Joint Working Group for a Common Data Model

Trial-Use Standard for Health Care Data Interchange — Information Model Methods

Data Model Framework

P1157.1, Draft 2 April 21,1996

ASTM
American Society for Testing and Materials

DICOM
Working groups of the American College of Radiology and the National Electrical Manufacturers Association

HL7
Health Level Seven

IEEE
Institute of Electrical and Electronics Engineers

NCPDP
National Council for Prescription Drug Programs

X12N
Insurance Subcommittee of ANSI Accredited Standards Committee X12

Secretariat: IEEE P1157 Medical Data Interchange Working Group

[This page has been intentionally left blank]

Introduction and committee list

The Joint Working Group for a Common Data Model (JWG-CDM) is an open standards effort to support the development of a common data model that can be shared by developers of health care informatics standards. The JWG-CDM was originally formed under the auspices of the Message Standards Developers Subcommittee (MSDS) of the ANSI Health Informatics Standards Planning Panel (HISPP). It continues to function as a cooperative effort among individuals who are participants in ASTM, DICOM (ACR-NEMA), HL7, IEEE, X12N and NCPDP. The IEEE P1157 Committee has secretariat responsibility for the activities of the JWG-CDM.

The following individuals have participated in two or more meetings of the JWG-CDM:

Rita Altamore
Soloman Appavu
George Beeler
Dean Bidgood, M.D.
Nicholas Brown
Juan Bustamante
Dave Carlson
Michael DeBellis
Gary Dickinson
Rob Drewniak
Robert Evola
Antonio Fernandez
Arden Forrey
James Fry
L. B. Golter
Jack Harrington
Jim Hoath
John Hoben
Stanley Huff, M.D.
Wayne Johnson
Karen Keeter
W. Ted Klein
Ric Light
Buck Locke
W. L. McMullen
Charles Meyer
Frank Olken
Linda Quade
Lawrence Reis
Virginia Riehl
Wes Rishel
Yzhak Ronen
Mark Shafarman
Abdul-Malik Shakir
Robert Sweet
Wayne Tracy
Mark Tucker
Thomas Vanderpool
Steve Wagner
Mead Walker

In addition, the following individuals worked on the MEDIX Information Model Methods document as part of the IEEE P1157 effort. These prior efforts were essential to the ability of the JWG-CDM to prepare this standard.

Arnold Ableman
Phil Bartleson
Tim Benson
Dave Bradley
Ellen Brox
George DeMoor
Dan Frank
Sigurd From
William Furniss
Mike Glickman
W. Ed Hammond
Gisle Hannemyr
Stephen Kay
Nils Halvard Lunde
Clement McDonald
Sean McLinden
Brian Molteno
Chuck Oleson
Kenneth Olsen
Helmut Orthner
David Ostler
Wesley Rishel
Tom Rutt
Samuel Schultz
Michael Shabot
Andrew Spector
Peter Spitzer
Mike Stern
Woody Trautman
Douglas Tucker
Chris White
Quinn Whiting-O'Keefe

Addresses for comments

This draft of the standard is being prepared for ballot. The JWG-CDM welcomes critical comment on the draft. Comments should be addressed to DRAFT-JWG@MAYO.EDU or may be sent to the document editor listed below.

Comments relating to the general concepts and approach found in the standard should refer to the clause in which the concept or approach is discussed. Detailed critique should refer to the page and line number(s) of the material being critiqued.[i]

NOTE -

Document Editor
Abdul-Malik Shakir
Kaiser Permanente
One Kaiser Plaza
Oakland, CA 94612

Phone: (510)271-6856
FAX : (510)271-6859
Internet: 74353.1431@Compuserve.COM

JWG-CDM Chair
George Beeler
Mayo Clinic
200 SW First St.
Rochester, MN, 55905, USA

Phone: (507)284-9129
FAX : (507)284-0796
Internet BEELER@MAYO.EDU

[i]Note that line numbering on a reader's copy of the document may differ from the numbering in the JWG-CDM's master copy if the reader's copy has been re-printed from a source file. Line numbers on paper copies from the JWG-CDM and on copies printed from PostScript files provided by the JWG-CDM should be consistent with the master copy.

Contents

[This page has been intentionally left blank]

1 Overview

1.1 Scope

This standard specifies the recommended approach for modeling the health care environment in an object-oriented paradigm. It is intended for use by organizations developing standards for information interchange between health care information systems. This standard specifies the graphical and textual notations used in defining the information model, procedures for registration of globally unique identifiers for the classes in these models, and procedures for defining conformance and compliance. This standard also specifies procedures for evolving the information model and for defining local specializations.

NOTE - The current draft of this standard does not contain either "procedures for registration of globally unique identifiers" or "procedures for defining local specializations." The JWG-CDM recognizes the value of developing these procedures and intends to pursue these as the standard moves through subsequent revisions and trial-use. The concerns for globally unique identifiers is to permit the identification of classes in different models and standards, not the instances of those classes.

1.2 Purpose

This standard defines notations and recommends procedures for development of standardized components of an overall object-oriented information model to be used in health care data interchange. The use of consistent notation and procedures facilitates modular development of the overall model and facilitates the development of a family of standards to provide for health care data interchange.

1.3 Reader's guide

This document is organized into eight clauses and six annexes. Clauses 5, 6, and 7 and annex A are normative while the remaining clauses and annexes are informative. This clause provides scope, purpose, and background for this standard. Clause 2: "References" contains references to related standards and references. Clause 3: "Definitions / glossary" provides definitions for the terms and abbreviations used in this standard. Clause 4: "Development and uses of Common Data Model" details the objective, development strategy, and development processes for the CDM. Clause 5: "Data model components" introduces and defines the components used to develop data models for health care informatics standards. Clause 6: "Expression of model content" describes the various ways the data model can be expressed. Clause 7: "Conformance to standard" describes three ways to consider conformance in the context of standards relating to a CDM. Clause 8: "Bibliography" includes a bibliography of data modeling and data representation references. The annexes contain literary and graphical expression for the meta model of this standard, a sample data model, recommended model development steps, a discussion on modeling tools, and a an approach to modeling application profiles, events and messages for messaging standards..

This standard is expected to be used by a diverse set of readers. The first time through, most readers will want to read the clauses in order. When the document is used as a reference, some clauses will likely be ignored and others will be referred to frequently. Because of these various needs the clauses are written to be read independently. Some information is repeated from clause to clause to account for the ad hoc nature in which the document will be used.

Those already familiar with the formation of the JWG-CDM can start with clause 4 to develop an understanding of the primary objective, strategy, and processes related to CDM development. Clause 5 "Data model components" is where the data model components are introduced. The components are described with short phrases which may be more easily understood if read together with the literary

expression specification in clause 6 and the sample data model in annex C. Once the reader has become familiar with the data modeling concepts, clause 6 "Expressions of model content" and annex C "Sample data model" will be the sections used most. Annex A "Meta model" is an easy-to-use quick reference to the data model components once the reader is familiar with the methodology and forms of expression

Readers who are interested in modeling additional concepts including the actors, use cases and interactions related to a data model can find recommended practices for such modeling in annex B.

This standard is not intended to provide a full discourse on object modeling. The reader is encouraged to consult the references listed in clause 8 for a more complete understanding of object-oriented analysis, design, and programming..

1.4 Background

1.4.1 Formation of the Joint Working Group for a Common Data Model

In 1991 there were at least six organizations in the U.S. developing health care messaging standards, of which three were accredited by the American National Standards Institute (ANSI). During that year, the European standards agencies asked ANSI to clarify with whom they could coordinate health informatics standards. As a result, ANSI formed the Health Informatics Standards Planning Panel (HISPP) to coordinate the development of health informatics standards. HISPP's membership included system vendors, professional organizations, standards developing organizations (SDOs), and various users of standards. HISPP remained in existence to the end of 1995 when it was replaced with an ANSI designated Health Informatics Standards Board.

In turn, HISPP formed a subcommittee of its members who were standards developing organizations. This was the Message Standards Developers Subcommittee (MSDS). The members of MSDS were SDOs developing health care message interchange standards. The objective of the MSDS was to achieve harmonization of the standards that they develop. In June 1992, MSDS established the following strategy:

a) Continue to support existing standards

b) Separate the consideration of message content from the details of interchange format

c) Divide the responsibility for message content among joint working groups (JWGs) for which the various standards groups have secretariat responsibilities

d) Provide mechanisms for balloting and reviewing all of the content definitions by the members of all of the standards bodies

e) Provide for coordination of a common data model

The IEEE was asked to provide the secretariat for the Joint Working Group for a Common Data Model (JWG-CDM). The JWG-CDM was to develop the techniques and tools by which such harmonization could be furthered. This work was taken on by the IEEE P1157 Medical Data Interchange Committee. The active participants of the JWG-CDM have included members of IEEE, HL7, ASC X12N, ASTM, DICOM, CEN, and NCPDP.

1.4.2 Requirements to achieve the standards harmonization objectives

In its first meetings, the JWG-CDM identified four basic requirements to achieve the harmonization objectives. They are as follows:

a) A framework document that details how data models to be shared should be represented in modeling tools and in a common syntax. This standard is that framework.

b) A high-level object model of the health care domain to serve as the basis for setting subject area assignments and managing the development of a Common Data Model.

c) A process to facilitate distributed development of portions of the data model as these portions are delegated to JWGs based in each of the standards bodies.

d) A repository that contains the elements of the Common Data Model and that documents both the evolution and final representation of the models that are developed by the various JWGs.

This standard meets the first of these requirements and recommends an approach to the third.

1.4.3 Responsibilities for harmonized standards development

Various parties have responsibility for portions of any development process that is used to achieve a harmonized set of standards based on a Common Data Model. One approach to such a harmonization process is outlined in Clause 4 of this standard. It proposes the following responsibilities:

a) The meta-model in this standard is only part of the meta-model needed to specify a complete standard. Each of the SDOs should develop a complete meta-model that defines how that SDO will define events and the communications associated with those events. These SDO meta-models should be directly linked to (or should incorporate) the meta-model specified in this standard.

b) The assignment of domains of interest within health care information systems should be agreed between the SDOs in order that there be minimal overlap in terms of event definitions.

c) As the SDOs develop specific transactions, the standard data model (SDM) for these should be provided to the JWG-CDM. The JWG-CDM would have the responsibility to review the proposed SDM for conflict and propose reconciliation of these conflicts.

The scope as well as the contents of the data model will be set by the SDOs themselves. The JWG-CDM's role will be to review the material that the SDOs generate. To the extent that there are gaps and omissions within the scope of SDO models, the JWG-CDM will seek to point this out.

1.5 Data models in standards specification

1.5.1 Background

Over the last six years, the concept of basing standards development on a data model of the application domain has evolved. Development of this approach began in the early meetings of the IEEE P1157 Committee . This led to very similar processes being developed in several U.S. and European standards organizations. In the European standards community, a methodology for mapping messages to the underlying data model has been developed. [R1] These approaches are based on the recognition that, regardless of the method of communication or information interchange, the subject matter for health care is

drawn from a data model of health care and health care processes. It follows that a common data model of the health care domain can be used as the starting point for any health care informatics standard.

1.5.2 Separation of common and standards-specific frameworks

The challenge to the JWG-CDM was to find a method for modeling the data content of the health care domain without imposing model components that relate to the standard-specific tasks of defining implementation details such as data structures, communication trigger events, messages, message containers, etc. The JWG-CDM expects that the technical committees of any standards body will need to develop a framework to describe that body's own approach to defining and developing standards. The art of the JWG-CDM's task is to provide a framework for the Common Data Model that can be readily extended by all of the standards bodies to meet their own standards-specific framework requirements.

In order to meet this objective, the JWG-CDM has continually wrestled with those components that are clearly part of a common frame work for data modeling, those components that are clearly peculiar to a single SDO as it develops standards, and those components that are similar in the frame works of several SDOs, but that may be difficult to harmonize owing to differences in the standards developing processes of the various SDOs.

In its initial efforts, including the first draft of this standard, the JWG-CDM sought to express only those elements that were clearly part of a common frame work for data modeling, and remained silent on the treatment of the other areas. In this draft of the standard, however, the JWG-CDM is providing a "recommended practices" section in annex B. This section provides guidance for modeling actors, use cases and interactions areas that the JWG-CDM believes are similar in the frame works of several SDOs. This section is not normative material, but might be advanced to normative status at a later date.

The relationship between the normative components, the recommended practices and model elements that are not part of this standard can best be understood by reference to a framework of models. Figure 1 provides a rudimentary schematic depicting the relationships between essential components of a Business Model, a Data/Information Model, a Business Case specification, and Message Standards. A business model describes functions in one or more business areas. Functions are activities performed by one or more roles depicted in the business model. The business model provides the business context for the data/information model and the resulting message standards. The business case specification identifies the information sharing interactions of interest between pairs of role/function combinations. The interactions depicted in the business case specification are supported by message types included in message standards. A use case is included in the business case specification to provide background and context for each interaction. The data/information model documents object classes and attributes needed to support message types, interactions, and use cases. The object classes are aggregated into subject areas. The attributes defined in the data/information model are used in the definition of data elements for standard data interchange transactions.

Figure 1 has been adapted to reflect the scope of this standard. The components underlain by the shaded figure along the bottom of figure 1 are those the JWG-CDM believes will be common to all SDOs. These are the focus for the normative portion of this standard. The components underlain by the L-shaped shaded figure in the upper-left of Figure 1 are those that the JWG-CDM expects will be similar, but not identical in two or more SDOs. The recommended practices in annex B addresses these components.

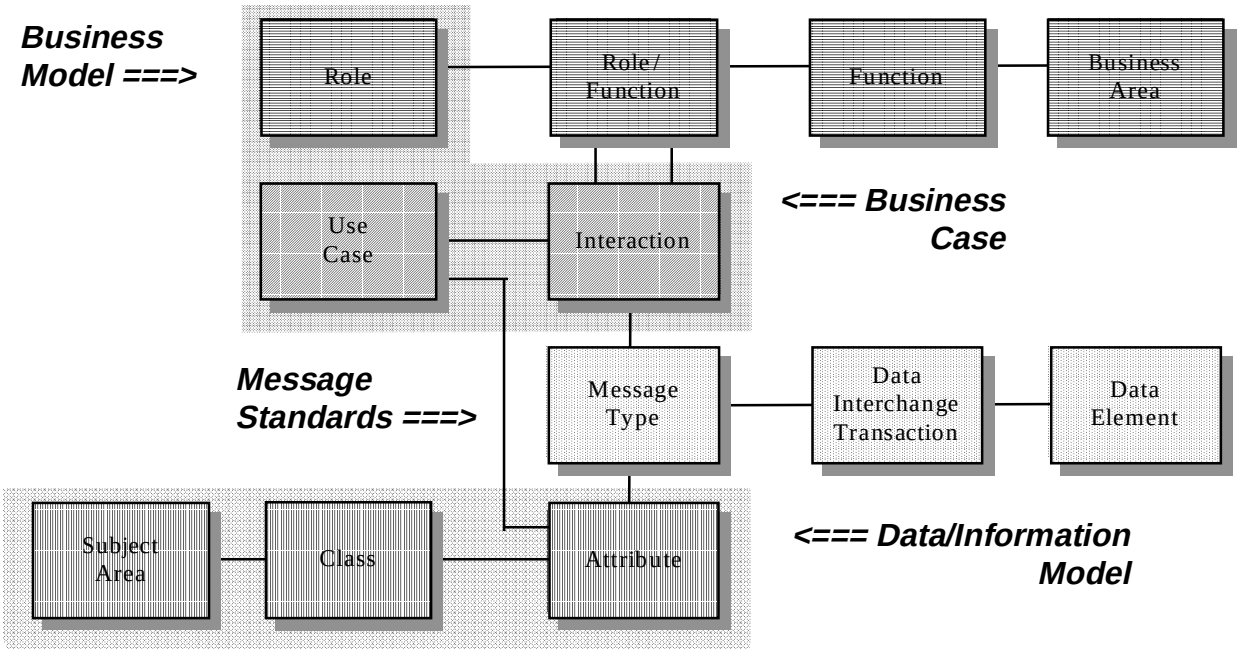

Figure 1. Diagram indicating the major models and specifications that are needed to support a generic messaging standard.

[This page has been intentionally left blank]

2 References

[R1] CEN prCR (out for formal vote) Methodology for the Development of Healthcare Messages

[R2] CEN CR 1350:1993 Investigation of Syntaxes for Existing Interchange Formats to be used in Healthcare

[R3] ISO-8601:1988 Data elements and interchange formats - Information Interchange - Representation of dates

[R4] ASTM E-1238-91 Standard Specification for Transferring Clinical Observations Between Independent Computer Systems

[R5] ASTM E1384-91, Guide for Content and Structure of an Automated Primary Record of Care

Note: ASTM publications are available from the Customer Service Department, American Society for Testing and Materials, 1916 Race Street, Philadelphia, PA 19103, USA.

[This page has been intentionally left blank]

3 Definitions / glossary

3.1 Definition of terms

3.1.1 framework: A structure of processes and specifications designed to support the accomplishment of a specific task. In this case, the task is the development of the CDM, and the framework includes both the specification of how data models should be developed and the processes for reviewing and validating these models between SDOs.

3.1.2 harmonization: A process designed to draw together the various health care informatics standards by ensuring that they are based upon a common specification of the information content they are using. Harmonization of data models is a process in which the contents of multiple models are united into a single, shared model.

3.1.3 instantiate: To create an instance of something. In the world of objects, a Class defines the attributes, relationships, and other characteristics of a class (group) of similar objects that might be created. When a particular occurrence (or member) of that Class is created it is said to be an instance of the class, and it has been instantiated.

3.1.4 joint working group: A group formed to advance standards. JWGs draw their membership from several standards developing organizations. The responsibility to provide the secretariat for a JWG rests with one of the standards developing organizations.

3.1.5 meta-model: A data model that describes a data model. The meta-model is defined in clause 5 and annex A of this standard. Health care data models will be developed using the specifications contained in this standard.

3.1.6 modeler: One who creates or builds models.

3.1.7 optionality: The characteristic of being optional.

3.1.8 secretariat: The administrative group of an organization that has oversight and responsibility for the correspondence, records, etc. of the organization. This includes the responsibility for balloting and publishing the standards produced by a JWG.

3.2 Special terms

A number of special terms appear in this standard. Each is an element of the meta-model that defines the specification of a data model. Each term is fully defined in the context of the standard. The terms are:

CardinalityString
ConditionString
Datatype
DatatypeString
DescriptiveText
IdentifierString (in Annex B)
LengthString
NameString
RealString

3.3 Glossary of abbreviations

ACR/NEMA	The American College of Radiology and the National Electrical Manufacturers Association
AHIMA	The American Health Information Management Association
ANSI	The American National Standards Institute
ASC	Accredited Standards Committee is an organization that is accredited by ANSI to develop and ballot standards
ASN.1	Abstract Syntax Notation 1, a standard language for specifying syntax
ASTM	The American Society for Testing and Materials
CDM	Common Data Model
CEN	European Committee for Standardization, the committee that oversees standards for the European Economic Community
DICOM	The standards developing working groups of the American College of Radiology and the National Electrical Manufacturers Association
FDL	Formal description language
HISB	The ANSI Health Informatics Standards Board
HISPP	The ANSI Health Informatics Standards Planning Panel
HL7	The Health Level Seven standards organization
IEEE	The Institute of Electrical and Electronics Engineers
JWG	Joint Working Group is the generic designation given to the groups that draw their membership from the SDOs in order to accomplish a particular task.
JWG-CDM	The Joint Working Group for a Common Data Model
MEDIX	An arbitrary acronym adopted for the IEEE P1157 Medical Data Interchange Working Group
MSDS	The Message Standards Developers Subcommittee of HISPP
NCPDP	The National Council for Prescription Drug Programs
OMG	The Object Management Group
OOA	Object-oriented analysis
SDM	Subset data model is a data model developed within an SDO and considered for harmonization with the CDM.
SDO	Standards Developing Organization, particularly those that are members of HISB.
X12N	The insurance subcommittee of the ASC X12 that develops electronic data interchange standards for the insurance business, including health care financial transactions

4 Development and uses of Common Data Model

4.1 Primary objective

The overall objective of this standard and of achieving a common data model is to promote harmonization of health care information standards by having those standards draw their data content from a common, shared information model. The approach that has been adopted separates the consideration of message content from the details of interchange format. The responsibility for message content is divided among standards development organizations (SDOs) and joint working groups (JWG) with secretariat responsibility.

In order to harmonize the data content of standards developed by independent SDOs and JWGs the data content must be expressed in a manner that facilitates comparison. A data model provides a structured method for detailing the data content including explicit specification of data definitions and statement of the semantic links. Data modeling therefore is the method chosen for specification of the data content of health informatics standards for the purpose of harmonization across the SDOs.

If this approach is to succeed the following steps must be undertaken:

a) A specification must be developed that lays out a common syntax for the expressions of data models for health informatics standards. This standard is that specification.

b) A process to facilitate distributed development of portions of the CDM among SDOs and JWGs must also be developed. The remainder of this clause describes such a process.

c) The SDOs must incorporate the task of data modeling into their own standards development process, and the resulting data models must conform to this standard. These tasks are the responsibility of each of the participating SDOs, and are not discussed here.

4.2 Model development strategy

A model development process is discussed later in this clause. This is a process in which subset data models (SDMs) are received from an SDO, are "harmonized" by the JWG-CDM, and recommendations for harmonization are returned to the SDO. The harmonized model components are also incorporated into a Common Data Model (CDM) for use in iterative harmonization cycles. This is an idealized process that cannot be fully implemented immediately. A fully harmonized model is not feasible until the CDM has incorporated sufficient content to be an effective influence on the harmonization process. Because of the dissimilarities in the data content of existing standards and because of the need to keep the data model used in each SDO compatible with the existing standards, several iterations of the harmonization process may be required before the effects of data content harmonization are realized in the standards. Figure 2 illustrates the evolutionary strategy that the JWG-CDM believes will lead to a harmonized CDM.

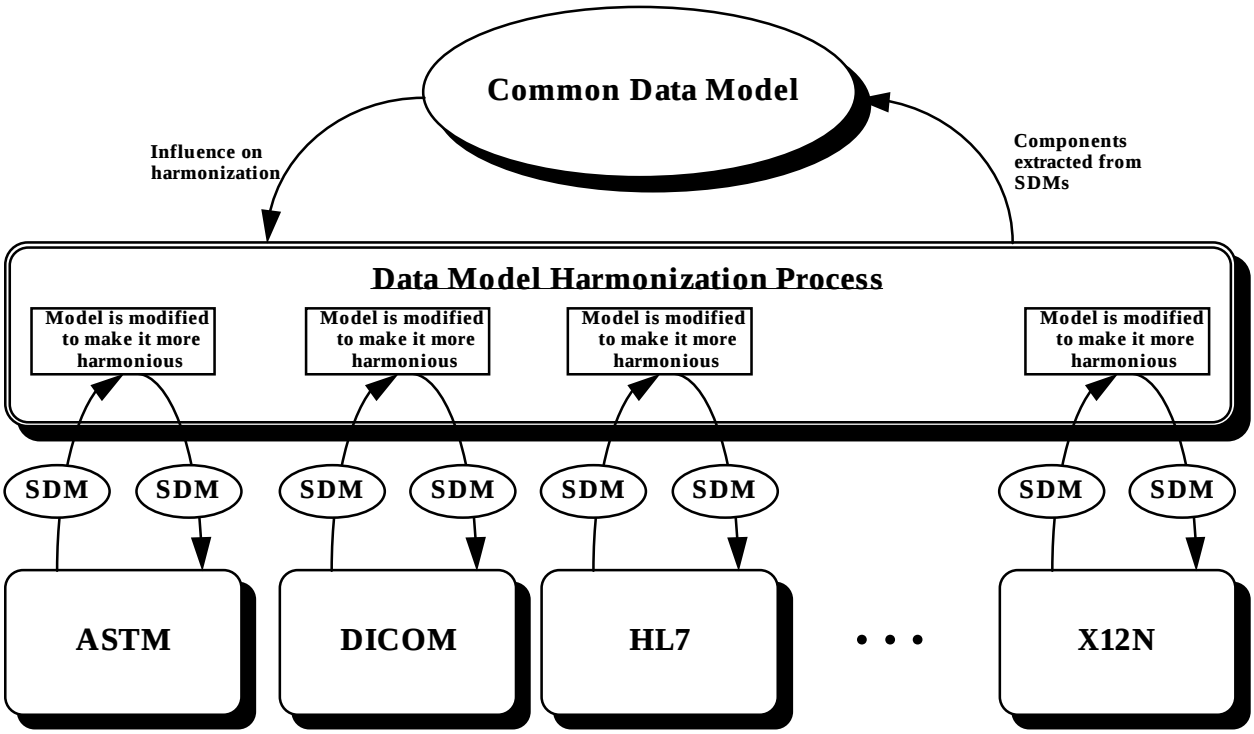

Figure 2 - Diagram of model development strategy.

The foundation of figure 2 is made up of the SDOs, each of which has its own independent subset data model (SDM) that has been developed either implicitly or explicitly as the standards of that SDO were created. The harmonization strategy is based on the fact that each of these data models undergoes many revisions. During each revision, the SDO's data model will be provided to the JWG-CDM which will review the model and suggest changes to reconcile that model with other data models. These changes might call for addition or modification of attributes, classes, class structures, and class connections.

With each iteration of the harmonization cycle the SDO's SDM moves closer to consistency with the SDMs of the other SDOs and with the CDM. At the same time, the JWG-CDM will be incorporating the harmonized elements of the individual SDMs into the CDM. The CDM is not just the aggregation of submitted SDMs; rather, it represents a synthesis of the best and most useful elements of the existing SDMs. Simultaneously, the JWG-CDM will make the CDM available to each of the SDOs for their use in guiding and managing their own data model development.

4.2.1 Role of standards development organizations

Within this strategy, the SDOs are the primary source of data models and of the domain experts whose knowledge is drawn upon to build these models. Although individual members of the JWG-CDM who are also members of the SDOs can act as modeling resources and can facilitate the development and coordination of models, the principle effort will fall to the SDOs. As these groups develop subset data models (SDMs), they will provide working and final drafts of these models to the JWG-CDM for review relative to the CDM and for harmonization with other SDMs. In turn, the SDO receives the recommendations for harmonization from the JWG-CDM.

4.2.2 Role of JWG-CDM

The primary tasks of the JWG-CDM in this strategy will be to maintain an overarching perspective of the data requirements of each of the SDOs and of the data models that each SDO is using. This will only be possible if the individuals who serve on the JWG-CDM are actively involved in the data modeling activities of their own SDOs.

The JWG-CDM will bring this knowledge and perspective to the task of providing prospective review of data models (SDMs) as they are developed by the SDOs. This will include data models from international health informatics standards development organizations, such as CEN. During its review, the JWG-CDM will identify inconsistencies between submitted SDMs and the CDM, and make changes to the CDM as indicated and recommend reconciliatory changes for the SDO to incorporate.

At the same time, the JWG-CDM will use the submitted SDMs to synthesize the CDM which represents a fully reconciled combination of ideas and model segments from the SDMs. Thus, the breadth and depth of the CDM will be determined primarily by the submission and analysis of the SDO-level models. However, the JWG-CDM might accept responsibility to develop new areas of the CDM if one of the JWGs, or an SDO wishes to delegate that responsibility.

4.2.3 Initial development of CDM

The JWG-CDM has also designed a strategy to expedite the initial development of a version of the CDM that is consistent with currently existing standards to the greatest extent possible. The strategy involves the following:

a) The JWG-CDM will build the initial version of the Common Data Model (CDM) by starting with data model expressions of the data content of the current SDO standards. These data models will come from three sources:

   1) Data models developed by the SDOs as a byproduct of their standards development activity.

   2) Data models developed by the SDO by transforming their current specification of data content into a data model consistent with this framework.

   3) Data models developed by the JWG-CDM by transforming the current specification of the data content of an SDO developed standard. This data model would be reviewed and approved by the SDO prior to use in the harmonization effort.

   The transformation will be carried out Subject area by Subject area using this framework, the High Level Data Model (Annex F), and the evolving CDM as a guide.

b) The JWG-CDM will identify inconsistencies between submitted SDO-level models, will document reconciliation proposals in the common data model, and will recommend changes to the SDO to reconcile remaining inconsistencies.

c) The JWG-CDM will report these inconsistencies along with recommendations for reconciliation to each SDO. These suggestions can then be incorporated in subsequent modeling efforts undertaken by each SDO.

d) Once the initial version of the CDM is done, the JWG-CDM will provide it to each of the SDOs to serve as a resource for their modeling efforts.

4.3 Model development process

At the core of these strategies is a model development process carried out in a cyclical fashion by each of the SDOs and the JWG-CDM. This process of model development, review, and harmonization is shown in figure 3 along with the critical information resources necessary to carry out the process:

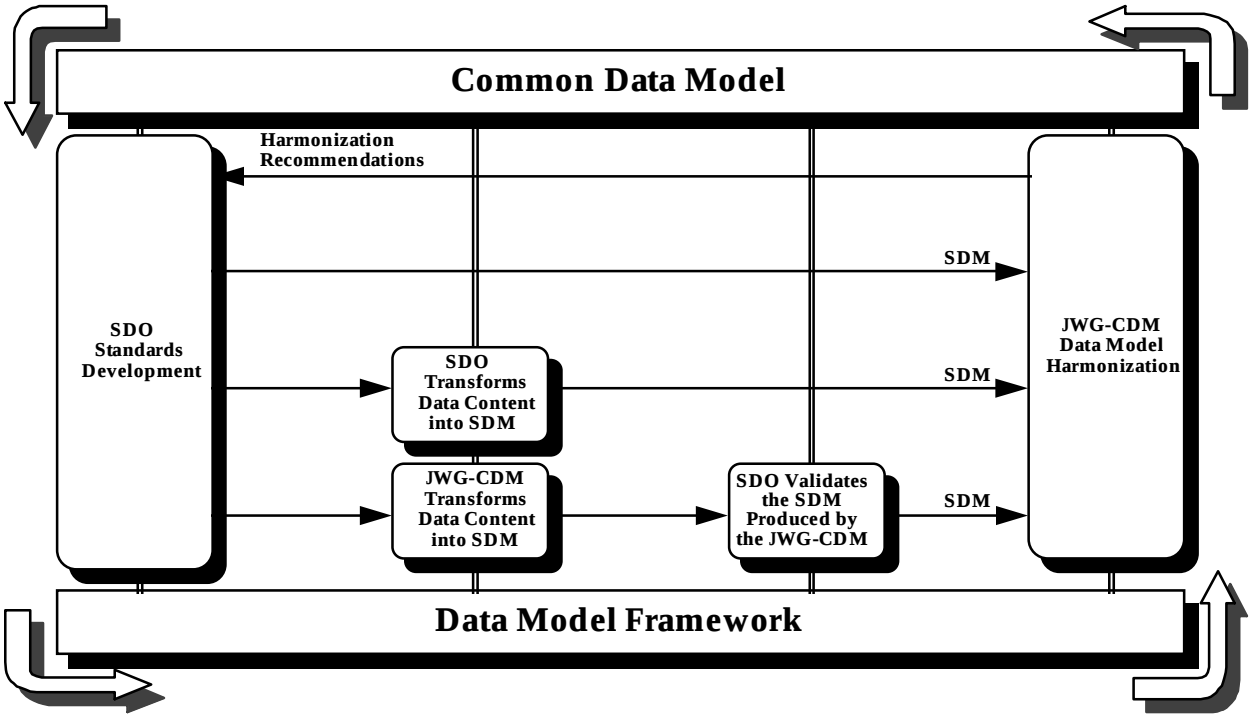

Figure 3 Diagram of model development processes

Figure 3 represents models; processes that create, affect, or modify models; information resources to support those processes; and flows of information. The process flows from SDO standards development activities, through transformation options, to harmonization, and then back to standards development. The Data Model Framework (this standard) is used in the SDO standards development effort as a guide to data modeling to ensure consistency in data modeling approaches and expressions used by the SDOs. The framework is also used during transformation and harmonization. The Common Data Model is updated as part of the harmonization process. It is also used as input during standards development and transformation.

The elements of this diagram are:

4.3.1 Standards development

The processes used by the SDOs to create standards. One of the products of that process is the data content specification for the standard. It is envisioned that SDOs will begin to use data models as one expression of their standards' data content. The Common Data Model and the Data Model Framework are used to assist the data modeling efforts. Harmonization recommendations from the JWG-CDM are used to help bring uniformity among the data content of the standards developed by the SDOs.

4.3.2 Transform Data Content into SDM

The initial cycle of subset data models (SDM) are to be developed using a transformation process. This process uses as its input the data content specification of the SDO-developed standard and uses this standard and the CDM to transform that specification into a consistent syntax and structure. The transformation process can be accomplished by the SDO or by the JWG-CDM. SDO sponsored transformations are preferred. SDMs resulting from transformations done by the JWG-CDM will be validated by the SDO prior to use in harmonization. Once data models become a natural byproduct of the standards development process, transformations will no longer be needed.

4.3.3 Subset Data Model (SDM)

The SDM is an expression of the data content of an SDO standard. The SDMs are subsets of the CDM. SDMs are submitted for harmonization either directly from the standards development effort or following a transformation process. SDMs may be developed by SDOs or joint working groups of SDOs and submitted to the JWG-CDM for harmonization.

4.3.4 Harmonization

Harmonization is a process carried out by the JWG-CDM . It takes as source information the existing CDM and one or more SDMs. The process is one of identifying overlaps, conflicts, new concepts, linkages, etc. that must be resolved or included in order to incorporate the contents of the SDM in the CDM. Harmonization will lead to changes in the CDM, and to recommendations for change being sent to the SDO that provided the content and to other SDOs whose content was already in the CDM.

4.3.5 Harmonization recommendations

These are recommendations developed by the JWG-CDM in the harmonization process. They are sent directly to the SDO for which the change is being recommended. If the SDM being harmonized is one that was transformed, the recommendations going back to the SDO will be transformed back into the terms of the content specification of the standard being modeled in order to be directly useful to the SDO.

4.3.6 Common Data Model

A product of the JWG-CDM and these processes. It is the representation of the information content of each of the standards that have been processed for harmonization. It can be drawn upon directly by any of the SDOs (and by others) for the creation of new standards and the modification of existing standards. Although its influence on the Standards Development process is indirect, the long-term goal is for each SDO to use the Common Data Model directly in the process of developing new standards.

4.3.7 Framework

This is the evolving specification contained in this standard, which states how the data content of SDO developed standards are to be represented in the form a data model. It provides the “rules” for the Transformation process. As the this standard evolves, particularly in the early stages, Transformation will be a critical source of feedback for changes to be made.

[This page has been intentionally left blank]

5 Data model components

This clause describes each component of a data model. The components are presented under three main headings. The following table lists the title for each subclause as a guide to finding a particular description.

Each component is presented with a short description and a list of information kept about that component. This clause describes the items needed to express the data content of health care informatics standards in a data model.

5.1 Model identification and scope

The following set of data model components provides information needed to identify the model and to determine the scope of the model content. The model object describes the model, the subject area identifies the major information categories in the model, and the scenarios provide text to explain the intent of the data modeler and the business case justification for model content.

5.1.1 Model

A model is a collection of subject areas, scenarios, classes, attributes, and services depicting the data needed in some aspect of a health care informatics standard. A model has several representations (graphical, textural, etc.). Information kept about a model includes:

5.1.1.1 Name

A descriptive title for the data model. Name may include qualifiers like version numbers, preliminary, draft, final, standard, and unauthorized. The name shall be unique within the set of models developed by any particular model developing organization.

5.1.1.2 Description

A short narrative describing the scope and intent of the data model.

5.1.1.3 Developing organization name

A short form name for the organization responsible for the publication and maintenance of the data model. This is usually formed by using the initials, abbreviation, or acronym associated with the organization.

5.1.1.4 Developing committee name

The name of the committee within the developing organization that actually undertakes the development of the data model (e.g. ADT Committee).

5.1.1.5 Version number

A number showing the release level of the data model. The version number shall be unique for all public releases of the model. Some organizations may wish to have this number correspond with the version designation of their standard.

5.1.1.6 Last modified date

The date the data model was last modified by the model developing organization. Last modified date is primarily a development organization internal control mechanism to distinguish different iterations of the model between public releases.

Model is the highest level qualifier used to decide uniqueness for model components. The combination of Model Name, Model Developing Organization Identifier, Version Number, and Last Modified Date must be unique.

5.1.1.7 Model Identifier

A unique identifier assigned to the model by the developing organization.

5.1.2 Scenario

A scenario is a statement of health care relevant events supported by the data model, related business models, and information interchange models. Scenarios provide a mechanism for explaining and justifying data modeling decisions that might not be otherwise obvious. Scenarios can be developed at varying levels of detail but should be sufficient to help in identifying required subject areas, classes, attributes, and services. One method for documenting scenarios is the Use Case. Use cases are described in annex B. The purpose of scenarios is to add clarity to the model and assist in understanding the model content. Scenarios

may be used to provide explanation for any portion of the data model. Information kept about a scenario includes:

5.1.2.1 Identification number

A unique number assigned to the scenario to simplify references to the scenario.

5.1.2.2 Title

A short phrase that provides a descriptive title for the scenario.

5.1.2.3 Content

The text, diagrams, and tables that constitute the scenario and provide the details necessary to assist in construction and interpretation of the data model. Use Cases can be used as documentation of scenarios. Refer to annex B for a recommendation on use cases.

5.1.2.4 Reference to parent model

A reference to the model this scenario is a part of.

5.1.2.5 List of class references

An optional reference to one or more model classes cited in the scenario. All classes used in the scenario should be referenced.

5.1.3 Subject area

A major category of information represented in the data model. An aggregation of interrelated classes. A subject area allows portions of a large model to be viewed as a whole thereby eliminating some complexity involved in understanding a large model. Subject areas contain classes and may be the primary subject area for a subset of the classes it contains. Subject area can overlap or be nested in other subject areas. Information kept about subject areas includes:

5.1.3.1 Name

The name given to the subject area. Generally the plural form of a noun. A subject area name is often the plural form of the name of the central or dominant class within the subject area.

5.1.3.2 Description

Short informative text describing the subject area to clarify what type of classes it includes.

5.1.3.3 Reference to parent subject area

A reference to a subject area that is the parent subject area for this nested subject area.

5.1.3.4 Reference to parent model

A reference to the model this subject area is a part of.

5.2 Classes, connections, attributes, and services

Classes, connections, attributes, and services are the primary building blocks of the data model. Classes provide abstractions of the objects represented by the model. The semantic relationships between classes are expressed using connections. The three types of connections are Generalization-specialization, Whole-part, and Instance. Attributes are the facts applicable to the objects of the class, and services describe the actions which affect the attributes of the class.

5.2.1 Class

An abstraction of a set of real-world things (objects) such that all of the objects have the same characteristics and all instances are subject to and conform to the same rules. Classes are the people, places, roles, things, and events about which information is kept. The term *Class* is used in this framework to represent both classes and objects. Classes which cannot be instantiated as objects are referred to as abstract classes. Information documented about a class includes:

5.2.1.1 Name

The classes in the data model are each given a unique name. The class name is a singular noun or noun phrase.

5.2.1.2 Description

A short informative statement that allows people to determine, with certainty, whether a particular real world thing is an instance of the class as conceptualized in the data model.

5.2.1.3 Reference to parent model

A reference to the model in which this class is formally defined and maintained.

5.2.1.4 Reference to primary subject area

An optional reference to the subject area that is the primary subject area for the class. A class can appear in many subject areas, this allows one subject area must be designated as primary.

5.2.1.5 Abstract class indicator

A indicator that specifies if the class is an abstract class. An abstract class is a class that can not be instantiated and is customarily the generalization class in a generalization/specialization structure.

5.2.2 Generalization-specialization connection

Generalization-specialization is an association between a class and a specialization of the class. A generalization class can be associated with more than one specialization class. Each specialization class is mutually exclusive. A specialization class may be associated with more than one generalization. The hierarchy or lattice of generalizations and specializations is called a generalization-specialization structure. The specialization inherits the attributes, connections, and services of all of the generalization classes with which it is associated. Information kept about the generalization-specialization connection includes:

5.2.2.1 Description

A short informative description of the generalization-specification connection.

5.2.2.2 Reference to generalization class

A reference to the generalization class of this connection.

5.2.2.3 Reference to specialization class

A reference to the specialization class of this connection.

5.2.3 Whole-part connection

An association between classes that depicts the relationship between a whole class and its component parts. Information kept about the whole-part structure includes:

5.2.3.1 Description

A short informative description of the whole-part connection.

5.2.3.2 Whole to part reference phrase

A short phrase representing the nature of the whole-part connection from the perspective of the whole class to the part class. Example phrases are: "include", "contains", "consist of", "has as parts". Other phrases may also be used. If no phrase is specified, the phrase "contains" will be assumed.

5.2.3.3 Part to whole reference phrase

A short phrase representing the nature of the whole-part connection from the perspective of the part class to the whole class. Example phrases are: "is included in", "is contained in", "is part of", and "is a component of". Other phrases may also be used. If no phrase is specified, "is part of" will be assumed.

5.2.3.4 Reference to whole class

A reference to the class that forms the whole of the whole-part connection.

5.2.3.5 Reference to part class

A reference to the class that is the part class in the whole-part association.

5.2.3.6 Part class cardinality

A set of values and value ranges indicating the number of part class instances involved in the whole class. In value ranges the minimum shall be zero or more and the maximum shall be equal to or greater than the minimum. The maximum number may be expressed as unlimited.

5.2.3.7 Whole class cardinality

The whole class cardinality is always a minimum and a maximum of one and does not have to be specified.

5.2.4 Instance connection

An association between classes that depicts the occurrence of a reference attribute used to connect class instances (objects). The class containing the reference is known as the "source class", the class referred to is the "destination class". Objects can be of the same or different classes. Information kept about an instance connection includes:

5.2.4.1 Name

A short action phrase that specifies the nature of the connection from the source class perspective. Each instance connection between the same pair of classes must have a unique name.

5.2.4.2 Inverse name

A short action phrase that specifies the nature of the connection from the destination class perspective. Each instance connection between the same pair of classes must have a unique name.

5.2.4.3 Description

A short informative statement that describes the relationship between the classes connected by the instance connection.

5.2.4.4 Source class cardinality

A set of values and value ranges indicating the number of source class instances involved in the connection. In value ranges the minimum shall be zero or more and the maximum shall be equal to or greater than the minimum. The maximum number may be expressed as unlimited.

5.2.4.5 Destination class cardinality

A set of values and value ranges indicating the number of destination class instances involved in the connection. In value ranges the minimum shall be zero or more and the maximum shall be equal to or greater than the minimum. The maximum number may be expressed as unlimited.

5.2.4.6 Reference to source class

A reference to the class from which the instance connection perspective is captured.

5.2.4.7 Reference to destination class

A reference to the class that is the target of the instance connection.

5.2.5 Attribute

Attributes in the data model are the major source of the data content used in health care information standards. Attributes are abstractions of the data captured about classes. Attributes capture separate aspects of the class and take their values independent of one another. Attribute domain specifications are captured in datatypes. Information documented about an attribute includes:

5.2.5.1 Name

Singular nouns are used for attribute names. Attribute names are unique within the class they describe and within the set of attributes inherited by the class they describe.

5.2.5.2 Description

A short informative description of the class characteristic captured by the attribute.

5.2.5.3 Reference to parent class

A reference to the class the attribute is a part of.

5.2.5.4 Reference to datatype

A reference to the datatype that describes the structure, format, and permitted values of the attribute. This reference includes a specification of the attribute's maximum length and a declaration that the attribute as optional or required.

5.2.6 Unique attribute set

A collection of attributes whose combinations of values must be unique among the class instances. A unique attribute set is also known as a candidate key. Unique attribute sets are not required for each or any of the classes in the model, but may be used where the modeler feels they add value. Information kept about a unique attribute set includes:

5.2.6.1 Attribute set identification number

An identifier for the unique attribute set to simplify references to the set.

5.2.6.2 List of attribute references

A reference to one or more attributes that comprise the unique attribute set. All attributes included in this list must be attributes of the parent class of the unique attribute set, or an attribute that has been inherited by the parent class.

5.2.6.3 Reference to parent class

A reference to the parent class the unique attribute set is a part of.

5.2.7 State

The identification of a unique combination of attribute value(s) and connections which are of interest about a class. The state is determined by the values of a specified subset of the attributes and connections of the class. State is not required for each or any of the classes in the model, but should be defined whenever the state affects permitted values of attributes or connections of the class. It may also be useful to document valid state transitions. Information kept about states includes:

5.2.7.1 Name

A word or short verbal phrase that depicts a stable state of a class.

5.2.7.2 Description

A short informative description of the state.

5.2.7.3 State predicate

The condition or set of conditions that when true about the attribute(s) and connections of the parent class places the class into the declared state.

5.2.7.4 Reference to parent class

A reference to the class this state is a part of.

5.2.8 State attribute condition

A specification of the allowable attribute values and connections for a specific state. Although all attributes can at sometime have the value of "none" or "unknown," there are certain states in which the concept of "none" or "unknown" would not be acceptable. Indeed, even among the set of permitted values for an attribute there are some states for which certain otherwise valid attribute values are not allowed. The state attribute condition is used to capture these types of state-related attribute value restrictions.

Information kept about a state attribute condition includes:

5.2.8.1 Condition statement

A statement that specifies constraints on the attribute values. The condition may specify a smaller set of permitted values for the attribute than is suggested by the datatype. A condition statement would also specify if values of "none" or "unknown" are allowed for the given attribute for the given state.

5.2.8.2 Reference to parent state

A reference to the state this state attribute condition is a part of.

5.2.8.3 Reference to constrained attribute

A reference to the attribute within the class referenced by the state to which this condition applies.

5.2.9 Service

A service is a specific behavior that a class is responsible for exhibiting. Services are the means by which state transitions occur, although not all services result in state transitions. The specification of services is beyond the scope of the common data model effort. However, it is important that each SDO define the relevant services for classes in the model. These should be defined if the model developing organization has a requirement for them. It is anticipated that services will be a key link between the data model and the other components of an SDO's standards framework. Information kept about a service includes:

5.2.9.1 Name

The name of the service, generally a descriptive verb or verb phrase that communicates the activity the service performs. A service name must be unique within the parent class. The name should make sense when combined with the class Name as the object of the verb or verb phrase.

5.2.9.2 Description

Descriptive text describing the service in terms of preconditions (events, triggers, predicate statements, states, etc.) which must be present to enable the service and the effect the service has on attributes of the class. The service description should describe what the service does (not how it does it, who does it, or what technology is used.) The how, who, and why of a service should be part of a scenario.

5.2.9.3 Reference to parent class

A reference to the class this service is a part of.

NOTE - Subclauses 5.2.9.4, 5.2.9.5, and 5.2.9.6 need only be specified if they are of use to the modeling organization. They are included here only as a guide to the SDOs to consider in their standards development framework.

5.2.9.4 List of affected attribute references

An optional reference to one or more attributes whose values are affected by the service. All attributes referenced must be part of the same class as the service. Affected attributes are those attributes whose values are set or removed by the service.

5.2.9.5 List of affected instance connection references

An optional reference to one or more instance connections that are established, transferred, or disconnected by the service. The parent class of the service must be a class participating in the instance connection.

5.2.9.6 List of accessed attribute references

An optional reference to one or more attributes whose values are assessed by the service. These include identifiers or unique attribute sets of classes that are the target of affected instance connections and attributes that influence process branches in the service. The values of attributes in this list are unaffected by the service. The attributes in this list may be part of any class in the model; they are not restricted to the parent class of the service the way that affected attributes are.

5.3 Datatype specifications

Datatype specifications are used to define the structure, format, and permitted values of attributes. The JWG-CDM has identified five classes of datatypes: Primitive, Enumerated, Compound, Collection, and Constrained. These five types are a beginning set that will be added to as need for additional datatypes are made known. Each of the five datatypes is described in this clause. Datatype specifications constitute a separate model within the data model. They are referenced by the attributes contained in the data model, but the specifications span data models and data modeling organizations. Datatypes are expected to be reused and should be developed with care to allow for maximum reuse capability without sacrificing utility.

5.3.1 Defined datatype

Defined datatype is an abstract class. It is specified here to allow those elements that are common to all defined datatypes to be specified once, instead of repeated in each. A defined datatype is always one of the three types: Enumerated, Compound, or Constrained. Information kept in common to all defined datatypes are:

5.3.1.1 Name

Each defined datatype is assigned a unique name. The name must be unique across all models developed by the modeling organization.

5.3.1.2 Description

A short narrative description of the datatype.

5.3.1.3 Reference to parent model

A reference to the model in which this datatype is formally defined and maintained.

5.3.1.4 Maintaining organization name

The name of the organization that maintains the datatype specification.

5.3.1.5 Last modified date

The date of the last change made to the datatype specification.

5.3.2 Primitive datatype

A primitive datatype is the simplest form of datatype. It declares the base form of an attribute value such as string, integer, real, or Boolean. Information kept about primitive datatypes includes:

5.3.2.1 Basic datatype

A specification of the basic datatype of a primitive datatype. A basic datatype value can be: string, integer, real, or Boolean.

5.3.2.2 Maximum length

The maximum length of the attribute, indicating the maximum number of characters needed in a textual representation of the attribute value. The maximum length can be expressed as unlimited. Maximum length must be greater than zero.

5.3.2.3 Precision

Specifies the number of decimal positions to the right of the decimal point of a real datatype component.

5.3.3 Enumerated datatype

An enumerated datatype is a specialization of a defined datatype. It specifies a set of values and their meaning. The permitted values of an attribute associated with this datatype are limited to this set of values. The individual values and their meanings are specified in the enumerated datatype elements that are a part of this enumerated datatype. Information kept about the enumerated datatypes includes all the information from defined datatype above and:

5.3.3.1 Responsible Organization

The identification of the organization or entity that has maintenance responsibility for the value table.

5.3.3.2 Reference to primitive datatype

A reference to the primitive datatype that describes the basic datatype of the value component of the enumerated datatype.

5.3.4 Enumerated datatype element

The specification of a value and its meaning. Information kept includes:

5.3.4.1 Value

The value being translated.

5.3.4.2 Name

The meaning of the value. A single word or phrase that reflects the meaning associated with the value.

5.3.4.3 Description

A short informative description of the value. This may include explanatory remarks, exceptions, and other comments to clarify the value meaning.

5.3.4.4 Reference to parent enumerated datatype

A reference to the enumerated datatype the enumerated datatype element is a part of.

5.3.5 Compound datatype

A compound datatype is a specialization of a defined datatype. It is a datatype that permits an attribute to be composed of multiple parts each of which has its own set of datatype constraints. Information kept about compound datatypes includes the information specified for defined datatypes and:

5.3.5.1 Maximum Length

The maximum aggregate length for the compound datatype, indicating the maximum number of characters needed in a textual representation of the attribute value. The maximum length may be less than the sum of the components.

5.3.6 Compound datatype component

A part of a compound datatype. Information kept about the compound datatype component includes:

5.3.6.1 Name

A singular noun used to identify the compound datatype component.

5.3.6.2 Preferred order

A positive integer that indicates the preferred order of presentation for this component.

5.3.6.3 Description

A short informative description of the compound datatype component.

5.3.6.4 Reference to datatype

A reference to the datatype that specifies the structure, format, and permitted values of the compound datatype component.

5.3.7 Collection datatype

A datatype specification that defines the cardinality and form of repeating attributes. Information kept about a collection datatype includes:

5.3.7.1 Number of occurrences

The maximum number of elements in the collection. The maximum occurrence may be specified as unlimited.

5.3.7.2 Collection Type

A specification of the type of collection. At this time the list type is the only type defined in this standard. Other candidate types that may be considered for future inclusion in the standard are array, ordered list, set, and bag.

5.3.7.3 Reference to datatype

A reference to the datatype specification for the elements in the collection. This may be a reference to a Defined datatype or to a Primitive datatype, but shall not be a reference to another Collection datatype.

5.3.8 Constrained datatype

A constrained datatype is a specialization of a defined datatype. It is a datatype specification which details specific criteria that values of the attribute must meet in order to be considered valid.

5.3.8.1 Predicate statement

An acceptance rule that states the conditions that an asserted value should have in order to be accepted as a legal value of the attribute.

5.3.8.2 Reference to datatype

A reference to the datatype that specifies the domain the attributes associated with the constrained datatype.

[This page has been intentionally left blank]

6 Expression of model content

This clause provides the rationale, formats, and specifications for expressing the content of data models developed using this framework. It is divided into four subclauses. It starts with the rationale for these expressions and a description of them. This is followed by the specification of formal representations and formats that are used in all of the expression forms. Finally, there is a detailed specification for the Literary expression, and the Graphical expression. Consideration was been given to developing an expression in a Formal Description Language (FDL). This was not undertaken for this version of the standard.

6.1 Types of expression in this standard

As data modelers develop and subsequently make use of their data models, they will need a standard way of expressing the content of those models to serve their own purposes and the purposes of others. Each of these purposes presents a different set of requirements. For this reason, this standard specifies a set of different, mutually compatible expressions to be used.

6.1.1 Literary expression

The literary expression of the data model should be included by data modelers when a model is submitted to other SDOs for discussion and when it is submitted to the JWG-CDM for harmonization with the CDM.

The literary expression is a structured document that contains the full semantics of the data model, which expresses all of the normative content of the model and includes all of the informative, descriptive information necessary for readers to understand the intent of the modelers as the latter developed the content.

The literary expression is designed to be readily maintained with simple word-processing tools. Even though it requires a standard structure, it is designed to be read and interpreted by individuals who understand the general concepts underlying object-oriented models, but who may not wish to submit to the rigor that an FDL would require.

6.1.2 Graphical expression

The graphical expression is a picture of the data model. It provides an alternate way to depict the core content of the model. It provides the viewer with the essential features of each class, the names of the attributes in the class, the connections between classes, the cardinality of those connections, and the inheritance structure within which each class is placed. Modelers submitting models for harmonization by the JWG-CDM and/or use by other SDOs need not provide a graphical expression, but if a graphical expression is used the expression defined in clause 6.4 of this standard shall be used.

6.2 Formal representations and formats

The elements that define the data model will ultimately be parsed and stored in data systems. Also, it is likely that the classes in the data model will need to undergo formal registration in order to be used with some standards. Therefore, it is necessary to restrict certain elements defined in the data model, or used in data model expression, to formal representations. These formal representations and the requirements for their usage follow.

6.2.1 Representation of unique names

6.2.1.1 NameString for unique names

Various elements of the data model are required to be unique throughout the model or within a limited scope within the model, as specified in subsequent clauses. These elements are listed in the following table, which shows the name of the restricted attribute, the meta-model component to which that attribute belongs, the range over which the attribute is required to be unique, and the preferred case for the initial character of the restricted attribute.

Restricted attribute	**Meta-model component**	**Range for uniqueness**	**Initial character**
name	Data_model	Models from any given SDO	Upper-case
name	Subject_area	Data_model	Upper-case
name	Class	Data_model	Upper-case
name	Attribute	Class	lower-case
name	Instance_connection	pairs of connected Classes	lower-case
inverse_name	Instance_connection	pairs of connected Classes	lower-case
name	Service	Class	Upper-case
name	State	Class	Upper-case
name	Datatype	Model	Upper-case

In a data model defined under this standard, the unique value used for the above meta-attributes shall be of datatype NameString. The NameString is a string that contains no embedded spaces and that is built from a limited character set. The NameString may include any number of the following characters: upper or lower case alphabetic characters (A-Z and a-z); the digits (0-9); and the underscore character (_). These characters may be in any order, except that the first character of the NameString shall be an alphabetic character.

The appropriate use of upper and lower case characters, and the inclusion of special characters allow data modelers to create easily readable strings for the noun- and verb-phrases required for many of these elements. (Examples might include "is_ordered_by" or "HealthCarePractitioner" or NameString.) For clarity of reading, the initial character of a NameString should be constrained as follows. The first character of all names in the above table, with the exception of attribute name, should begin with an upper case (capital) letter, as indicated in the right-hand column of the table. Conversely, attribute name and the two phrases for Instance connection should begin with a lower case character. No matter what conventions are used with respect to capitalization, when NameStrings are compared for uniqueness, all alphabetic characters shall be treated as though they are lower case.

The naming and listing of the meta-model elements in this clause of the standard follow these rules and use the preferred capitalization convention presented above.

6.2.1.2 Specification of fully qualified names

Although this standard does not require such references, there may be times in documenting a particular data model when the modeler will need to refer to a particular attribute of a particular class and need to distinguish this from an attribute of a different class that has the same attribute name. In that instance, the qualified name of the attribute shall be expressed by concatenating the NameString names of the class name and attribute name, in that sequence, using the dot character (.) as a separator. In this terminology, the qualified name of the date and time for a lab procedure in the Sample Data Model (Annex C) would be "Lab_Procedure.date_time".

If a modeler needs to distinguish between elements drawn from more than one data model, then the model name must also be included in the qualified name with the element concatenated to it. If a unique model identifier has been assigned to the model, this identifier may be used as the fully qualified name of the model. If a unique identifier has not been assigned, then the fully qualified model name shall be the concatenation in the following sequence of: the Model.developing_organization_identifier; the Model.name; the Model.version_number expressed as "Vn-m", where 'n' is the version number and 'm' is the release number; and the Model.effective_date expressed in the standard Date datatype (CCYYMMDD). The dot character (.) shall be used as a separator in this concatenation.

For example, the version of the Sample Data Model provided in annex C has a fully qualified name of: "IEEE.Sample_Data_Model.V1-0.19940425" and the fully qualified name of the date and time for a lab procedure in that model would be "IEEE.Sample_Data_Model.V1-0.19940425.Lab_Procedure.date_time". If the same model had been assigned the identifier "IEEE_SMP_0001" then the same element could be referred to as "IEEE_SMP_0001.Lab_procedure.date_time".

6.2.2 Representation of descriptive text

In most instances, the information to be kept about model components, as specified in Clause 5, includes provision for a textual description of the component. Initial efforts to document data models using this draft standard have revealed the need to structure these descriptions in order to provide for cross reference to particular standards, identification of open issues, explanation of modeling rationale, etc.

Descriptions of model components shall be of type DescriptiveText, as specified in this sub-clause.

6.2.2.1 Descriptive text components

In order to accommodate these requirements without extending the number of components to be specified, this standard provides for the representation of up to five different components within each description. The basic description component is contained in the first paragraph(s) of the descriptive text. The remaining components, if they are included, are identified by the appearance of a reserved keyword at the beginning of the paragraphs. The following table lists the key words that are used to introduce each component, and the rules governing the content and frequency of occurrence of each component.

Component	Key word	Purpose
Basic description		The basic description component is made up one or more paragraphs which serve the primary purpose of describing the data model component in question. The basic description has no keyword; shall be the first paragraphs in the DescriptiveText; and includes all paragraphs up to the first occurrence of a paragraph that starts with one of the key words specified in this table. Paragraphs that are part of the basic description shall not begin with any of the four reserved keywords specified in this table. The basic description component may not be repeated within a DescriptiveText.
Standards reference	StdRef	The standards reference component provides a vehicle by which individual SDOs may reference data model components to particular elements of their standards. Each standards reference component shall be a single paragraph that begins with the keyword "StdRef." There is no restriction on what may follow this key-word. Thus the SDOs can extend this component with their own reserved words in order to define sub-components. A DescriptiveText may include multiple standards reference components.
Modeling rationale	Rationale:	This component is designed to allow the data modeler to document the rationale or justification for the specification of a particular data model component. This rationale may occupy one or more paragraphs in the DescriptiveText. The first paragraph of the modeling rationale component shall begin with the keyword "Rationale:". Subsequent paragraphs of the modeling rationale component, if any, shall not begin with any of the four reserved keywords specified in this table. Only one modeling rationale component should appear in each DescriptiveText.
Open issue	OpenIssue:	This component of the DescriptiveText is intended to allow the data modelers to identify and discuss any open issues that remain to be resolved with respect to the model component being described. This issue statement may occupy one or more paragraphs in the DescriptiveText. The first paragraph of an open issue component shall begin with the keyword "OpenIssue:". Subsequent paragraphs of each open issue component, if any, shall not begin with any of the four reserved keywords specified in this table. There may be multiple open issue components in a DescriptiveText.
Scenario reference	Scenario:	This component permits the data modeler to link a data model component to a specific Scenario in the same data model. This component shall be a single paragraph that opens with the keyword "Scenario:". Following the keyword, the component shall be made up of a comma-delimited list of the scenario identification numbers for the scenarios in which the element is discussed. There shall be no more than one scenario reference component in each DescriptiveText.

NOTE - The key-word "StdRef" does not include a colon as do the other key-words. This is done because it is expected that each SDO will extend this keyword with its own sets of keywords, and that these extensions will include a colon. An example from HL7 are the reference key phrases "StdRef HL7 Table:" and "StdRef HL7 Segment:"

6.2.2.2 Deprecated usage

It is intended that there be few restrictions on what a data modeler may enter in descriptive fields. In order that the expression of these models can be automatically parsed, however, there are a few restrictions that should be observed in writing descriptions. These restrictions are:

a) The following special characters should not be used in descriptions: left curly bracket ({), right curly bracket (}), and vertical bar (|) .

b) The character combination of a colon surrounded by spaces (' : ') should not be used in descriptions.

c) The reserved words CONDITIONAL, MANDATORY, OPT, OPTIONAL, and REQUIRED expressed in all capital letters should not be used in descriptions.

d) The special 'standard phrases' specified as part of the line patterns in the literary expression should not be used in descriptions.

6.2.3 Representation of cardinality and datatype elements

6.2.3.1 Representation of connection cardinality

A set of values and value ranges including the minimum and maximum occurrence are required for instance connections and whole/part relations in the meta-model. A CardinalityString shall be used to represent this set in both the literary and graphical expressions of the model. The CardinalityString is a constrained string. It is built according to the following rules:

6.2.3.1.1 A CardinalityString shall have at least a minimum and a maximum value.

6.2.3.1.2 A CardinalityString may also include ranges or individual values to define a set.

6.2.3.1.3 A CardinalityString may also include an open ended range at the upper end.

6.2.3.1.4 A CardinalityString shall be expressed as a series of one or more elements enclosed in parentheses.

6.2.3.1.5 The elements in the series making up a CardinalityString shall be separated by a comma (,).

6.2.3.1.6 The elements making up a CardinalityString shall be either a range, zero, a positive integer, or the letters "m" or "n".

6.2.3.1.7 A range element in a CardinalityString shall be represented by a pair of periods (..) separating a lower value that must be an integer and an upper value that may either be an integer or the letter "m" or the letter "n". The range includes both its upper and lower values.

6.2.3.1.8 If the letter "m" (preferred) or "n" appears in a CardinalityString, there must be only a single occurrence, and that occurrence shall represent the set of all positive integers that are greater than the largest of the other integers in the same CardinalityString.

6.2.3.1.9 The minimum value for the cardinality shall be the smallest integer in the CardinalityString.

6.2.3.1.10 The maximum value for the cardinality shall be the largest integer in the CardinalityString, and must be greater than zero.

6.2.3.1.11 The elements making up a CardinalityString should be ordered in ascending order, but are not required to be.

6.2.3.2 Representation of data element length

A representation of the maximum length of the basic datatypes String and Integer, and a representation of the maximum number of elements in a List are required as part of the datatype specification for attributes. The LengthString shall be used in these cases. The LengthString is a constrained String. It is built according to the pattern "(x)". The element, "x", shall be either a positive integer or the character 'm.' This value expresses the maximum length of the component, or the maximum number of elements in a List. If an integer is used, it is the maximum. If the character 'm' is used it represents an indeterminate maximum. If the LengthString is omitted at any place, the default value shall be "(m)".

6.2.3.3 Representation of precision

A representation of precision expressed as the number of digits required to the right of a decimal point is part of the specification of the Real datatype. The RealString shall be used in these cases. The RealString is a constrained String. It is built according to the pattern "(x.s)". The first element ('x') shall be either a positive integer or the character 'm.' This value expresses the maximum length of the number. If an integer is used, it is the maximum. If the character 'm' is used, it represents an indeterminate maximum. The second element in the pattern ('s') shall be a positive integer specifying the minimum number of digits to the right of the decimal point in the real number.

6.2.3.4 DatatypeString entries for attributes

In the literary and graphical expression for data models, it is necessary to specify the datatype for each attribute in the data model. This specification is represented by a constrained String known as a DatatypeString. The rules for constructing this string follow.

6.2.3.4.1 Optionality indicator

The first component in a DatatypeString is a field that indicates whether the DatatypeString is optional, required, or conditional. This component itself is optional. The values that may be assigned to this component are: 'OPTIONAL', 'OPT', and '?' (meaning the component is optional); 'REQUIRED', 'MANDATORY', and 'M' (meaning the component is required); and 'CONDITIONAL' (meaning the optionality is determined by a state of the class to which the attribute belongs).

The presence of this first component in a DatatypeString is optional. If the component is omitted, the default assumption shall be that the attribute or attribute part being specified is required. If this component of the DatatypeString is present, it shall be separated from the remainder of the DatatypeString by a space.

6.2.3.4.2 Datatype name and modifiers

The second component of the DatatypeString shall be the name of a particular datatype including, where appropriate, the LengthString or RealString for that datatype. Allowable datatype names are the names of

any datatypes that are defined within the model; the names of datatypes defined in other models; the names of the Basic types; and the datatype name List. Thus, the datatype name and modifier must come from the following list:

a) Datatype.name :: The name of a datatype defined within the model being expressed, or the fully qualified name of a datatype defined in some external model. In the latter case, the name is the concatenation of the fully qualified model name of the source model and the datatype name, as specified above.

b) A name of a Basic datatype which shall be one of the following:

1) Boolean :: The name of the Boolean Basic datatype.

2) Real (x.s) :: The name of the real Basic datatype combined with the RealString that represents the length and precision of that real number.

3) Integer (x) :: The name of the integer Basic datatype combined with the LengthString that represents the length of the integer. The LengthString may be omitted in which case the default value "(m)" will be assumed.

4) String (x) :: The name of the string Basic datatype combined with the LengthString that represents the length of the string. The LengthString may be omitted in which case the default value "(m)" will be assumed.

c) List (x) of: :: The name of the list datatype including the LengthString that represents the maximum number of occurrences within that list. The LengthString may be omitted in which case the default value "(m)" will be assumed.

If the datatype name is List ('c' in the list), then there shall be a third component of the DatatypeString as specified in the following subclause.

6.2.3.4.3 DatatypeString for Lists

If the datatype name in the second component of the DatatypeString being defined is List, then the DatatypeString shall have a third component. The third component shall consist of a properly defined DatatypeString that specifies the datatype of the entries in the List except that the DatatypeString following a List(x) of: shall have a required optionality. This is a recursive definition. The definition may recur as often as necessary but the specification "List of List of" should not be used.

6.2.4 Literary expression format

The literary expression of a data model is a structured document that presents all of the normative and informative information that is contained in the model. The document is built in outline form and includes four major sections. Each of the sections contains one or more subsections that may, in turn, be subdivided. The maximum depth of this structure is four levels, where level one is the section, level two is the subsection, etc.

This hierarchical structure, the indented format in which it should be documented, and the rules applied to each section are designed to provide a document that can be easily read and within which specific information can be found rapidly. Secondarily, the specifications permit the document to be automatically

generated from the output of graphical modeling tools and to be parsed in order to provide source data for such graphical tools and for tools that can validate and document the model.

6.2.4.1 Indentation

The literary document is intended to be created with simple word processing or text editing programs. The document structure, which allows a reader to find specific information rapidly, is made evident through indentation where each succeeding level is indented to the right of the preceding level. The amount by which each succeeding level is indented should be constant, and the indentation increment should be between 0.3 in and 0.7 in on a printed page. The typeface(s) used in this document shall be 10 (ten) point size or larger.

6.2.4.2 String separators and line spacing

At several places in the definition of both the literary and graphical expression for data models, this standard calls for the concatenation of two or more NameStrings to form a compound expression. When this is called for, any combination of one or more spaces and/or one or more colons (:) may be used as the separator between each pair of NameStrings. The use of two colons surrounded by spaces (::) is preferred as a separator. The Sample Data Model in annex C demonstrates a selection of separators that are designed to enhance readability.

A blank line should be used to separate individual lines and paragraphs in the literary expression.

6.2.4.3 Text attributes

This standard specifies no requirement for typeface, or the use of bold-face and underlined text styles. However, authors are encouraged to use these attributes to enhance the legibility of the literary expression when possible. One example of this usage can be seen in the literary expression for the Sample Data Model contained in annex C.

6.2.5 Literary expression pattern notation

As each section of the literary expression is specified, this standard will specify the level of indentation for that section or subsection and a format by which the lines in that subsection are to be constructed. A pattern is used to convey this information. The remainder of this clause specifies the notation used in the patterns.

The pattern for a line or a set of lines in the expression contains: the indentation level, a standard phrase, and the content to be placed in the line. The pattern for the beginning of a class definition section will be used as an example for the discussion in this clause. The pattern is:

L1 Class: <Class name>

6.2.5.1 Indentation level

Each pattern starts with the letter "L" followed by a single digit. The digit specifies the indentation level, where L0 is placed at the left margin with no indentation, L1 is indented one step, L2 is indented two steps, etc.

6.2.5.2 Standard phrase

A standard phrase follows the indentation level in most, but not all of the patterns. Wherever a pattern is used in the document, the specified standard phrase shall begin the line, except in the circumstance where a pattern is used to specify a sequence of single lines. In the latter case, the standard phrase in the pattern should not be used in the second and subsequent lines of the sequence.

These phrases are shown as underlined text in the pattern. When the phrases are used in expressing a data model they may be underlined, but they are not required to be. The standard phrase for the example pattern above is "Class:".

6.2.5.3 Line content

The third element of a pattern represents the line content that shall appear on that line. Most, but not all patterns call for such values. These values may be single elements as in the class name called for in the example pattern, or they may be compound values built by combining individual values according to simple rules. A descriptive phrase for the line content is placed in the pattern enclosed in a pair of angled brackets (< >). A precise definition for the descriptive phrase is provided in the paragraph that follows the pattern.

6.3 Literary expression specification

The structured elements that make up the literary expression of a data model are specified in this clause. The sequence and listing are very similar to that used in Clause 5. Annex C provides a proper literary expression of a sample data model. Readers will find it helpful to refer to annex C for illustrations of each of the specifications in the subclauses that follow. Further, the following table lists the title for each subclause as a guide to finding a particular specification.

6.3.1 Model identification and scope section

- 6.3.1.1 Model identification
- 6.3.1.2 Scenario definitions
- 6.3.1.3 Subject area definitions

6.3.2 Classes, attributes, and services

- 6.3.2.1 Class summary
- 6.3.2.2 Generalization-specialization
- 6.3.2.3 Whole
- 6.3.2.4 Parts
- 6.3.2.5 Instance connections
- 6.3.2.6 Attributes
- 6.3.2.7 Unique attribute sets
- 6.3.2.8 States
- 6.3.2.9 Services

6.3.3 Datatype specifications

- 6.3.3.1 Enumerated datatype
- 6.3.3.2 Compound datatype
- 6.3.3.3 Constrained datatype

6.3.4 External references

6.3.4.1 Referenced classes
6.3.4.2 Referenced datatypes

6.3.1 Model identification and scope section

The model identification and scope section shall be the first section of a literary expression document. This section provides information about the model including supporting scenarios and subject areas, a description of the model's scope, and the organization responsible for developing the model. The pattern for the first line is:

L0 Data model: <Model name>

The line content shall be name of the model expressed as a NameString.

6.3.1.1 Model identification

The model identification subsection shall be the first subsection under model identification and scope. It provides the name of the model, the organizations responsible for developing the model, the model version, and a description of the model's scope. The pattern for the first line is:

L1 Identifications: <null>

This line has no content. It shall be followed by:

L2 Organization: <Developing organization name>

The line content shall be the name of the standards developing organization responsible for developing the model.

L2 Version: <Version number and last modified date>

The line content is the version and release number of the model followed by the last modified date. The version shall be expressed as "Vn-m", where 'n' is the version number and 'm' is the release number. The last modified date which follows shall be expressed in the standard Date datatype (CCYYMMDD). A space shall separate these values in the content.

L2 ModelID: <Assigned model identifier>

If a specific model identifier has been assigned by the developing organization, or some other group seeking to manage multiple models, this line shall appear. The line content is the assigned model identifier.

L2 Developed by: <Developing Committee name>

The line content shall be a string that contains the name and other identifiers of the committee within the developing organization that developed the model.

L2 Description of: <Model name>

The name of the model shall be expressed as a NameString.

L3 <Model description>

The textual description of the intent and scope of the model shall follow and be expressed as DescriptiveText. This concludes the model identification subsection.

6.3.1.2 Scenario definitions

The scenarios definitions subsection shall be the second subsection in the model identification and scope section. The first line of this subsection shall be:

L1 Scenarios for: <Model name>

The name of the model shall be expressed as a NameString.

Following the opening line there shall be one sub-subsection for each scenario defined in the data model. The scenarios shall be listed in the order of their Identification numbers. Each sub-subsection shall open with:

L2 Scenario: <Scenario Identification_number and Title>

The line content in this pattern shall be the identification number of the scenario followed by its Title. A period and a space should separate the number from the title string.

L3 <Scenario Content>

The scenario content that describes the scenario shall follow in one or more paragraphs. Note that scenarios may be more complex than simple text, and may include diagrams, tables, etc. The scenario content concludes the sub-subsection for each scenario.

NOTE - In order to retain the ability to parse the literary expression, a standard separator will need to be defined to surround scenario content that contains other than simple text. This specification will be added in later versions of the standard.

L3 Class references: <Class name>

This line shall be repeated for each of the classes that is referenced in the scenario. The line content shall be the name of the contained class as a NameString. The lines shall be listed in alphabetic order by the names of the classes in the list.

6.3.1.3 Subject area definitions

The subject area definition subsection shall be the third subsection in the model identification and scope section. The first line of this subsection shall be:

L1 Subject areas for: <Model name>

The name of the model shall be expressed as a NameString.

Following the opening line there shall be one sub-subsection for each subject area defined in the data model. The subject areas shall be listed in alphabetic order by their names. Each sub-subsection shall open with:

L2 Subject area: <Subject_area name>

The name of the subject area shall be expressed as a NameString.

L3 <Subject_area description>

The text that describes the subject area shall follow expressed as DescriptiveText.

L3 Parent area: <Parent_subject_area name>

If a parent subject area is identified for this subject area, this line shall be used with the NameString name of the parent as the line content.

L3 Contains classes: <Class name>

This line shall be repeated for each of the classes that is contained in the class list for this subject area. The line content shall be the name of the contained class as a NameString. The lines shall be listed in alphabetic order by the names of the classes in the list.

6.3.2 Classes, attributes, and services

The class definitions section shall be the second section in the literary expression. It opens with:

L0 Classes in: <Model name>

The name of the model shall be expressed as a NameString.

Following the opening line there shall be one subsection for each class defined in this data model. The subsection for each class shall occur in alphabetic order by the class name. Each subsection shall open with:

L1 Class: <Class name>

The name of the class shall be expressed as a NameString.

Following the class identification line, the subsection for each class is further broken down into sub-subsections. Not all of these sub-subsections will be present for each class in the model. The specification for each of these sub-subsections along with the rules that determine when they are required are given in the next eight subclauses. Throughout these subclauses the phrase "subject class" will be used to refer to the class being defined.

6.3.2.1 Class summary

The sub-subsection for class summary shall be part of each class definition. The first pattern in this sub-subsection is:

L2 Primary model: <Parent model and class name>

This line shall be used only if the subject class has its primary definition in a model other than the one being documented. The line content shall be the fully qualified name of the class as it is known in the

parent model. The class name in the model being documented should be the same as that used in the parent model, but the two names are not required to be the same. Authors may choose to copy the class definition from the parent model in order to present a complete expression of the model, but are not required to do so. All references to classes in other models are also expressed in a separate section.

L2 In Subject area: <Primary_subject_area name>

If the subject class is in one or more subject areas, then one of those subject areas shall be the primary subject area for the subject class, and this line shall be used. It contains the NameString that is the name of the primary subject area for the subject class.

L2 Is Abstract class <null>

If the subject class is an abstract class, this line shall be used. There is no other line content.

L2 Specialization of: <Generalized Class name>

If the subject class is a specialization of another class, this line shall be used. The line content is the name of the parent class expressed as a NameString.

L2 Generalization of: <Specialized class name>

If the subject class is a generalization of one or more specialized classes in a generalization-specialization relationship, this line shall be listed once for each of the specialized classes. The order of listing should be the alphabetic sequence of the specialized class names. The line content shall be the NameString that expresses the specialized class name.

Note - The following three elements of the class summary subsection are indicated as optional. In aggregate, they complete the thumb-nail view of the class being described, and thus their inclusion is encouraged. They are, however, redundant in that the same information content is included in subsequent required elements of the expression. Thus, their omission would not lead to incomplete specification of the model being expressed.

L2 Is part of: <Whole class name>

If the subject class is a part of another class in the model, this line may be used and the line content shall be the NameString that is the name of the other class of which the subject class is a part.

L2 Contains: <Part class name>

If the subject class has one or more part classes in a whole/part relationship, this line may be listed once for each of the part classes. The order of listing shall be the alphabetic sequence of the part class names. The line content shall be the NameString that expresses the part class name.

L2 Connected to: <Target class name>

If the subject class has instance connection(s) to one or more other classes in the model, this line may be listed once for each instance connection. The order of listing should be the alphabetic sequence of the names of the classes at the other (target) end of the instance connection. The line content shall be the NameString that is the name of the class at the other (target) end of the instance connection.

L2 Description of: <Class name>

If a description is provided for the subject class, this line shall be used. The line content is the name of the subject class expressed as a NameString.

L3 <Class description>

If a description is provided for the subject class, the text that describes the subject class shall follow expressed as DescriptiveText.

There are no further elements in the class Summary sub-subsection.

6.3.2.2 Generalization-specialization

If the subject class participates in a generalization-specialization connection with another class, and if there is a description provided for that connection, this sub-subsection shall be placed in the document starting with the following line:

L2 Gen-Spec description for: <Class name>

The line content is the name of the subject class expressed as a NameString. This line shall be followed by:

L3 <Gen-spec description>

The text of the description for the generalization specialization connection shall follow expressed as DescriptiveText.

6.3.2.3 Whole

If the subject class is a part of another class, the whole sub-subsection shall be placed in the document starting with the following line:

L2 Whole/part for: <Class name>

The line content is the name of the subject class expressed as a NameString. This line shall be followed by two more lines to describe the relationship:

L3 <Part to whole reference phrase (1,1) :: Whole class name ::
Whole to part reference phrase and cardinality>

The line content shall be a compound built of five components. The first component shall be the reference phrase from the perspective of the part class, if such a phrase has been defined. Otherwise, the default reference phase "is_part_of" shall be used. Following a space as the separator, the second component shall be the default cardinality for the whole "(1,1)". Following a separator, the third component shall be the NameString that is the name of the class of which the subject class is a part. Following a separator, the fourth component shall be the inverse phrase from the perspective of the whole class, if such has been defined. Otherwise, the default phrase "contains" shall be used. Following a space as the separator, the fifth component shall be the cardinality of the part relationship expressed as a CardinalityString.

L4 <Whole/part description>

If a description is provided for the whole/part relationship, the text of the whole/part description shall follow expressed as DescriptiveText.

6.3.2.4 Parts

If the subject class is a whole (composite) element in a whole/part relationship, the parts sub-subsection shall be placed in the document starting with the following line:

L2 Whole/part for: <Class name>

The line content is the name of the subject class expressed as a NameString. Following this opening line, the parts sub-subsection shall include a pair of entries for each of the classes that are parts of the subject class. The order of listing shall be the alphabetic sequence of the part class names. The pair of entries are specified by the next two line patterns.

L3 <Whole to part reference phrase and cardinality :: Part class name ::
Part to whole reference phrase (1,1)>

The line content shall be a compound built of five components. The first component shall be the reference phrase from the perspective of the whole class, if such a phrase has been defined. Otherwise, the default reference phase "contains" shall be used. Following a space as the separator, the second component shall be the cardinality of the part relationship expressed as a CardinalityString. Following a separator, the third component shall be the NameString that is the name of the class for which the subject class is a whole. Following a separator, the fourth component shall be the inverse phrase from the perspective of the part class, if such has been defined. Otherwise, the default phrase "is_part_of" shall be used. Following a space as the separator, the fifth component shall be the default cardinality for the whole "(1,1)".

L4 <Whole/part description>

If a description is provided for the whole/part relationship specified in the preceding line, the text of the description shall follow expressed as DescriptiveText. The description is the second of the two line patterns that are used for each part in the subject class.

6.3.2.5 Instance connections

If the subject class partakes in one or more instance connections, this sub-section shall appear and shall begin with:

L2 Connections for: <Class name>

The line content is the name of the subject class expressed as a NameString. Following this opening line, the connections sub-subsection shall include a pair of entries for each of the instance connections in which the subject class participates. The order of listing shall be the alphabetic sequence of the target class names. The pair of entries for each connection are specified by the next two line patterns.

L3 <Reference phrase and cardinality :: Target class name :: Inverse phrase and cardinality>

The line content shall be a compound built of five components. The "target class" is the class to which the subject class is connected. The first component shall be the reference phase for the instance connection when read from the subject class to the target class. Following a space as the separator, the second component shall be the cardinality for the target class in this connection expressed as a CardinalityString. Following a separator, the third component shall be the NameString that is the name of the target class. Following a separator, the fourth component shall be the inverse phrase used to describe the returning connection from the target class to the subject class. Following a space as the separator, the fifth component shall be the cardinality of the subject class in this connection expressed as a CardinalityString.

Note that this phrase is designed to be read in a circular fashion starting and ending with the name of the subject class which is contained on the preceding line. This will result in a crude sentence of the form "Subject_class connects_to (1,m) :: Target_class :: connects_back_to (1,1) Subject_class". A similar phrase construct is used in the graphical expression.

L4 <Instance connection description>

If a description is provided for the instance connection, the text of that description shall follow expressed as DescriptiveText. The description is the second of the two line patterns that are used for each instance connection in the subject class.

6.3.2.6 Attributes

If the subject class contains one or more attributes, this sub-subsection shall be used, and it shall start with:

L2 Attributes of: <Class name>

The line content is the name of the subject class expressed as a NameString. Following this opening line, the attributes sub-subsection shall include a pair of entries for each attribute. The order of listing shall be alphabetic sequence of the attribute names. The pair of entries are specified by the next two line patterns.

L3 <name/DatatypeString>

The line content is a compound string that combines the name of the attribute and the DatatypeString that is the domain specification. The first element in the compound string shall be the attribute name expressed as a NameString. This shall be followed by a separator. The second element shall be the DatatypeString that is the domain specification.

L4 <Attribute description>

If a description is provided for the attribute, the text of the description shall follow expressed as DescriptiveText. The description is the second of the two line patterns that are used for each attribute in the subject class.

6.3.2.7 Unique attribute sets

If the subject class contains the specification for one or more unique attribute sets, this sub-subsection shall appear, and it shall start with:

L2 Unique attribute sets for: <Class name>

The line content is the name of the subject class expressed as a NameString. Following this opening line, each attribute set in the subject class will be expressed with its identification number and the list of attributes within the set. The order of listing shall be the numeric sequence of the attribute set identification numbers. The expression of the entries is specified by the following two patterns.

L3 Unique set: <Attribute_set ID>

The line content is the integer that is the identification number for the set.

L4 <Attribute name>

One line containing the NameString for an attribute name shall be listed for each attribute in the set. The order of listing shall be the alphabetic order of the attribute names.

6.3.2.8 States

If one or more states are defined for the subject class, the states sub-subsection shall be placed in the document starting with the following line:

L2 States for: <Class name>

The line content is the name of the subject class expressed as a NameString. Following the opening line, the states sub-subsection shall consist of an entry for each of the states listed in any order. Each state entry will be built from the following four patterns:

L3 State: <State name>

The line content shall be the NameString that is the state name.

L4 <State description>

If a description is provided for this state, the text of that description shall follow expressed as DescriptiveText.

L4 Predicate: <Predicate text>

The text that is the predicate for the state shall be line content.

L4 Condition on: <Attribute name : ConditionString>

If conditions (limitations) on attributes of the subject class are defined for the state, this line shall be repeated once for each attribute condition, listed in alphabetic sequence by the names of the conditioned attributes. The line content shall be the NameString that expresses the attribute name followed by the ConditionString that represents the attribute condition.

NOTES

1 - The JWG-CDM acknowledges that semantics by which the attribute condition should be expressed are needed and have not been defined. This issue will be deferred for resolution in later versions of the standard.

2 - In subsequent versions, the JWG-CDM will introduce condition statements for instance connections. These will be similar to the conditions on attributes above.

3 - The JWG-CDM considered the specification of state transition tables as part of this standard. This specification was not sufficiently developed and tested to include in the normative material of this version of the standard. A preliminary recommendation for this representation is included in annex B.

6.3.2.9 Services

If one or more services are defined for the subject class, the services sub-subsection shall be placed in the document starting with the following line:

L2 Services for: <Class name>

The line content is the name of the subject class expressed as a NameString. Following the opening line, the services sub-subsection shall consist of an entry for each of the services listed in alphabetic sequence by the service name. Each service entry will be built from the following three patterns:

L3 Service: <Service name>

The line content shall be the name of the service expressed as a NameString.

L4 <Service description>

If a description is provided for the service, the text of that description shall follow expressed as DescriptiveText.

L4 Affected attribute: <Attribute name>

This line shall be repeated once for each affected attribute, and the attributes shall be listed in alphabetic order by their names. The line content is the NameString that is the name of the affected attribute.

L4 Affected connection: <Subject Class :: reference phrase :: Target class>

This line shall be repeated once for each affected instance connection. The listing order is alphabetic by target class. The line content shall be built of three components. The "target class" is the class to which the subject class is connected. The first component shall be the NameString name of the subject class. Following a separator, the second component shall be the reference phase for the affected connection when read from the subject class to the target class. Following a separator, the third component shall be the NameString that is the name of the target class.

L4 Accessed attribute: <Class and Attribute name>

This line provides the references to any attributes whose values are accessed by the service in order to complete the service's activity. This line shall be repeated once for each accessed attribute. Since these attributes will not usually be in the subject class, the line content shall be the qualified name of the attribute that combines the name of the attribute with the name of its class. The attributes shall be listed in alphabetic order by their qualified names.

6.3.3 Datatype specifications

The datatype definitions section shall be the third section in the document. It opens with:

L0 Defined datatypes in: <Model name>

The name of the model shall be expressed as a NameString.

Following the opening line there shall be one subsection for each datatype defined in the data model. Although Clause 5 describes several kinds of datatypes that may be used in specifying attributes, not all of these kinds of datatypes need to be formally defined in this section. Specifically, the primitive datatypes (Boolean, Real, Integer, and String) and the List datatype are specified directly with the attributes in question and need no further definition here. Thus the kinds of datatypes that must be defined in this section of the literary expression are Enumerated, Compound, and Constrained datatypes.

The subsection for each datatype shall occur in alphabetic order by the datatype name. Each subsection shall open with:

L1 Datatype: <Datatype name>

The name of the datatype shall be expressed as a NameString. Throughout this subclause and its own subclauses that define the patterns for the definition of a particular datatype, the phrase "subject datatype" will be used to refer to the datatype being defined.

The first pattern in this subsection is:

L2 Primary model: <Parent model name>

This line shall be used only if the subject datatype has its primary definition in a model other than the one being documented. The line content shall be the fully qualified name of the datatype as it is known in the parent model. The datatype name in the model being documented should be the same as that used in the parent model, but the two names are not required to be the same. Authors may choose to copy the datatype definition from the parent model in order to present a complete expression of the model, but are not required to do so. All references to datatypes in other models are also expressed in a separate section.

The next line of the subsection shall be:

L2 Maintained by: <Maintaining organization / last modified date>

The line content shall be the maintaining organization name for the subject datatype followed by the last modified date of the definition expressed as a Date.

L3 <Datatype description>

If a description is provided for the subject datatype, the text that describes the subject datatype shall follow expressed as DescriptiveText.

The remaining elements in the datatype definition subsection are dependent upon the kind of datatype being defined. Each of the following three subclauses provides the specification for one of these kinds.

6.3.3.1 Enumerated datatype

Enumerated datatypes may either be defined within the expression of the model, or defined by reference to an external publishing authority.

6.3.3.1.1 Defined within the model

If the subject datatype is an enumerated datatype that is defined within the model, the next line in the definition of the subject datatype shall be:

L2 Enumerated :: <DatatypeString>

The line content shall be the DatatypeString that specifies the domain for the enumerated datatype value that is used to encode the subject datatype. This DatatypeString shall be further constrained in that it may only contain one of the four basic types (Boolean, real, integer, or string).

L2 Responsible: <Responsible organization>

The line content is the text that identifies the organization or entity that has maintenance responsibility for the value table. This may or may not be the same as the maintaining organization for this datatype definition.

The next line in the subsection should be a table header line defined by the following pattern:

L2 { Value { Name } Description } <null>

If it is used, this line has no content. If preferred, the vertical bar character (|) may be used in place of the curly brackets ({}) in the pattern above.

Following the opening lines, the subsection shall include an entry for each defined value specified for the subject datatype being enumerated. These entries are specified by the line pattern:

L2 <Value/Name/Description>

The line content is a compound string that fully defines each enumerated datatype element. For each element, it combines the value of the element, the name of the element, and the description of the element. The compound string shall have three segments. A left curly bracket ({) or a vertical bar (|) shall precede the first segment. The first segment shall contain the value of the element. A left curly bracket ({) or a vertical bar (|) shall follow the first segment and precede the second segment. The second segment of the compound string shall be either the element name, if one is provided, or one or more spaces if there is no name. A right curly bracket (}) or a vertical bar (|) shall follow the second segment and precede the third segment. The third segment of the compound string shall be either the element description, if one is provided, or one or more spaces if there is no description. The element description may contain more than one paragraph. A right curly bracket (}) or a vertical bar (|) shall follow the third segment of the compound string.

The set of lines constitutes a table of definitions for the enumerated datatype. Authors should not mix the use of vertical bars and curly brackets as separators within a single table. As can be seen in annex C, the segments of these strings can be spaced so as to appear as an aligned table in the literary expression. The

table entry for the last element of the enumerated datatype ends the datatype definition subsection for a particular enumerated datatype.

6.3.3.1.2 Referenced to external authority

If the subject datatype is an enumerated datatype that is referenced rather than being defined within the model, the next line in the definition of the subject datatype shall be:

L2 Enumerated :: <DatatypeString>

The line content shall be the DatatypeString that specifies the domain for the enumerated datatype value that is used to encode the subject datatype. This DatatypeString shall be further constrained in that it may only contain one of the four basic types (Boolean, real, integer, or string).

L2 Reference source: <Maintaining organization>

The line content is the text that identifies the organization or entity that has maintenance responsibility for the value table. This may or may not be the same as the maintaining organization for this datatype definition. This line shall be followed by:

L3 <Reference description>

The line content shall be an explicit reference to the enumerated table that is published or maintained elsewhere. The description shall include explicit information as to which portions of the referenced table correspond to the value, name and description fields that would be part of an internally defined enumerated datatype.

6.3.3.2 Compound datatype

If the subject datatype is a compound datatype, the next line in the definition of the subject datatype shall be:

L2 Compound datatype has parts: <null>

There is no line content for this line. Following this opening line, the subsection shall include a pair of entries for each compound datatype component for the subject datatype. The components shall be listed according to their preferred order of presentation. The pair of entries are specified by the next two line patterns.

L3 <name/DatatypeString>

The line content is a compound string that combines the name of the compound datatype part and the DatatypeString that is the domain specification of the part. The first element in the compound string shall be the compound datatype part name expressed as a NameString. This shall be followed by a separator. The second element shall be the DatatypeString that is the domain specification of the part.

L4 <Compound_datatype_part description>

If a description is provided for the compound datatype part, the text of the description shall follow expressed as DescriptiveText. The description is the second of the two line patterns that are used for each part in the subject datatype.

The line pair for the last part of the compound datatype ends the datatype definition subsection for a particular compound datatype.

6.3.3.3 Constrained datatype

If the subject datatype is a constrained datatype, the next line in the definition of the subject datatype shall be:

L2 Constrained :: <DatatypeString>

The line content shall be the DatatypeString that is the domain specification of the nominal (unconstrained) attribute datatype of the subject datatype. This line shall be followed by:

L2 Predicate statement for: <Datatype name>

The line content shall be the NameString name of the subject datatype. This line shall be followed by:

L3 <Predicate statement>

The line content shall be one or more paragraphs that express the rules of constraint to be applied to the subject datatype. These paragraphs end the datatype definition subsection for a particular constrained datatype.

6.3.4 External references

The external references section shall be the fourth and final section in the document. This section collects all references to classes or datatypes that have their primary definition in a model other than the one being documented. All external references must be listed in this section. The classes and datatypes listed here may also be defined in earlier sections of the document, if desired. If they are listed there, the fully qualified name of the element must also be listed with that definition.

The External references section opens with:

L0 External references for: <Model name>

The name of the model shall be expressed as a NameString.

Following the opening line there shall be one subsection for referenced classes and one for referenced datatypes.

6.3.4.1 Referenced classes

L1 Referenced classes: <null>

If any external classes are referenced in the model, this line shall be used to open the referenced classes subsection. There is no line content.

L2 <External class name :: Class name>

This line shall be used for each class in the model that has its primary definition in a model other than the one being documented. The line content is a compound string combining the external reference with the internal name of the class, if it is different. The first segment of the line content shall be the fully qualified name of the class as it is known in the parent model. If a different name has been assigned to the class within the model being documented, the NameString internal name shall be the second segment of the line content with a separator in between. The order of listing shall be the alphabetic sequence of the fully qualified external names of the classes.

6.3.4.2 Referenced datatypes

L1 Referenced datatypes: <null>

If any external datatypes are referenced in the model, this line shall be used to open the referenced datatypes subsection. There is no line content.

L2 <External datatype name :: Datatype name>

This line shall be used for each datatype in the model that has its primary definition in a model other than the one being documented. The line content is a compound string combining the external reference with the internal name of the datatype, if it is different. The first segment of the line content shall be the fully qualified name of the datatype as it is known in the parent model. If a different name has been assigned to the datatype within the model being documented, the NameString internal name shall be the second segment of the line content with a separator in between. The order of listing shall be the alphabetic sequence of the fully qualified external names of the datatypes.

6.4 Graphical expression specification

The graphical expression of a data model is the pictorial representation of the most critical elements of the data model. It can express the classes in the model, the various connections between those classes, the attributes of the classes, the name of the attribute datatype for each attribute, and the services for the classes. It cannot, however, express the model scenarios, the unique attribute sets and states for the classes, nor the specification for the named datatypes defined in the model. Moreover, the graphical expression contains none of the descriptive text that may be associated with each element of the data model.

The principle role of the graphical expression is to provide an overview of the model, to make clear the important connections that exist between classes, and to provide a map by which readers can navigate through the complete specification contained in the literary expression.

The graphical expression is made up of graphic symbols for the classes and connections in the model, and of textual labels that are placed within or adjacent to those symbols. The subclauses that follow will specify the form of these symbols and the format and location of the labels. This standard cannot,

however, specify the layout of these elements, and good layout is an essential feature of a useful graphical expression. Thus layout is left to the artistic skill of the modeler.

The following graphic is provided to exemplify the symbols being described in the following subclauses. The graphic does not represent a complete or even a logical data model. Its sole purpose is to demonstrate each element of the graphical expression.

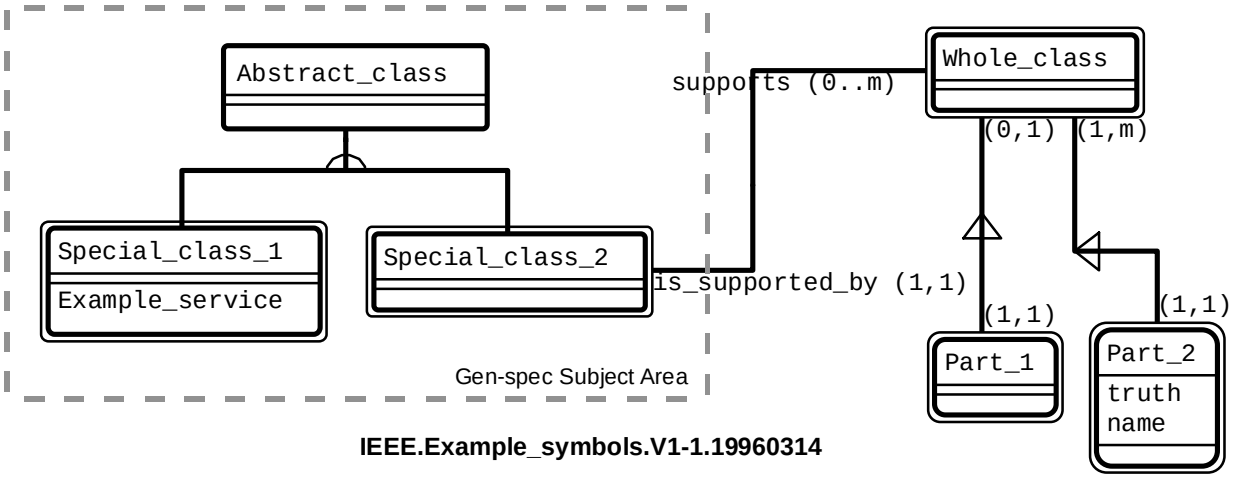

Figure 4 - Example of graphical expression conventions

The conventions for expressing the components of a data model in the graphical symbols are specified in the following subclauses. These conventions are those documented by Coad and Yourdon [B1]. The order of presentation is the same order that was used in the literary expression.

6.4.1 Model identification

Each page of the graphical expression shall contain the identification of the model being represented. This identification shall be a textual label containing the fully qualified name of the model, which includes the developing organization, and the name, version, and last modified date of the model. In figure 4, this identification is at the bottom of the diagram.

6.4.2 Subject areas

Subject areas may be depicted in the graphic expression if the authors desire. Subject areas should be indicated by a gray or dashed line surrounding the symbols of the classes that are part of the subject area. An example subject area is shown on the right side of the diagram above. At times, the subject area lines confuse a drawing because they may obscure important connecting lines. It is for this reason that subject areas are not depicted in the graphical expression of the Sample Data Model contained in annex C.

6.4.3 Classes

The classes in the model may be represented in the graphical expression by its own symbol. The class symbol is a rounded rectangle nested inside a similar rectangle. The inner rectangle should be drawn with a bolder line than the outer rectangle. The inner rectangle represents the class and terminates all connections that apply to every instance of the class, while the outer rectangle represents the instances of the class and terminates the connections that apply only to specific instances of the class.

The class symbol shall be divided into three sections by horizontal lines. The NameString that is the name of the class shall be placed in the upper-most section of the symbol.

An abstract class is one that will not be instantiated. Therefore it is drawn without the outer rectangle which represents the instances of a class. Figure 4 contains an abstract class in the upper left, and five classes that can be instantiated.

6.4.4 Generalization-specialization connections

The generalization-specialization structure is a connection between a more general class and one or more specializations of that class. This connection shall be drawn as a line between the bottom of the general class rectangle (inner rectangle) and the top of the specialized class rectangle (inner rectangle). A semicircle shall be drawn somewhere along the line such that the line to the general class connects to the upper, circular portion of the semicircle, while the line(s) to the specialized class(es) connect to the flat bottom of the semicircle. The generalization-specialization connection is not labeled.

The three classes on the left of figure 4 are connected by a generalization-specialization structure in which the abstract class is the general class.

6.4.5 Whole-part connections

Whole-part connections are connections between the instances of a whole class and the instances of its parts. Each part class shall be connected to its whole class by a connection from the top of the instance (outer) rectangle of the part to the bottom of the instance (outer) rectangle of the whole. Somewhere along the line shall be placed a triangle with the line to the whole connecting to the apex of the triangle and the line to the part connecting to the base of the triangle. Each part shall have its own connecting line to the whole.

The whole-part connection shall be labeled with the cardinality of the relationship. The CardinalityString that represents the cardinality of the part shall be placed near where the connection line meets the whole. The cardinality of the whole is always expressed by the CardinalityString "(1,1)". This string should be placed near where the connection meets the part.

The three classes on the right of figure 4 participate in a whole-part structure.

6.4.6 Instance connections

Instance connections express a relationship between the instances of two classes. Instance connections shall be drawn from one side of the instance (outer) rectangle of one of the classes to a side of the instance (outer) rectangle of the other.

The instance connection line shall be labeled with the cardinalities of the connection and should also be labeled with the reference phrases. The labels should be placed near the connection points at either end of the line. The labels are built of two parts. Each label should start with the reference phrase that expresses the nature of the relationship from the perspective of the class that is adjacent to the label. Following the phrase should be the cardinality of the class at the other end of the connection.

The location of the cardinality is that recommended by [B1]. Although it is the reverse of what is normally done in entity-relationship diagrams, it provides an easily read diagram. An instance connection is drawn

between the Special_Class_2 and Whole_Class in the example. Starting from Special_Class_2 this connection can be read as "Special_Class_2 is_supported_by one and only one Whole_Class." Starting from the other direction, this connection can be read as "Whole_Class may support many Special_Class_2."

6.4.7 Attributes

The attributes of each class may be listed by including their NameStrings in the center section of the class rectangle.

The Part_2 class in figure 4 shows two attributes.

6.4.8 Services

The services of each class that has defined services may be listed in the bottom-most section of the class rectangle. These shall be listed as the NameString name of the service.

Special_Class_1 has an example of a service.

6.4.9 Compound Datatypes

Compound datatypes are a structured datatype made up of multiple parts each of which has its own set of datatype constraints. These may be represented graphically along with the elements of the data models they apply to. If some of the compound datatypes for a model are represented graphically, all of them should be.

Compound datatypes shall be contained within a subject area named "Compound Datatypes." Each compound datatype shall be represented using the same symbol as that specified for an abstract class. (See 6.4.3.) This symbol is a rectangle divided into three sections by horizontal lines. The NameString that is the name of the compound datatype shall be placed in the upper-most section of the symbol.

The names of the compound datatype components that are the parts of the compound datatype may be represented by their NameStrings in the center section of the compound datatype rectangle.

Although they are diagrammed using class symbols, the compound datatypes shall not diagrammed as: being structured in generalization-specialization structures; being connected by whole-part or instance connection relationships; or possessing services, unique attribute sets or states.

7 Conformance to standard

Ultimately, there are three criteria of conformance in the context of standards relating to a Common Data Model to support harmonization of health care standards:

a) Adoption of data modeling as a basis for standards development

b) Appropriate use of the framework for a Common Data Model (this standard)

c) Conformance to the content of the Common Data Model

Although only items a) and b) above are directly relevant to this standard, all three are presented to provide a background for consideration of this standard.

7.1 Adoption of data modeling

Use of data modeling is the fundamental foundation for adopting the Data Model Framework. If an SDO does not adopt data modeling as a basis for its standards, it will not be possible to include its standards in the harmonization of a CDM. Although transformation of data content from standards that are not built upon data models will be used as an initial strategy, it will not be possible to maintain such transformations over time. Eventually, participating SDOs must undertake data modeling on their own.

7.2 Use of this standard in data modeling

This standard specifies a meta-model to which data models shall conform. It also specifies the expression that must be used to represent the content of these models. Conformance to this standard is considered below in terms of models that are submitted by SDOs (SDMs) for harmonization and incorporation into the CDM, and in terms of the incorporation of this meta-model into the SDOs' own frameworks.

7.2.1 Conformance in published models

In order for SDMs to be harmonized and considered for incorporation into the CDM, they should conform to the specifications of this standard. Conformance in this context means that:

a) The data model shall be constructed according to the meta-model expressed in clauses 5 and 6. This includes the definition of all required meta-model elements identified in those clauses.

b) The data model shall be expressed in a Literary expression as specified in clause 6, and shall include all of the required elements of that expression.

c) The data model should be expressed in a Graphical expression as specified in clause 6.

Claims of conformance shall state that they adhere to items (a) and (b) above and state whether or not they adhere to item (c).

NOTES - The requirement in the first paragraph of this clause is stated as a "should" rather than a "shall." The JWG-CDM expects there will need to be a protracted period of use of this standard before it becomes an absolute requirement for submissions to the CDM. Certainly, it will remain a preferred (rather than mandatory) requirement throughout the trial-use period of this standard.

7.2.2 Adoption of framework specification

Each SDO is strongly encouraged to adopt a framework that incorporates the framework of this standard as a subset of the SDO's complete framework. If this is done, the use of CDM components in the standards of that SDO become a seamless undertaking. Short of such adoption, however, the SDO can participate fully in the CDM efforts if it undertakes to represent its own data models in the form specified in this standard.

NOTE - If an SDO incorporates the meta-model from this standard into its own framework, then conformance to this standard is implicit in that SDO's development of standards. A statement of such conformance remains to be defined.

7.2.3 Conformance testing and validation

This standard does not specify the following:

a) testing or validation procedures to assess a particular SDO's conformance to the framework of this standard

b) testing or validation procedures to assess whether an SDM matches to its conformance statement

c) what optional features should be supported for a given application of this standard.

7.3 Adoption of Common Data Model components

The third form of conformance is the use of CDM specifications as the information content for the standards developed by each SDO. Consideration of this form of conformance is premature, until this standard is in trial use and the harmonization of submitted models has been attempted in several cases. In any event, JWG-CDM will be able to do no more than monitor the compliance of each SDO.

8 Bibliography

[B1] Coad, Peter and Yourdon, Edward, *Object-Oriented Analysis*. 2d ed. Prentice Hall 1991, New Jersey

[B2] Harel, D., "Statecharts: a visual formalism for complex systems". in *Science Of Computer Programming*, Vol 8 (1987), pp 231-274

[B3] Jacobson, Ivar; Christerson, Magnus; Jonsson, Patrik and Overgaard, Gunnar, *Object Oriented Software Engineering - A Use Case Driven Approach*. ACM Press - Addison Wesley 1992 (ISBN 0-201-54435-0)

[B4] Booch, Grady, "Object Oriented Design with applications", Benjamin/Cummings, 1991 (ISBN-0-8053-0091-0)

[B5] Rumbaugh, James; Blaha, Michael; Premerlani, William; Eddy, Frederick; Lorenson, William; "Object-Oriented Modelling and Design", Prentice Hall, 1991 (ISBN-0-13-629841-9)

[B6] Bidgood, W. Dean; Tracy, Wayne R.; "ANSI HISPP MSDS: Common Data Types - For Harmonization of Communication Standards in Medical Informatics", Final Draft : June 18, 1993

[This page has been intentionally left blank]