Northwestern University
The Feinberg School of Medicine
Department of Radiology
Imaging Informatics

Stanford University
School of Medicine
Department of Medical Informatics

Annotation and Image Markup Version 3 Project: Requirements, Design, Implementation and Usage

Document Number : NU-SU-PR-008-09-20-2010

Abstract

The Annotation and Image Markup (AIM) version 3 revision 11 information model is an extension of the AIM version 2 revision 15. AIM version 3 has evolved in response to the feedback and changing demands of the imaging community. Feedback was collected as part of requirements for improvement. The model captures anatomic entity characteristic, inference, annotation role, AIM status as well as characteristic quantification. An inference provides a conclusion derived by interpreting an imaging study and/or medical history. Annotation role describes the role of referenced annotation. AIM status captures a status of an annotation instance using a coded term, a version of annotation instance and update authorization. A quantification can be a numerical value, an interval, a scale, a quantile and a non-quantifiable value. The project has produced the AIM information model, AIM library, and a validation and transformation tool. The AIM library is written in C++, using DCMTK and Xerces for DICOM and XML creation and manipulation. The library has two logical components: implementation of the AIM Schema as an object model and definition of transformations, which can be performed on the AIM object model. The ANIVATR tool is used for validating AIM annotations and transcoding between AIM XML and DICOM SR.

0. Executive Summary

AIM version 3 is an evolutionary response to the changing demands of the imaging community. The AIM standard has demonstrated its usefulness by integrating the descriptive information of an image with user-generated graphical symbols and textual descriptors placed on the image into a single common information source. The project has accomplished the project's requirements as set forth by the caBIG™ In Vivo Imaging Workspace. The AIM information model provides a supporting infrastructure for creation and collection of medical image annotation needs. Annotation of findings and objects of interest in large clinical or research data collections are fully supported in the model. The AIM project focuses on annotation and markup of DICOM images. These annotations and image markups are information objects that are linked to (but separate from) the images. The AIM information model is, however, compatible with other image formats beside DICOM. This approach is forward-looking, providing an infrastructure for future extension of base functionality that will enable future caBIG™ projects, such as annotation for pathology and genomic data, as well as documentation and tracking of quantitative changes to image features.

The AIM information model or AIM schema is described using an UML class diagram. The model is used to express and capture image annotation and markup information relevant to images. An annotation can be explanatory or descriptive information, generated by humans or machines, that directly relates to the content of a referenced image or images. It describes information regarding the meaning of pixel information in images. Annotations also become a collection of image semantic content that can be used for data mining purposes. An image markup is the graphical symbols or textual descriptions associated with an image. Markups can be used to depict textual information and regions-of-interest visually along side of, or more typically when overlaid upon, an image. Information from annotations and markups are used to populate AIM schema via the AIM software library for the purpose of generating AIM DICOM SR objects and AIM native XML documents.

The AIM 3.0 captures anatomic entity characteristic, inference, annotation role, AIM status as well as characteristic quantification. An inference provides a conclusion derived via interpreting an imaging study and/or medical history. Annotation role describes the role of referenced annotation. AIM status captures a status of an annotation instance using a coded term, a version of annotation instance and an update authorization. A quantification can be a numerical value, an interval (e.g. 34-67%), a scale (e.g. 1:None, 2:Mild), a quantile (e.g. 1(1-50), 2(51-100)) and a non-quantifiable (e.g. none, mild, mark).

The AIM library is created using C++ and the open source DCMTK DICOM toolkit. The AIM library has a collection of public mutators and accessor methods or APIs to manipulate and retrieve information in the AIM model. It also has a set of APIs to read, write and transcode AIM artifacts to and from AIM DICOM SR objects and AIM XML documents.

The AIM deliverables consist of a UML AIM model, the AIM toolkit library to create AIM annotations in AIM XML, DICOM SR and ANIVATR, which is a tool for validating AIM annotations and transcoding between AIM XML and DICOM SR. AIM XML to HL7 CDA conversion tool will be provided in a separate package.

1. Requirements

1.1 Background and Purpose

Modern medical images contain vast amounts of information captured in standard DICOM format. While this information may include meta-data about the image, such as how or when the image was acquired, the majority of image information remains in pixel data. This data contains rich content that is neither explicit nor easily accessible by programs. The information about how images are perceived by human or machine observers is not currently captured in a form that is directly tied to the images in a structured manner. A wealth of data pertaining to image content is thus segregated from the images, limiting the value of radiology images for use with other non-imaging data, such as cancer clinical trials.

There are neither widely adopted standard terminology nor syntax used to capture annotation, markup and computational descriptions of image features or non-imaging biomedical data. This results in limited interoperability between imaging and health information system. The majority of the human image features and descriptions in the biomedical domain are captured only as free text. This free text often has no association with the spatial location of the feature, making it difficult to relate image features that are described in that text to the corresponding image locations. Free text is also cumbersome, in both the lay and technical sense, as indexing, querying, and searching in order to retrieve images or their features based on free text descriptions is labor burdensome.

Our purpose is to address the requirements of the caBIG™ In Vivo Imaging Workspace's Annotation and Imaging Markup (AIM) [1] Development project and to provide an information model and reference implementation of the information model that supports the annotation needs such as the annotation of findings and objects of interest in large clinical or research data collections. Our approach is forward-looking, providing an infrastructure for future extensibility of our base functionality that will enable future caBIG™ projects, such as pathology, genomics, documentation and tracking of quantitative changes to image features.

1.2 Description and Scope

Image annotation information obtained in cancer clinical trial research is collected in both a clinical setting of commercial information systems and a research world of more flexible and tailored software provenance. This diversity of systems has led to the development of a variety of technical frameworks and standards. However, we still need tools that will allow both human and machine image annotations to be created and stored in a standard format that is at the same time syntactically and semantically interoperable with the infrastructure of caBIG™ as well as the widespread healthcare standards, DICOM [2] and HL7.

Selecting a single standard format to store image annotations will streamline software development, and enable the work to focus on providing rich annotation features and functionality. Designing the software library and tools to be compatible with other standards will enable a high degree of interoperability and allow the incorporation of the annotation standard into commercial and clinical information systems. This has the potential to open many existing resources and databases of cancer-related image data and metadata for exploitation not only by caBIG™, but also by the broader research and clinical radiology community.

The scaffold of AIM is the creation of UML class diagram information model, use of AIM software library and tools. The model is used to express and capture image annotation and markup information relevant to images. ***An annotation can be explanatory or descriptive information, generated by humans or machines, directly related to the content of a referenced image or images. It describes information about the meaning of pixel information in images. Annotations become a collection of image semantic content that can be used for data mining purposes. An image markup is the graphical symbols or textual descriptions associated with an image. Markups can be used to depict textual information and region-of-interest visually along side of an image. An AIM annotation is a collection of associated annotations and markups. They are used to populate AIM information model in the AIM software library for the purpose of generating AIM DICOM SR objects, AIM native XML documents, and AIM HL7 CDA documents.***

The AIM software library is a collection of C++ application programming interfaces used to construct AIM information model (also known as "AIM schema" [3]) based on UML class diagram.

A reference implementation of the AIM schema can be found in ANIVATR. The ANIVATR software application validates AIM annotations and transcodes them into different artifacts, namely native AIM XML and DICOM SR. We have developed AIM annotations in DICOM SR such that they can be created and displayed in a variety of medical imaging workstations, notably the eXtensible Imaging Platform (XIP) as well as in clinical imaging devices. ANIVATR reads and transcodes DICOM SR into AIM XML representation so that AIM annotation and markup can be included in appropriate HL7 CDA documents (XSLT translation to be provided in a separate package after the official release of AIM 3.0 model and toolkit).

1.3 Information Model Requirements

Our overall goal is to create an AIM schema that has the required information necessary to record an image annotation and markup and to create an AIM DICOM SR object. As described in the previous section, AIM distinguishes between image annotation and markup. Annotations provide information about images, while markups depict visual graphic and textual representation of the annotations applied to the images.

Since the AIM schema is used to create AIM DICOM SR, it is required to have the minimum information necessary to create a DICOM SR object. We elected to construct AIM SR objects based on the DICOM Comprehensive SR information object definition [2], part three. We also used Mammography CAD SR content tree structures [2], part 17, and DICOM Structure Reporting and Cancer Clinical Trials Results [4,5] as guiding examples of how to create a functional SR object. These documents also served as sources for AIM requirements.

In general, the model shall encompass the below information:

Author
Equipment and software version used to create AIM objects
Annotation
Image annotation
Annotation of annotation
Role of an annotation
Markup (graphical and textual)
DICOM SR graphic types

DICOM SR spatial coordinates
Calculation results from the markup
Associate a calculation result with a geometric shape
Patient demographic information
Anatomic structure
Annotation of Image finding (or Image Semantic Content)
Characteristic quantification of anatomic structure and annotation of image finding
Image inference
Image segmentation
Image presentation state
Referenced image
DICOM (study, series, image)
Web location

Based on the existing requirements in the AIM version 1 and 2, the table below summarises complete requirements for AIM 3.

Requirement Number	Description
R1.	**"caGridId"**
R1.1.	It shall be used as one of the key attribute for all classes except subclasses.
R2.	**Annotation**
R2.1.	It shall have a type as controlled terminology. Type shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.
R2.2.	It shall have model version, data time created, name or description of annotation, unique identifier.
R2.3.	It shall have a reference to user or machine created the annotation.
R2.4.	It shall have a reference to the previous annotation.
R2.5.	It shall have a reference to calculation.
R2.6.	It shall have a reference to anatomic entity.
R2.7.	It shall have a reference to imaging observation.
R2.8.	It shall have a reference to inference and AIM status (to keep track of AIM object).
R2.9.	It shall have information about the equipment (a separate class) that generate an annotation.
R2.9.1.	It shall have manufacture name.
R2.9.2.	It shall have manufacture model name.
R2.9.3.	It shall have software version.
R2.10.	It shall have information about user (a separate class) who create an annotation.
R2.10.1.	It shall have author name.
R2.10.2.	It shall have login name.
R2.10.3.	It shall have role in trial.
R2.10.4.	It shall have number within role of a trial.
R3.	**Image Annotation**
R3.1.	It shall inherit all properties from Annotation.
R3.2.	It shall have information about a patient (Person on NBIA model and a separate class).
R2.2.1.	It shall have patient name.
R2.2.2.	It shall have patient ID.
R2.2.3.	It shall have patient date of birth.

Requirement Number	Description
R2.2.4.	It shall have patient sex.
R2.2.5.	It shall have patient ethnicity.
R3.3.	It shall have a relationship with DICOM segmentation.
R3.4.	It shall have a relationship with Image Reference.
R3.5.	It shall have a relationship with Geometric Shape.
R3.6.	It shall have a relationship with Text Annotation.
R4.	**Annotation of Annotation**
R4.1.	It shall inherit all properties from Annotation.
R4.3.	It shall be able to reference to another Annotation of Annotation (referred as referencedAnnotationID)
R4.4.	A reference annotation shall have a role of the annotation, e.g. baseline, 2nd scan, etc.
R5.	**Calculation**
R5.1.	It shall have a unique identification.
R5.2.	It shall have a description.
R5.3.	It shall allow mathML description.
R5.4.	It shall have type of calculation that is captured as a coded term.
R5.5.	It shall have an unique identification.
R6.	It shall have algorithm name and version number.
R7.	A calculation shall be able to associate with a particular geometric shape
R8.	Type shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.
R9.	**Calculation Result**
R5.1.	It shall have a calculation result for each calculation.
R5.2.	It shall have type of calculation result such as scalar, vector, histogram, matrix or array.
R5.3.	It shall have number of dimension.
R5.4.	It shall have unit of measure.
R5.5.	It shall associate with an object used to contain data.
R5.5.1.	For each data, it shall associate with an object used to identify dimension index and the position within the dimension.
R5.6.	It shall associate with an object contained number of dimension a calculation result has.
R6.	**Annotation Role**
R6.1.	It shall have controlled vocabulary describing a role of an annotation in particular set of clinical trial.
R6.2.	It shall have a number assigned for each role.
R7.	**Reference Geometric Shape**
R7.1.	It shall have an attribute used to hold a value of referenced shape identifier.
R8.	**Anatomic Entity**
R8.1.	It shall have an anatomic location term from a recognized controlled vocabulary (RadLex, SNOMED-CT, UMLS, etc.). It shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.
R8.2.	It shall have a flag to capture if an anatomic entity is present or not.
R8.3.	It shall have an attribute that capture human readable description of the entity. This information is more than code meaning description.
R8.4.	It shall have an annotator confidence, described in term of percentage.
R9.	**Anatomic Entity Characteristic**
R9.1.	It shall have an anatomic location term from a recognized controlled vocabulary. It shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.

Requirement Number	Description
R9.2.	It shall have an anatomic characteristic describing Anatomic Entity.
R9.3.	It shall have a reference to characteristic quantification.
R9.4.	It shall have an annotator confidence, described in term of percentage.
R10.	**Capture AIM status**
R10.1.	It shall capture AIM status as a coded value (e.g preliminary, final, addendum, etc.).
R10.2.	It shall capture version number.
R10.3.	It shall record the name of the person who make changes to an AIM instance.
R11.	**Imaging Observation**
R11.1.	It shall have an anatomic location term from a recognized controlled vocabulary (RadLex, SNOMED-CT, UMLS, etc.). It shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.
R11.2.	It shall have a flag to capture if an imaging observation is present or not.
R11.3.	It shall have an attribute that capture human readable description of the entity. This information is more than code meaning description.
R11.4.	It shall have an annotator confidence, described in term of percentage.
R12.	**Imaging Observation Characteristic**
R12.1.	It shall have an anatomic location term from a recognized controlled vocabulary. It shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.
R12.2.	It shall have an anatomic characteristic describing Anatomic Entity.
R12.3.	It shall have a reference to characteristic quantification.
R12.4.	It shall have an annotator confidence, described in term of percentage.
R13.	**Characteristic Quantification**
R13.1.	It shall capture one or more identifiable and nonidentifiable quantifications.
R13.1.1.	It shall capture numerical value with a UCUM unit. Optional comparison operator may be used to associate the value, e.g. > 3 mm.
R13.1.2.	It shall capture quantile.
R13.1.3.	It shall capture interval value with a UCUM unit. A value falls in between maximum and minimum value. A maximum and minimum value has a comparison operator associate with it.
R13.1.4.	It shall capture scale value.
R13.1.5.	It shall capture nonquantifiable coded concept.
R14.	**Inference**
R14.1.	It shall have controlled vocabulary described an inference that are described as controlled terminology. It shall be captured as code value, code meaning and coding scheme designator with coding scheme version as an option.
R14.2.	It shall have annotator confidence.
R14.3.	It shall have imaging evidence.
R15.	**Segmentation**
R15.1.	It shall have its own instance UID.
R15.2.	It shall have SOP Class UID.
R15.3.	It shall have reference instance UID.
R15.4.	It shall have segmentation number.
R16.	**Presentation State**
R16.1.	It shall have a reference to a DICOM presentation state.
R17.	**Image Study (formerly known as Study)**
R17.1.	It shall have study instance identifier.
R17.2.	Start time of an exam shall be added.

Requirement Number	Description
R17.3.	The name of the class shall be ImageStudy
R18.	**Image Series (formerly known as Image)**
R18.1.	It shall have series instance identifier.
R19.	**Image**
R19.1.	It shall have SOP class uid.
R19.2.	It shall have SOP instance uid.
R20.	**Equipment**
R20.1.	It shall have manufacturer name.
R20.2.	It shall have manufacturer model name.
R20.3.	It shall have software version information.
R21.	**Text Annotation**
R21.1.	It shall have font, font color, font effect, font size, font style, font opacity, and text justify as optional.
R21.2.	It shall have text to store textual information.
R21.3.	It shall associate with Multi Point class to specify the location of the text to be displayed.
R22.	**Geometric Shape**
R22.1.	It shall have line color, line opacity, line style, and line thickness as optional.
R22.2.	It shall have a notion of inclusion and exclusion, e.g. for a donut case.
R22.3.	It shall have a number to identify each shape.
R22.4.	It shall have Multi Point, Point, Circle, Ellipse, and Polyline subtype per DICOM SR.
R22.5.	It shall have an association with Special Coordinate
R23.	**Spatial Coordinate**
R23.1.	It shall be ordered.
R23.2.	It shall have Two Dimension Coordinate and Three Dimension Coordinate as subtype.
R23.3.	Two Dimension Coordinate shall have x and y coordinate, DICOM Image Reference UID and DICOM Reference Frame Number.
R23.4.	Three Dimension Coordinate shall have x, y and z coordinate and DICOM Frame of Reference UID.

Table 1. Requirements

1.4 Tools

The list indicates the tools used in this project.

a. Altova XMLSpy 2010 (Altova)
b. caCORE SDK 3.2.1 (caBIG) [5]
c. DCMTK version 3.5.4 (OFFIS e. V.)
d. Eclipse (Eclipse Foundation, Inc.)
e. Enterprise Architect version 6.5.804 (Sparx Systems)
f. Java JDK 6 (Sun Microsystems)
g. Microsoft Visual Studio 2008 (Microsoft) with STL
h. Xerces XML library version 3.0.1 (Apache Software Foundation)
i. Boost C++ library version 1.34.1 (Boost.org)

1.5 Constraints

The AIM library has been developed on Microsoft C++ Visual Studio 2008 without using any specific or proprietary Microsoft C++ statements. The library has not been compiled with other development environments on other platforms such as Linux or Mac OS X.

2. Design

The AIM schema is used to capture information about how images are perceived by human or machine observers that is directly tied to medical images. Our design process started with understanding the initial requirements and identifying a set of objects that are used to collect information about imaging annotation and markup. The requirements are in section 1.3. Classes are divided into image semantic content, calculation, markup and image reference. If there are classes that do not pertain to a specific group, we classify them in the general information group. These classes contain information about the workstation used to create the AIM annotations, the user that creates the AIM annotations, patient identification, DICOM segmentation, AIM status, AIM annotation, and annotation of annotation.

The AIM schema, shown in figure 1, has evolved through an iterative feedback process since the release of AIM version 1 revision 12 silver compatible. The model has gone through many reviews and recommendation processes. The current AIM model is version 3 and revision 10.

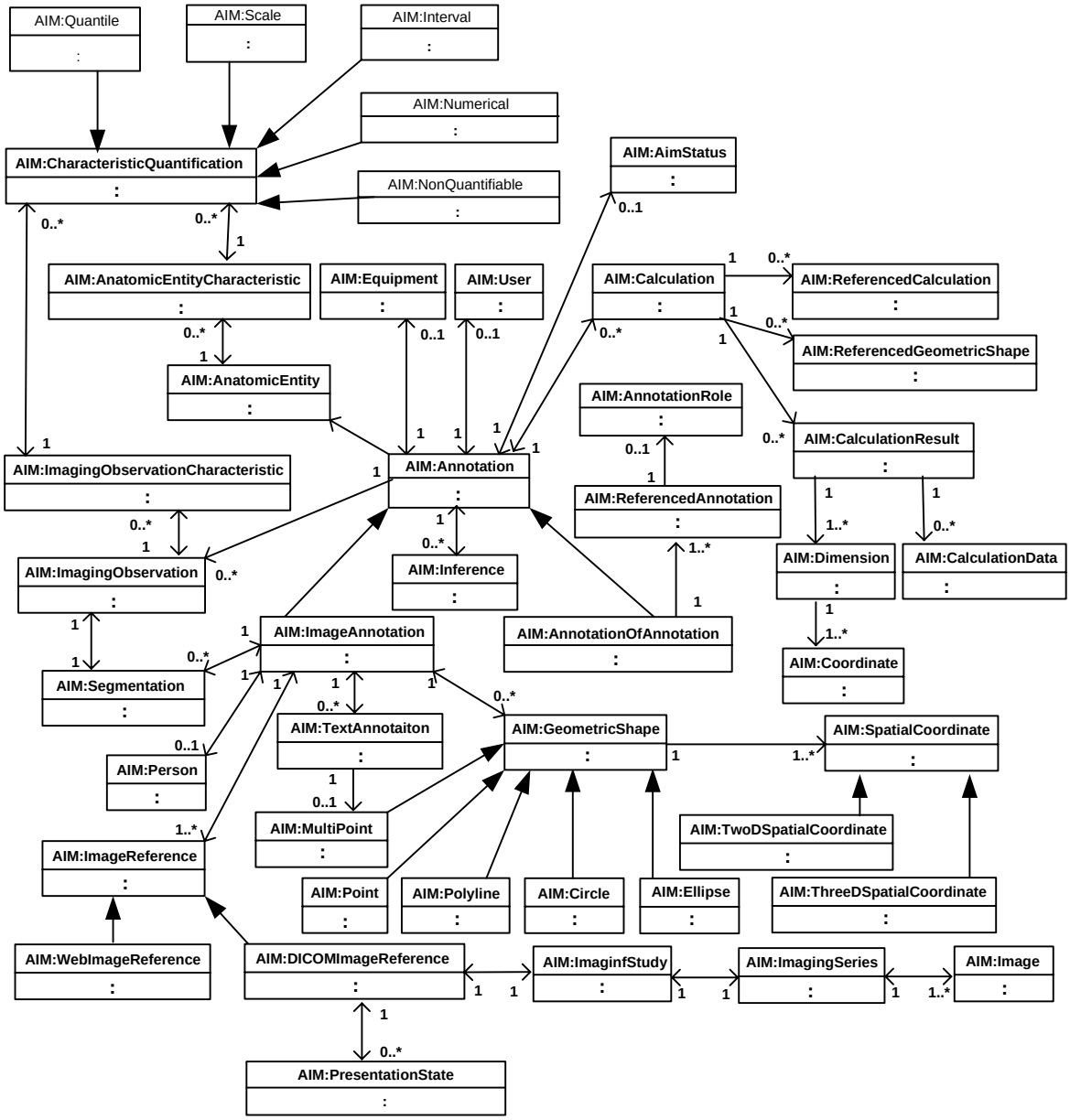

Figure 1. UML Class Diagram for AIM 3.0 Schema

Enterprise Architect (Sparx Systems) can be used to view the AIM UML class diagram from approved_annotated_AIM_v3_rv11.EAP. One can also view this diagram in JPEG format using the file named approved_annotated_AIM_v3_rv11.jpg.

2.1 Understanding The AIM schema

AIM annotations can only be either ImageAnnotation or AnnotationOfAnnotation. These are the two root objects that inherit all properties of the abstract class, Annotation. These are the two kinds of annotation that can be instantiated. ImageAnnotation class annotates images. AnnotationOfAnnotation class annotates other AIM annotations for comparison and reference purposes.

An instance of ImageAnnotation object has annotation and markup information on one or more images in the same series and study. ***Annotation and markup information describes particular fidings of a single thing found on an image or images. For instance, if there are two nodules found on an axial image, two ImageAnnotation instances must be created.*** Image markups of the same

nodule put on different images in the same study can be captured in a single ImageAnnotation instance. ImageAnnotation must have one or more ImageReference objects, which can be either DICOM image objects or web image objects uniquely identified by an URI (Uniform Resource Identifier). It may have only one Equipment, Inference, User and Person object. ImageAnnotation may have zero or more, AimStatus, Segmentation, TextAnnotation, GeometricShape, AnatomicEntity, ImagingObservation , Inferenceand Calculation objects.

An AnnotationOfAnnotation has annotation information about one or more AIM annotations, which can be ImageAnnotation or AnnotationOfAnnotation. *An example of this type of annotation could aggregate two studies from time points one and two with calculation result of the difference in size of the two tumors.* An AnnotationOfAnnotation object must have a single unique AimStatus, Equipment and User object. It must contains one ReferencedAnnotation object. It may associate with AnnotationRole, which defines the role of referenced annotation. AnnotationOfAnnotation may have zero or more AnatomicEntity, ImagingObservation, Inference and Calculation objects.

An object type is defined by controlled terminology that is being captured by a code value, code meaning and coding scheme designator. The three attributes are used in combination to identify an object's type of image annotation or annotation of annotation. A controlled terminology can be found from RadLex, SNOMED CT, DICOM, UCUM, user defined terms or other lexicons.

Based on the information in Figure 1, we can categorize the collection of classes into five groups, General Information, Calculation, Image Semantic Content, Markup and Image References. Note that the AIM model has an attribute called "cagridId" in all of its concrete classes. The attributes are not being used directly by the AIM model as they do not have any meaningful representation in the AIM model. They are used for caGRID deployment to uniquely identify an instance.

The General Information group, shown in Figure 2, contains a collection of classes that do not belong to any group. Each class is self-described and does not have dependency to work with other classes to provide a larger concept. We begin with the Annotation class. It is an abstract base class for the AIM schema. The Annotation class captures name and general description of the AIM annotation; version of AIM schema, type of annotation via controlled terminology, creation date and time, and AIM annotation UID. *The ImageAnnotation class annotates images. The AnnotationOfAnnotation class annotates other AIM annotations for comparison and reference purposes.* The class uses ReferencedAnnotation class, which has a UID of ImageAnnotation or AnnotationOfAnnotation. Both ImageAnnotation and AnnotationOfAnnotation have AimStatus, User, Equipment and Person classes. The User class represents a person or a computing resource, which creates an AIM annotation. The User class is composed of a user's full name, login name with optional role in trial and order within the trial. The Equipment class provides information about the system that is used to create AIM annotations. The Equipment class collects manufacture name, model description and software versions. The Person class contains basic patient demographic information: patient name, identification string, birth date and sex. The Segmentation class represents a multi-frame image representing a classification of pixels in one or more referenced images. Segmentations are either binary or fractional. The Segmenation class contains DICOM SOP class UID that defines the type of segmentation, and references to its own instance UID and referenced instance UID of the image to which the segmenation is applied. It also has an identification number of the segment that shall be unique within the Segmentation instance in which it is created. An ImageAnnotation may have zero or more Segmentation objects. The ReferencedAnnotation class provides reference to another annotation

instance. Annotation role describes the role of referenced annotation. Each instance can have a role associate with it, e.g. a baseline case. A role of a ReferenceAnnotation is captured by AnnotationRole class. An Inference class provides a conclusion derived through interpreting an imaging study and/or medical history. AIM Status class captures a status of an annotation instance using a coded term, a version of annotation instance and an update authorization.

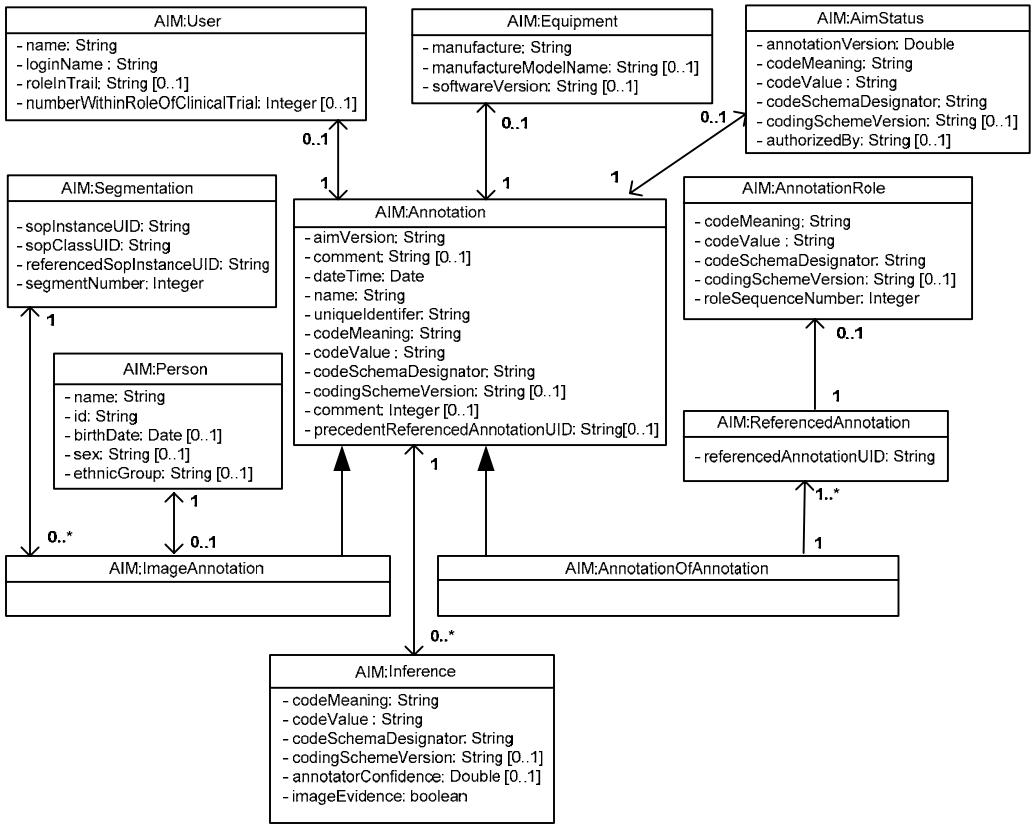

Figure 2. General Information Group

The Calculation group, shown in Figure 3, represents the calculation results of an AIM annotation. Calculation results may or may not be directly associated with graphical symbols or markups. For example, given an image with a single ellipse markup, calculation results can be an area in square millimeters and references maximum and minimum pixel values. As another example, an image has an arrow pointing to a specific location and two concentric circles, with an area measurement of the larger circle minus the smaller circle. The computation result is based on the independence calculation made upon each circle. The AIM schema allows calculation results that are not directly related to markups. Calculation class has overall information about a calculation performed as free textual description of calculations performed on AIM annotation. It defines a type of calculation, such as area, height, radius and volume of ellipsoid from UCUM, as controlled terminology that can be captured in code value, code meaning and coding scheme designator. It also captures MathML as a string attribute within the class. The ReferencedCalculation class is used as a mechanism for referencing other calculations that the current calculation is based upon. The CalculationResult class contains total number of dimensions, a string representation of UCUM unit for the dimensions and the type of result such as scalar, vector, histogram or array. The Dimension class states how many dimensions a CalculationResult has. It has a description of the dimension, the index of the current dimension and the size of the dimension. The Data class is used to store result value. The Coordinate class identifies location within a dimension for the Data class. An ImagingObservation

instance may have a reference to a particular markup or graphical drawing on an image. This reference is referred to the shapeIdentifier attribute in the GeometricShape class. Within a given AIM ImageAnnotation instance, each value in shapeIdentifier attribute has to be uniqued. A Calculation instance may also have a reference to a markup or a collection of markups.

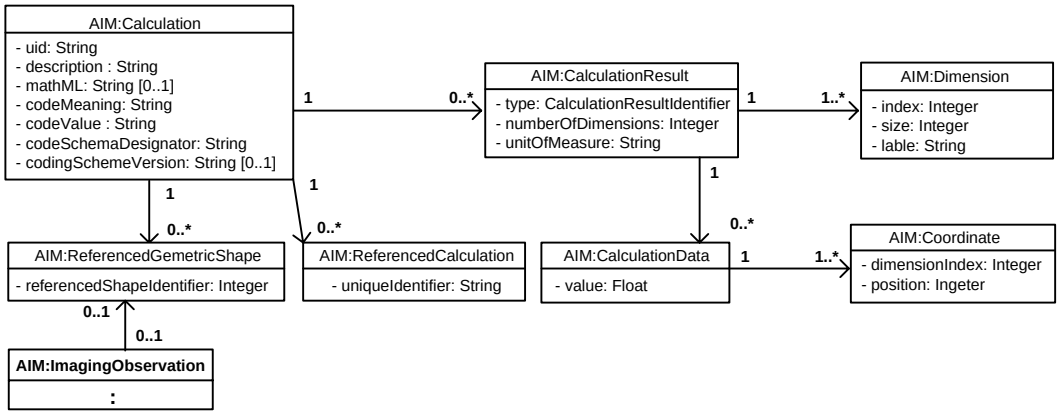

Figure 3. Calculation Group

The classes in Image Semantic Content group, shown in Figure 4, are used to gather clinical finding of images. The AnatomicEntity class has an anatomical location term from a recognized controlled vocabulary (RadLex, SNOMED-CT, UMLS, etc). The AnatomicEntityCharacteristic class further describes AnatomicEntity class such as “fracture”. The ImagingObservation class is the description of things that are seen in an image. "Mass", "Radiographic evidence of pleural effusion", "Foreign Body", Artifact", are all examples of ImagingObservations. The ImagingObservationCharacteristic class is descriptors of the ImagingObservation class such as "dense", "heterogeneous", "hypoechoic" and "spiculated". Only the final selected values of AnatomicEntity, AnatomicEntityCharacteristic, ImagingObservation and ImagingObservationCharacteristic are stored in ImageAnnotation instances. Both AnatomicEntityCharacteristic and ImagingObservationCharacteristic may have CharacteristicQuantification associated with. A quantification can be a numerical value, an interval (e.g. 34-67%), a scale (e.g. 1:None, 2:Mild), a quantile (e.g. 1(1-50), 2(51-100)) and a non-quantifiable (e.g. none, mild, mark).

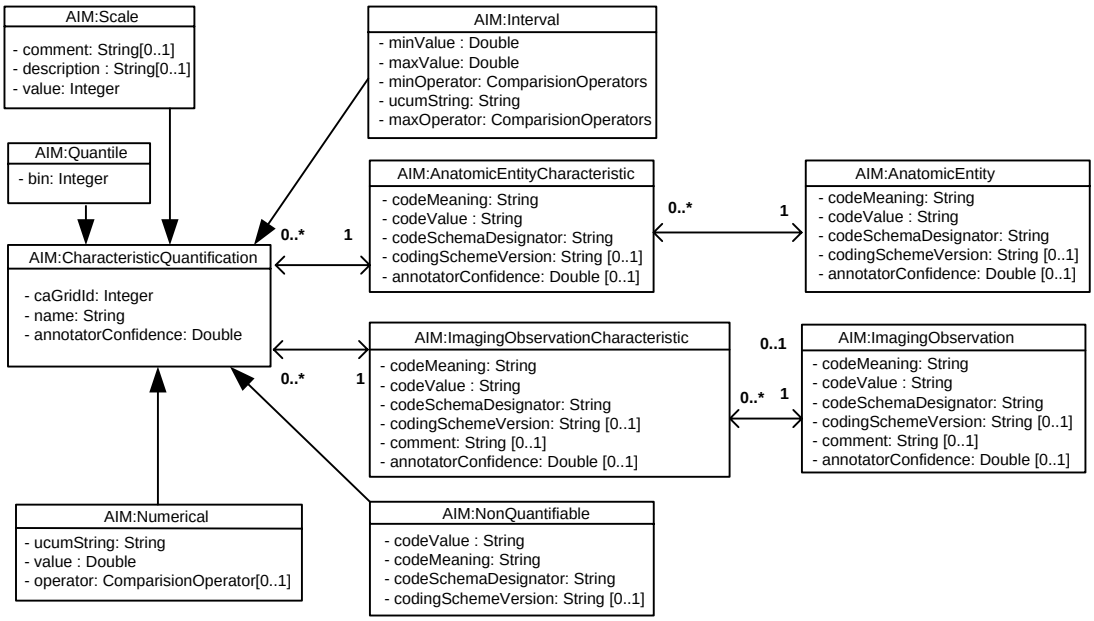

Figure 4. Image Semantic Content Group

The Markup Group, shown in Figure 5, captures textual information and graphical representation as DICOM SR value type SCOORD. The available graphic types are Point, Multipoint, Polyline, Circle, and Ellipse. Each drawing has SpatialCoordinate abstract class, which can be in two or three dimensional space. The TwoDSpatialCoordinate class has x and y coordinate, the SOPInstance UID of the image that contains the pixel and the frame number within the referenced SOP Instance to which the reference applies. The first frame shall be denoted as frame number 1. In the case of a multi-frame image, we can use the frame number from the DICOM header. The ThreeDSpatialCoordinate class has x, y and z coordinate as well as the frame of reference for a Series. The TextAnnotation class has coordinate captured as SCOORD graphic type MultiPoint. A TextAnnotation's MultiPoint implementation is expected to have no more than two coordinates which can be represented as an arrow connecting TextAnnotation to a point on an image. Only the ImageAnnotation class can have the Markup group.

POINT = a single pixel denoted by a single (column,row) pair
MULTIPOINT = multiple pixels each denoted by an (column,row) pair
POLYLINE = a series of connected line segments with ordered vertices denoted by (column,row) pairs
CIRCLE = a circle defined by two (column,row) pairs. The first point is the central pixel. The second point is a pixel on the perimeter of the circle.
ELLIPSE = an ellipse defined by four pixel (column,row) pairs, the first two points specifying the endpoints of the major axis and the second two points specifying the endpoints of the minor axis of an ellipse

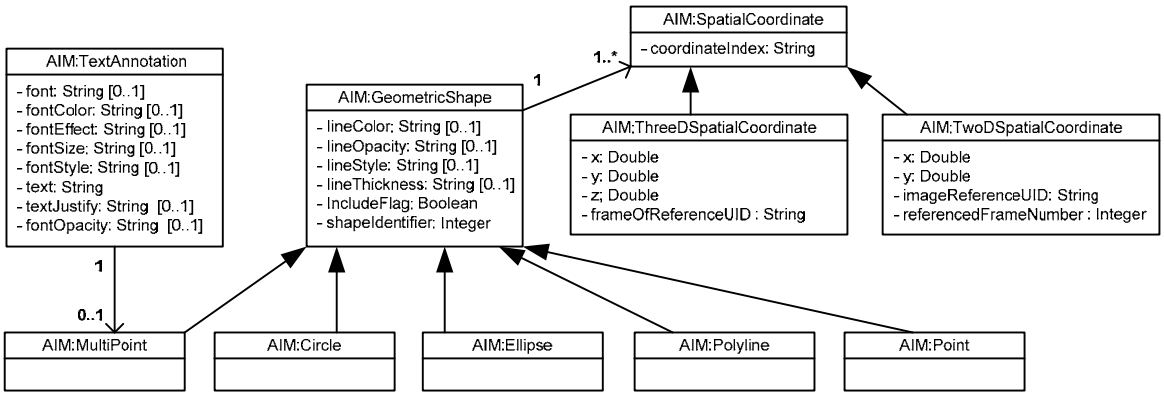

Figure 5. Markup Group

The ImageReference group, as shown in Figure 6, represents an image or collection of images being annotated. The two possible types of references are DICOM and URI or web image reference. First, DICOMImageReference object mimics the DICOM information model. It has one ImageStudy object that has one ImageSeries object, which in turn has one or more Image objects. ImageStudy class has study instance UID, start date and start time. The ImageSeries class has series instance UID. The Image class has SOP class UID and SOP instance UID. The second image reference type is WebImageReference that contains a URI to an image.

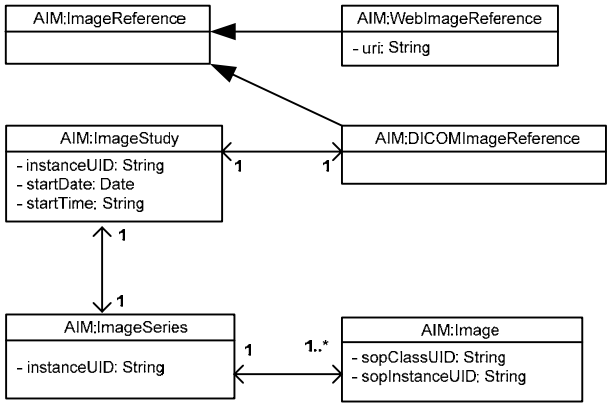

Figure 6. ImageReference Group

Note that abstract classes in the AIM schema are Annotation, GeometricShape, ImageReference and SpatialCoordinate. [0..1] denotes an optional occurrence of an attribute.

2.2 AIM Schema for AIM XML Document

The AIM UML class diagram, Figure 1, contains bi-directional associations between many classes for caGRID data services such as storage, query and retrieve. The model in Figure 1 generates XML schema that works perfectly for such requirements. In general, each class in the UML class diagram can be mapped to an XSD element and to a complex type in an XML schema. Each of these elements can be a root element in an XML document. A directional association relationship in the UML class diagram is translated (to a containing or has-a relationship). For example, the Annotation class and the User class in Figure 1 have bi-directional association between them. This means that the User class can have the Annotation class as its child in XML and vice versa. However, the AIM

schema requires an XML schema that will precisely enforce the AIM schema structure. Thus, the only two root elements in the AIM schema are ImageAnnotation and AnnotationOfAnnotation. This means that the Annotation class of type ImageAnnotation can have a User class element and not the other way around. Therefore, there is a need to create a separate UML class diagram that contains uni-directional associations. The AIM schema, Figure 7, for creating AIM XML document is used to generate an XMI file for the XML schema generation by caCORE SDK 3.2.1. The new XML schema is used by AIM library to create and validate AIM XML documents.

If the AIM library were to use the XML schema that contains bi-directional associations between classes and were not to limit the root elements to ImageAnnotation and AnnotationOfAnnotation, any class in the XML schema could server as a root element. Any class that has a directed association to another class could have that other class as its child element in XML. This behaviour is not desirable for the information model that the AIM schema has to capture and represent.

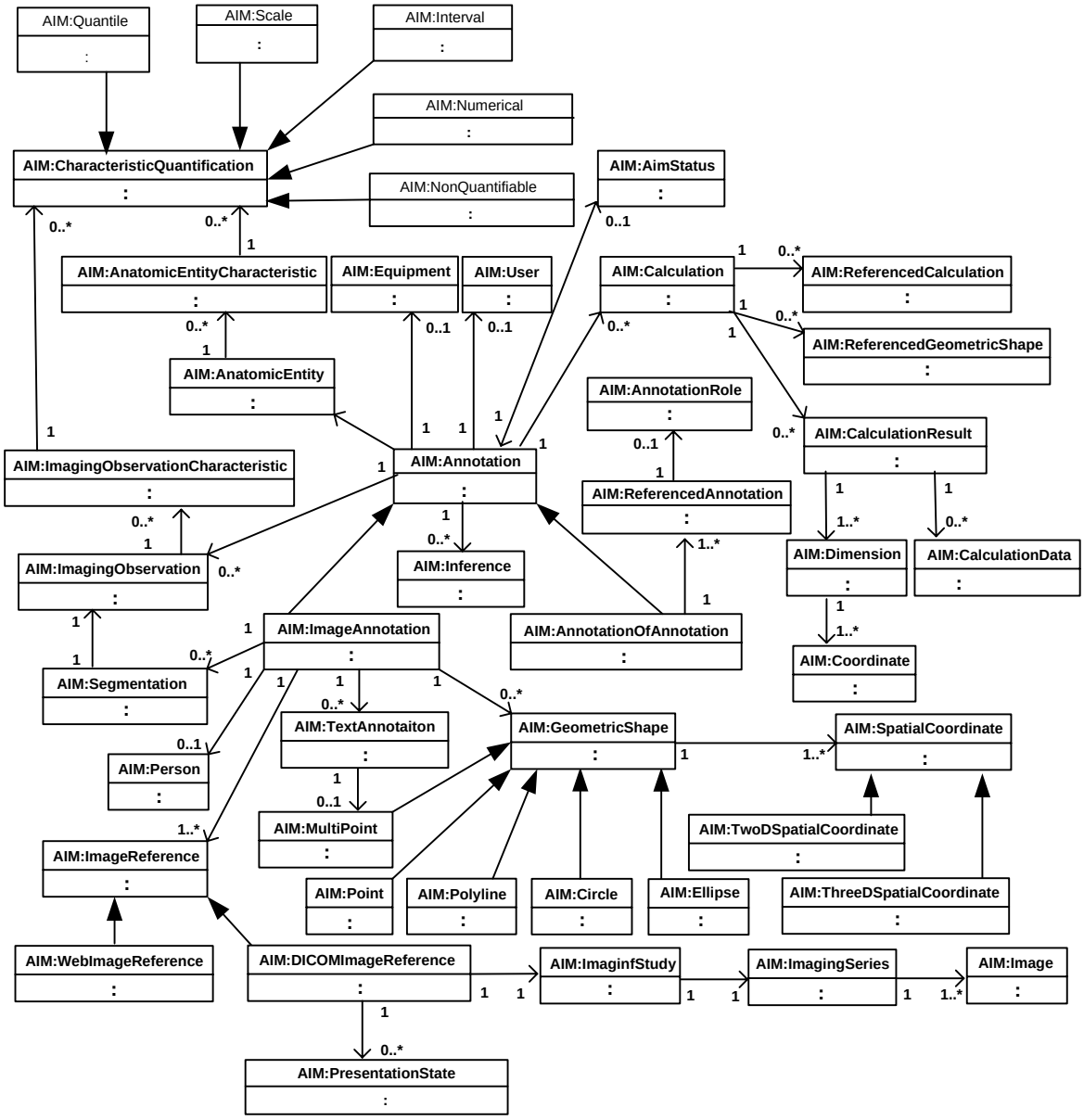

Figure 7. AIM UML Class Diagram for Creating AIM XML Document

Any valid AIM XML documents generated by the AIM schema for AIM XML document will be validated by the XML schema generated using the AIM schema described in section 2.1.

2.3 Correcting AIM XML Schema

Due to differences in caGRID and AIM schema requirements and targets, the AIM model requires two types of schemas; one for creation and validation of AIM XML documents and another for caGRID services. There are five main reasons for modifying a caCORE toolkit generated XSD. Note: caCORE_SDK_32_Programmers_Guide.pdf contains step-by-step instructions for creating an XSD.

1. caCORE toolkit 3.2.1 does not currently support "value domain" mapping to XSD. Therefore, an AIM defined value domain objects are missing from a generated XML schema.
2. The toolkit changes an association with n-to-Many or 1-to-Many to 0-to-Many. The AIM model has many association relationships that require association relationships to remain as they are in the AIM model. A visual inspection and manual modification to a correct value are required to be performed.
3. The toolkit also maps class attributes as XSD optional attributes. AIM requires most of its attributes to be present. Thus, we need to add *use="required"* to all XML attributes that do not have minimum and maximum number of occurrence setting.
4. We use SDK 3.2.1 to generate XML schema. However, it does not include an attribute abstract="true" when a class in UML Class Diagram is defined as an abstract class. We have to change it manually in XSD file for all abstract classes, for example:
   a) <xs:complexType name="Annotation">
      to
      <xs:complexType name="Annotation" abstract="true">
   b) Remove all elements except ImageAnnotation and AnnotationOfAnnotaiton. You will need to change "<xs:element ref=ClassName…/>" to "<xs:element name=ClassName type=ClassName…/>". It needs to be done to allow only two type of elements be a root node of an AIM document.

3. Implementation and Usage

Based on the above model, an AIM programming library has been constructed to create, validate and transform between AIM XML document and DICOM SR. Implementors should familiarize themselves with the AIM schema and be able to deduce relationships between classes in the schema. The library is a set of APIs. It is independent from the Graphical User Interface (GUI) and workflow of an application. All interactions with the AIM library are done through AIM library APIs. In addtion to the AIM library, the application would need to provide for:

- creating, displaying and converting application markups to AIM markups
- presenting and collecting AIM annotation of image findings
- capturing and displaying annotation information
- computing and translating its computation to AIM calculations.

3.1 AIM Library

The AIM library is an C++ [7] module. It consists of two logical parts: implementation of the AIM Schema as an object model and definition of a set of operations, which can be performed on the object model.

The AIM library can be used as a linked dependency of another application. All exported library APIs are thought to conform to ANSI C++ (1998/2003). Various STL containers are used extensively throughout the library and in the public APIs.

The object model implementation creates an ANSI C++ class for each class in the AIM Schema. The Class hierarchy closely follows AIM Schema. Each object's model class provides mutation methods (Set and Get method) for every attribute in the corresponding AIM Schema class. All changes to the class' states are done via those mutation methods. The whole AIM Schema is represented by the object model through containment and inheritance.

The set of object model operations supported by the AIM library includes persisting the model in XML and DICOM SR formats. The reverse set of operations of reading XML and DICOM SR instances into the object model is supported as well.

3.2 Using AIM Library

When a software developer wants to instantiate an AIM model, they should start with creating either AnnotationOfAnnotation or ImageAnnotation object and populating its related objects' content. All required attributes need to be populated with valid data. Optional attributes are depicted in the AIM schema with [0..1].

3.2.1 ImageAnnotation

The ImageAnnotation object is required to have at least one ImageReference object of type DICOMImageReference or WebImageReference object. DICOMImageReference object must have one Study object. The ImageStudy object may have one or more Series objects. Each ImageSeries object may have one or more Image objects. *It is implied in the model that all images originate from the same study of the same patient.*

ImageAnnotation object may have a Person object. An ImageAnnotation object may have Segmentation objects, which contain references to its own instance UID and referenced instance UID of the image to which the segmenation is applied. It also has a type of segmentation as DICOM SOP class UID and an identification number of the segment. The identification of the segment shall be unique within the Segmentation instance in which it is created. ImagingObservation object is being captured as DICOM code sequence with a possible textual comment. An ImagingObservation may have zero or more ImagingObservationCharacteristic objects, which are captured as DICOM code sequences. An ImageAnnotation may store conclusions derived by interpreting an imaging study and/or medical history in a collection of Inference objects that store the information as code sequence based on a controlled terminology.

ImageAnnotation object may have zero or more TextAnnotation objects. Each TextAnnotation object may have a two or three dimensional cartesian coordinate set defined as a MultiPoint object. TextAnnotation is used as a text markup that can be shown on an image. Graphic markups are stored as GeometricShape objects. A MultiPoint, Point, Circle, Ellipse and Polyline objects inherit the GeometricShape abstract class properties and methods. Each GeometricShape has one or more SpatialCoordinate object of type TwoDSpatialCoordinate or ThreeDSpatialCoordinate. GeometricShape can have only one type of coordinates at a time. The coordinateIndex attribute in SpatialCoordinate class signifies the order in which a coordinate appears in the shape. GeometricShape class closely follows DICOM 3.0 part 3, C.18.6.1.2 Graphic Type. TwoDSpatialCoordinate contains SOP Instance UID and frame number (multi-frame image) to identify which image a GeometricShape object belongs to.

ImageAnnotation inherits properties and methods from the Annotation class. It may have at most one Equipment and User objects. It may have zero or more AnatomicEntity, ImagingObservation and Calculation objects.

A Calculation object can be related to a single markup, a collection of markups and other calculations, which are not related to markup. A calculation may reference other calculations by having ReferencedCalculation objects, which contains a referenced Calculation object UID. A Calculation object may have zero or more CalculationResult objects. It is possible for a Calculation to have no CalculationResult. This means that the information provided in the Calculation object is sufficient to describe the calculation.

A calculation result can be a scalar, vector, matrix, histogram or array. Dimensionality of calculation results is represented by Dimension objects. A CalculationResult object must have at least one Dimension object. The Index attribute in the Dimension object is a zero based unique index of the dimension. Size attribute in Dimension object says how many members a dimension has. Label attribute provides textual meaning to a dimension.

A CalculaitonResult object may have zero or more Data objects. Absence of any Data object means that result is an empty set. Each Coordinate object specifies a dimension index and a position within the dimension. The number of Coordinate objects for each Data object cannot exceed the total number of Dimension objects in a CalculationResult. Data object cannot have more than one Coordinate object with the same dimensionIndex.

3.2.2 AnnotationOfAnnotation

AnnotationOfAnnotation works very much the same way as ImageAnnotation for calculation group, image semantic content group, Equipment class and User class, see section 2.1. AnnotationOfAnnotation object must have at least one ReferencedAnnotation object that contains a UID of ImageAnnotation or AnnotationOfAnnotation object. AnnotationOfAnnotation may store a conclusions derived by interpreting an imaging study and/or medical history in a collection of Inference object, which stores the information as a code sequence based on a controlled terminology.

AnnotationOfAnnotation may refer to a collection of ImageAnnotation object thatcan come from different studies.

The AIM model and DICOM templates do not explicitly address the issue of Study Instance UID, Series Instance UID and SOP Instance UID creation of an AnnotationOfAnnotation object. These three UIDs can be generated by AIM implementers for the purpose of creating AIM DICOM object. When a transformation process from AIM DICOM object to AIM XML or HL7 CDA, these three UIDs are not being used.

3.3 AIM Library Objects

AIM library is created using C#. The library objects and operations are in the *aim_lib* namespace. All object model files are located in *model/* sub-directory. For convenience purposes, there is *model/AimHeaders.h* header file that includes all headers of the object model. Below is a sample of how one can populate parts of the object model.

```
#include "AIMLib/model/AimHeaders.h"

……

// Calculation and its dependents
aim_lib::Dimension dim;
dim.SetIndex(0);
dim.SetLabel("Centimeters");
dim.SetSize(1);
aim_lib::DimensionVector dimColl;
dimColl.push_back(dim);
aim_lib::CalculationResult calcResult;
calcResult.SetUnitOfMeasure("cm");
calcResult.SetType("CalculationResultTYpe::Scalar");
calcResult.SetNumberOfDimensions(1);
calcResult.SetDimensionCollection(dimColl);
aim_lib::Coordinate coordinate;
coordinate.SetDimensionIndex(0);
coordinate.SetPosition(0);
aim_lib::CoordinateVector coordColl;
coordColl.push_back(coordinate);
aim_lib::Data data;
data.SetValue(150.0);
data.SetCoordinateCollection(coordColl);
aim_lib::DataVector dataColl;
dataColl.push_back(data);
calcResult.SetDataCollection(dataColl);
aim_lib::CalcResultVector calcResults;
calcResults.push_back(calcResult);
aim_lib::ReferencedCalculation refCalc;
refCalc.SetReferencedCalculationUID("1.23.5698.24546.231365.74654");
aim_lib::ReferencedCalcVector refCalcs;
refCalcs.push_back(refCalc);
aim_lib::Calculation calc;
calc.SetUID(AimUidGenerator::GenerateNewUID("33.333"));
calc.SetCodeValue("Value::Length");
calc.SetCodeMeaning("length");
calc.SetCodingSchemeDesignator("CALC_SCHEME");
calc.SetDescription("Description of the Calculation One - Length");
calc.SetMathML("MathML for the Calc One goes here");
```



```
calc.SetCalculationResultCollection(calcResults);
calc.SetReferencedCalculationCollection(refCalcs);
calculations.push_back(calc);
```

3.4 AIM Library Operations

Operations provided by the AIM library reside in the *operations/* sub-folder of the library. All operations are performed on the object model via *DCMModel* and *XMLModel* classes.

AIMLib/operations/DCMModel.h contains operation for DICOM SR.
To read AIM library object from DICOM SR file use one of the two available APIs:

1. *ReadAnnotationsFromFile* / *GetNextAnnotation* pair will read all available annotations from a file and will make the annotations available as an object model one by one.
2. *ReadAnnotationFromFile* will read a single annotation object from a file.

Two more APIs are available to write AIM library object(s) to a DICOM SR file:

1. *WriteAnnotationToFile* / *WriteAnnotationsToFile* will serialize AIM library object(s) into a DICOM SR.

AIMLib/operations/XMLModel/.h contains operation for AIM XML.
To read AIM XML file into an AIM library object use:

1. *ReadAnnotationsFromFile* / *GetNextAnnotation* pair will read all available annotations from a file and will make the annotations available as an object model one by one.
2. *ReadFromXmlFile* will read a single annotation object from a file.
3. *ReadFromXmlString* will read a single annotation object from a string buffer.

To write AIM library object to an AIM XML file use:

1. *WriteAnnotationToFile* / *WriteAnnotationsToFile* will write AIM library object(s) to file.
2. *WriteAnnotationToString* / *WriteAnnotationsToString* will write AIM library object(s) to a string buffer.

3.5 Environment Configuration

AIM development environment on Windows systems requires a few configurations. The below is an example of how an environment may be set up.

Environment variables:
JAVA_HOME=c:\jdk6
BOOST_ROOT=C:\Program Files\boost\boost_1_34_1

3.6 Sample Code

AIMTestLib project contains file *AIMLibTest/AIMLibTest.cpp* that shows examples of how to use AIM Library for generating/reading/writing AIM library objects and files.

REFERENCES

1. Channin, D. S., Mongkolwat, P., Kleper, V., Sepukar, K. Rubin, D. L., The caBIG™ Annotation and Image Markup Project. Journal of Digital Imaging, Vol 23, No. 2, April, 2010.
2. DICOM 2008. Digital Imaging and Communications in Medicine (DICOM) [online] URL: ftp://medical.nema.org/dicom/2010/ Accessed 2010-31-08
3. Rubin, D. L., Mongkolwat, P., Kleper, V., Supekar, K., Channin, D. S., Medical Imaging on the Semantic Web: Annotation and Image Markup, Association for the Advancement of Artificial Intelligence, 2008 Spring Symposium Series, Stanford, CA, 2008
4. Clunie, D. A., DICOM Structured Reporting and Cancer Clinical Trials Results. Cancer Informatics, pp. 33-56, 2007.
5. Clunie, D. A., DICOM Structured Reporting, PixelMed Publishing, 2000.
6. caCORE Software Development Kit 3.2.1 Programmer's Guide, National Cancer Institute, Center for Bioinformatics, 2007-07-16.
7. C++: URL: http://www.cplusplus.com/

