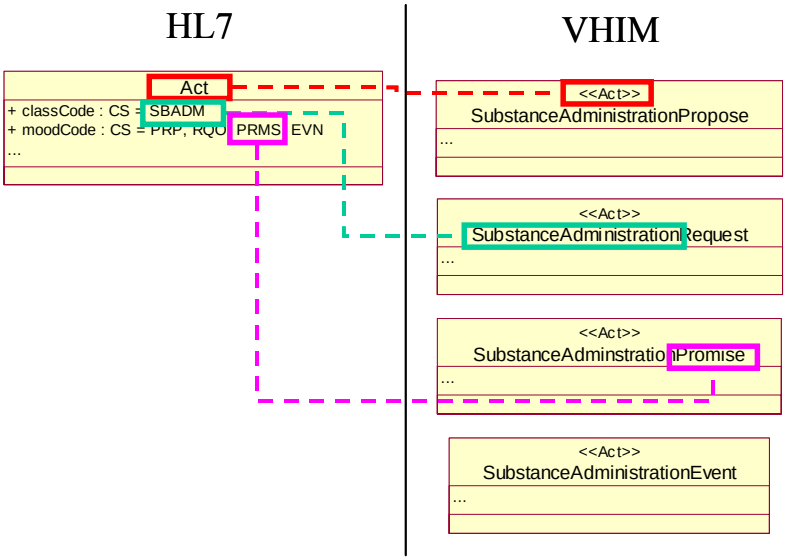

Figure 4: Relationship between HL7 RIM and VHIM Classes

In addition, the Act foundation class has a Mood Code which further differentiates between subclasses. For the *SubstanceAdministration* class, the four allowable mood codes are PRP (Propose), RQO (Request), PRMS (Promise), and EVN (Event). The VHIM does not carry the Mood Code attribute; instead, those classes will be declared explicitly. The Class Code attribute is also not carried in the VHIM class; the business name of the class code will become the name of the VHIM class. If the class is an Act, the business name of the mood code will be appended to the name.

In the SubstanceAdministration example, this will be represented as four separate classes: SubstanceAdministrationPropose, SubstanceAdministrationRequest, SubstanceAdministrationPromise, and SubstanceAdministrationEvent.

2.2.3. HL7 Attributes in the VHIM

The attributes of the HL7 RIM classes will guide the attributes of the corresponding VHIM classes. The names of the attributes of the HL7 RIM classes are purposefully generic to handle the needs of all potential subclasses. Additionally, HL7 data types are complex and may contain multiple values (e.g., Person.addr may contain a home address and a work address).

Thus the generic names combine with the multiplicity of concepts contained in the attribute to create a model that is not as expressive or as rigorous as it otherwise might be. The user of the model is forced to resort to other tools to properly define the concepts being modeled and to constrain the values that the attributes may contain. For example Act.priorityCode may contain *UD* (Use as Directed). This code is useful for SubstanceAdministration Acts, but is useless for a FinancialTransaction Act.

Another issue encountered with the HL7 RIM is that attributes represented by a codeset must be manually constrained to ensure an appropriate codeset is used. For example, if the State in an address is a code, there is no automated way to prevent one from entering a code from another codeset in that attribute. “Boston, Magenta” makes no sense, but it is legal. To prevent this, the

VHIM Profile maintains a property which can be applied to an attribute in the model, indicating its corresponding vocabulary domain that may legally be used, and the VUID, if applicable.

3. Models

The VHIM model is able to meet the variety of its implementation requirements via the model transformation process. The VHIM may be transformed into a platform specific model that is more conducive to XML, Java, HL7, Database Definition Language (DDL), etc. This allows the VHIM to remain computationally independent, with no built-in bias towards a particular implementation platform. Most importantly, the derived models will all retain the same semantic meaning, thus preserving the integrity of the data regardless of how or where it is used.

The artifacts of the model transformation process are (see Figure 5):

- Computationally Independent Model (CIM) – conceptual model
- Platform Specific Model (PSM) – logic model
- Model implementation – physical model

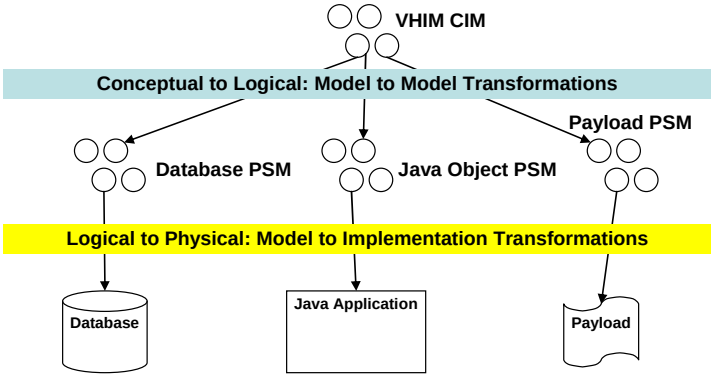

Figure 5: VHIM Model Transformation Artifacts

The VHIM model consists of domain packages, data types, and the UML Profile. In addition, project-specific models (such as the CDS Authoritative Repository Master Schema (ARMS)) are created as separate packages within the VHIM.

The corresponding filename for the VHIM model is *VHIM.emx*.

4. Packages

As the healthcare domain is very large, it is convenient to partition the domain into smaller portions for consideration. These portions correlate with UML packages, in which related classes will be stored. Additional packages are used to store VHIM-defined data types (*Datatypes*) or other shared constructs (*Common*).

It is important to consider that the packages are an artificial grouping of portions of the healthcare domain; the VHIM is to be considered the union of all of the packages. Packages also form the basis of version control; that is, users will check out and check in work at the package level.

- All packages must have simple generic names (e.g., “Lab”)

- Package names may not contain spaces, underscores, or other punctuation.
- Package names are to be in UpperCamelCase. That is, separate words are concatenated, and are all lower case, except that the first letter of each sub-word is capitalized (e.g., "VitalSigns").
- Each package shall be saved in its own file. The name of the file shall be identical to the package name. In RSA, the primary model file is given the *.emx* extension (e.g., "VHIM.emx"). VHIM domain packages are saved as linked subfiles. RSA gives these subfiles the *.efx* extension (e.g., "Lab.efx").

5. Diagrams

- The text font used for all diagram objects should be Arial 10.
- Each diagram should have a label in the top-left corner of the diagram with the name of the diagram, the version, and the release date (see Figure 6).

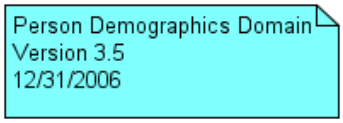

Figure 6: VHIM Diagram Label

- Although RSA names the first diagram created within a package *Main* by default, the main diagram in each VHIM package should be renamed to carry the same name as the package. When multiple diagrams from multiple packages are simultaneously open in RSA, having unique diagram names prevents confusion when switching back and forth between diagrams. Figure 7 shows three open diagrams in RSA: The VHIM *Main* diagram, the *Common* diagram (owned by the *Common* package), and the *Allergies* diagram (owned by the *Allergies* package).

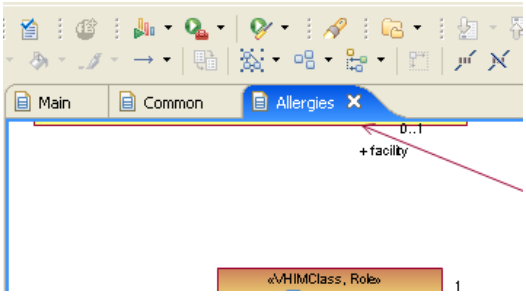

Figure 7: Unique Diagram Names in RSA

- Domain classes are color-coded based on the HL7 Act-Role-Entity pattern. Entities are green, Roles are yellow, Participations are blue, and Acts are light red. All other classes use the default color, light yellow-brown (see Figure 8). To automatically assign those colors in RSA when the HL7 Development Framework (HDF) plug-in is installed, right-click on the diagram and choose "Assign RIM Colors".

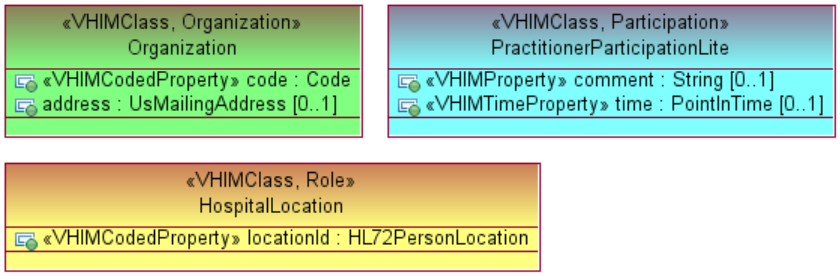

Figure 8: VHIM Classes Showing HL7 Act-Role-Entity Color Coding

6. Classes

- Classes must be uniquely named between domain partitions; in other words, two classes in two different domains may not have the same name. It is understood that the converse is true when speaking of two different versions of the same domain; in general, the classes must have the same names, unless there was a need to rename a class. Note that while classes may appear in multiple domain diagrams, they only belong to one package. This rule is concerned with naming of different classes, not the re-use of a class by multiple partitions.
- Class names must be nouns and singular.
- Names of classes should be no more than 30 characters in length.
- A class name may not contain spaces, underscores, or other punctuation.
- Class names are to be in UpperCamelCase. That is, separate words are concatenated, and are all lower case, except that the first letter of each sub-word is capitalized (e.g., “EpisodeOfCare”).
- Attributes are to be arranged alphabetically within the class, unless it makes more sense to arrange them in a different manner (e.g, city, state, zip in an address class). If a class has an “id” attribute, that attribute may be placed first.
- Classes are to be stereotyped to indicate which HL7 RIM class is being used as a pattern for the class.
- Multiple-inheritance is not allowed.
- When sub-typing, care should be taken to ensure that the generalization makes sense (i.e., object A "is a" object B).

7. Attributes

- Attributes should be named so as to minimize ambiguity regarding the purpose of the attribute. Ideally, any user of the model should be able to deduce what the attribute represents based upon the name. For example, the term “entry” is used often in CPRS. The reader may ask, “Entry to what? Is this when a patient enters a hospital?” The term “dataEntry” clarifies what the attribute represents.

- An attribute name should not contain spaces or other punctuation.
- Attribute names are to be in lowerCamelCase. That is, separate words are concatenated, and are all lower case, except that the first letter of each sub-word following the first sub-word is capitalized. The first sub-word is lower case (e.g., “dateOfBirth”). This rule applies to acronyms as well (“deaNumber” is easier to read than “DEANumber”).
- The use of abbreviations is discouraged, but may be used when they are commonly used in healthcare. However, the use of “id” is encouraged instead of “identifier”. Note that commonly used acronyms (e.g., “deaNumber”) are acceptable.
- Attribute names must be unique within the context of a class and its supertypes.
- Non-boolean attribute names should be nouns and singular. Boolean attribute names should read like a question, and should start with a verb (e.g., “isActive”, “needsSpecialApproval”). The verb “is” is preferred.
- Unlike common practice in Entity-Relationship modeling, as a rule of thumb, the class name should not be pre-pended to the attribute name, as the class name is required to access the attribute. For Example, the attribute for “name” in the “Person” class should be called “name” vice “personName” as it is accessed using *Person.name*. *Person.personName* is redundant.
- The use of “code” or “flag” is discouraged. These terms imply a particular implementation, and do not impart any additional meaning, as the fact that an attribute is a code is known from its data type.
- The use of “type” and “class” is not allowed. These are both reserved words in some programming languages. Use the word “category” instead.
- The use of “number” should be avoided unless the attribute is and always will be a number. For example, “patientNumber” implies that the attribute is numeric. “patientId” better describes the attribute, as it is an identifier for the patient, and does not imply a particular implementation.
- Attributes should have public visibility.
- The data type assigned to an attribute must inherit from a RIM attribute appropriate to the RIM class being used as a pattern. For example, RepeatNumber is a valid data type, but is only valid for Act classes.

8. Associations

- Association names are optional.
- Uni-directional associations are preferred. However, bi-directional associations are allowed if both associationEnds must be maintained simultaneously (e.g., in a “marriage” relationship, the “husband” and the “wife” roles are both created and both destroyed with the relationship).
- Circular associations (e.g., A → B → C → A) are not allowed, unless C associates with a new and unique instance of A.

- Recursive associations (e.g. A → A) are allowed. It is understood that the association is between two instances of A.
- The associationEnd name on the target end is required. associationEnd names follow the same rules as attribute names. If the association is bidirectional, both associationEnd names must be specified.
- Association cardinality must be specified at the target class (i.e., at the arrowhead). There is no relevance for cardinality at the tail in uni-directional associations, and therefore, no cardinality will be used on the tail end. This is because the instance must exist for the association to exist, and therefore the cardinality on the tail end is always one.
- In those situations where a relationship may be modeled as either an association or as an aggregation, the use of the association is preferred. Nevertheless, aggregations are not discouraged, and may be used if they convey more meaning (e.g., panels). The usefulness of the aggregation is that the semantic meaning of the association means assembly or membership. Note that aggregate associations are different than composite associations. In a composite association, the parent is responsible for creating and destroying the child(ren). When the parent is destroyed, the child(ren) are destroyed. An example is Order and OrderLineItem.
- Composite associations are acceptable.

9. UML Profile

As the VHIM is now able to leverage the features of UML 2, the modeling style has been enhanced to remove metadata (descriptive information about the packages, classes, attributes, associations, etc.) that are only needed for a single purpose from the model proper. The preferred method for storing these metadata is through the use of the UML Profile. Profiles are used to extend the UML using stereotypes, tagged values (properties), and constraints, in order to facilitate the development of models within particular domains (e.g., CDS, HL7, VHA, etc.). A UML Profile will be maintained for the VHIM (see Figure 9), allowing the VHIM to further define semantics and well-formedness rules beyond what is specified by the UML metamodel.

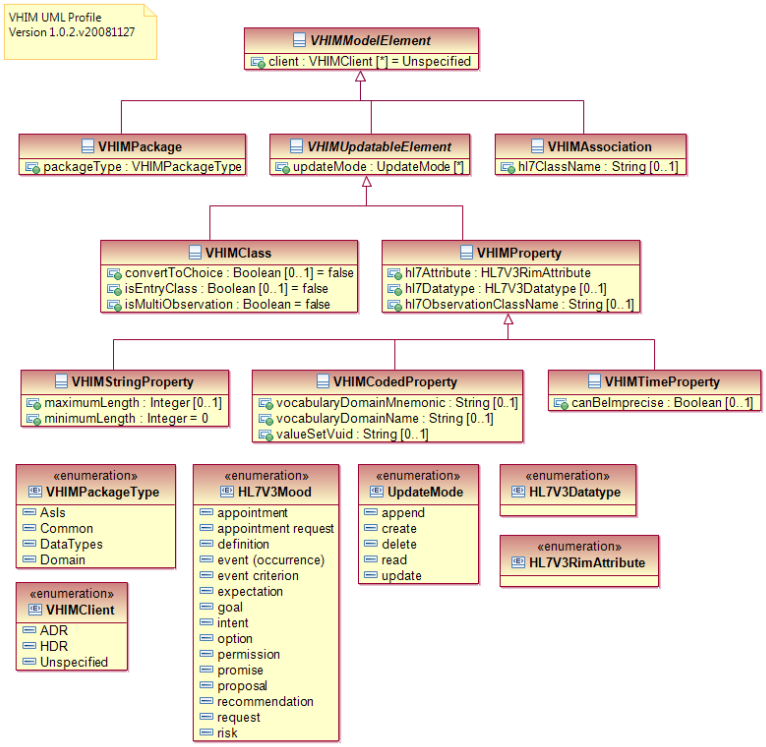

Figure 9: The VHIM UML Profile

9.1. VHIM UML Profile Elements

This section defines each of the VHIM UML Profile elements and describes its intended use. Where applicable, the corresponding enumeration class is also identified.

9.1.1. VHIMModelElement.client

Identifies the specific VHIM consumer for whom the model was developed.

Data that is of interest to VHA is stored in various repositories. Those repositories are only concerned with the subset of VHIM data which they maintain. The *VHIMModelElement.client* property determines which elements are included in XSD generation for a particular VHIM consumer. For example, one consumer may opt to identify the authoritative source for a particular VHIM attribute, but another consumer may not.

Since an element may be needed in multiple repositories, this property allows for multiple entries (e.g., ADR and HDR may be included).

Enumeration: *VHIMClient* (ADR, HDR, Unspecified)

9.1.2. VHIMPackage.packageType

Indicates whether the model package maintains information for a particular VHIM domain (Domain), is shared by all domains (Common), defines the VHIM data types (DataTypes), or represents the structure for an existing VistA file (AsIs).

The model transformation process will use the package type to determine how and when to use the information within.

Enumeration: *VHIMPackageType* (AsIs, Common, DataTypes, Domain)

9.1.3. VHIMUpdatableElement.updateMode

Currently used by CDS and for HL7 Version 3 message generation. Indicates how the data contained in the message should be used.

Enumeration: *UpdateMode* (append, create, delete, read, update)

9.1.4. VHIMClass.convertToChoice

As mentioned before, the VHIM is a computationally independent model. However, subclasses are not conducive to xml implementation. so it is desireable to convert the superclass/subclass relationship into a choice group. Yet this is not wanted in the model, because it would be xml specific. Set on the superclass so that an xml-specific model can convert the structure into a choice group. Indicates whether the class and its subclasses are suited to conversion to an XML choice group, which ensures that only one of the child elements can be instantiated.

The model transformation process will transform the class and its subclasses into a structure from which an XML choice group can be generated.

9.1.5. VHIMClass.isEntryClass

Indicates whether the VHIM class should be considered an entry class for the model.

The model transformation process will use a class marked as an entry class as the starting point for XSD generation.

9.1.6. VHIMClass.isMultiObservation

Indicates whether the VHIM class contains multiple observations.

HL7 requires each observation to be modeled in a separate class. The VHIM simplifies this structure by combining simultaneous or related observations into one class. When the XSD for an HL7 Version 3 message is created, the model transformation process will convert the VHIM observation class into multiple HL7 observation classes.

9.1.7. VHIMClass.hl7Mood

Classes in the VHIM which are based upon the HL7 Act class must be linked to their corresponding HL7 Mood Code.

Enumeration: *Hl7V3Mood*

9.1.8. VHIMProperty.hl7Attribute

Provides a link from the VHIM attribute to its corresponding HL7 RIM attribute.

When the XSD for an HL7 Version 3 message is created, the transformation process will use the attribute name as the XML tag name.

Enumeration: *HL7V3RimAttribute*

9.1.9. VHIMProperty.hl7Datatype

Provides a link from the VHIM data type to its corresponding HL7 RIM data type.

This information enables the model transformation process to build the appropriate data type structure for HL7 messages.

Enumeration: *HL7V3RimDatatype*

9.1.10. VHIMProperty.hl7ObservationClassName

Identifies the corresponding HL7 observation class for an attribute that is not modeled as an observation in the VHIM.

In some cases the VHIM uses a simplified modeling structure to represent information which is modeled as an observation class in HL7 Version 3. When the XSD for an HL7 Version 3 message is created, the model transformation process will convert the VHIM attribute into an HL7 observation class.

9.1.11. VHIMStringProperty.maximumLength

Defines the maximum length allowed for an attribute of type String.

9.1.12. VHIMStringProperty.minimumLength

Defines the minimum length allowed for an attribute of type String.

9.1.13. VHIMCodedProperty.vocabularyDomainMnemonic

Identifies the corresponding HL7 vocabulary table, which contains the list of valid values for an attribute.

9.1.14. VHIMCodedProperty.vocabularyDomainName

Identifies the corresponding HL7 vocabulary table, which contains the list of valid values for an attribute.

9.1.15. VHIMCodedProperty.valueSetVuid

Indicates the corresponding VHA Unique Identifier (VUID), thus providing a link from the VHIM coded concept to the corresponding Enterprise Terminology Service (ETS) terminology concept.

9.1.16. VHIMTimeProperty.canBeImprecise

A Boolean value indicating whether the time captured in the attribute value must be precise. For example, information such as birth date, death date, or date of last menstrual cycle can be imprecise; however, a machine generated timestamp must be precise.

9.1.17. VHIMAssociation.hl7ClassName

Provides a link from the VHIM association to its corresponding HL7 RIM class. In some cases the VHIM uses a simplified modeling structure to represent information which is modeled as a class in HL7 Version 3.

When the XSD for an HL7 Version 3 message is created, the model transformation process will convert the VHIM association into the corresponding HL7 class.

10. Stereotypes

All allowable stereotypes other than standard UML stereotypes (e.g., Actor, Boundary) will be defined in the VHIM UML Profile. The UML Profile defines the allowable set of stereotypes and the types of model elements to which the stereotype may be applied. RSA enforces the stereotypes that are used, and the modeler will select the stereotype from a pick-list.

11. Definitions

All classes, attributes, and associations must have precise definitions documented in the model and validated by the Subject Matter Experts. Where appropriate, it is preferable that the definition be quoted or derived from a medical dictionary, HL7 documentation, or the Enterprise Reference Terminology.

- The definition should concisely and precisely describe the concept represented by the model element. The definition should stand on its own, and not force the reader to rely on any context. The definitions will be published in other forums, such as the MetaData Repository (MDR), where the user will not have the benefit of the UML diagram and does not necessarily have any knowledge of HL7. Ideally, the reader of the definition should be able to immediately understand what the concept is, and, if appropriate, what it is not.
- If the definition is lifted verbatim from an outside source (e.g., HL7, National Council for Prescription Drug Programs (NCPDP), Merriam-Webster), it should be placed in quotation marks and the source cited in parenthesis. For example, "A code specifying the modality by which the Entity playing the Role is participating in the Act." (HL7 3.0)
- In general, the definition should *not* directly quote the HL7 RIM data type definition, as the HL7 RIM definition is generic, does not adequately describe the specific concept, and can be found from the data type anyway.
- If an example is given, use the abbreviation “e.g.” within parentheses. For example, “This represents the day of the week (e.g., Monday).”
- If a clarification of a concept is needed, use “i.e.,” within parenthesis. For example, “This is the duration of the illness (i.e., it includes a start date and an end date, either of which may be empty).”

- When describing a coded attribute, it is helpful to list a few (not all) sample concepts that would be expressed in the code. However, the actual codes should *not* be cited, as they may change and thus break the definition. For example, for OrderHold.reason, the VistA definition is:

  > *This field tells why the prescription was put on hold.1:Insufficient qty in stock; 2:Drug-Drug Interaction; 3:Patient Reaction; 4:Physician to be contacted; 5:Allergy Reactions; 6:Drug Reaction; 99:Other--See Comments*

  The corresponding VHIM definition is:

  > *Indicates why the order is/was put on hold. Possible values include: Out of stock, Drug-Drug interaction, Patient adverse reaction, etc.*

  Notice the "Possible values include" at the beginning, and the ", etc." at the end. This prevents the description from being too restrictive, and allows VHA Enterprise Reference Terminology to design the codeset as they see fit.

  The following definition for InPatientMedicationRequest.infusionRate is illustrative:

  > *Infusion: "Introduction of a solution into the body through a vein for therapeutic purposes." (American Heritage Stedman's Medical Dictionary). This is the rate at which the solution is introduced into the body. Expressed as volume/time (e.g., 500 ml/hr)*

  This definition defines the concept of Infusion Rate by first defining Infusion, which is directly quoted from a medical dictionary. The concept of Infusion Rate is then described by building on the definition of Infusion. Finally, an example is given as to how the concept would be used.

12. Naming Conventions

To determine a name for a concept being modeled, the primary source will be the HL7 version 3 Domain Message Information Model (D-MIM), Refined Message Information Model (R-MIM), Vocabulary, Code Sets, and explanatory material in the Ballots. As detailed above, the class name will come from the business name of the code used in the ClassCode or TypeCode attribute. If the class is an Act, the business name of the mood code will be appended to the class name. Note that the name chosen may need to be modified to conform to the naming conventions above. This having all been said, it is noted that the data standardization and terminology efforts with VHA may identify a different name for the same concept. In these cases, the VHA name will be used instead of the HL7 name. The VHA name will then be provided to the VHA HL7 Liaison team to be brought back to HL7 for harmonization.

13. VistA (As-Is) Modeling Style

VHIM has developed a set of VistA (As-Is) models to document the existing VistA FileMan files, and to provide a reference point for VHIM requirements analysis. These models are maintained in the *VistA* model in RSA (see Figure 10). In order to apply UML constructs to files which are not object-oriented, the following modeling conventions have been employed.

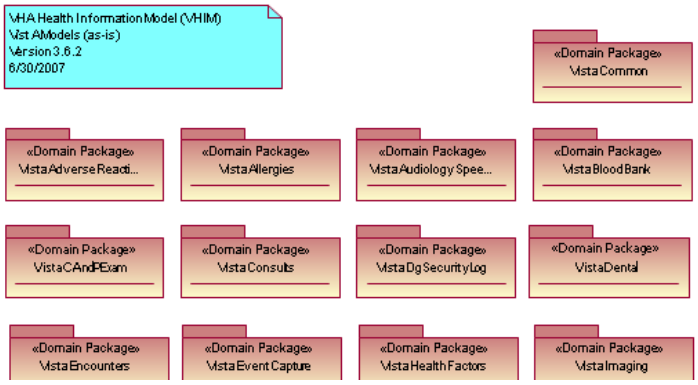

Figure 10: Sample VistA (As-Is) Packages

- As-is modeling begins with creating a single class to hold the FileMan file in question (e.g., for Orders, the file is file 100), with the following exceptions:
  - When a multiple is encountered, it is modeled as its own class.
  - When a pointer to another file is encountered, it may be modeled as an attribute (if the file is not referenced by another class), or the other file is modeled as its own class with an association to it.
  - References to File200 are modeled as associations to classes named for HL7 participation types, which then point to Provider.
- Definitions are not required, except for classes which represent pointers to other files. These classes will contain the referenced file number in their definition.
- The attributes are arranged in the order in which they appear in FileMan.
- The attribute data types will reflect the actual data type in FileMan, except that attributes that are clearly Yes/No fields are modeled as Boolean.
- Attributes that are sets should end in the word “code” (e.g., statusCode).
- Attributes that are date/times should end in the word “time”.
- Attributes that are Booleans should begin with the word “is”.
- File IENs should be named “ien”, and will typically be an integer.
- Class, attribute, and association names should be singular nouns.
- Class names should be in UpperCamelCase. That is, separate words are concatenated, and are all lower case, except that the first letter of each sub-word is capitalized (e.g., “VistaPatient”).
- Attribute and association names should be in lowerCamelCase. That is, separate words are concatenated, and are all lower case, except that the first letter of each sub-word following the first sub-word is capitalized. The first sub-word is lower case (e.g., “appointmentTime”). This rule applies to acronyms as well (“ien” is easier to read than “iEN”).

- Rather than model a class to hold an event that just needs an author participation, use an association to Author with a compound association name that contains the name of the Act that would have been created. For example, if we have an OrderFlag event that just has an author, just have an association directly to Author called orderFlagAuthor.

14. Known Issues

Several of the requirements in this document are due to or are influenced by characteristics of the RSA tool currently in use. This section details those issues

- RSA does not support qualified associations. (?)
- RSA does not permit grouping of attributes within a class. (?)
- RSA does not support constraints. (?)
- RSA Gotchas:
    A. Regardless of package you are using when an association is drawn, the association is owned by the package above it. Have to manually move the association to the lower package.
    B. If you move a class from one package to another, by dragging the class in project explorer, associations to that class sometimes lose their role name, thus causing an orphan association. Use refactor/move instead.

15. Acronyms

Acronym	Definition
ARMS	Authoritative Repository Master Schema
CDS	Clinical Data Services
CIM	Computationally Independent Model
DDL	Database Definition Language
D-MIM	Domain Message Information Model (from HL7 3.0)
HDF	HL7 Development Framework
HL7	Health Level 7
MDR	Metadata Repository
NCPDP	National Council for Prescription Drug Programs
PIM	Platform Independent Model
RDT	VHIM Refined Data Type
RIM	Reference Information Model (from HL7 3.0)
R-MIM	Refined Message Information Model (from HL7 3.0)
SRDT	Specialized Refined Data Type
UML	Unified Modeling Language
VHIM	VHA Health Information Model
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema Definition