

**Veterans Health Information Model (VHIM) 3.3
Modeling Style Guide**



**Veterans Health Administration
Office of Information
Health Information Architecture**

December 2005

Revision History

Date	Revision	Description	Author
12/1/05	1.0	Final Draft	Galen Mulrooney

TABLE OF CONTENTS

1. Purpose.....	3
2. General Approach.....	3
3. Models	11
4. Packages.....	11
5. Classes	12
6. Attributes.....	12
7. Associations	13
8. UML Profile.....	14
9. Stereotypes.....	14
10. Definitions	14
11. Naming	15
12. Aesthetics	16
13. Issues	16
14. Acronyms	16

Veterans Health Information Model (VHIM) 3.3 Modeling Style Guide

1. Purpose

As the VHIM will be prepared by multiple individuals, the use of a consistent modeling style is desirable. In addition, because the VHIM is an enterprise model to be used by multiple projects, a common style is needed to ensure consistency. A common style enhances readability, reduces complexity, and reduces the learning curve of new readers of the model. This section defines the elements of the style to be used in VHIM models.

All models conformant to this Style Guide are required to be developed with the most recent version of the toolset currently in use (currently Rational Rose, which implements UML version 1.4). In addition, the style selected should leverage the power of the tool to create code in Java or XML/XMI.

2. General Approach

One of the goals of the VHIM modeling project is to align with the emerging Health Level 7 (HL7) 3.0 standard. In addition, we wish to leverage the expertise and efforts put into HL7's modeling effort. HL7's development methodology currently employs some non-UML tools and processes which the VHIM effort wishes to emulate but in a manner consistent with UML modeling practice.

The HL7 process begins with approximately 60 high-level classes called the Reference Information Model (RIM). These classes serve as a pattern or a metamodel for modeling efforts with HL7. The RIM classes are further categorized into what HL7 calls the "Act-Role-Entity pattern" (see figure 1). *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: Entities are green, Roles are yellow, Participations are blue, and Acts are light red. The VHIM follows the same color-coding.

The various groups within HL7 model their area of interest using copies of the RIM Classes. Such a model is called a Domain Message Information Model (DMIM). For example, there is a RIM class called Act, which represents any act or event in which we are interested. The pharmacy DMIM may thus contain a SubstanceAdministrationEvent class that is a copy of the Act class. Importantly, RIM attributes that are not needed by the DMIM are removed.

Once the DMIM is completed, Refined Message Information Models (RMIMs) may be derived. The RMIM is a subset of the DMIM that is used to model an individual message specification. Class attributes that are not needed by the RMIM are removed. In addition, the data type of the attribute may be restricted. For example, the RIM and the DMIM may allow an attribute to be a number, but the RMIM may require it to be an integer. Similarly, the RIM and the DMIM may allow an attribute to have zero to many occurrences, the RMIM may require it to exist only once.

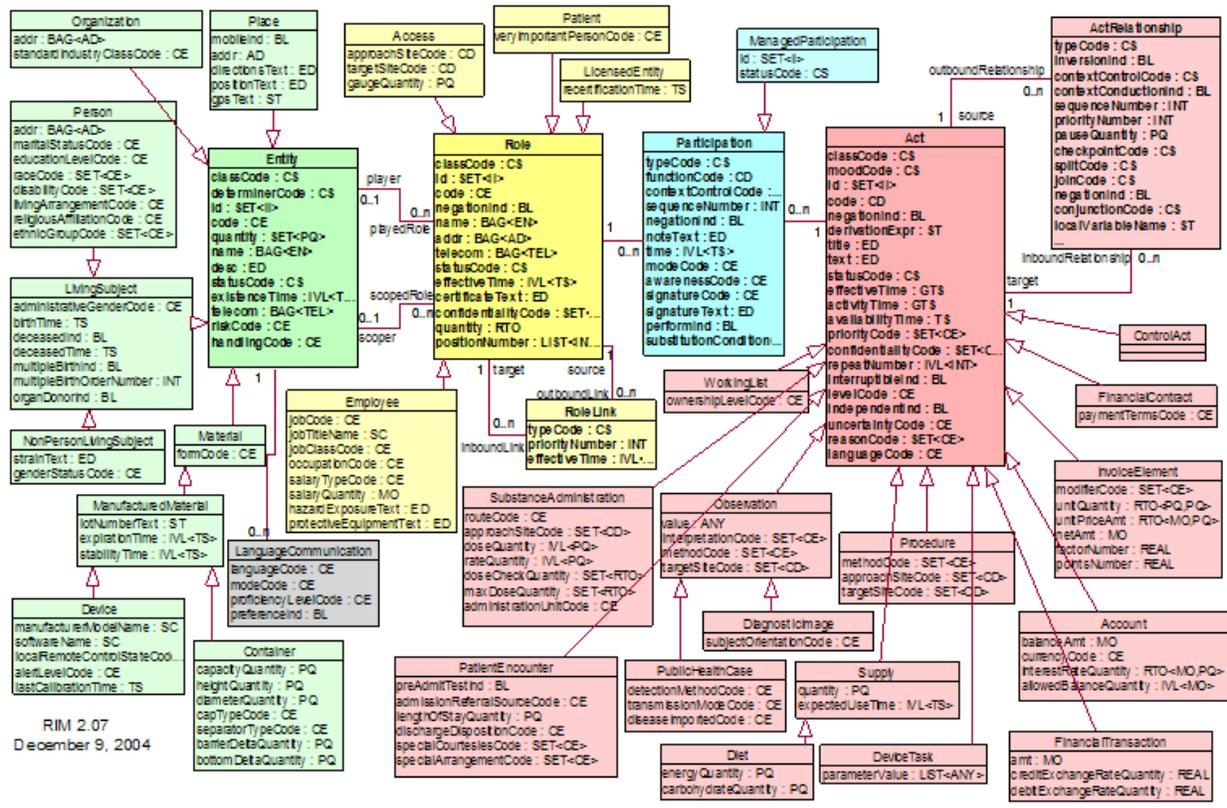


Figure 1: The HL7 Reference Information Model (RIM)

HL7 has defined a set of complex data types. As in the above process, HL7 removes unneeded attributes in subclasses. The VHIM uses the HL7 data types. But because the VHIM is UML-compliant, the inheritance relationships of the HL7 data types have been modified. For example, Entity Name (EN) defines 5 attributes. It has three subtypes, two of which (Person Name (PN) and Organization Name (ON)) have all 5 attributes, but one subtype (Trivial Name (TN)) has only one of the attributes. In the VHIM, EN becomes a subtype of TN, which then only has two subtypes: ON and PN.

The VHIM thus contains a Data types package which describe the HL7 data types in rigorous UML. The VHIM also contains a package containing a replica of the HL7 RIM, but using the VHIM data types instead of the HL7 data types. This package is called the VHIM RIM.

Unlike the HL7 process, wherein RIM classes are copied to create “subclasses”, the VHIM process uses the VHIM RIM classes as a metamodel. The VHIM RIM classes become a UML Profile (see section 8 below), which dictates the structure of the VHIM classes. The VHIM classes are stereotyped to refer back to the VHIM RIM class.

In HL7, top-level classes contain a Class Code (or, in some cases, a Type Code), which indicates what “subclass” of the top level class is being modeled. In addition, the Act class has a Mood Code which also effectively acts to differentiate between subclasses. The VHIM will declare the classes explicitly (see figure 2). 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. The class code and the mood code will not appear in the derived classes.

For example, in figure 2, the Act with the class code SBADM (business name = SubstanceAdministration) has four allowable mood codes: a) PRP (Propose); b) RQO (Request); c) PRMS (Promise); and d) EVN (Event). In the VHIM, this will be represented as four separate classes: SubstanceAdministrationPropose, SubstanceAdministrationRequest, SubstanceAdministrationPromise, and SubstanceAdministrationEvent.

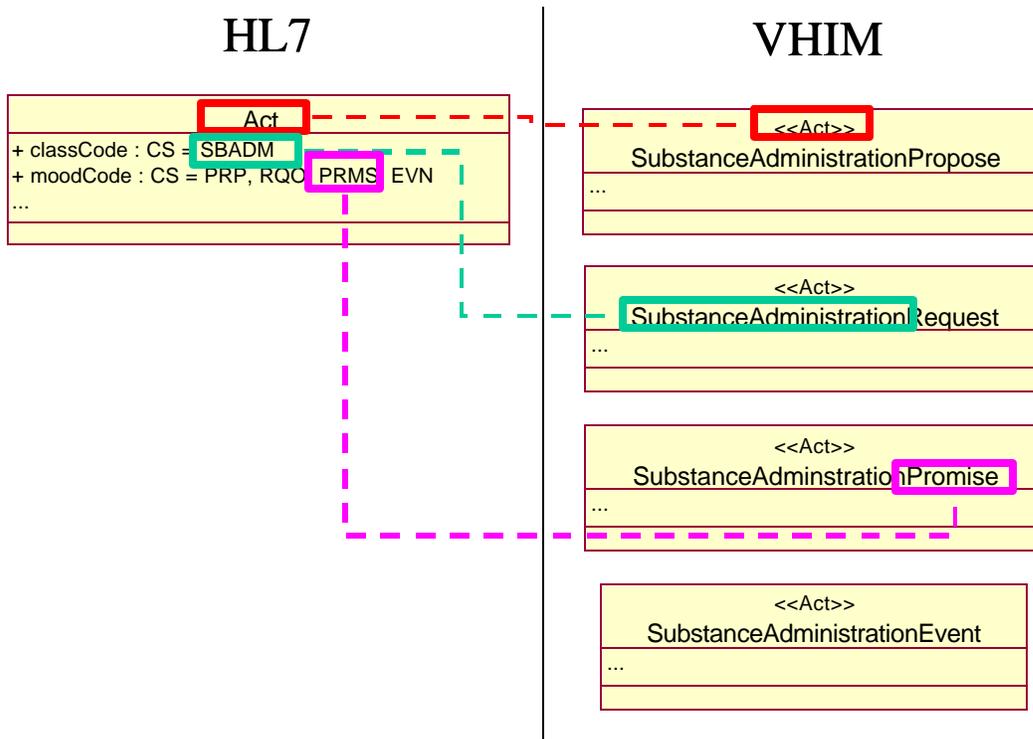


Figure 2: Relationship between VHIM RIM and VHIM Classes

Similarly, the attributes of the VHIM RIM classes will guide the attributes of the corresponding VHIM classes. The names of the attributes of the HL7 RIM classes are purposefully generic so they can 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 for Use as Directed. This code is useful for SubstanceAdministration Acts, but is useless for a FinancialTransaction Act.

The VHIM solves this problem in two ways. First, each of the HL7 RIM class attributes will be modeled as classes, which are called Refined Data Types (RDTs). These classes will be subclasses of the data type (see figure 3). For example, the HL7 Act class has an effectiveTime attribute, which is of type GTS. The VHIM defines a class called EffectiveTime, which is a subtype of GTS.

This allows us to split out the different concepts that are “buried” in the data type. For example, in HL7, the SubstanceAdministrationRequest.effectiveTime contains dozens of times, including dose frequency (e.g., 2 times a day), duration (e.g., 30 minutes), Start Date (e.g. Jan. 1st), End Date (e.g., Jan 16th), etc. In the VHIM, these concepts are modeled explicitly (see figure 4). Note that instead of having one attribute called effectiveTime, the VHIM now has four attributes which are of type EffectiveTime. This model is more readable, and easier to implement.

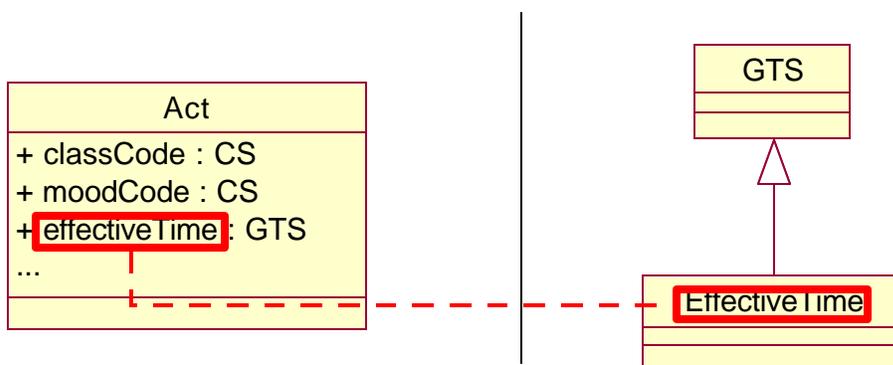


Figure 3: Transformation of RIM attributes to VHIM Refined Data Types

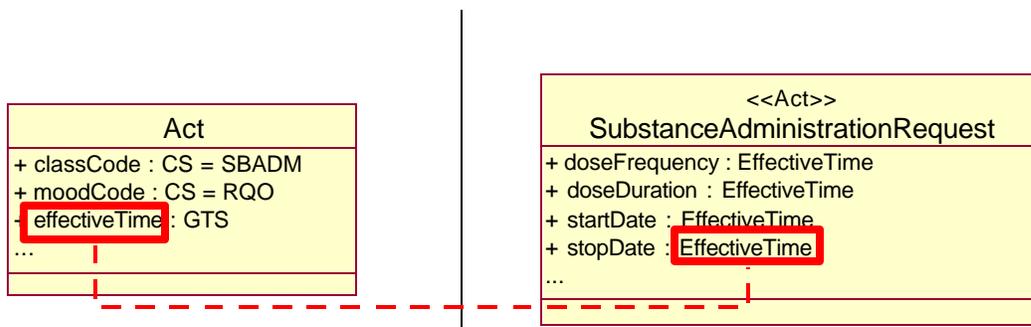


Figure 4: Using Refined Data Types to explicitly convey hidden concepts

The second problem encountered with the HL7 RIM is that attributes represented by a codeset must be constrained so that only an appropriate codeset is used. If the State in an address is a code, there is no automated way to prevent someone from inserting a code from another codeset in that attribute. “Boston, Magenta” makes no sense, but is legal. To prevent this, the VHIM subclasses both the concept of CodeSet (also known as Coding Scheme, or Code System) and of

the CodedValue (also known as a Code or Coded Term). The subclassing allows one to explicitly indicate the codeset(s) that may legally be used. Note in figure 5 how Address.state is of type StateCodedValue, therefore the assignment of a CountryCodeValue to Address.state would be illegal.

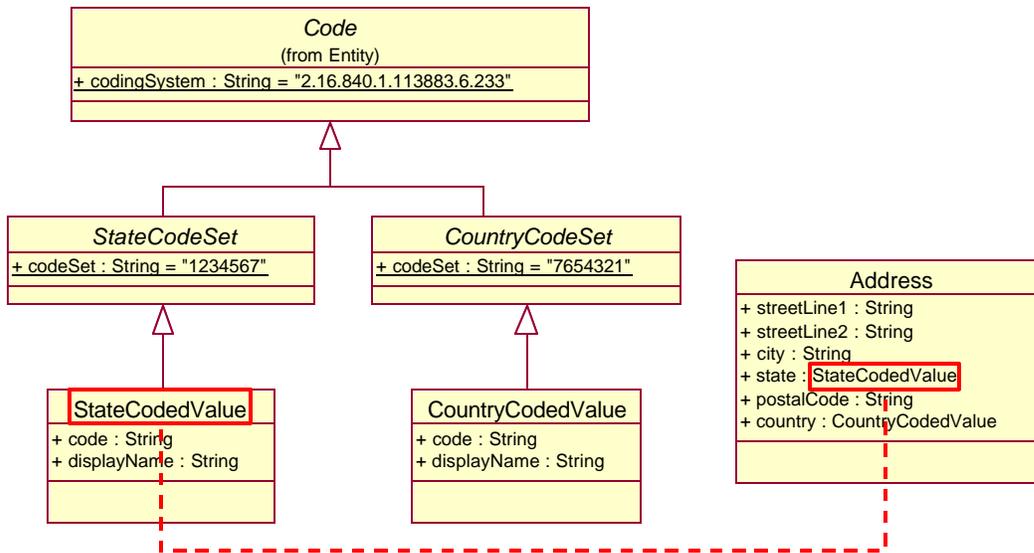


Figure 5: Explicitly declaring acceptable coded values

Also note that `Code.codingSystem` represents the OID for the VHA VUID registry. The value of this element is initialized and frozen within the VHIM. The `codeSet` is also initialized and frozen in the VHIM. This `codeSet` is the VUID for the corresponding Code Set. Thus, the `codingSystem` and the `codeSet` are defined at design-time by the VHIM. The actual code (e.g., Virginia or Utah) is determined at run-time. Note that, although VUIDs are integers, they are declared as Strings. This allows us to substitute an OID for the VUID when communicating outside of the VHA.

These explicitly defined codesets are called Specialized Refined Data Types (SRDTs). The SRDTs created for a given domain are placed in a package within the Domain package. Thus, when a partition is being modeled, the modeler does not need to check out multiple packages. In addition, this structure prevents the domain model from being cluttered. The RDTs are stored within the VHIM RIM package.

Please note that, while this example is showing how `Entity.Code` is being subclassed, such subclassing must be performed for each coded RDT. For example, a surgery Act might require a surgery type codeset. In this case, the top-level class would be `Act.Code`, not `Entity.Code`. Note also that those coded RDTs that have been subclassed into SRDTs are abstract in the VHIM RIM, and contain only the `codingSystem` attribute. Other RDTs, which have not yet been subclassed into RDTs are concrete in the VHIM RIM, and contain all four attributes.

As mentioned previously, the HL7 process involves taking a general specification and constraining the specification as needed for individual domains. This process applies to data types as well. For example, an element that is defined as a GTS (General Time Specification) can be constrained to be a TS (Time Stamp) for a particular instance. This is handled by the VHIM by subtyping the RDT into type-specific RDTs. For example, in HL7, `Observation.value` is defined as an ANY. Figure 6 shows how the corresponding RDT (the `Value` class) is subtyped into specifically-typed classes. Thus, if it is known that a particular observation value attribute should be a real number (e.g., a temperature reading), we would specify `RealValue` as the datatype for the attribute vice `Value`.

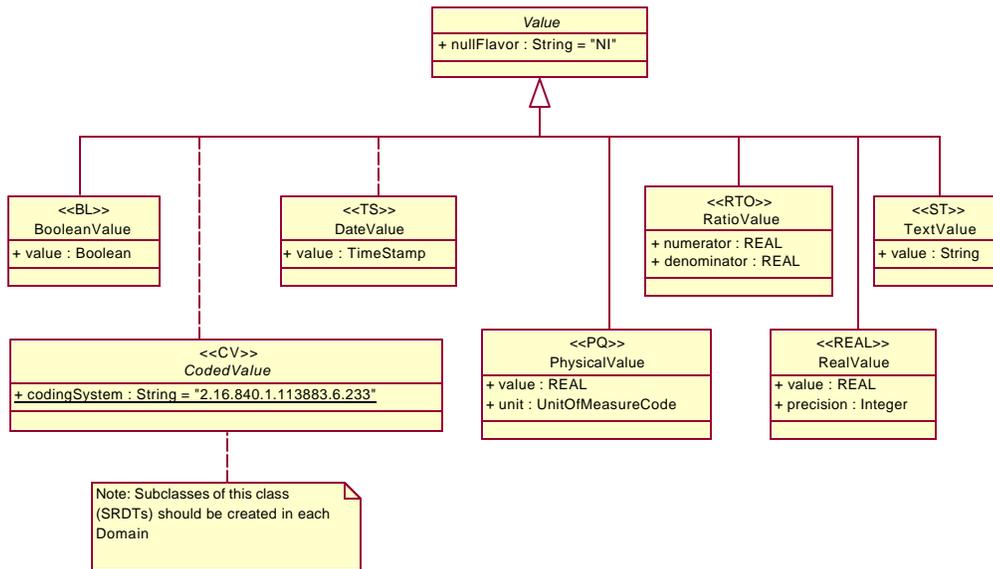


Figure 6: Explicitly declaring constrained datatypes

Two style changes have been introduced in version 3.2: 1) separate classes have been created for each PractitionerParticipation that we are using, and 2) date/time RDTs have been subclassed to explicitly contain either a single date/time (a timestamp), or two date/times (a time range).

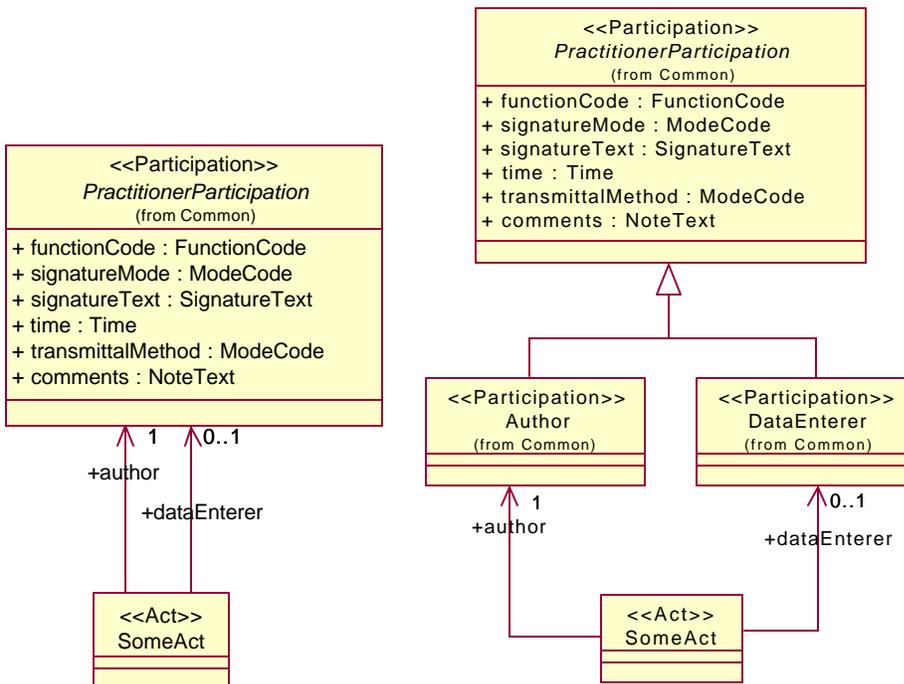


Figure 7: Practitioner Participation Style: version 3.1 to version 3.2

In previous versions, practitioner participations were modeled as associations from an Act to PractitionerParticipation. The association end name indicated what kind of participation was involved. This caused the VHIM XSDs to differ significantly from the HL7 XSDs. In version 3.2, explicitly named subclasses have been created for each kind of participation. A side effect of this style is that the association end name may be different than that of the target class, so that a more common name may be substituted for the HL7 name (e.g., we could use “co-signor” instead of “legalAuthenticator”). See figure 7.

The date/time RDTs are modeled as time intervals, and thus may contain two date/times. In some cases, it doesn't make sense to have two date/times. In HL7, datatypes may be constrained. In the VHIM we do this by subclassing. Figure 8 shows how Act.EffectiveTime has been subclassed into EffectiveTimeSpan and EffectiveTimeStamp. Note that EffectiveTime is now abstract.

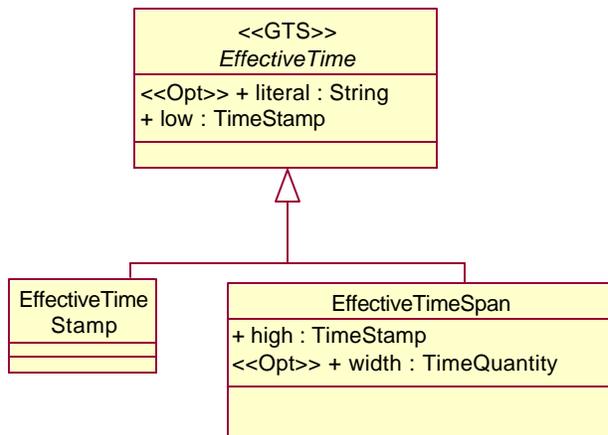


Figure 8: Effective Time subclassing

3. Models

Two series of models will be created and maintained: the Computationally Independent Models (CIM), and the Platform Independent Models (PIM). The CIM model will be named VHIMxxxx, where xxxx is the release number (e.g., VHIM3000.mdl, VHIM3100.mdl). The PIM model will be named VHPIMxxxx, where xxxx is the release number. The CIM models will define the domain packages, data types, RIM classes, RDTs, and the UML Profile. The PIM models will reuse these elements, plus add any project-specific elements, such as message template packages. It is noted that when using Rational Rose, a .cat file from one model may be re-used in another model.

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. Other packages, such as common data types will also be employed. 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 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. For example, VitalSigns.
- Each package shall be saved in its own file. The name of the file shall be identical to the package name. In Rational Rose, these files are .cat files.
- Package names will contain a version number. If any change is made to a versioned instance of a package, such as an element name, the package name shall be updated. The version number will be four digits appended to the end of the package name; the first two digits indicate the model version in which the package was **last modified**, the last two digits are an incremental number which indicate the number of revisions made to the package **since inception of the package**. A package that has not changed will retain its name. For example, say that a Lab package was created in VHIM 3.0, was unchanged in VHIM 3.1, but modified in VHIM 3.2. The package name would be Lab3001, Lab3001, and Lab3202 in VHIM 3.0, 3.1, and 3.2, respectively.
- A package will be frozen once the VHIM version in which it was constructed is released. Any defect change that causes a maintenance release of the VHIM will cause the package name to be changed. For example, if a defect was discovered in Lab3201 within VHIM 3.2, the Lab package may be corrected and called Lab3202.
- Project-specific packages (such as a Message Template) will not refer to more than one version of a domain package for the same topic and a domain package will not refer to more than one version of the data types package. For example Labs3001 has been released with VHIM 3.0 and new concepts need to be added for VHIM 3.2. A new

package Labs3201 is constructed which starts as a complete clone of Lab3001 from which features are added or removed as necessary.

- While desirable, it is not required that a new version of a domain package be backward compatible with its previous version. It is up to the individual projects to determine to which version they will conform.

5. 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. For example, MedicationOrder.
- 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 VHIM 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).
- Association classes are not acceptable due to tooling issues. If an association class is needed, transform it into an intermediate class between the connected classes.

6. Attributes

- Attributes should be named so that there is as little ambiguity as possible in the purpose of the attribute. Ideally, any user of the model should be able to deduce what the attribute represents based simply on the name. For example, the term “entry” is used often in CPRS. Entry to what? Is this when a patient enters a hospital? By using the term “dataEntry”, it is now clear 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. For example, “numberOfRefillsAuthorized”. 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 are acceptable. Thus, deaNumber is 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. For example, “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 by 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 of an attribute must be a Refined Data Type (RDT), or a Specialized Refined Data Type (SRDT). Furthermore, the data type must be (or inherit from) a VHIM RIM attribute within the class being used as a pattern. For example, RepeatNumber is a valid data type, but is only valid for Act classes.
- If the data type of an attribute is another VHIM class (i.e., not a RDT or SRDT), it should be modeled as an association to the other class.
- If an attribute is optional, apply the stereotype <<Opt>>.
- If an attribute has a cardinality of more than one (i.e., 0..n or 1..n), use the stereotype <<Set>>, <<List>>, or <<Bag>>

7. 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 (i.e. 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.
- Composite associations are acceptable. 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.

8. UML Profile

A UML Profile will be maintained for the VHIM in the VHIM RIM package. All stereotypes used in the VHIM, other than standard UML stereotypes (e.g., Actor, Boundary), will be defined in that package. Note that because Rose doesn't support UML Profiles, it is up to the individual modeler to manually make sure that the Stereotypes used are defined in the VHIM RIM.

9. 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. Because the current tool in use, Rational Rose, does not enforce this rule automatically, care must be taken to ensure that the stereotypes applied are indeed in the UML Profile and that the model elements to which the stereotype is applied is a legal element for that stereotype. Stereotype names will be in UpperCamelCase.

10. 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. Bear in mind that the definitions will be published in other forums, such as the MetaData Repository, 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, NCPDP, Webster's), 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 datatype definition, as the HL7 RIM definition is generic, does not adequately describe the specific concept, and can be found from the datatype anyway.
- If an example is given, use "e.g.," within parenthesis. 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 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.

11.Naming

To determine a name for a concept being modeled, the primary source will be the HL7 version 3 Domain Message Information Model (DMIM), Refined Message Information Model (RMIM), 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 Liason team to be brought back to HL7 for harmonization.

12. Aesthetics

- Each diagram should contain a label in the top left corner identifying the diagram name, version, and date. The label should be light blue.
- 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).
- The text font used should be Arial 10.

13. Issues

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

- Rational Rose does not display cardinality of attributes. This forces the use of the <<Opt>> stereotype for optional attributes, and the use <<Set>>, <<List>>, and <<Bag>> for attributes with a cardinality of more than one.
- Rational Rose does not properly support tagged values. While User Defined Properties can be used, these are not visible in the diagrams. Tagged values may be used to track certain metadata.
- Rational Rose does not yet support UML Profiles. While the VHIM UML Profile has been defined and will be used, the tool will not enforce the profile. A manual effort will be needed to ensure that the profile is properly applied.
- Rational Rose does not support qualified associations.
- Rational Rose does not permit grouping of attributes within a class.
- Rational Rose does not support constraints.

14. Acronyms

CIM	Computationally Independent Model
DMIM	Domain Message Information Model (from HL7 3.0)
HL7	Health Level 7
PIM	Platform Independent Model
RDT	VHIM Refined Data Type
RIM	Reference Information Model (from HL7 3.0)
RMIM	Refined Message Information Model (from HL7 3.0)
SRDT	Specialized Refined Data Type
UML	Unified Modeling Language
VHIM	VHA Health Information Model