INTERNATIONAL STANDARD

ISO 13606-2

First edition
2008-12-01

Health informatics — Electronic health record communication —

Part 2: Archetype interchange specification

Informatique de santé — Communication du dossier de santé informatisé —

Partie 2: Spécification d'échange d'archétype

Reference number
ISO 13606-2:2008(E)

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO 2008

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

COPYRIGHT PROTECTED DOCUMENT

© ISO 2008

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Contents

Page

Foreword iv
Introduction v
1 Scope 1
2 Conformance 1
3 Normative references 1
4 Terms and definitions 2
5 Symbols and abbreviations 3
6 Archetype representation requirements 4
6.1 General 4
6.2 Archetype definition, description and publication information 4
6.3 Archetype node constraints 6
6.4 Data value constraints 8
6.5 Profile in relation to EN 13606-1 Reference Model 10
7 Archetype model 11
7.1 Introduction 11
7.2 Overview 14
7.3 The archetype package 18
7.4 The archetype description package 20
7.5 The constraint model package 24
7.6 The assertion package 31
7.7 The primitive package 35
7.8 The ontology package 42
7.9 The domain extensions package 44
7.10 The support package 47
7.11 Generic types package 56
7.12 Domain-specific extensions (informative) 57
8 Archetype Definition Language (ADL) 58
8.1 dADL — Data ADL 58
8.2 cADL — Constraint ADL 79
8.3 Assertions 106
8.4 ADL paths 110
8.5 ADL — Archetype definition language 111
Bibliography 123

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 13606-2 was prepared by Technical Committee ISO/TC 215, *Health informatics*.

ISO 13606 consists of the following parts, under the general title *Health informatics — Electronic health record communication*:

— *Part 1: Reference model*

— *Part 2: Archetype interchange specification*

— *Part 3: Reference archetypes and term lists*

— *Part 5: Interface specification*

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

Comprehensive, multi-enterprise and longitudinal electronic health records will often in practice be achieved through the joining up of multiple clinical applications, databases (and increasingly devices) that are each tailored to the needs of individual conditions, specialties or enterprises.

This requires that Electronic Health Record (EHR) data from diverse systems be capable of being mapped to and from a single comprehensive representation, which is used to underpin interfaces and messages within a distributed network (federation) of EHR systems and services. This common representation has to be sufficiently generic and rich to represent any conceivable health record data, comprising part or all of an EHR (or a set of EHRs) being communicated.

The approach adopted in the ISO 13606 series of International Standards, underpinned by international research on the EHR, has been to define a rigorous and generic Reference Model that is suitable for all kinds of data and data structures within an EHR, and in which all labelling and context information is an integral part of each construct. An EHR Extract (as defined in ISO 13606-1) will contain all the names, structure and context required for it to be interpreted faithfully on receipt, even if its organization and the nature of the clinical content have not been "agreed" in advance.

However, the wide-scale sharing of health records, and their meaningful analysis across distributed sites, also requires that a consistent approach be used for the clinical (semantic) data structures that will be communicated via the Reference Model, so that equivalent clinical information is represented consistently. This is necessary in order for clinical applications and analysis tools to safely process EHR data that have come from heterogeneous sources.

Archetypes

The challenge for EHR interoperability is therefore to devise a generalized approach to representing every conceivable kind of health record data structure in a consistent way. This needs to cater for records arising from any profession, speciality or service, whilst recognising that the clinical data sets, value sets, templates, etc., required by different health care domains will be diverse, complex and will change frequently as clinical practice and medical knowledge advance. This requirement is part of the widely acknowledged health informatics challenge of semantic interoperability.

The approach adopted by this part of ISO 13606 distinguishes a Reference Model, used to represent the generic properties of health record information, and Archetypes (conforming to an Archetype Model), which are metadata used to define patterns for the specific characteristics of the clinical data that represent the requirements of each particular profession, speciality or service.

The Reference Model is specified as an Open Distributed Processing (ODP) Information Viewpoint Model, representing the global characteristics of health record components, how they are aggregated, and the context information required to meet ethical, legal and provenance requirements. In the ISO 13606 series of International Standards, the Reference Model is defined in ISO 13606-1. This model defines the set of classes that form the generic building blocks of the EHR. It reflects the stable characteristics of an electronic health record, and would be embedded in a distributed (federated) EHR environment as specific messages or interfaces (as specified in ISO 13606-5).

Archetypes are effectively pre-coordinated combinations of named RECORD_COMPONENT hierarchies that are agreed within a community in order to ensure semantic interoperability, data consistency and data quality.

For an EHR_Extract, as defined in ISO 13606-1, an archetype specifies (and effectively constrains) a particular hierarchy of RECORD_COMPONENT subclasses, defining or constraining their names and other relevant attribute values, optionality and multiplicity at any point in the hierarchy, the data types and value ranges that ELEMENT data values may take, and may include other dependency constraints. Archetype

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

instances themselves conform to a formal model, known as an Archetype Model (which is a constraint model, also specified as an ODP Information Viewpoint Model). Although the Archetype Model is stable, individual archetype instances may be revised or succeeded by others as clinical practice evolves. Version control ensures that new revisions do not invalidate data created with previous revisions.

Archetypes may be used within EHR systems to govern the EHR data committed to a repository. However, for the purposes of this interoperability standard, no assumption is made about the use of archetypes within the EHR provider system whenever this standard is used for EHR communication. It is assumed that the original EHR data, if not already archetyped, may be mapped to a set of archetypes, if desired, when generating the EHR_EXTRACT.

The Reference Model defined in ISO 13606-1 has attributes that can be used to specify the archetype to which any RECORD_COMPONENT within an EHR_EXTRACT conforms. The class RECORD_COMPONENT includes an attribute *archetype_id* to identify the archetype and node to which that RECORD_COMPONENT conforms. The *meaning* attribute, in the case of archetyped data, refers to the primary concept to which the corresponding archetype node relates. However, it should be noted that ISO 13606-1 does not require that archetypes be used to govern the hierarchy of RECORD_COMPONENTS within an EHR_EXTRACT; the archetype-related attributes are optional in that model. It is recognised that the international adoption of an archetype approach will be gradual, and may take some years.

Archetype repositories

The range of archetypes required within a shared EHR community will depend upon its range of clinical activities. The total set needed on a national basis is currently unknown, but there might eventually be several thousand archetypes globally. The ideal sources of knowledge for developing such archetype definitions will be clinical guidelines, care pathways, scientific publications and other embodiments of best practice. However, *de facto* sources of agreed clinical data structures might also include:

— the data schemata (models) of existing clinical systems;

— the lay-out of computer screen forms used by these systems for data entry and for the display of analyses performed;

— data-entry templates, pop-up lists and look-up tables used by these systems;

— shared-care data sets, messages and reports used locally and nationally;

— the structure of forms used for the documentation of clinical consultations or summaries within paper records;

— health information used in secondary data collections;

— the pre-coordinated terms in terminology systems.

Despite this list of *de facto* ways in which clinical data structures are currently represented, these formats are very rarely interoperable. The use of standardized archetypes provides an interoperable way of representing and sharing these specifications, in support of consistent (good quality) health care record-keeping and the semantic interoperability of shared EHRs.

The involvement of national health services, academic organizations and professional bodies in the development of archetypes will enable this approach to contribute to the pursuit of quality evidence-based clinical practice. The next key challenge is to foster communities to build up libraries of archetypes. It is beyond the scope of this part of ISO 13606 to assert how this work should be advanced, but, in several countries so far it would appear that national health programmes are beginning to organize clinical-informatics-vendor teams to develop and operationalize sets of archetypes to meet the needs of specific healthcare domains. In the future, regional or national public domain libraries of archetype definitions might be accessed via the Internet, and downloaded for local use within EHR systems. Such useage will also require processes to verify and certify the quality of shared archetypes, which are also beyond the scope of this part

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

of ISO 13606 but are being taken forward by non-profit-making organizaitons such as the openEHR Foundation and the EuroRec Institute.

Communicating archetypes

This part of ISO 13606 specifies the requirements for a comprehensive and interoperable archetype representation, and defines the ODP Information Viewpoint representation for the Archetype Model and an optional archetype interchange format called Archetype Definition Language (ADL).

This part of ISO 13606 does not require that any particular model be adopted as the internal architecture of archetype repositories, services or components used to author, store or deploy archetypes in collaboration with EHR services. It does require that these archetypes be capable of being mapped to the Archetype Model defined in this part of ISO 13606 in order to support EHR communication and interoperability within an EHR-sharing community.

Overview of the archetype model

This section provides a general informative description of the model that is specified in Clause 7.

The overall archetype model consists of identifying information, a description (its metadata), a definition (expressed in terms of constraints on instances of an object model), and an ontology. Identifying information and lifecycle state are part of the *ARCHETYPE* class. The archetype description is separated into revision history information and descriptive information about the archetype. Revision history information is concerned with the committal of the archetype to a repository, and takes the form of a list of audit trail items, while descriptive information describes the archetype itself (regardless of whether it has been committed to a repository of any kind).

The archetype definition, the “main” part of the archetype model, is an instance of a *C_COMPLEX_OBJECT*, since the root of the constraint structure of an archetype shall always take the form of a constraint on a non-primitive object type. The fourth main part of the archetype model, the ontology, is represented by its own class, and is what allows the archetypes to be natural-language- and terminology-neutral.

An enumeration class, *VALIDITY_KIND*, is also included in the archetype package. This is intended to be used as the type of any attribute in this constraint model whose value is logically “mandatory”, “optional”, or “disallowed”. It is used in this model in the classes *C_Date*, *C_Time* and *C_Date_Time*.

Archetypes contain some natural language elements, including the description and ontology definitions. Every archetype is therefore created in some original language, which is recorded in the *original_language* attribute of the *ARCHETYPE* class. An archetype is translated by doing the following:

— translating every language-dependent element into the new language;

— adding a new *TRANSLATION_DETAILS* instance to *ARCHETYPE.translations*, containing details about the translator, organization, quality assurance and so on.

The *languages_available* function provides a complete list of languages in the archetype.

The archetype definition

The main definitional part of an archetype consists of alternate layers of object- and attribute-constraining nodes, each containing the next level of nodes. In this section, the word “attribute” refers to any data property of a class, regardless of whether it is regarded as a “relationship” (i.e. association, aggregation, or composition) or a “primitive” (i.e. value) attribute. At the leaves are primitive object constrainer nodes constraining primitive types such as String, Integer, etc. There are also nodes that represent internal references to other nodes, constraint reference nodes that refer to a text constraint in the constraint binding part of the archetype ontology, and archetype constraint nodes, which represent constraints on other archetypes allowed to appear at a given point.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The full list of node types is as follows:

— C_complex_object: any interior node representing a constraint on instances of a non-primitive type, e.g. ENTRY, SECTION;

— C_attribute: a node representing a constraint on an attribute (i.e. UML "relationship" or "primitive attribute") in an object type;

— C_primitive_object: a node representing a constraint on a primitive (built-in) object type;

— Archetype_internal_ref: a node that refers to a previously defined object node in the same archetype; the reference is made using a path;

— Constraint_ref: a node that refers to a constraint on (usually) a text or coded term entity, which appears in the ontology section of the archetype, and in ADL, and is referred to using an "acNNNN" code; the constraint is expressed in terms of a query on an external entity, usually a terminology or ontology;

— Archetype_slot: a node whose statements define a constraint that determines which other archetypes may appear at that point in the current archetype; logically it has the same semantics as a C_COMPLEX_OBJECT, except that the constraints are expressed in another archetype, not the current one.

The archetype description

What is normally considered the "metadata" of an archetype, i.e. its author, date of creation, purpose, and other descriptive items, is described by the *ARCHETYPE_DESCRIPTION* and *ARCHETYPE_DESCRIPTION_ITEM* classes. The parts of this that are in natural language, and therefore may require translated versions, are represented in instances of the *ARCHETYPE_DESCRIPTION_ITEM* class. If an *ARCHETYPE_DESCRIPTION* has more than one *ARCHETYPE_DESCRIPTION_ITEM*, each of these should carry exactly the same information in a different natural language.

When an archetype is translated for use in another language environment, each ARCHETYPE_DESCRIPTION_ITEM needs to be copied and translated into the new language.

The *AUDIT_DETAILS* class is concerned with the creation and modification of the archetype in a repository. Each instance of this class in an actual archetype represents one act of committal to the repository, with the attributes documenting who, when and why.

NOTE Revision of an archetype should be limited to modifying the descriptive information and adding language translations and/or term bindings. If the definition part of an archetype is no longer valid it should instead be replaced with a new archetype to ensure that corresponding EHR data instances each conform to the same archetype specification.

The archetype ontology

All linguistic entities are defined in the ontology part of the archetype. There are four major parts in an archetype ontology: term definitions, constraint definitions, term bindings and constraint bindings. The former two define the meanings of various terms and textual constraints which occur in the archetype; they are indexed with unique identifiers that are used within the archetype definition body. The latter two ontology sections describe the mappings of terms used internally to external terminologies.

Archetype specialization

Archetypes may be specialized: an archetype is considered a specialization of another archetype if it specifies that archetype as its parent, and only makes changes to its definition such that its constraints are "narrower" than those of the parent. Any data created via the use of the specialized archetype shall be conformant both to it and to its parent.

Every archetype has a "specialization depth". Archetypes with no specialization parent have depth 0, and specialized archetypes add one level to their depth for each step down a hierarchy required to reach them.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Archetype composition

Archetypes may be composed to form larger structures semantically equivalent to a single large archetype. Archetype slots are the means of composition, and are themselves defined in terms of constraints.

Data types and the support package

The model is dependent on three groups of assumed types, whose names and assumed semantics are described by ISO/IEC 11404.

The first group comprises the most basic types:

- Any
- Boolean
- Character
- Integer
- Real
- Double
- String

The second comprises the assumed library types:

- date
- time
- date_time
- duration

These types are supported in most implementation technologies, including XML, Java and other programming languages. They are not defined in this specification, allowing them to be mapped to the most appropriate concrete types in each implementation technology.

The third group comprises the generic types:

- List<T> (ordered, duplicates allowed)
- Set<T> (unordered, no duplicates)
- Hash <T, K > (keyed list of items of any type)
- Interval <T> (interval of instances of any ordered type)

Although these types are supported in most implementation technologies, they are not yet represented in UML. The semantics of these types are therefore defined in the Generic_Types package of the UML model.

The remaining necessary types are defined in the Support Package of the Archetype Model.

- ARCHETYPE_ID
- HIER_OBJECT_ID
- TERMINOLOGY_ID

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— CODE_PHRASE

— CODED_TEXT

The support package also includes two enumeration classes to provide controlled data sets needed by this part of ISO 13606.

The constraint model package

Any archetype definition is an instance of a *C_COMPLEX_OBJECT*, which expresses constraints on a class in the underlying Reference Model (see ISO 13606-1), recorded in the attribute *rm_type_name.* A *C_COMPLEX_OBJECT* consists of attributes of the type *C_ATTRIBUTE*, which are constraints on the attributes (i.e. any property, including relationships) of that Reference Model class. Accordingly, each *C_ATTRIBUTE* records the name of the constrained attribute (in *rm_attribute_name)*, the existence and cardinality expressed by the constraint (depending on whether the attribute it constrains is a multiple or single relationship), and the constraint on the object to which this *C_ATTRIBUTE* refers via its *children* attribute (according to its reference model) in the form of further *C_OBJECTs*.

The key subtypes of *C_OBJECT* are:

— *C_COMPLEX_OBJECT*

— *C_PRIMITIVE_OBJECT*

— *ARCHETYPE_SLOT*

ARCHETYPE_INTERNAL_REF and *CONSTRAINT_REF* are used to express, respectively, a “slot” where further archetypes may be used to continue describing constraints; a reference to a part of the current archetype that expresses exactly the same constraints needed at another point; a reference to a constraint on a constraint defined in the archetype ontology, which in turn points to an external knowledge resource, such as a terminology.

The effect of the model is to create archetype description structures that are a hierarchical alternation of object and attribute constraints. The repeated object/attribute hierarchical structure of an archetype provides the basis for using *paths* to reference any node in an archetype. Archetype paths follow a syntax that is a subset of the W3C Xpath syntax.

All node types

All nodes in an archetype constraint structure are instances of the supertype *ARCHETYPE_CONSTRAINT*, which provides a number of important common features to all nodes.

The any_allowed Boolean, if true, indicates that any value permitted by the reference model for that attribute is allowed by the archetype; its use permits the logical idea of a completely “open” constraint being simply expressed, avoiding the need for any further substructure.

Attribute nodes

Constraints on attributes are represented by instances of the two subtypes of *C_ATTRIBUTE*: *C_SINGLE_ATTRIBUTE* and *C_MULTIPLE_ATTRIBUTE*. For both subtypes, the common constraint is whether the corresponding instance (defined by the *rm_attribute_name* attribute) must exist. Both subtypes have a list of children, representing constraints on the object value(s) of the attribute.

Single-valued attributes are constrained by instances of the type *C_SINGLE_ATTRIBUTE*, which uses the children to represent multiple alternative object constraints for the attribute value.

Multiple-valued attributes are constrained by an instance of *C_MULTIPLE_ATTRIBUTE*, which allows multiple co-existing member objects of the container value of the attribute to be constrained, along with a cardinality constraint, describing ordering and uniqueness of the container.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO 2008 – All rights reserved
Not for Resale

Cardinality is only required for container objects such as *List<T>*, *Set<T>*, *Bag<T>* and so on, whereas *existence* is always required. If both are used, the meaning is as follows: the existence constraint says whether the container object will be there (at all), while the cardinality constraint says how many items shall be in the container, and whether it acts logically as a list, set or bag.

Primitive types

Constraints on primitive types are defined by the classes inheriting from *C_PRIMITIVE*, namely *C_STRING*, *C_INTEGER* and so on.

Constraint references

A *CONSTRAINT_REF* is a proxy for a set of constraints on an object that would normally occur at a particular point in the archetype as a *C_COMPLEX_OBJECT*, but where the actual definition of the constraint is expressed as the binding to a query or expression into an external service (such as an ontology or terminology service), e.g.:

— a set of allowed *CODED_TERMs,* e.g. the types of hepatitis;

— an *INTERVAL<QUANTITY>* forming a reference range;

— a set of units or properties or other numerical items.

Assertions

The *C_ATTRIBUTE* and subtypes of *C_OBJECT* enable constraints to be expressed in a structural fashion. In addition to this, any instance of a *C_COMPLEX_OBJECT* may include one or more *invariants*. Invariants are statements in a form of predicate logic, which may be used to state constraints on parts of an object. They are not needed to state constraints on a single attribute (since this can be done with an appropriate *C_ATTRIBUTE*), but are necessary to state constraints on more than one attribute, such as a constraint that “systolic pressure should be ⩾ diastolic pressure” in a blood pressure measurement archetype. Such invariants may be expressed using a syntax derived from the Object Management Group's (OMG) OCL syntax.

Assertions are also used in *ARCHETYPE_SLOTs*, in order to express the “included” and “excluded” archetypes for the slot.

Assertions are modelled as a generic expression tree of unary prefix (e.g. *not* p) and binary infix (e.g. p *and* q) operators.

Node_id and paths

The node_id attribute in the class *C_OBJECT* and inherited to all subtypes has two functions:

— it allows archetype object constraint nodes to be individually identified and, in particular, guarantees sibling node unique identification;

— it is the main link between the archetype definition (i.e. the constraints) and the archetype ontology, because each *node_id* is a “term code” in the ontology.

The existence of *node_ids* in an archetype is what allows archetype paths to be created, which refer to each node.

Domain-specific extensions

The main part of the archetype constraint model allows any type in a reference model to be archetyped — i.e. constrained — in a standard way by a regular cascade of *C_COMPLEX_OBJECT* / *C_ATTRIBUTE* / *C_PRIMITIVE_OBJECT* objects. However, lower level logical “leaf” types may need special constraint semantics that are not conveniently achieved with the standard approach. To enable such classes to be

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

integrated into the generic constraint model, the class *C_DOMAIN_TYPE* is included. This enables the creation of specific "*C_*" classes, inheriting from *C_DOMAIN_TYPE*, which represent custom semantics for particular reference model types. For example, a class called *C_QUANTITY* might be created which has different constraint semantics from the default effect of a *C_COMPLEX_OBJECT* / *C_ATTRIBUTE* cascade representing such constraints in the generic way (i.e. systematically based on the reference model).

Assumed values

When archetypes are defined to have optional parts, an ability to define "assumed" values is useful. For example, an archetype for the concept "blood pressure measurement" might contain an optional fragment describing the patient position, with choices "lying", "sitting" and "standing". Since that part of the ENTRY is optional, data could be created according to the archetype that does not contain this information. However, a blood pressure cannot be taken without the patient in some position, so it may be clinically valid to define an implied or "assumed" value. The archetype allows this to be explicitly stated so that all users/systems know what value to assume when optional items are not included in the data. Assumed values are definable at the leaf level only and may be specified in the *C_PRIMITIVE* classes.

The notion of assumed values is distinct from that of "default values"; default values do appear in data, while assumed values do not.

The assertion package

Assertions are expressed in archetypes in typed first-order predicate logic (FOL). They are used in two places: to express archetype slot constraints, and to express invariants in complex object constraints. In both of these places, their role is to constrain something inside the archetype. Constraints on external resources such as terminologies are expressed in the constraint binding part of the archetype ontology.

The concrete syntax of assertion statements in archetypes is designed to be compatible with the OMG Object Constraint Language (OCL). Archetype assertions are statements that contain the following elements:

- variables that are attribute names or ADL paths terminating in attribute names (i.e. equivalent of referencing class feature in a programming language);
- manifest constants of any primitive type, plus date/time types;
- arithmetic operators: +, *, -, /, ^ (exponent);
- relational operators: >, <, >=, <=, =, !=, matches;
- Boolean operators: not, and, or, xor;
- quantifiers applied to container variables: for_all, exists.

The primitives package

Ultimately, any archetype definition will devolve down to leaf node constraints on instances of primitive types. The primitives package defines the semantics of constraint on such types. Most of the types provide at least two alternative ways to represent the constraint; for example, the *C_DATE* type allows the constraint to be expressed in the form of a pattern or an *Interval<Date>*.

The ontology package

All linguistic and terminological entities in an archetype are represented in the ontology part of an archetype, whose semantics are given in the ontology package.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

An archetype ontology comprises the following things.

- A list of terms defined local to the archetype. These are identified by "atNNNN" codes, and perform the function of archetype node identifiers from which paths are created. There is one such list for each natural language in the archetype.
- A list of external constraint definitions, identified by "acNNNN" codes, for constraints defined external to the archetype, and referenced using an instance of a *CONSTRAINT_REF*. There is one such list for each natural language in the archetype.
- Optionally, a set of one or more bindings of term definitions to term codes from external terminologies.
- Optionally, a set of one or more bindings of the external constraint definitions to external resources such as terminologies.

Specialization depth

Any given archetype occurs at some point in a hierarchy of archetypes related by specialization, where the depth is indicated by the *specialisation_depth* attribute. An archetype which is not a specialization of another has a specialization depth of 0. Term and constraint codes introduced in the ontology of specialized archetypes (i.e. which did not exist in the ontology of the parent archetype) are defined in a strict way using "." (period) markers. For example, an archetype of specialization depth 2 will use term definition codes like the following:

- "at0.0.1" — a new term introduced in this archetype, which is not a specialization of any previous term in any of the parent archetypes;
- "at0001.0.1" — a term which specializes the "at0001" term from the top parent. An intervening ".0" is required to show that the new term is at depth 2, not depth 1;
- "at0001.1.1" — a term which specializes the term "at0001.1" from the immediate parent, which itself specializes the term "at0001" from the top parent.

This systematic definition of codes enables software to use the structure of the codes to make inferences more quickly and accurately about term definitions up and down specialization hierarchies. Constraint codes on the other hand do not follow these rules, and exist in a flat code space instead.

Term and constraint definitions

Local term and constraint definitions are modelled as instances of the class *ARCHETYPE_TERM*, which is a code associated with a list of name-value pairs. For any term or constraint definition, this list shall at least include the name-value pairs for the names "text" and "description". It might also include such things as "provenance", which would be used to indicate that a term was sourced from an external terminology. The attribute term_attribute_names in *ARCHETYPE_ONTOLOGY* provides a list of attribute names used in term and constraint definitions in the archetype, including "text" and "description", as well as any others that are used in various places.

Generic types package

This package is included to confirm the semantics of the generic types used in this part of ISO 13606. Although List<T>, Set<T>, Hash<T,K>, and Interval<T> are generic types supported by many programming environments, they are not directly supported in UML. In this package, new types such as List<String> are defined using Binding Dependencies between a new Basic Type such as List<String> and a Class (LIST in this example) that defines the minimum required semantics for all Lists.

Domain-specific extension (informative)

Domain-specific classes can be added to the archetype constraint model by inheriting from the class *C_DOMAIN_TYPE*. Subclause 7.12.1 (scientific/clinical computing types) shows the general approach used to

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

add constraint classes for commonly used concepts in scientific and clinical computing, such as “ordinal”, “coded term” and “quantity”. The constraint types shown are *C_ORDINAL*, *C_CODED_TEXT* and *C_QUANTITY* which can optionally be used in archetypes to replace the default constraint semantics represented by the use of instances of *C_OBJECT* / *C_ATTRIBUTE.*

Overview of ADL

Archetype Definition Language (ADL) is a formal language for expressing archetypes. ADL uses two other syntaxes, cADL (constraint form of ADL) and dADL (data definition form of ADL) to describe constraints on data that are instances of the information model specified in Clause 7 of this part of ISO 13606.

Archetypes expressed in ADL resemble programming language files, and have a defined syntax. ADL itself is a very simple *glue* syntax, which uses two other syntaxes for expressing structured constraints and data, respectively. The cADL syntax is used to express the archetype definition, while the dADL syntax is used to express data, which appears in the language, description, ontology, and revision_history sections of an ADL archetype. The top-level structure of an ADL archetype is shown in Figure 1. The abbreviation FOPL stands for First-Order Predicate Logic.)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

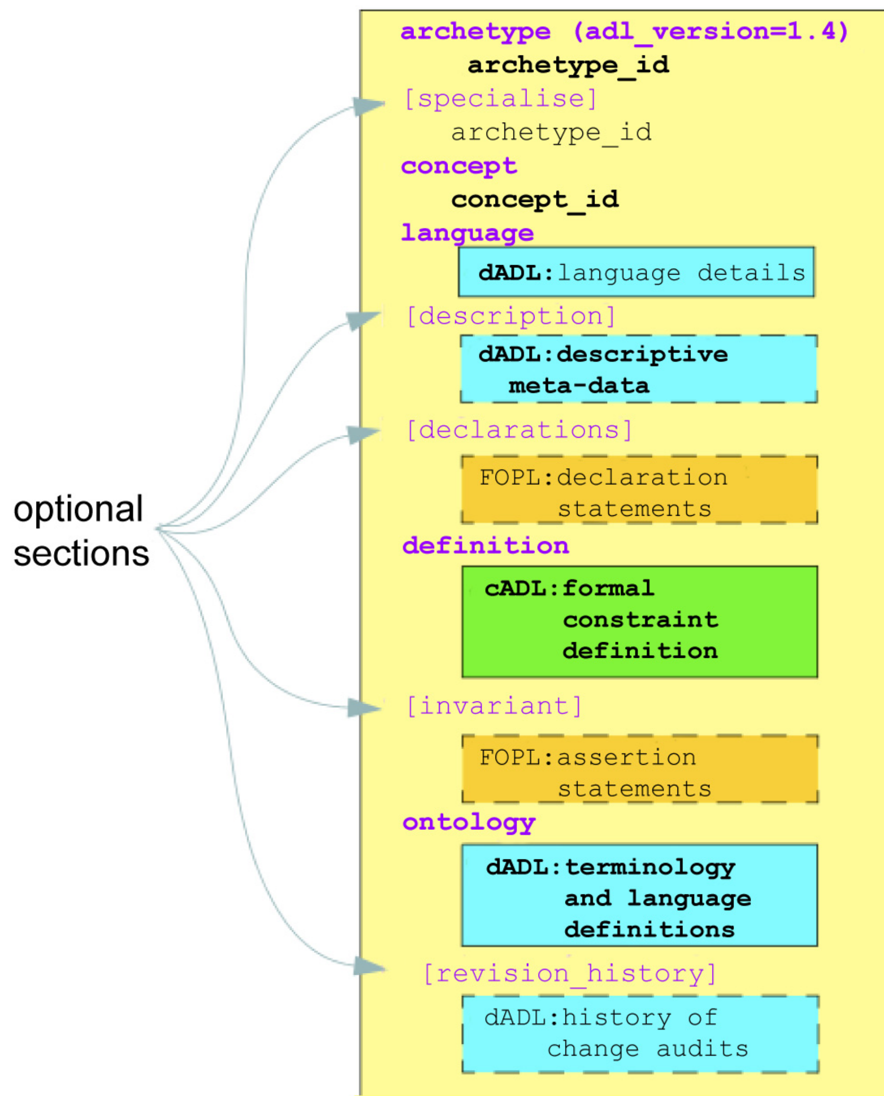


Figure 1 — ADL archetype structure

Clause 8 of this partof ISO 13606 specifies dADL, cADL, ADL path syntax, and the combined ADL syntax, archetypes and domain-specific type libraries.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

EXAMPLE

The following is an example of a simple archetype. The notion of *guitar* is defined in terms of *constraints* on a *generic* model of the concept **INSTRUMENT**. The names mentioned down the left-hand side of the definition section (INSTRUMENT, size, etc.) are alternately class and attribute names from an object model. Each block of braces encloses a specification for some particular set of instances that conform to a specific concept, such as guitar or neck, defined in terms of constraints on types from a generic class model. The leaf pairs of braces enclose constraints on primitive types such as Integer, String and Boolean.

```
archetype (adl_version=1.4)
      adl-test-instrument.guitar.draft
concept
      [at0000]                                        -- guitar
language
      original_language = <"en">
      translations = <"de", ...>
definition
      INSTRUMENT[at0000] matches {
          size matches {|60..120|    }                -- size in cm
          date_of_manufacture matches {yyyy-mm-??}
                       -- year & month ok
          parts cardinality matches {0..*} matches {
             PART[at0001] matches {                   -- neck
                material matches {[local::at0003]}    -- timber
             }
          PART[at0002] matches {                      -- body
             material matches {[local::at0003]}       -- timber

          }
       }
    }
ontology
     term_definitions = <
        [en] = <
          items = <
             ["at0000"] = <
                text = <"guitar">;
                description = <"stringed instrument">
             >
             ["at0001"] = <
                text = <"neck">;
                description = <"neck of guitar">
             >
             ["at0002"] = <
                text = <"timber">;
                description = <"straight, seasoned timber">
             >
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                  ["at0003"] = <
                     text = <"nickel alloy">;
                     description = <"frets">
                  >
               >
            >
         >
```

Clinical examples of archetypes

NOTE 1 Clause 8 of this part of ISO 13606 contains many example code fragments of ADL, which are used to illustrate specific features of the formalism. These are not to be considered normative clinical data specifications and are treated only for illustrative purposes.

It is not feasible to include full clinical examples of archetypes within this part of ISO 13606 since they are quite voluminous in document form, but the reader is encouraged to review a selection of archetypes that are available on-line from:

http://svn.openehr.org/knowledge/archetypes/dev/index.html

This site offers both an ADL representation and an html view of a wide range of archetypes. These examples include language translations and terminology bindings.

NOTE 2 The internal links given in this part of ISO 13606 will only function if http-prefixed.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Health informatics — Electronic health record communication —

Part 2: Archetype interchange specification

1 Scope

This part of ISO 13606 specifies the information architecture required for interoperable communications between systems and services that need or provide EHR data. This part of ISO 13606 is not intended to specify the internal architecture or database design of such systems.

The subject of the record or record extract to be communicated is an individual person, and the scope of the communication is predominantly with respect to that person's care.

Uses of healthcare records for other purposes such as administration, management, research and epidemiology, which require aggregations of individual people's records, are not the focus of this part of ISO 13606 but such secondary uses could also find this document useful.

This part of ISO 13606 defines an archetype model to be used to represent archetypes when communicated between repositories, and between archetype services. It defines an optional serialized representation, which may be used as an exchange format for communicating individual archetypes. Such communication might, for example, be between archetype libraries or between an archetype service and an EHR persistence or validation service.

2 Conformance

The communication of an archetype that is used to constrain part of an EHR_EXTRACT shall conform to the information model defined in Clause 7, and may optionally conform to the specification of Archetype Definition Language defined in Clause 8.

This part of ISO 13606 does not prescribe any particular representation of archetypes to be used internally within an archetype repository, server or EHR system. However, it is recommended that any representation used meet the requirements listed in Clause 6.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 639 (all parts), *Codes for the representation of names of languages*

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 10646*, Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
abstract class
⟨in Unified Modelling Language⟩ a "virtual" common parent to two or more classes

NOTE The abstract class will never be instantiated. Its value in modelling terms is to provide a container for attributes and associations that might apply to several other classes (its subclasses).

4.2
archetype instance
individual metadata class instance of an archetype model, specifying the clinical concept and the value constraints that apply to one class of record component instances in an electronic health record extract

4.3
archetype model
information model of the metadata to represent the domain-specific characteristics of electronic health record entries by specifying values or value constraints for classes and attributes in the electronic health record reference model

4.4
archetype repository
persistent repository of archetype definitions, accessed by a client-authoring tool or by a run-time component within an electronic health record service

4.5
audit trail
chronological record of activities of information system users which enables prior states of the information to be faithfully reconstructed

4.6
concept
unit of knowledge created by a unique combination of characteristics

[ISO 1087-1:2000, definition 3.2.1]

NOTE Concepts are not necessarily bound to particular languages. They are, however, influenced by the social or cultural background, often leading to different categorizations.

4.7
electronic health record
repository of information regarding the health of a subject of care, in computer processable form

NOTE Adapted from ISO/TR 20514:2005, definition 2.11.

4.8
electronic health record entry
health record data in general

EXAMPLE Clinical observations, statements, reasoning, intentions, plans or actions, without particular specification of their formal representation, hierarchical organization or particular record component class(es) that might be used to represent them.

4.9
electronic health record extract
part or all of the electronic health record of a subject of care, communicated in compliance with EN 13606

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.10
electronic health record system
system for recording, retrieving and manipulating information in electronic health records

4.11
generic
applicable to requirements or information models across health care professions, domains and countries

4.12
metadata
data that define and describes other data

[ISO/IEC 11179-3:2003, definition 3.2.18]

4.13
patient
subject of care

4.14
semantic
relating to meaning in language

4.15
semantic interoperability
ability for data shared by systems to be understood at the level of fully defined domain concepts

[ISO/TS 18308:2004, definition 3.38]

4.16
shareable electronic health record
electronic health record with a standardized information model which is independent of electronic health record systems and accessible by multiple authorized users

4.17
subject of care
person scheduled to receive, receiving or having received health care

NOTE Adapted from EN 14822-2:2005.

5 Symbols and abbreviations

ADL Archetype Definition Language

EHR Electronic Health Record

ODP Open Distributed Processing

OWL Ontology Web Language

UML Unified Modelling Language

XML Extended Markup Language

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6 Archetype representation requirements

6.1 General

This clause lists a set of formal requirements for an archetype representation. This provides the basis on which the archetype model specified in Clause 7 has been designed. It has been necessary to define these requirements within this part of ISO 13606 as there is little published work on requirements for such a model, unlike the EHR itself for which ISO/TS 18308 has been adopted.

6.2 Archetype definition, description and publication information

6.2.1 The definition of an archetype shall include the following information.

6.2.1.1 The globally unique identifier of this archetype definition.

6.2.1.2 The identifier of the repository in which this archetype originated or is now primarily held, or of the authority responsible for maintaining it. This repository will be the one in which the definitive publication status of this archetype will be managed.

6.2.1.3 The concept that best defines the overall clinical scope of instances conforming to this archetype as a whole, expressed as a coded term or as free text in a given natural language.

6.2.1.4 The health informatics domain to which this archetype applies (e.g. EHR). This will map to a set of Reference Models with which this archetype may be used.

6.2.1.5 The underlying Reference Model for which this archetype was ideally fashioned.

NOTE An archetype might be suitable for use with more than one relevant Reference Model within a given health informatics domain, but it is expected that the archetype will be optimized for one Reference Model only.

6.2.1.6 The natural language in which this archetype was originally defined, represented by its ISO 639 code. In the event of imprecise translations, this is the definitive language for interpretation of the archetype.

6.2.2 The definition of an archetype may include the following information, if applicable.

6.2.2.1 The globally unique identifier for the archetype of which this archetype is a specialization and to which it shall also conform.

6.2.2.2 The globally unique identifier of the former archetype that this definition replaces, if it not the first version of an archetype.

6.2.2.3 The reason for defining this new version of a pre-existing archetype.

6.2.2.4 The identifier of the replacement for this archetype, if it has been superseded.

NOTE It might only be possible to add this information by reference within a version-controlled repository; how this is effected is not within the scope of this part of ISO13606.

6.2.2.5 An archetype shall have one or more description sets, defining its usage and purpose. Multiple versions of this information may be included, represented in different natural languages or to inform different kinds of potential user.

6.2.3 An archetype description set shall include the following information.

6.2.3.1 The uniquely identified party responsible for providing this description set. This identification might optionally include the organization which that party represents or the authority on which he or she is acting. This may include contact information for that party.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2.3.2 The natural language in which this description set is provided, represented by its ISO 639 code.

6.2.3.3 A formal statement defining the scope and clinical purpose of this archetype, expressed as a coded term or as free text in a given natural language.

It is recommended that these criteria be expressed as coded terms to improve queries for relevant archetypes from the repository.

EXAMPLE The clinical scope and purpose might specify:

1) the principal clinical specialty for which it is intended;

2) a list of clinical, medical or procedural terms (keywords): diagnoses, acts, drugs, findings, etc.;

3) the kind of patient in whom it is intended to be used (age, gender, etc.).

6.2.4 An archetype description set may include the following information, if applicable.

6.2.4.1 A formal statement of the intended use of this archetype.

NOTE Ideally, this would be a coded expression, although a suitable terminology for this is not yet available.

6.2.4.2 A formal statement of situations in which users might erroneously believe this archetype should be used. This may also stipulate any kinds of Reference Model for which it is unsuitable.

6.2.4.3 A detailed explanation of the purpose of this archetype, including any features of particular interest or note. This may include an indication of the persons for which this definition is intended, e.g. for students. This information might be included explicitly and/or by reference (e.g. via a URL).

6.2.4.4 A description, reference or link to the published medical knowledge that has underpinned the definition of this archetype.

6.2.5 An archetype definition shall include a statement of its publication status.

An archetype definition may evolve through a series of publication states, for example an approval process, without otherwise being changed. These successive states shall be retained as part of the archetype, for audit purposes. However, the modification of the publication status of an archetype shall not itself constitute a formal revision of the identifier by which the archetype is referenced within an EHR_EXTRACT, since the constraint specification will not have been changed.

6.2.6 The publication status of an archetype shall specify the following information.

6.2.6.1 The publication status of this archetype, taken from the following list:

- test/demo;
- tentative;
- draft;
- private;
- public;
- preferred;
- deprecated.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2.6.2 The date when this particular publication status applied.

NOTE The first instance of a publication status for this archetype will also be the date when it was first composed.

6.2.6.3 The unique identifier of the party committing this archetype to the repository and thereby asserting this publication status. This identification might optionally include the organization which that party represents.

6.2.6.4 The unique identifier of the body authorizing this change in publication status.

6.2.6.5 The date when it is anticipated that the present publication status, and the archetype content itself, ought to be reviewed to confirm that it remains clinically valid.

6.3 Archetype node constraints

6.3.1 General

An archetype definition will include a specification of the hierarchical schema to which instances of data (e.g. EHR data) shall conform. This schema defines the hierarchical organization of a set of nodes, the relationships between them, and constraints on the permitted values of attributes and data values. These will also conform to the underlying Reference Model(s) for which this definition is applicable.

6.3.2 Archetype node references

6.3.2.1 Any node in the archetype hierarchy might be defined explicitly or by reference, or be specified to be part or whole of a pre-existing archetype.

6.3.2.2 A reference to a pre-existing archetype or archetype fragment may be explicit, by specifying the archetype identifier, and optionally the archetype node of the archetype fragment insertion point.

6.3.2.3 A reference to an archetype fragment may be internal to (i.e. part of) the current archetype.

6.3.2.4 An archetype node may be specified to be one of a set of possible archetypes, by defining an explicit list of candidates and/or by specifying a set of constraints on any of the attributes of an archetype definition.

6.3.2.5 In addition to specifying one or more archetype fragments by reference or constraint, it shall be possible to include an explanation of the rationale for incorporating that specification at the given point in the current archetype hierarchy.

6.3.3 Specification of an archetype node

6.3.3.1 The specification of an archetype node (if not by reference) shall include the following information.

6.3.3.2 An internally unique identifier of this archetype node. When combined with the globally unique identifier of this archetype definition, it shall be a globally unique reference to the node itself.

6.3.3.3 The class in the instance hierarchy, mapping to the underlying Reference Model for which this archetype was ideally fashioned, that shall be instantiated in order to conform to this archetype node. For an EHR hierarchy conforming to this part of ISO 13606, this class shall be specified using one of the following values:

— FOLDER

— COMPOSITION

— SECTION

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— ENTRY

— CLUSTER

— ELEMENT

The number of occurrences, expressed as a range, that may be instantiated corresponding to this archetype node within an instance hierarchy.

6.3.3.4 Other constraints and rules may optionally be specified to govern the creation of instances corresponding to this archetype node.

6.3.3.5 Constraint rules may be expressed as logical conditions, and may include reference to environment parameters such as the current time or location or participants, or be related to the (pre-existing) values of other nodes in the instance hierarchy. Constraint rules might be used to represent the relationship between EHR data and workflow or care pathway processes.

6.3.3.6 Constraint rules may be expressed as inclusion or exclusion criteria.

6.3.3.7 Any constraint rule specification shall identify the formalism (including version) in which it is expressed (e.g. ADL, OWL).

6.3.4 Binding archetype nodes to terms

6.3.4.1 Every node of an archetype schema hierarchy shall be associated with at least one clinical term, which most accurately expresses the intended concept to be represented by that node on instantiation in the corresponding instance hierarchy. This term will usually be included or referenced within the instance.

6.3.4.2 Any node of an archetype may be mapped to any number of additional concepts, terms and synonyms from terminology systems, to support either the interrogation of the archetype repository or that of the corresponding instances.

6.3.4.3 Any concept mapping term or text shall specify the purpose that this mapping serves from the following list of values:

— principal concept;

— term binding;

— synonym;

— language translation.

6.3.4.4 Any reference to a coded term shall include the code, the rubric and the coding system (including version) from which the code and rubric have been taken. In addition, it shall be possible to specify the natural language in which this term was mapped, or in which a translation is expressed.

6.3.5 Attribute and association constraints

An archetype node may specify constraints on any attributes or associations that correspond to the attributes and associations of that node in the underlying Reference Model.

These constraints may pre-determine or restrict some or all of the contextual information that is included within the corresponding instance, as represented within the Reference Model. Context information, such as the person to whom a particular observation or inference relates, is formally represented in most generic EHR-like models to facilitate safe querying and retrieval, even if that information might be inferred from the archetype name or an axis within a terminology system used for the data value. Some archetypes or fragments will pre-determine the values of some of these, which shall be capable of specification within the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

archetype definition (for example, to constrain the subject of information to be a relative of the patient, and not the patient, in an archetype for family history). The applicability of any aspect of context to a given archetype node will be determined by the context attributes in the corresponding node of the target EHR Reference Models. For example, in the EN 13606 Reference Model, the subject of information is represented at the Entry level. There is significant commonality within the set of context information between reference models (for example between EN 13606 and HL7 CDA Release 2), and, ideally, a common set of labels for these should be adopted to permit relevant context constraints to be applied to more than one Reference Model.

6.3.6 Further information

6.3.6.1 For any given Reference Model attribute or association, it shall be possible to specify the following information.

6.3.6.2 The name of the attribute or association, mapping to the underlying Reference Model for which this archetype was ideally fashioned, to which this constraint applies.

6.3.6.3 A common label for this aspect of context. For EHR archetypes, this value is to be taken from a (provisional) list of values as specified in Table 1.

6.3.6.4 For a given Reference Model, whether the inclusion within a valid EHR instance of an attribute or association corresponding to this aspect of context is mandatory.

6.3.6.5 The number of instances (expressed as a range) corresponding to this aspect of context that may be instantiated.

6.3.6.6 If multiple instances are permitted, it shall be possible to specify if these are to be represented as an ordered or unordered list.

6.3.6.7 If multiple instances are permitted, it shall be possible to specify if the corresponding data values (of leaf nodes or attributes) shall be unique.

6.3.6.8 Constraints may be specified for the data values of leaf nodes or leaf attributes.

6.3.6.9 Other constraints and rules may optionally be specified to govern the creation of instances corresponding to a Reference Model attribute or association.

6.4 Data value constraints

6.4.1 It shall be possible to specify constraints and rules for the data values of leaf nodes in the Reference Model hierarchy, or for any other attributes of any archetype node.

6.4.2 It shall be possible to specify the following data value constraint information.

6.4.2.1 If the data value is permitted to have a null value, and optionally to specify a reason (e.g. to specify a null flavour value).

6.4.2.2 If the constraint or rule is an inclusion or exclusion criterion.

6.4.2.3 The formalism (including version) in which this constraint specification is represented.

6.4.2.4 The intended fixed (prescribed) value for conforming instances.

6.4.2.5 The intended default value for conforming instances.

6.4.2.6 A list of permitted candidate values for conforming instances (i.e. to be a subset of those values legally permissible in the underlying Reference Model).

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.4.3 For quantity data types, it shall be possible to specify:

- a range within which values for conforming instances shall lie;
- a range within which values are considered clinically exceptional or critical;
- the intended measurement units for conforming instances.

6.4.4 For date and time data types, it shall be possible to specify:

- a range within which values for conforming instances shall lie;
- the intended measurement units for conforming instances.

6.4.5 For textual data types, it shall be possible to specify:

- a string pattern defining a range of possible values;
- the intended coding scheme to be used for conforming instances.

6.4.6 Constraint rules may be expressed as logical conditions, and may include reference to environment parameters such as the current time or location or participants, or be related to the (pre-existing) values of other nodes in the instance hierarchy.

6.4.7 The reference to a pre-existing value shall specify that instance precisely and unambiguously. For example, it may be necessary to include a reference to:

- the archetype identifier;
- the archetype node identifier;
- the attribute or association name;
- the occurrence in the instance hierarchy, for example:
  - first;
  - most-recent;
  - any;
  - n ordered by y (the nth element of a set of instances ordered on y);
  - highest value;
  - lowest value;
  - one or more instances within a (definable) recent time interval;
- the intended relationship between this specified instance value and the data value being constrained, for example:
  - the same value as;
  - a subset or substring of;
  - greater than, greater than or equal to, less than, less than or equal to;

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— earlier than, later than, etc.;

— if ... then...;

— shall not be the same as.

6.4.8 These relative constraints may be nested, and include logical or set operators in order to represent compound rules.

6.5 Profile in relation to EN 13606-1 Reference Model

This set of tables defines a generic set of contexts that are used within the EN 13606-1 EHR Reference Model to specify how observed values, intentions and inferences relate to the subject of care or to other parts of that subject's EHR. Table 1 summarises these areas of context, and Table 2 maps these to the EN 13606 Reference Model.

Table 1 — List of context areas and elements

Context area	Context element	Description
Meaning	Meaning	Formal concept defining the EHR object to be instantiated at a given node in the EHR.
Subject of information	Subject of information	Rules for the permitted values of the subject of information, for example to specify that the subject of information may not be the patient or shall be a genetic relative.
Act status	Act status	Rules for the permitted values of the act status, for example to specify that the act status shall be a planned/ordered activity.
Temporal relationship	Temporal relationship	Specification of the temporal relationship of the information to the time of its recording, e.g. former, ongoing, future.
Structure	Structure	The spatial structure to be used to represent (render) a data structure, e.g. list, table, tree.
Observation time	Observation time	Rules for the permitted values of the observed/intended time (or time interval) for this observation set.
Link		Most links (13606 LINK, HL7 Act Relationship) are defined on an *ad hoc* basis within individual instances of EHR data. However, there may be times when particular kinds of clinical data shall always have certain links defined. For example, certain care acts might always need to reference a pre-existing consent document in the EHR, or a pre-existing clinical finding that justifies the activity.
	Nature	The kind of link (CEN nature, HL7 ActRelationship class code) that shall or may be composed.
	Role	The role that the target plays in this link.
	Follow_link	Rules for when the link is required to be followed when the source or target component is retrieved from an EHR system (CEN follow_link, HL7 separatableInd).
Participation		If certain participants may or shall be defined in the EHR node instance.
	Function	The functional role that shall be held by the participant.
	Mandatory attestation	If this participant is required to attest the instance; this requirement does not specify if this node shall be individually attested or if it may be attested as part of a larger collection of EHR nodes, for example at the document level.
	Attestation reason	Specifies a fixed reason for the attestation, e.g. if the attestation is performing a particular legal function.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 2 — EN 13606 Reference Model — Context profile

13606 RM class	Context area	Corresponding 13606 RM attribute
FOLDER	Meaning	Meaning
	Link	LINK
COMPOSITION	Meaning	Meaning
	Link	LINK
	Participation	composer other_participations
SECTION	Meaning	Meaning
	Link	LINK
ENTRY	Meaning	Meaning
	Link	LINK
	Participation	other_participations
	Subject of information	subject_of_information
	Act status	act_status
CLUSTER	Meaning	Meaning
	Link	LINK
	Structure	structure_type
	Observation time	obs-time
ELEMENT	Meaning	Meaning
	Link	LINK
	Observation time	obs-time

7 Archetype model

7.1 Introduction

7.1.1 General

The model is presented using a constrained form of UML diagrams, described below in a UML profile.

7.1.2 UML profile

The classes of the model together with their associations and inheritances are grouped into packages, and these are shown in a number of separate diagrams hereafter. Package boundaries are shown as blue lines, with the name of the package, also shown in blue, in the top left box of the package outline.

Class boxes usually have three compartments.

The top compartment contains the name of the class in upper case, and may also show the owning class in parentheses when the class belongs to a package other than the package that is the subject of the diagram. Some diagrams also show class constraints in the top compartment.

The second compartment, if present, contains attributes, showing attribute name, attribute type and multiplicity. Multiplicity may also be further qualified by the “ordered” marker. Attribute names are shown in lower case. Attribute types are shown in title case if the type is one of the basic types, and in upper case if the type is of another class.

The third compartment, if present, contains operations, showing operation name, return type and parameters passed. Operation names and types follow the same casing rules as attributes.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A class box with only two compartments has a class name and attributes; a class box with only one compartment just has a name.

Colour itself has no significance, but is used to enhance readability. Inheritance lines are shown in black, and association lines in maroon. Colour fills of class boxes are sometimes used to highlight particular groupings of classes. Grey class boxes are used to indicate that the class details are shown on another diagram.

Associations between classes are always "single ended" with the association name and multiplicity placed at the far end of the line. If a double-ended association is required, it is shown as two single-ended associations, one in each direction, between the two classes. This restriction has been applied to make it possible to document each association automatically, as though it were an attribute of the near-end class. Navigation arrows are not used.

7.1.3 Detailed documentation of the model

The order of documentation is by package and, within package, by class.

Each class has a starting section showing the owning package, any inheritance, inner elements, and any internal model documentation, and is followed by up to four tabular sections for:

a) attributes;

b) attributes derived from associations;

c) operations;

d) constraints.

The associations themselves are shown in the diagrams using UML notation, but are documented as derived associations using the following transformations:

Association far-end name	becomes	Attribute name
Association far-end class	becomes	Attribute type

Association multiplicity	**Generates**	**Container type**	**and**	**Attribute optionality**	**Original multiplicity**
0..*		Set<far end CLASS>		0..1	0..*
0..* {ordered}		List<far end CLASS>		0..1	0..* {ordered}
1..*		Set<far end CLASS>		1	1..*
1..* {ordered}		List<far end CLASS>		1	1..* {ordered}
*		Set<far end CLASS>		0..1	*
0..1		Not a container		0..1	N/A
1		Not a container		1	N/A

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.1.4 Package structure

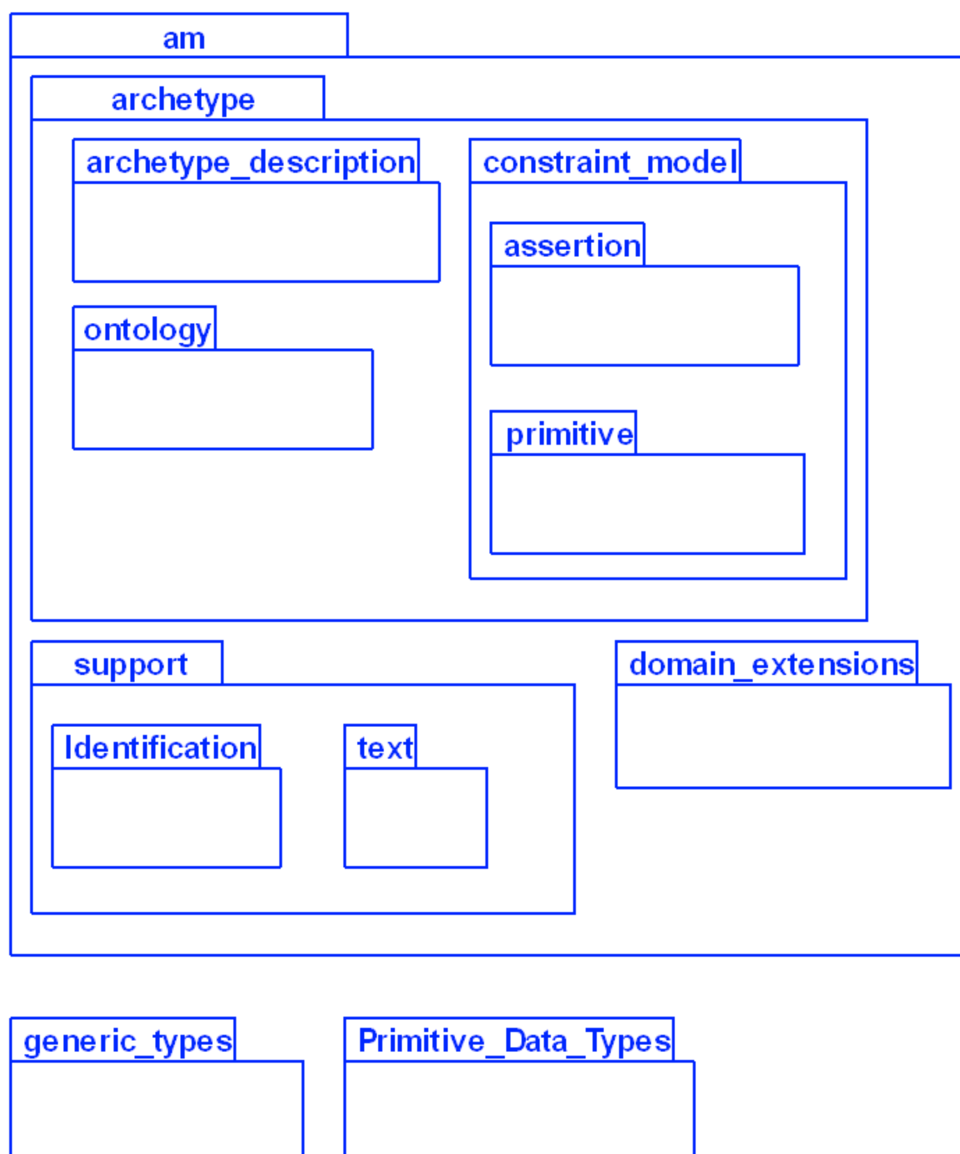


Figure 2 — Package structure

The overall Archetype Model shown in Figure 3 and Figure 4 defines the generic representation of archetypes for interoperability and communication purposes.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.2 Overview

7.2.1 General

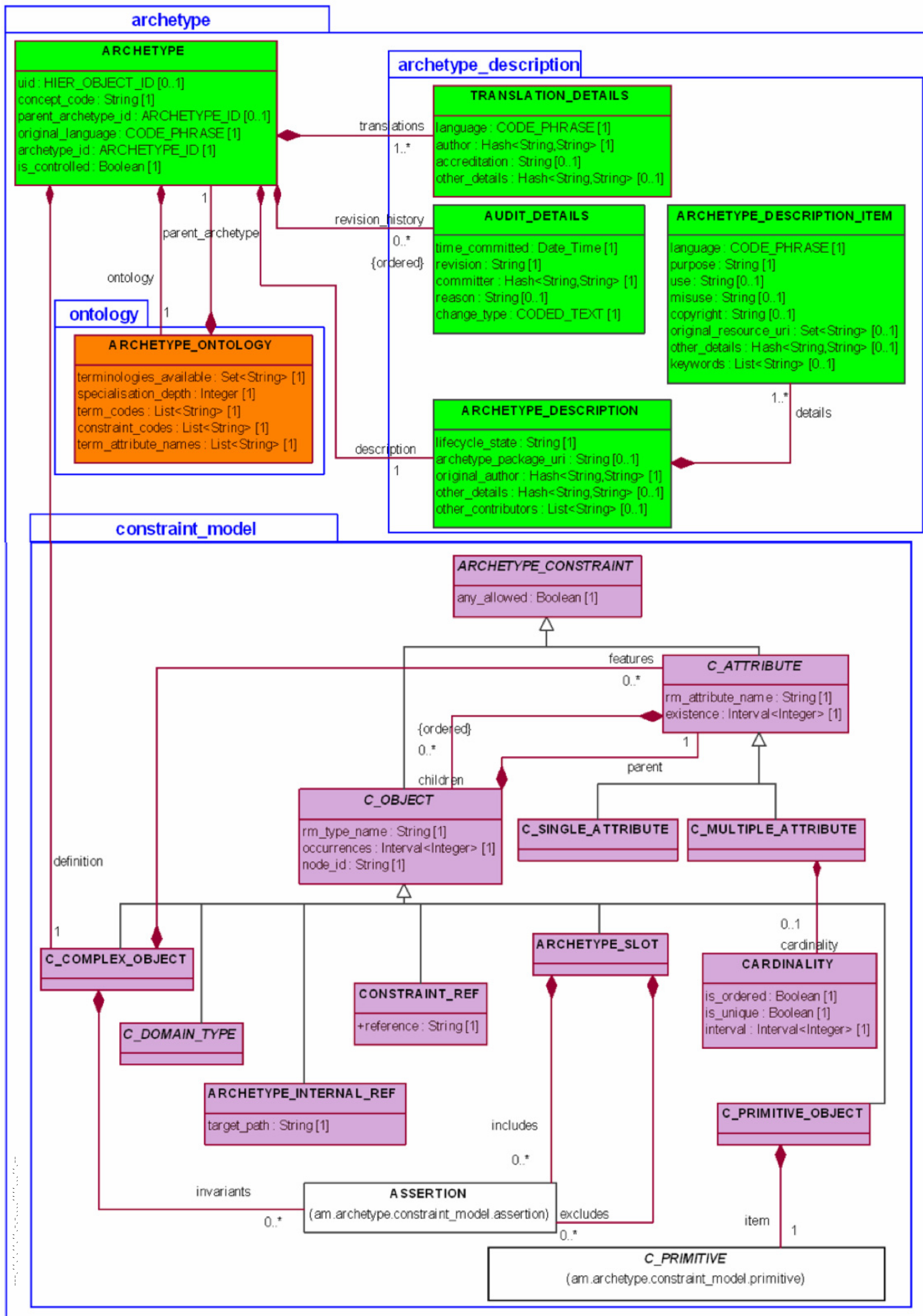


Figure 3 — Overview of the main part of the Archetype Model — Part 1

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

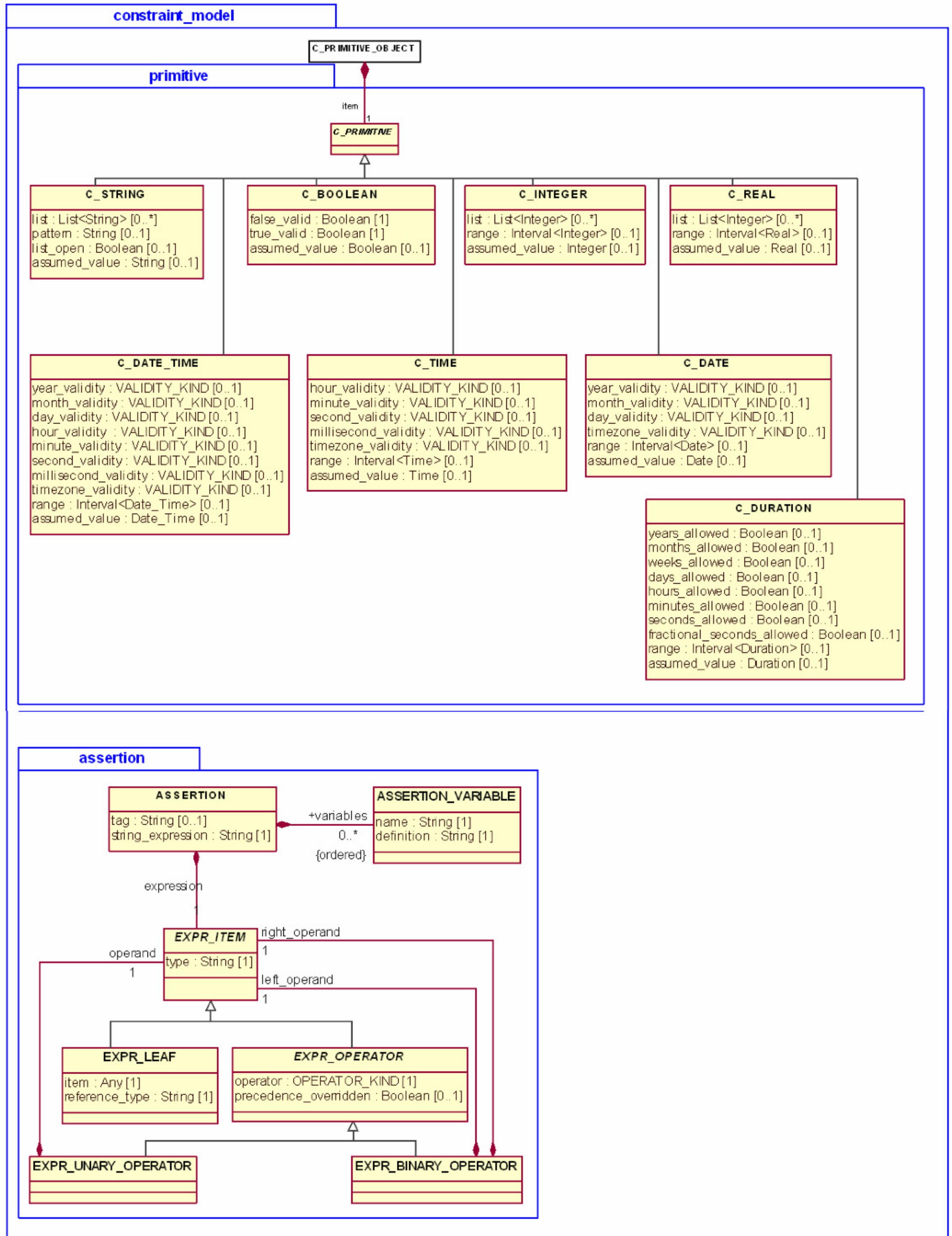


Figure 4 — Overview of the Archetype Model — Part 2

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Documentation derived from the UML model

NOTE The following list of package and class names is included to provide a high-level overview of the model as organized within this subclause.

am

archetype

ARCHETYPE

archetype_description

ARCHETYPE_DESCRIPTION

ARCHETYPE_DESCRIPTION_ITEM

AUDIT_DETAILS

TRANSLATION_DETAILS

constraint_model

ARCHETYPE_CONSTRAINT

ARCHETYPE_INTERNAL_REF

ARCHETYPE_SLOT

C_ATTRIBUTE

C_COMPLEX_OBJECT

C_DOMAIN_TYPE

C_MULTIPLE_ATTRIBUTE

C_OBJECT

C_PRIMITIVE_OBJECT

C_SINGLE_ATTRIBUTE

CARDINALITY

CONSTRAINT_REF

assertion

ASSERTION

ASSERTION_VARIABLE

EXPR_BINARY_OPERATOR

EXPR_ITEM

EXPR_LEAF

EXPR_OPERATOR

EXPR_UNARY_OPERATOR

primitive

C_BOOLEAN

C_DATE

C_DATE_TIME

C_DURATION

C_INTEGER

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

- C_PRIMITIVE
- C_REAL
- C_STRING
- C_TIME

- ontology
  - ARCHETYPE_ONTOLOGY
  - ARCHETYPE_TERM

- domain_extensions
  - C_CODED_TEXT
  - C_ORDINAL
  - C_QUANTITY
  - C_QUANTITY_ITEM
  - ORDINAL

- support
  - OPERATOR_KIND
  - VALIDITY_KIND

- Identification
  - ARCHETYPE_ID
  - HIER_OBJECT_ID
  - OBJECT_ID
  - TERMINOLOGY_ID

- text
  - CODE_PHRASE
  - CODED_TEXT
  - TERM_MAPPING
  - TEXT

7.2.2 Package :: am

Inner elements	
Name	**Type**
archetype	Package
domain_extensions	Package
support	Package

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3 The archetype package

7.3.1 General

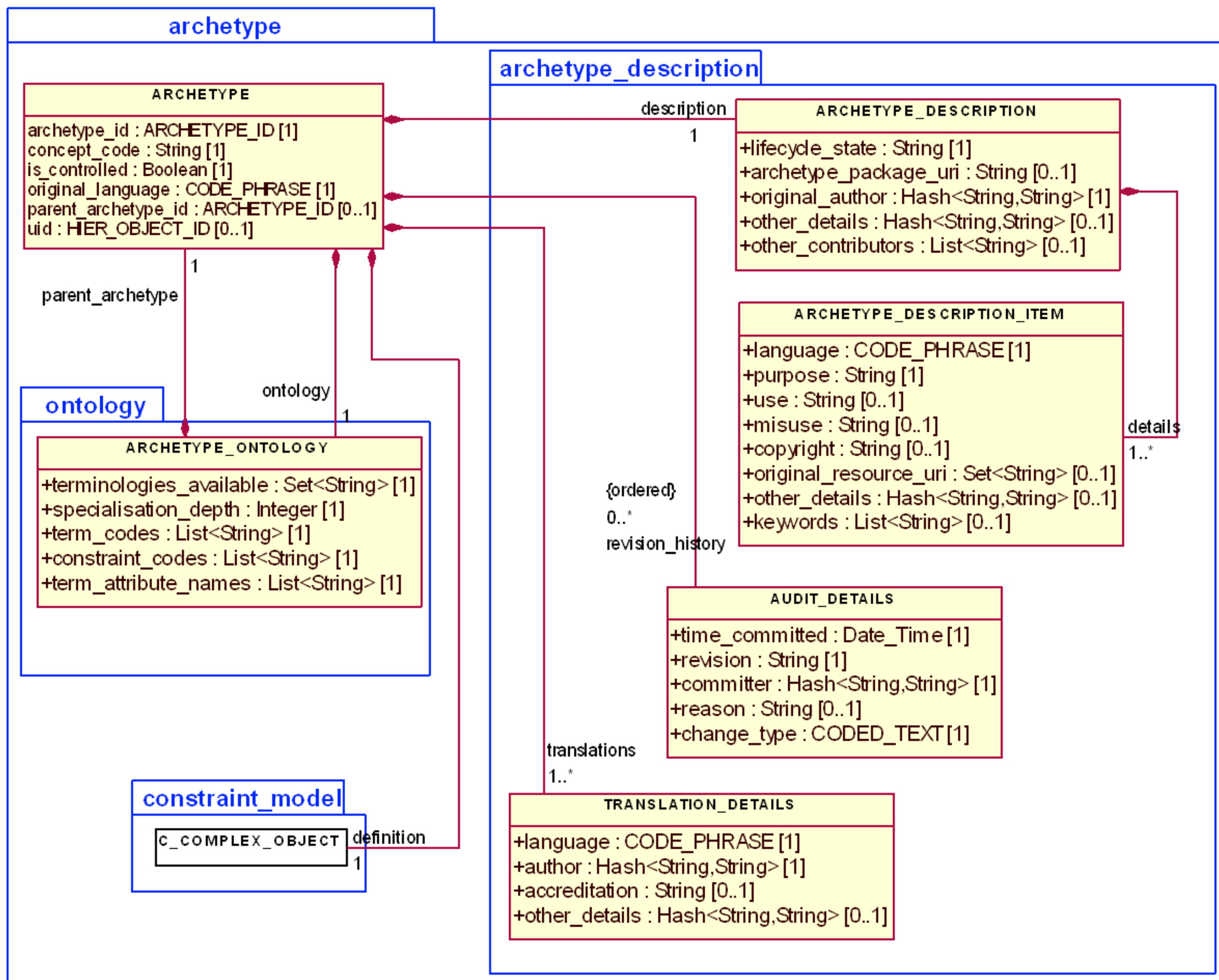


Figure 5 — Archetype package

7.3.2 Package :: archetype

Inner elements	
Name	**Type**
ARCHETYPE	Class
archetype_description	Package
constraint_model	Package
ontology	Package

Package: archetype
Class archetype

The main class of the archetype package.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
archetype_id : ARCHETYPE_ID	1	--	Multi-axial identifier of this archetype in archetype space.
concept_code : String	1	--	Normative meaning of the archetype as a whole.
is_controlled : Boolean	1	--	True if this archetype is under change control in which case revision history is created.
original_language : CODE_PHRASE	1	--	Language in which this archetype was initially authored.
parent_archetype_id : ARCHETYPE_ID	0..1	--	Identifier of the specialization parent of this archetype.
uid : HIER_OBJECT_ID	0..1	--	OID identifier of this archetype.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
revision_history : List<AUDIT_DETAILS>	0..1	0..* ordered	Revision history of the archetype; only required if is_controlled = True
description : ARCHETYPE_DESCRIPTION	1	--	Description and lifecycle information of the archetype.
ontology : ARCHETYPE_ONTOLOGY	1	--	Ontology of the archetype.
definition : C_COMPLEX_OBJECT	1	--	Root node of this archetype.
translations : Set<TRANSLATION_DETAILS>	1	1..*	List of details for each natural translation included in this archetype.

Constraints	
Name	**Expression**
revision_history_validity	inv: is_controlled implies (revision_history <> Void and revision_history.is_empty)
archetype_id_validity	inv: archetype_id <> Void
description_exists	inv: description <> Void
ontology_exists	inv: ontology <> Void
definition_exists	inv: definition <> Void
uid_validity	inv: uid <> Void implies not uid.is_empty
original_language_valid	inv: original_language <> Void and translations. language <> Void and terminology_service.code_set('languages').has(original_language)
has_parent	post: is_specialised implies parent_archetype_id <>Void

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.4 The archetype description package

7.4.1 General

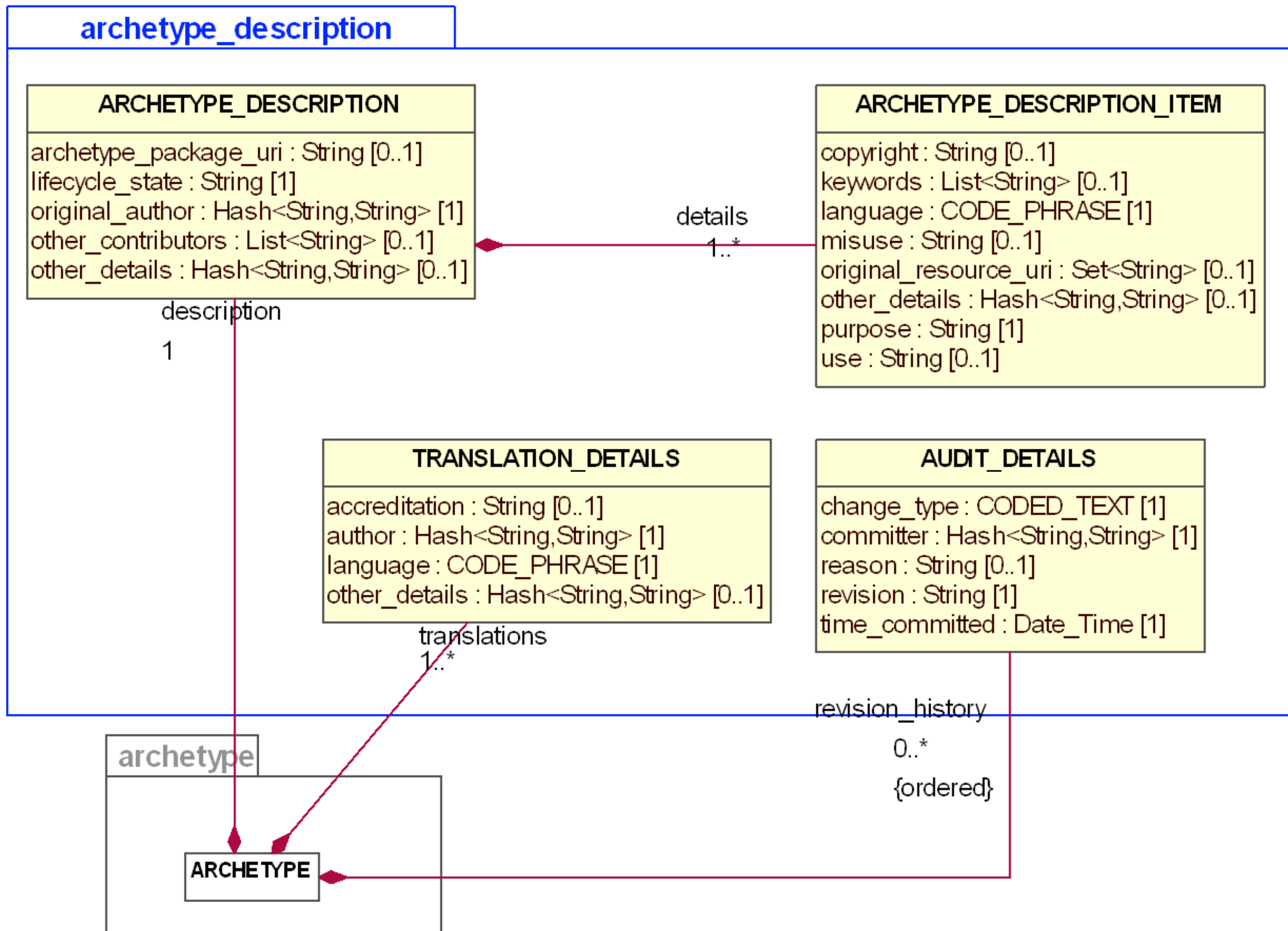


Figure 6 — Archetype description package

7.4.2 Package :: archetype_description

The "metadata" of an archetype.

Inner elements	
Name	**Type**
ARCHETYPE_DESCRIPTION	Class
ARCHETYPE_DESCRIPTION_ITEM	Class
AUDIT_DETAILS	Class
TRANSLATION_DETAILS	Class

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: archetype_description
Class AUDIT_DETAILS

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
change_type : CODED_TEXT	1	--	Type of change.
committer : Hash<String,String>	1	--	Identification details of the author of the main content of this archetype, expressed as a list of name-value pairs.
reason : String	0..1	--	Natural language reason for change.
revision : String	1	--	Revision corresponding to this change.
time_committed : Date_Time	1	--	Date/time of this change.

Constraints	
Name	**Expression**
committer_validity	inv: committer <> Void and not committer.is_empty
committer_organisation_validity	inv: committer_organisation <> Void implies not committer_organisation.is_empty
time_committed_exists	inv: time_committed <> Void
reason_valid	inv: reason <> Void implies not reason.is_empty
revision_valid	inv: revision <> Void and not revision.is_empty
change_type_exists	inv: change_type <> Void and terminology_service.terminology('openehr').codes_for_group_name('audit_change_type', 'en').has(change_type.defining_code)

Package: archetype_description
Class ARCHETYPE_DESCRIPTION

Defines the descriptive metadata of an archetype.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
archetype_package_uri : String	0..1	--	URI of package to which this archetype belongs.
lifecycle_state : String	1	--	Lifecycle state of the archetype: initial, submitted, experimental, awaiting_approval, approved, superseded, obsolete.
original_author : Hash<String,String>	1	--	Original author of this archetype, expressed as a list of name-value pairs.
other_contributors : List<String>	0..1	--	Names of other contributors to the archetype.
other_details : Hash<String,String>	0..1	--	Additional non-language-sensitive archetype metadata, as a list of name-value pairs.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
details : Set<ARCHETYPE_DESCRIPTION_ITEM>	1	1..*	The descriptive metadata of an archetype.

Constraints	
Name	**Expression**
original_author_validity	inv: original_author <> Void and not original_author.is_empty
details_exists	inv: details <> Void and not details.is_empty
original_author_organisation_validity	inv: original_author_organisation <> Void implies not original_author_organisation.is_empty
language_validity	inv: details->for_all(d| parent_archetype.languages_available.has(d.language))
parent_archetype_valid	inv: parent_archetype <> Void and parent_archetype.description = Current

Package: archetype_description
Class ARCHETYPE_DESCRIPTION_ITEM

Language-specific detail of archetype description. When an archetype is translated for use in another language environment, each ARCHETYPE_DESCRIPTION_ITEM needs to be copied and translated into the new language.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
copyright : String	0..1	--	Optional copyright statement for the archetype as a knowledge resource.
keywords : List<String>	0..1	--	Keywords by which this Archetype may be referenced.
language : CODE_PHRASE	1	--	The localized language in which the items in this description item are written.
misuse : String	0..1	--	Description of any contexts in which it should not be used.
original_resource_uri : Set<String>	0..1	--	URI of original clinical document(s) or description of which archetype is a formalization, in the language of this description item.
other_details : Hash<String,String>	0..1	--	Additional language-sensitive archetype metadata, as a list of name-value pairs.
purpose : String	1	--	Purpose of the archetype.
use : String	0..1	--	Description of the uses of the archetype, i.e. contexts in which it could be used.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Constraints	
Name	**Expression**
use_valid	inv: use <> Void implies not use.is_empty
language_valid	inv: language <> Void and code_set('languages').has(language)
misuse_valid	inv: misuse <> Void implies not misuse.is_empty
copyright_valid	inv: copyright <> Void implies not copyright.is_empty
purpose_exists	inv: purpose <> Void and not purpose.is_empty

Package: archetype_description
Class TRANSLATION_DETAILS

Class providing details of a natural language translation.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
accreditation : String	0..1	--	Accreditation of translator, e.g a national translator's association id.
author : Hash<String,String>	1	--	Translator name and other demographic details, expressed as a list of name-value pairs.
language : CODE_PHRASE	1	--	Language of translation.
other_details : Hash<String,String>	0..1	--	Any other metadata.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5 The constraint model package

7.5.1 General

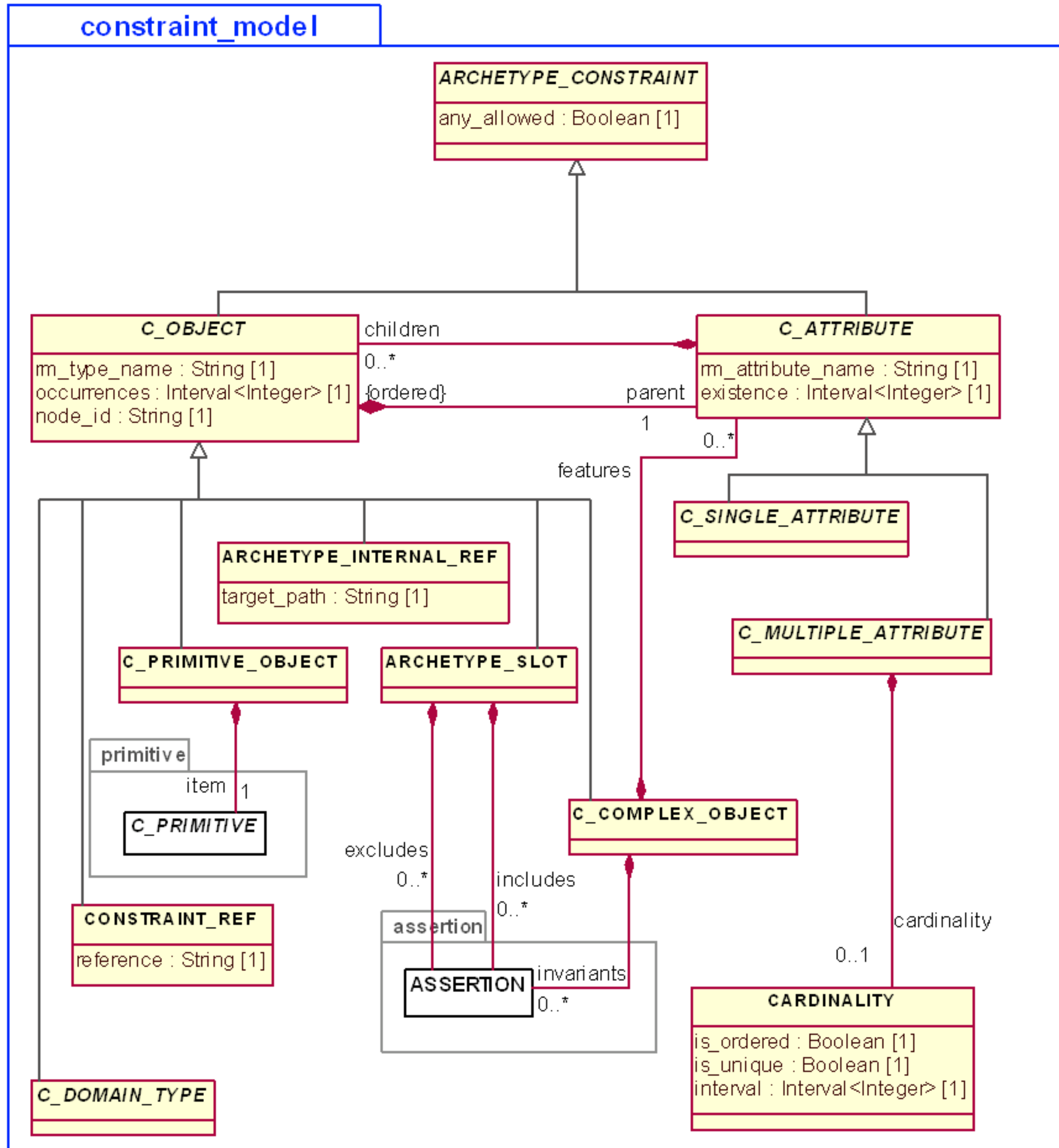


Figure 7 — Constraint model package

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5.2 Package :: constraint_model

Inner elements	
Name	**Type**
ARCHETYPE_CONSTRAINT	Class
ARCHETYPE_INTERNAL_REF	Class
ARCHETYPE_SLOT	Class
C_ATTRIBUTE	Class
C_COMPLEX_OBJECT	Class
C_DOMAIN_TYPE	Class
C_MULTIPLE_ATTRIBUTE	Class
C_OBJECT	Class
C_PRIMITIVE_OBJECT	Class
C_SINGLE_ATTRIBUTE	Class
CARDINALITY	Class
CONSTRAINT_REF	Class
assertion	Package
primitive	Package

Package: constraint_model
Class ARCHETYPE_CONSTRAINT{Abstract}

Direct subclassifiers:
C_OBJECT, C_ATTRIBUTE

Defines common constraints for any archetypeable class in any reference model.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
any_allowed : Boolean	1	--	True if no additional constraints are defined in the archetype, beyond those defined in the underlying Reference Model

Package: constraint_model
Class C_ATTRIBUTE{Abstract}

```
ARCHETYPE_CONSTRAINT
   |
   +--C_ATTRIBUTE
```

Direct subclassifiers:
C_MULTIPLE_ATTRIBUTE, C_SINGLE_ATTRIBUTE

Abstract model of constraint on any kind of attribute node.

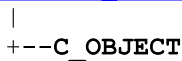
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
existence : Interval<Integer>	1	--	Constraint on every attribute, regardless of whether it is singular or of a container type, which indicates whether its target object exists or not (i.e. is mandatory or not).
rm_attribute_name : String	1	--	Reference model attribute within the enclosing type represented by a C_OBJECT.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
children : List<C_OBJECT>	0..1	0..* ordered	Child C_OBJECT nodes. Each such node represents a constraint on the type of this attribute in its reference model.

Constraints	
Name	**Expression**
existence_set	inv: existence <> Void and (existence.lower >= 0 and existence.upper <= 1)
rm_attribute_name_valid	inv: rm_attribute_name <> Void and not rm_attribute_name.is_empty
Children_validity	inv: any_allowed xor children <> Void

Package: constraint_model
Class C_OBJECT{Abstract}



Direct subclassifiers:
ARCHETYPE_INTERNAL_REF, C_PRIMITIVE_OBJECT, C_COMPLEX_OBJECT, ARCHETYPE_SLOT, CONSTRAINT_REF, C_DOMAIN_TYPE

Abstract model of constraint on any kind of object node.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
node_id : String	1	--	Semantic id of this node, used to differentiate sibling nodes of the same type. (Previously called “meaning”). Each node_id shall be defined in the archetype ontology as a term code.
occurrences : Interval<Integer>	1	--	Occurrences of this object node in the data, under the owning attribute. Upper limit may only be greater than 1 if owning attribute has a cardinality of more than 1.
rm_type_name : String	1	--	Reference model type to which this node corresponds.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
Parent : C_ATTRIBUTE	1	--	C_ATTRIBUTE that owns this C_OBJECT.

Constraints	
Name	**Expression**
rm_type_name_valid	inv: rm_type_name <> Void and not rm_type_name.is_empty
node_id_valid	inv: node_id <> Void and not node_id.is_empty

Package: constraint_model
Class CARDINALITY

Expresses constraints on the cardinality of container objects which are the values of multiple-valued attributes, including uniqueness and ordering, providing the means to state that a container acts like a logical list, set or bag. The cardinality cannot contradict the cardinality of the corresponding attribute within the relevant reference model.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
interval : Interval<Integer>	1	--	The interval (range) of this cardinality.
is_ordered : Boolean	1	--	True if the members of the container attribute to which this cardinality refers are ordered.
is_unique : Boolean	1	--	True if the members of the container attribute to which this cardinality refers are unique.

Constraints	
Name	**Expression**
Validity	inv: not interval.lower_unbounded

Package: constraint_model
Class CONSTRAINT_REF

```
C_OBJECT
    |
    +--CONSTRAINT_REF
```

Reference to a constraint described in the same archetype, but outside the main constraint structure. This is used to refer to constraints expressed in terms of external resources, such as constraints on terminology value sets.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
reference : String	1	--	Reference to a constraint in the archetype local ontology.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Constraints	
Name	**Expression**
Consistency	inv: not any_allowed
reference_valid	inv: reference <> Void and not reference.is_empty and archetype.ontology.has_constraint(reference)

Package: constraint_model
Class ARCHETYPE_INTERNAL_REF

```
C_OBJECT
    |
    +--ARCHETYPE_INTERNAL_REF
```

A constraint defined by proxy, using a reference to an object constraint defined elsewhere in the same archetype.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
target_path : String	1	--	Reference to an object node using archetype path notation.

Constraints	
Name	**Expression**
Consistency	inv: not any_allowed
target_path_valid	inv: target_path <> Void and not target_path.is_empty and ultimate_root.has_path(target_path)

Package: constraint_model
Class ARCHETYPE_SLOT

```
C_OBJECT
    |
    +--ARCHETYPE_SLOT
```

Constraint describing a “slot” where another archetype may occur.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
excludes : Set<ASSERTION>	0..1	0..*	List of constraints defining other archetypes that cannot be included at this point.
includes : Set<ASSERTION>	0..1	0..*	List of constraints defining other archetypes that could be included at this point.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Constraints	
Name	**Expression**
includes_valid	inv: includes <> Void implies not includes.is_empty
excludes_valid	inv: excludes <> Void implies not excludes.is_empty
Validity	inv: any_allowed xor includes <> Void or excludes <> Void

Package: constraint_model
Class C_SINGLE_ATTRIBUTE

```
C_ATTRIBUTE
   |
   +--C_SINGLE_ATTRIBUTE
```

Concrete model of constraint on a single-valued attribute node. The meaning of the inherited children attribute is that they are alternatives.

Package: constraint_model
Class C_MULTIPLE_ATTRIBUTE

```
C_ATTRIBUTE
   |
   +--C_MULTIPLE_ATTRIBUTE
```

Abstract model of constraint on any kind of attribute node.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
cardinality : CARDINALITY	0..1	--	Cardinality of this attribute constraint, if it constrains a container attribute.

Constraints	
Name	**Expression**
members_valid	inv: members <> Void and members->for_all(co: C_OBJECT | co.occurrences.upper <= 1)
cardinality_validity	inv: cardinality <> Void

Package: constraint_model
Class C_DOMAIN_TYPE{Abstract}

```
C_OBJECT
   |
   +--C_DOMAIN_TYPE
```

Direct subclassifiers:
C_ORDINAL, C_QUANTITY, C_CODED_TEXT

Abstract parent type of domain-specific constrainer types, to be defined in external packages.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: constraint_model
Class C_COMPLEX_OBJECT

C_OBJECT

Constraint on complex objects, i.e. any object that consists of other object constraints.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
invariants : Set<ASSERTION>	0..1	0..*	Invariant statements about this object. Statements are expressed in first-order predicate logic, and usually refer to at least two attributes.
features : Set<C_ATTRIBUTE>	0..1	0..*	List of constraints on attributes of the reference model type represented by this object.

Constraints	
Name	**Expression**
attributes_valid	inv: any_allowed xor (attributes <> Void and not attributes.is_empty)
invariants_valid	inv: invariants <> Void implies not invariants.is_empty
invariant_consistency	inv: any_allowed implies invariants = Void

Package: constraint_model
Class C_PRIMITIVE_OBJECT

C_OBJECT

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
item : C_PRIMITIVE	1	--	Object actually defining the constraint.

Constraints	
Name	**Expression**
item_exists	inv: any_allowed xor item <> Void

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.6 The assertion package

7.6.1 General

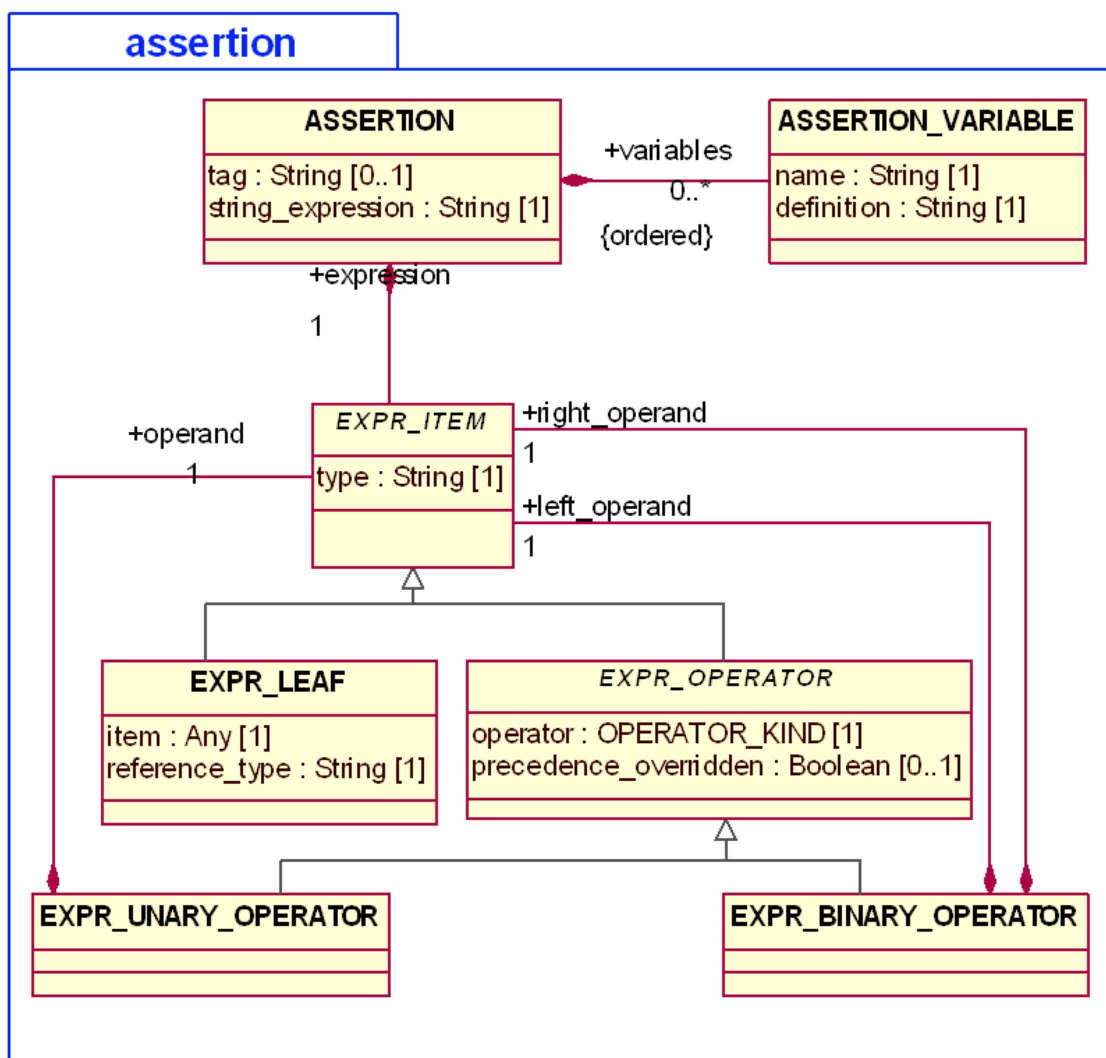


Figure 8 — Assertion Package

7.6.2 Package :: assertion

Inner elements	
Name	**Type**
ASSERTION	Class
ASSERTION_VARIABLE	Class
EXPR_BINARY_OPERATOR	Class
EXPR_ITEM	Class
EXPR_LEAF	Class
EXPR_OPERATOR	Class
EXPR_UNARY_OPERATOR	Class

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

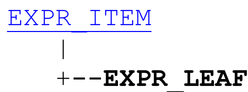
Package: assertion
Class EXPR_ITEM{Abstract}

Direct subclassifiers:
EXPR_OPERATOR, EXPR_LEAF

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
type : String	1	--	(none)

Constraints	
Name	**Expression**
type_valid	inv: type <> Void and not type.is_empty

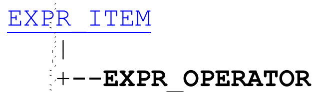
Package: assertion
Class EXPR_LEAF



Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
item : Any	1	--	(none)
reference_type : String	1	--	(none)

Constraints	
Name	**Expression**
item_valid	inv: item <> Void

Package: assertion
Class EXPR_OPERATOR{Abstract}



Direct subclassifiers:
EXPR_BINARY_OPERATOR, EXPR_UNARY_OPERATOR

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
operator : OPERATOR_KIND	1	--	(none)
precedence_overridden : Boolean	0..1	--	(none)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: assertion
Class EXPR_UNARY_OPERATOR

```
EXPR_OPERATOR
   |
   +--EXPR_UNARY_OPERATOR
```

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
operand : EXPR_ITEM	1	--	(none)

Constraints	
Name	**Expression**
operand_valid	inv: operand <> Void

Package: assertion
Class EXPR_BINARY_OPERATOR

```
EXPR_OPERATOR
   |
   +--EXPR_BINARY_OPERATOR
```

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
left_operand : EXPR_ITEM	1	--	(none)
Right_operand : EXPR_ITEM	1	--	(none)

Constraints	
Name	**Expression**
left_operand_valid	inv: left_operand <> Void
right_operand_valid	inv: right_operand <> Void

Package: assertion
Class ASSERTION_VARIABLE

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
definition : String	1	--	(none)
Name : String	1	--	(none)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: assertion
Class ASSERTION

Structural model of a typed first-order predicate logic assertion, in the form of an expression tree, including optional variable definitions.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
string_expression : String	1	--	(none)
tag : String	0..1	--	(none)

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
variables : List<ASSERTION_VARIABLE>	0..1	0..* ordered	(none)
expression : EXPR_ITEM	1	--	(none)

Constraints	
Name	**Expression**
expression_valid	inv: expression <> Void and expression.type.is_equal("Boolean")
tag_valid	inv: tag <> Void implies not tag.is_empty

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.7 The primitive package

7.7.1 General

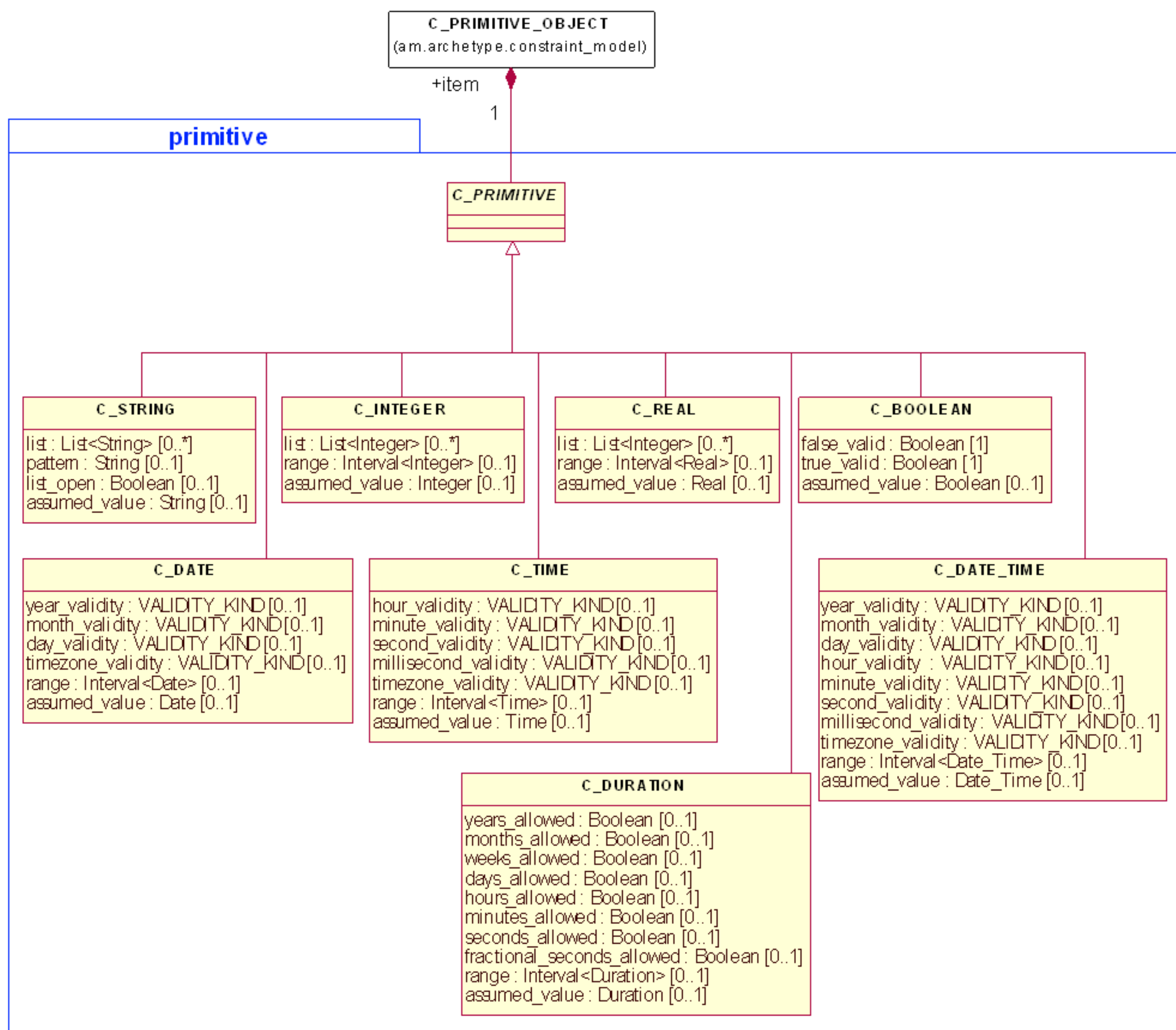


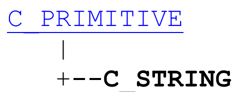
Figure 9 — Primitive package

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.7.2 Package :: primitive

Inner elements	
Name	**Type**
C_BOOLEAN	Class
C_DATE	Class
C_DATE_TIME	Class
C_DURATION	Class
C_INTEGER	Class
C_PRIMITIVE	Class
C_REAL	Class
C_STRING	Class
C_TIME	Class

Package: primitive
Class C_STRING

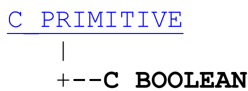


Constraint on instances of String.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : String	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
list : Set<List<String>>	0..1	0..*	List of Strings specifying constraint.
list_open : Boolean	0..1	--	True if the list is being used to specify the constraint but is not considered exhaustive.
pattern : String	0..1	--	Regular expression pattern for proposed instances of String to match.

Constraints	
Name	**Expression**
pattern_exists	inv: pattern <> Void implies not pattern.is_empty
Consistency	inv: pattern <> Void xor list <> Void

Package: primitive
Class C_BOOLEAN



Constraint on instances of Boolean. Both attributes cannot be set to False, since this would mean that the Boolean value being constrained cannot be True or False.

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Boolean	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
false_valid : Boolean	1	--	True if the value False is allowed.
true_valid : Boolean	1	--	True if the value True is allowed.

Constraints	
Name	**Expression**
Default_value_consistency	inv: (default_value.value and true_valid) or (not default_value.value and false_valid)
Binary_consistency	inv: true_valid or false_valid

Package: primitive
Class C_DURATION

```
C_PRIMITIVE
    |
    +--C_DURATION
```

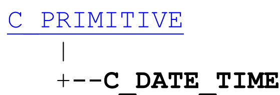
Constraint on instances of Duration.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Duration	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
days_allowed : Boolean	0..1	--	True if days are allowed in the constrained Duration.
fractional_seconds_allowed : Boolean	0..1	--	True if fractional seconds are allowed in the constrained Duration.
hours_allowed : Boolean	0..1	--	True if hours are allowed in the constrained Duration.
minutes_allowed : Boolean	0..1	--	True if minutes are allowed in the constrained Duration.
months_allowed : Boolean	0..1	--	True if months are allowed in the constrained Duration.
range : Interval<Duration>	0..1	--	Constraint on instances of Duration.
seconds_allowed : Boolean	0..1	--	True if seconds are allowed in the constrained Duration.
Weeks_allowed : Boolean	0..1	--	True if weeks are allowed in the constrained Duration.
years_allowed : Boolean	0..1	--	True if years are allowed in the constrained Duration.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Constraints	
Name	**Expression**
Range_valid	inv: range <> Void

Package: primitive
Class C_DATE_TIME



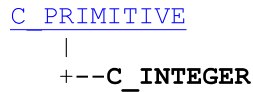
Constraint on instances of Date_Time. There is no validity flag for “year”, since it shall always be by definition mandatory in order to have a sensible date/time.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Date_Time	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
day_validity : VALIDITY_KIND	0..1	--	Validity of day in constrained date.
hour_validity : VALIDITY_KIND	0..1	--	Validity of hour in constrained time.
millisecond_validity : VALIDITY_KIND	0..1	--	Validity of millisecond in constrained time.
minute_validity : VALIDITY_KIND	0..1	--	Validity of minute in constrained time.
month_validity : VALIDITY_KIND	0..1	--	Validity of month in constrained date.
range : Interval<Date_Time>	0..1	--	Range of Date_times specifying constraint.
second_validity : VALIDITY_KIND	0..1	--	Validity of second in constrained time.
timezone_validity : VALIDITY_KIND	0..1	--	Validity of timezone in constrained date.
year_validity : VALIDITY_KIND	0..1	--	(none)

Constraints	
Name	**Expression**
second_validity_disallowed	inv: second_validity = 'disallowed' implies millisecond_validity = 'disallowed'
second_validity_optional	inv: second_validity = 'optional' implies (millisecond_validity = 'optional' or millisecond_validity = 'disallowed')
minute_validity_optional	inv: minute_validity = 'optional' implies (second_validity = 'optional' or second_validity = 'disallowed')
minute_validity_disallowed	inv: minute_validity = 'disallowed' implies second_validity = 'disallowed'
hour_validity_disallowed	inv: hour_validity = 'disallowed' implies minute_validity = 'disallowed'
day_validity_disallowed	inv: day_validity = 'disallowed' implies hour_validity = 'disallowed'
month_validity_disallowed	inv: month_validity = 'disallowed' implies day_validity = 'disallowed'
day_validity_optional	inv: day_validity = 'optional' implies (hour_validity = 'optional' or hour_validity = 'disallowed')
hour_validity_optional	inv: hour_validity = 'optional' implies (minute_validity = 'optional' or minute_validity = 'disallowed')
validity_is_range	inv: validity_is_range = (range <> Void)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: primitive
Class C_INTEGER

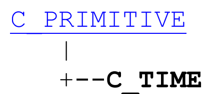


Constraint on instances of Integer.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Integer	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
list : Set<List<Integer>>	0..1	0..*	Set of Integers specifying constraint.
range : Interval<Integer>	0..1	--	Range of Integers specifying constraint.

Constraints	
Name	**Expression**
consistency	inv: list <> Void xor range <> Void

Package: primitive
Class C_TIME



Constraint on instances of Time. There is no validity flag for “hour”, since it shall always be by definition mandatory in order to have a sensible time at all.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Time	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
hour_validity : VALIDITY_KIND	0..1	--	(none)
millisecond_validity : VALIDITY_KIND	0..1	--	Validity of millisecond in constrained time.
Minute_validity : VALIDITY_KIND	0..1	--	Validity of minute in constrained time.
range : Interval<Time>	0..1	--	Interval of Times specifying constraint.
second_validity : VALIDITY_KIND	0..1	--	Validity of second in constrained time.
timezone_validity : VALIDITY_KIND	0..1	--	Validity of timezone in constrained date.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Constraints	
Name	**Expression**
Minute_validity_optional	inv: minute_validity = 'optional' implies (second_validity = 'optional' or second_validity = 'disallowed')
second_validity_disallowed	inv: second_validity = 'disallowed' implies millisecond_validity = 'disallowed'
second_validity_optional	inv: second_validity = 'optional' implies (millisecond_validity = 'optional' or millisecond_validity = 'disallowed')
minute_validity_disallowed	inv: minute_validity = 'disallowed' implies second_validity = 'disallowed'
validity_is_range	inv: validity_is_range = (range <> Void)

Package: primitive
Class C_REAL

```
C_PRIMITIVE
   |
   +--C_REAL
```

Constraint on instances of Real.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Real	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
list : Set<List<Integer>>	0..1	0..*	Set of Reals specifying constraint.
range : Interval<Real>	0..1	--	Range of Real specifying constraint.

Constraints	
Name	**Expression**
consistency	inv: list <> Void xor range <> Void

Package: primitive
Class C_DATE

```
C_PRIMITIVE
   |
   +--C_DATE
```

Constraint on instances of Date in the form either of a set of validity values, or an actual date range. There is no validity flag for “year”, since it shall always be by definition mandatory in order to have a sensible date at all.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
assumed_value : Date	0..1	--	The value to assume if this item is not included in data, if it is part of an optional structure.
day_validity : VALIDITY_KIND	0..1	--	Validity of day in constrained date.
Month_validity : VALIDITY_KIND	0..1	--	Validity of month in constrained date.
range : Interval<Date>	0..1	--	Interval of Dates specifying constraint.
timezone_validity : VALIDITY_KIND	0..1	--	Validity of timezone in constrained date.
year_validity : VALIDITY_KIND	0..1	--	(none)

Constraints	
Name	**Expression**
Validity_is_range	inv: validity_is_range = (range <> Void)
Month_validity_optional	inv: month_validity = 'optional' implies (day_validity = 'optional' or day_validity = 'disallowed')
Month_validity_disallowed	inv: month_validity = 'disallowed' implies day_validity = 'disallowed'

Package: primitive
Class C_PRIMITIVE{Abstract}

Direct subclassifiers:
C_REAL, C_BOOLEAN, C_STRING, C_DATE, C_DURATION, C_INTEGER, C_TIME, C_DATE_TIME

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.8 The ontology package

7.8.1 General

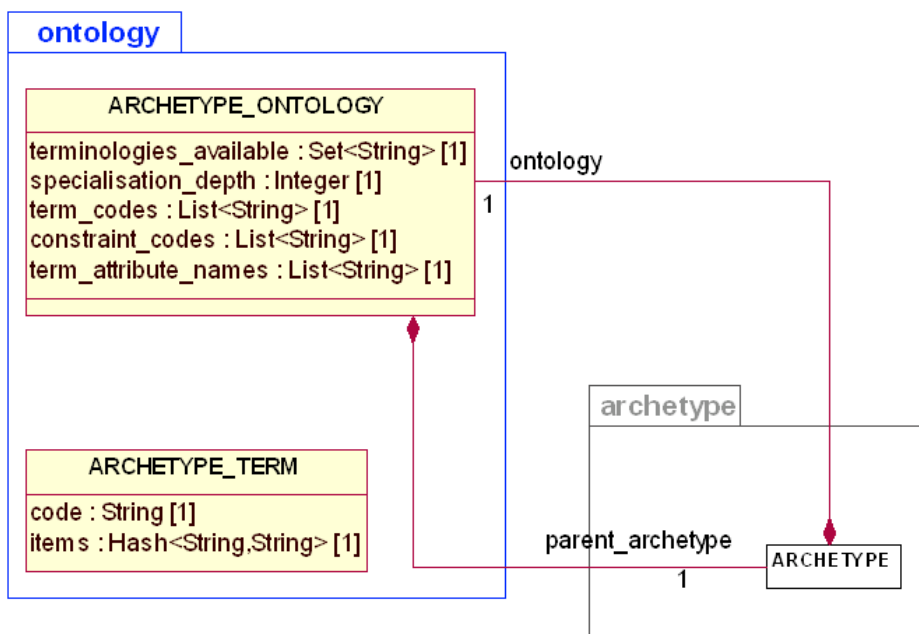


Figure 10 — Ontology package

7.8.2 Package :: ontology

Inner elements	
Name	**Type**
ARCHETYPE_ONTOLOGY	Class
ARCHETYPE_TERM	Class

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: ontology
Class ARCHETYPE_ONTOLOGY

Local ontology of an archetype.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
constraint_codes : List<String>	1	--	List of all constraint codes in the ontology.
specialisation_depth : Integer	1	--	Specialization depth of this archetype. Unspecialized archetypes have depth 0, with each additional level of specialization adding 1 to the specialisation_depth.
term_attribute_names : List<String>	1	--	List of “attribute” names in ontology terms, typically includes “text”, “description”, “provenance”, etc.
term_codes : List<String>	1	--	List of all term codes in the ontology. Most of these correspond to “at” codes in an ADL archetype, which are the node_ids on C_OBJECT descendants. There may be an extra one, if a different term is used as the overall archetype concept_code from that used as the node_id of the outermost C_OBJECT in the definition part.
terminologies_available : Set<String>	1	--	List of terminologies to which term or constraint bindings exist in this terminology.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
parent_archetype : ARCHETYPE	1	--	Archetype that owns this ontology.

Constraints	
Name	**Expression**
terminologies_available_exists	inv: terminologies_available <> Void
term_attribute_names_valid	inv: term_attribute_names <> Void and term_attribute_names.has('text') and term_attribute_names.has('description')
Parent_archetype_valid	inv: parent_archetype <> Void and parent_archetype.description = Current
constraint_codes_exists	inv: constraint_codes <> Void
concept_code_valid	inv: term_codes.has (concept_code)
term_codes_exists	inv: term_codes <> Void

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: ontology
Class ARCHETYPE_TERM

Representation of any coded entity (term or constraint) in the archetype ontology.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
code : String	1	--	Code of this term.
items : Hash<String,String>	1	--	Hash of keys (“text”, “description”, etc) and corresponding values.

Constraints	
Name	**Expression**
code_valid	inv: code <> Void and not code.is_empty

7.9 The domain extensions package

7.9.1 General

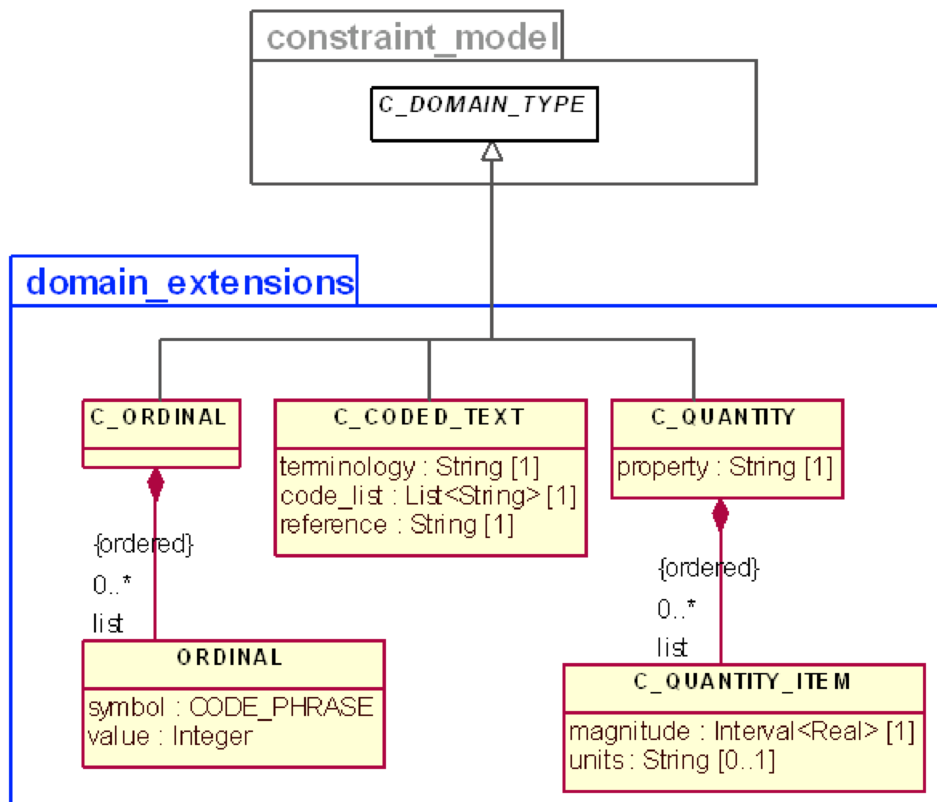


Figure 11 — Domain extensions package

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.9.2 Package :: domain_extensions

Inner elements	
Name	**Type**
C_CODED_TEXT	Class
C_ORDINAL	Class
C_QUANTITY	Class
C_QUANTITY_ITEM	Class
ORDINAL	Class

Package: domain_extensions
Class C_ORDINAL

```
C_DOMAIN_TYPE
   |
   +--C_ORDINAL
```

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
list : List<ORDINAL>	0..1	0..* ordered	(none)

Package: domain_extensions
Class C_CODED_TEXT

```
C_DOMAIN_TYPE
   |
   +--C_CODED_TEXT
```

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
code_list : List<String>	1	--	(none)
reference : String	1	--	(none)
terminology : String	1	--	(none)

Package: domain_extensions
Class C_QUANTITY

```
C_DOMAIN_TYPE
   |
   +--C_QUANTITY
```

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
property : String	1	--	(none)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
list : List<C_QUANTITY_ITEM>	0..1	0..* ordered	(none)

Package: domain_extensions
Class C_QUANTITY_ITEM

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
magnitude : Interval<Real>	1	--	(none)
units : String	0..1	--	(none)

Package: domain_extensions
Class ORDINAL

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
symbol : CODE_PHRASE	0..1	--	(none)
value : Integer	0..1	--	(none)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.10 The support package

7.10.1 General

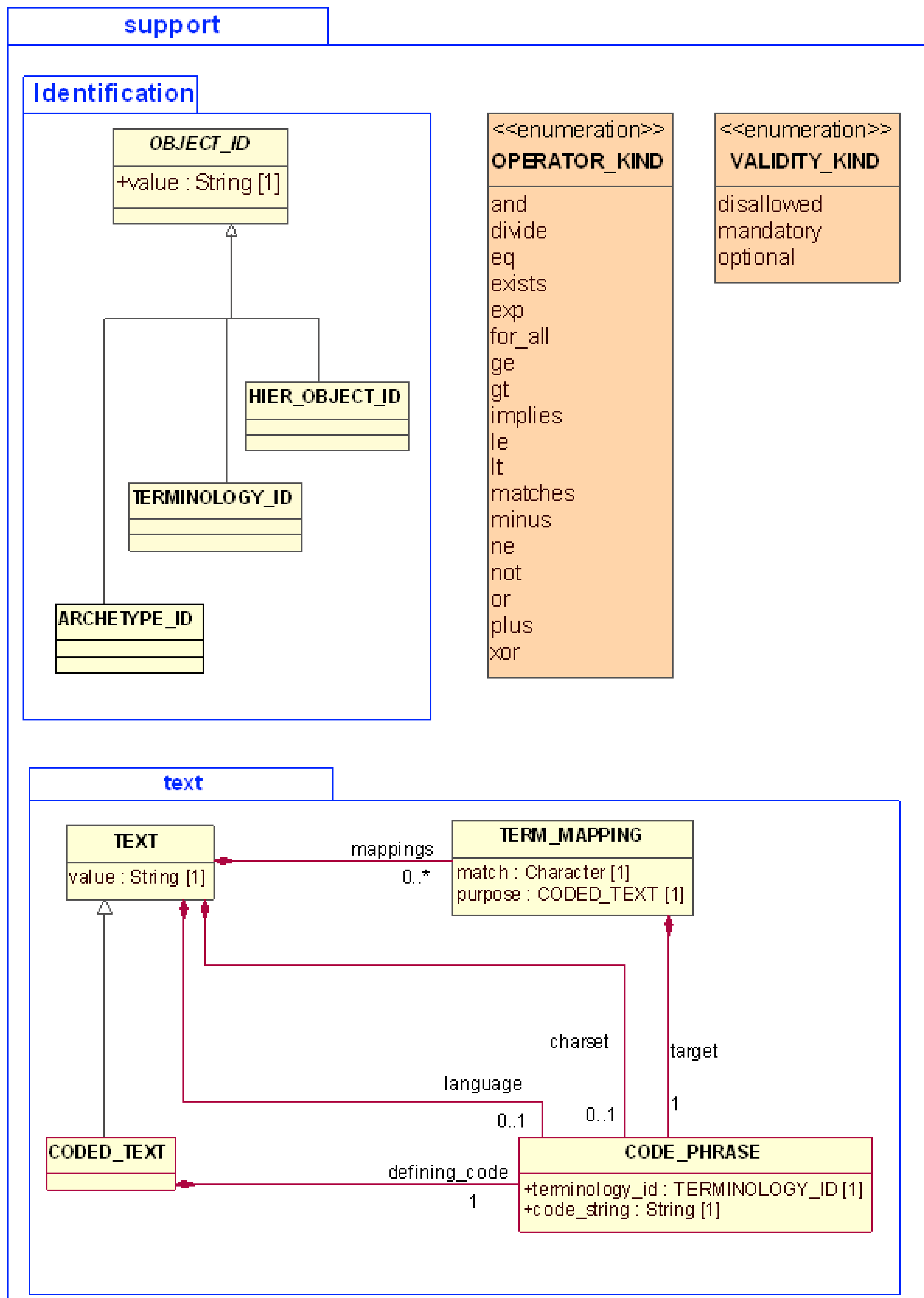


Figure 12 — Support package

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.10.2 Package :: support

Inner elements	
Name	**Type**
OPERATOR_KIND	Enumeration
VALIDITY_KIND	Enumeration
Identification	Package
text	Package

support
Enumeration OPERATOR_KIND

Enumeration literals
And
Divide
Eq
Exists
Exp
for_all
Ge
Gt
Implies
Le
Lt
Matches
Minus
Ne
Not
Or
Plus
Xor

support
Enumeration VALIDITY_KIND

Enumeration literals
Disallowed
Mandatory
Optional

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.10.3 Package :: Identification

Inner elements	
Name	**Type**
ARCHETYPE_ID	Class
HIER_OBJECT_ID	Class
OBJECT_ID	Class
TERMINOLOGY_ID	Class

Package: Identification
Class TERMINOLOGY_ID

```
OBJECT_ID
   |
   +--TERMINOLOGY_ID
```

Package: Identification
Class OBJECT_ID{Abstract}

Direct subclassifiers:
HIER_OBJECT_ID, TERMINOLOGY_ID, ARCHETYPE_ID

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
value : String	1	--	(none)

Package: Identification
Class HIER_OBJECT_ID

```
OBJECT_ID
   |
   +--HIER_OBJECT_ID
```

Package: Identification
Class ARCHETYPE_ID

```
OBJECT_ID
   |
   +--ARCHETYPE_ID
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.10.4 Package :: text

Inner elements	
Name	**Type**
CODE_PHRASE	Class
CODED_TEXT	Class
TERM_MAPPING	Class
TEXT	Class

Package: text
Class CODED_TEXT

A text item whose value shall be the rubric from a controlled terminology, the key (i.e. the "code") of which is the defining_code attribute. In other words: a CODED_TEXT is a combination of a CODE_PHRASE (effectively a code) and the rubric of that term, from a terminology service, in the language in which the data were authored.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
defining_code : CODE_PHRASE	1	--	(none)

Package: text
Class TEXT

Direct subclassifiers:
CODED_TEXT

A plain text item, which may contain any amount of legal characters arranged as words, sentences, etc. (i.e. one TEXT may be more than one word). Any TEXT may be "coded" by adding mappings to it.

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
value : String	1	--	Displayable rendition of the item, regardless of its underlying structure. For CODED_TEXT, this is the rubric of the complete term as provided by the terminology service. No carriage returns, line feeds, or other permitted non-printing characters.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
charset : CODE_PHRASE	0..1	--	Name of character set in which this value is encoded.
language : CODE_PHRASE	0..1	--	Optional indicator of the localized language in which the value is written.
mappings : Set<TERM_MAPPING>	0..1	0..*	Terms from other terminologies most closely matching this term.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: text
Class TERM_MAPPING

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
match : Character	1	--	The relative match of the target term with respect to the mapped text item. Result meanings: “>”: the mapping is to a broader term “=”: the mapping is to a (supposedly) equivalent to the original item “<”: the mapping is to a narrower term “?”: the kind of mapping is unknown
purpose : CODED_TEXT	1	--	Purpose of the mapping, e.g. “automated data mining”, “billing”, “interoperability”.

Attributes from associations			
Signature	**Optionality**	**Multiplicity**	**Documentation**
target : CODE_PHRASE	1	--	The target term of the mapping.

Package: text
Class CODE_PHRASE

A fully coordinated (i.e. all “coordination” has been performed) term from a terminology service (as distinct from a particular terminology).

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
code_string : String	1	--	The key used by the terminology service to identify a concept or coordination of concepts.
terminology_id : TERMINOLOGY_ID	1	--	Identifier of the distinct terminology from which the code_string (or its elements) was extracted.

7.10.5 Package :: generic_types

Inner elements	
Name	**Type**
Aggregate	Class
Hash	Class
Interval	Class
List	Class
Set	Class

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: generic_types
Class list

```
Aggregate
   |
   +--List
```

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
content : List<T>	0..1	0..* ordered	(none)

Template parameters		
Name	**Type**	**Default value**
T		(none)

Package: generic_types
Class set

```
Aggregate
   |
   +--Set
```

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
content : Set<T>	0..1	0..*	(none)

Template parameters		
Name	**Type**	**Default value**
T	(none)	(none)

Package: generic_types
Class interval

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
lower : T	1	--	(none)
lower_unbounded : Boolean	1	--	(none)
upper : T	1	--	(none)
upper_unbounded : Boolean	1	--	(none)

Template parameters		
Name	**Type**	**Default value**
T	(none)	(none)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Package: generic_types
Class hash

```
Aggregate
    |
    +--Hash
```

Attributes			
Signature	**Optionality**	**Multiplicity**	**Documentation**
key : S	1	--	(none)
value : T	1	--	(none)

Template parameters		
Name	**Type**	**Default value**
S	(none)	(none)
T	(none)	(none)

Package: generic_types
Class aggregate{Abstract}

Direct subclassifiers:
List, Hash, Set

Template parameters		
Name	**Type**	**Default value**
T	(none)	(none)

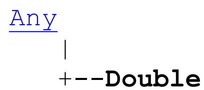
Operations		
Signature	**Constraints**	**Documentation**
count() : Integer	(none)	(none)
has() : Boolean	(none)	(none)
is_empty() : Boolean	(none)	(none)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.10.6 Package :: primitive_data_types

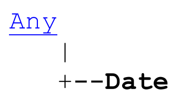
Inner elements	
Name	**Type**
Any	Data Type
Boolean	Data Type
Character	Data Type
Date	Data Type
Date_Time	Data Type
Double	Data Type
Duration	Data Type
Integer	Data Type
Real	Data Type
String	Data Type
Time	Data Type

Primitive_Data_Types
Data Type Double



Primitive_Data_Types
Data Type Character

Primitive_Data_Types
Data Type Date



Primitive_Data_Types
Data Type Duration

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Primitive_Data_Types
Data Type Date_Time

```
Any
   |
   +--Date_Time
```

Primitive_Data_Types
Data Type Time

```
Any
   |
   +--Time
```

Primitive_Data_Types
Data Type String

```
Any
   |
   +--String
```

Operations		
Signature	**Constraints**	**Documentation**
is_empty() : Boolean	(none)	(none)
is_equal(other : String) : Boolean	(none)	(none)

Primitive_Data_Types
Data Type Integer

```
Any
   |
   +--Integer
```

Primitive_Data_Types
Data Type Boolean

```
Any
   |
   +--Boolean
```

Primitive_Data_Types
Data Type Real

```
Any
   |
   +--Real
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Primitive_Data_Types
Data Type Any

Direct subclassifiers:
Date_Time, Real, Time, Integer, Duration, Character, String, Double, Boolean, Date

7.11 Generic types package

This package is included to confirm the semantics of the generic types used in this part of ISO 13606. Although List<T>, Set<T>, Bag(not used), Hash<T,K>, and Interval<T> are generic types supported by many programming environments, they are not directly supported in UML. In this package, new types such as List<String> are defined using binding dependencies between a new basic type such as List<String> and a Class (LIST in this example) that defines the minimum required semantics for all Lists.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

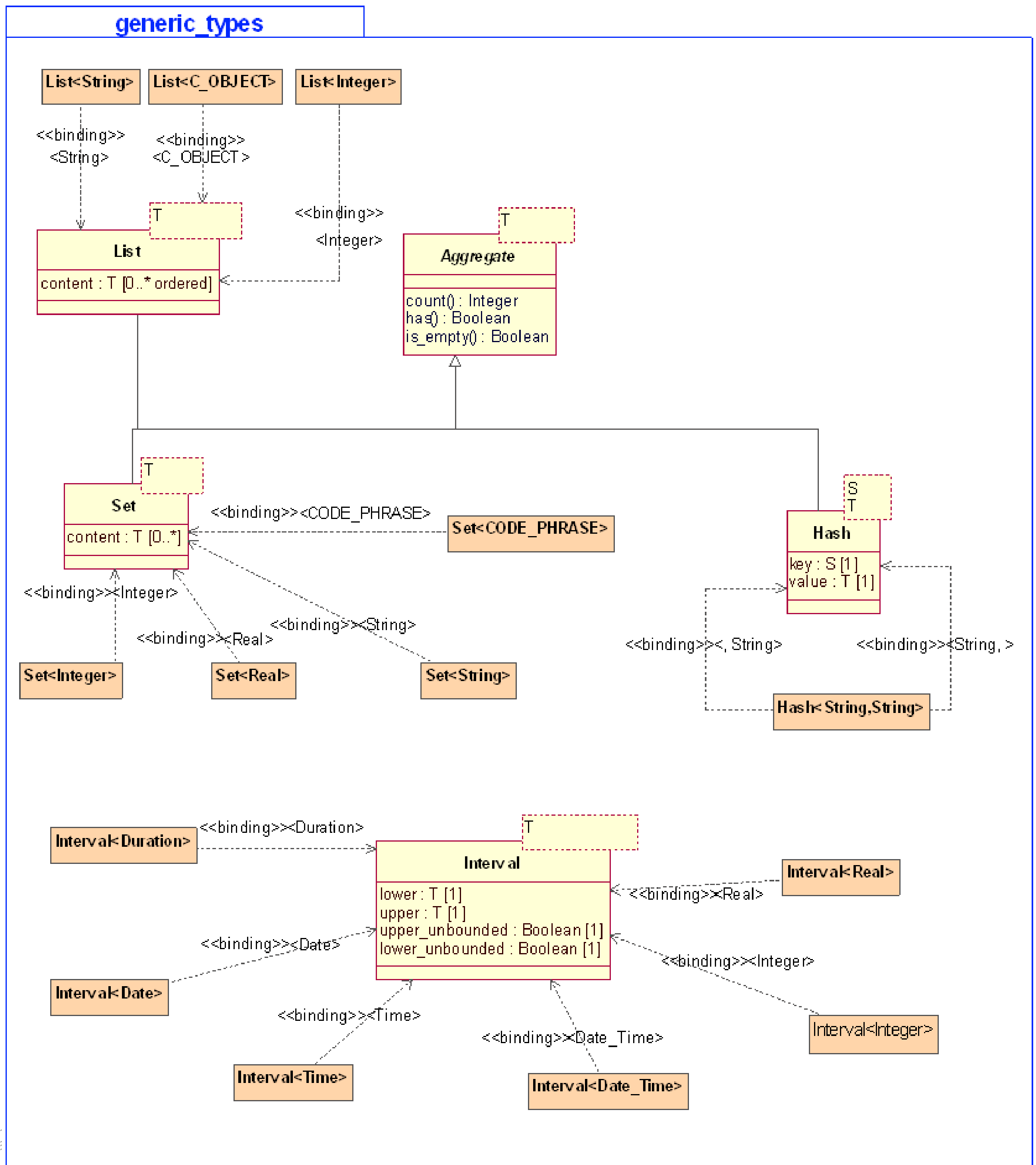


Figure 13 — Generic types package

7.12 Domain-specific extensions (informative)

7.12.1 General

Domain-specific classes may be added to the archetype constraint model by inheriting from the class *C_DOMAIN_TYPE*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.12.2 Scientific/clinical computing types

Figure 14 shows the general approach used to add constraint classes for commonly used concepts in scientific and clinical computing, such as “ordinal”, “coded term” and “quantity”. The constraint types shown are *C_ORDINAL*, *C_CODED_TEXT* and *C_QUANTITY* which may optionally be used in archetypes to replace the default constraint semantics represented by the use of instances of *C_OBJECT* / *C_ATTRIBUTE.*

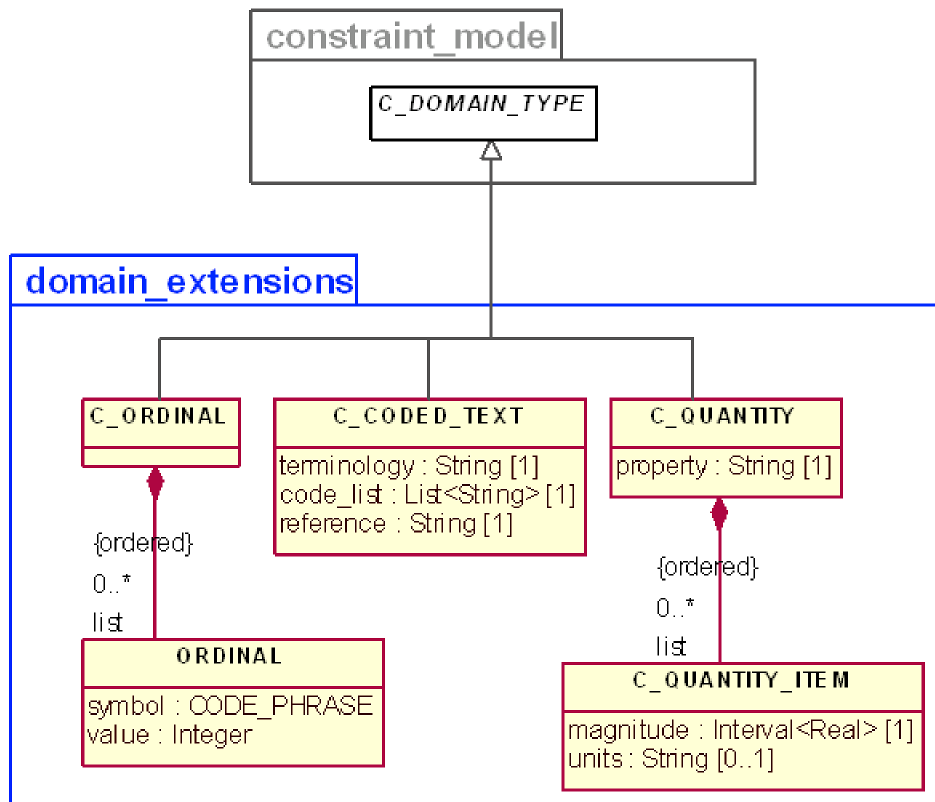


Figure 14 — Example domain-specific package

8 Archetype Definition Language (ADL)

8.1 dADL — Data ADL

8.1.1 Overview

8.1.1.1 Preamble

The dADL syntax provides a formal means of expressing *instance data* based on an underlying information model, which is readable both by humans and machines.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

EXAMPLE

```
person = List<PERSON> <
    [01234] = <
        name = <                    -- persons name
        forenames =                 <"Sherlock">
        family_name =                   <"Holmes">
        salutation =                    <"Mr">
        >
        address = <                     -- persons address
        habitation_number =              <"221B">
        street_name =                   <Baker St>
        city =                  <London>
        country =                   <England>
      >
    >
    [01235] = < -- etc
    >
>
```

NOTE In the above, the identifiers PERSON, name, address, etc. are all assumed to come from an information model. The basic design principle of dADL is to be able to represent data in a way that is both machine-processible and human-readable, while making the fewest assumptions possible about the information model to which the data conform. To this end, type names are optional; often, only attribute names and values are explicitly shown. More than one information model can be compatible with the same dADL-expressed data. The UML semantics of composition/aggregation and association are expressible, as are shared objects. Literal leaf values are only of widely recognised types, i.e. Integer, Real, Boolean, String, Character and a range of Date/Time types; all complex types are expressed structurally.

8.1.1.2 Scope of a dADL document

A dADL document may contain one or more objects from the same object model.

8.1.1.3 Keywords

dADL has no keywords of its own — all identifiers are assumed to come from an information model.

8.1.1.4 Reserved characters

In dADL, some characters are reserved and have the following meanings:

‘<’: open an object block;

‘>’: close an object block;

‘=’: indicate attribute value = object block;

‘(’, ‘)’: type name or plug-in syntax-type delimiters;

‘<#’: open an object block expressed in a plug-in syntax;

‘#>’: close an object block expressed in a plug-in syntax.

Within <> delimiters, the following characters are used to indicate primitive values:

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

“”: double quote characters are used to delimit string values;

‘’: single quote characters are used to delimit single character values;

‘|’: bar characters are used to delimit intervals;

[ ]: brackets are used to delimit coded terms.

8.1.1.5 Comments

Comments are indicated by the -- characters. Multiline comments are achieved using the -- leader on each line where the comment continues.

8.1.1.6 Information model identifiers

A type name is any identifier with an initial upper case letter, followed by any combination of letters, digits and underscores. A generic type name (including nested forms) may additionally include commas and angle brackets, but no spaces, and shall be syntactically correct as per the UML. An attribute name is any identifier with an initial lower case letter, followed by any combination of letters, digits and underscores.

8.1.1.7 Semicolons

Semicolons are optionally used to support readability.

NOTE The following examples are equivalent:

```
term = <text = <"plan">;        description = <"The clinician's advice">>

term = <text = <"plan">         description = <"The clinician's advice">>

term = <
    text = <"plan">
    description = <"The clinician's advice">
>
```

8.1.2 Paths

Because dADL data are hierarchical, and all nodes are uniquely identified, a unique path can be determined for every node in a dADL text. The syntax of paths in dADL is the standard ADL path syntax. Paths are directly convertible to XPath expressions for use in XML-encoded data.

NOTE A typical ADL path used to refer to a node in a dADL text is as follows:

```
/term_definitions[en]/items[at0001]/text/
```

8.1.3 Structure

8.1.3.1 General form

8.1.3.1.1 General

A dADL document expresses serialized instances of one or more complex objects. Each such instance is a hierarchy of attribute names and object values.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

NOTE 1 In its simplest form, a dADL text consists of repetitions of the following pattern:

```
attribute_name = <value>
```

In the most basic form of dADL, each attribute name is the name of an attribute in an implied or actual object or relational model. Each value is either a literal value of a primitive type (see 7.10.6) or a further nesting of attribute names and values, terminating in leaf nodes of primitive type values. Where sibling attribute nodes occur, the attribute names shall be unique.

NOTE 2 The following shows a typical structure:

```
attr_1 =   <
    attr_2 = <
        attr_3 = <leaf_value>
        attr_4 = <leaf_value>
    >
    attr_5 = <
        attr_3 = <
            attr_6 = <leaf_value>
        >
        attr_7 = <leaf_value>
    >
>
attr_8 =   <>
```

NOTE 3 In the above structure, each <> encloses an instance of some type. The hierarchical structure corresponds to the part-of relationship between objects: composition and *aggregation* relationships in UML. Associations between instances in dADL are also representable by references, and are described in 8.1.3.5.

8.1.3.1.2 Outer delimiters

Outer <> delimiters in a dADL text are optional.

8.1.3.2 Empty sections

Empty sections are permitted at both internal and leaf node levels, enabling the author to express the fact that there are, in some particular instance, no data for an attribute, while still showing that the attribute itself is expected to exist in the underlying information model. Nested empty sections may be used.

EXAMPLE

```
address =       <>          -- persons address
```

8.1.3.3 Container objects

8.1.3.3.1 General

Container instances are expressed using repetitions of a block introduced by an arbitrary container attribute name, contained in square brackets, and qualified in each case by a unique manifest value. The qualifiers are arbitrary unique keys, which need not be drawn from the set of contained values. These keys need not be sequential and do not in themselves imply an ordering. Container structures may appear anywhere in an overall instance structure.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

EXAMPLE

```
school_schedule =   <
    lesson_times = <08:30:00, 09:30:00, 10:30:00, ...>

    locations = <
        [1] = <under the big plane tree>
        [2] = <under the north arch>
        [3] = <in a garden>
    >

    subjects = <
        [philosophy:plato] = <    -- note construction of qualifier
            name = <philosophy>
            teacher = <plato>
            topics = <meta-physics, natural science>
            weighting = <76%>
        >
        [philosophy:kant] = <
            name = <philosophy>
            teacher = <kant>
            topics = <meaning and reason, meta-physics, ethics>
            weighting = <80%>
        >
        [art] = <
            name = <art>
            teacher = <goya>
            topics = <technique, portraiture, satire>
            weighting = <78%>
        >
    >
```

8.1.3.3.2 Paths

Paths through container objects are formed in the same way as paths in other structured data, with the addition of the key to ensure uniqueness. The key is included syntactically enclosed in brackets.

EXAMPLE

```
/school_schedule/locations[1]/
       -- path to under the big...
/school_schedule/subjects[philosophy:kant]/
                       -- path to kant
```

8.1.3.4 Adding type information

Type information may be included optionally on any node immediately before the opening < of any block, in the form of a UML-style type identifier which optionally includes dot-separated namespace identifiers and

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

template parameters. Type information may be added to instance data by including the type name in parentheses after the = sign.

EXAMPLE 1

```
destinations =  <
      [seville] = (TOURIST_DESTINATION) <
          profile = (DESTINATION_PROFILE) <>
          hotels = <
             [gran sevilla] = (HISTORIC_HOTEL) <>
             [sofitel] = (LUXURY_HOTEL) <>
             [hotel real] = (PENSION) <>
          >
          attractions = <
             [la corrida] = (ATTRACTION) <>
             [Alc·zar] = (HISTORIC_SITE) <>
          >
      >
>
```

NOTE In the above, no type identifiers are included after the hotels and attractions attributes. However, the complete typing information may be included as follows.

```
hotels = (List<HOTEL>) <
     [gran sevilla] = (HISTORIC_HOTEL) <>
>
```

Type identifiers may also include namespace information, which is necessary when same-named types appear in different packages of a model. Namespaces are included by pre-pending package names, separated by the character.

EXAMPLE 2

`RM.EHR.CONTENT.ENTRY`

and

`Core.Abstractions.Relationships.Relationship`.

8.1.3.5 Associations and shared objects

Shared objects are referenced using paths. Objects in other dADL documents can be referred to using normal URIs whose path section conforms to dADL path syntax.

EXAMPLE Hotel objects may be shared objects, referred to by association.

```
destinations = <
    ["seville"] = <
       hotels = <
          ["gran sevilla"] = </hotels["gran sevilla"]>
          ["sofitel"] = </hotels["sofitel"]>
          ["hotel real"] = </hotels["hotel real"]>
       >
    >
>
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
bookings = <
    ["seville:0134"] = <
        customer_id = <"0134">
        period = <...>
        hotel = </hotels["sofitel"]>
    >
>

hotels = <
    ["gran sevilla"] = (HISTORIC_HOTEL) <>
    ["sofitel"] = (LUXURY_HOTEL) <>
    ["hotel real"] = (PENSION) <>
>
```

8.1.4 Leaf data

8.1.4.1 General

All dADL data devolve to instances of the primitive types **String**, **Integer**, **Real**, **Double**, **String**, **Character**, various date/time types, lists or intervals of these types, and a few special types. dADL does not use type or attribute names for instances of primitive types, only manifest values.

8.1.4.2 Primitive types

8.1.4.2.1 Character data

Characters are shown in a number of ways. In the literal form, a character is shown in single quotes. Characters outside the low ASCII (0-127) range shall be UTF-8 encoded.

EXAMPLE

```
'&ohgr;'            -- greek omega
```

8.1.4.2.2 String data

All strings are enclosed in double quotes. Quotes are encoded using ISO/IEC 10646 codes.

EXAMPLE 1

```
"this is a much longer string, what one might call a &quot;phrase&quot;."
```

Line extension of strings is done simply by including returns in the string. The exact contents of the string are computed as being the characters between the double quote characters, with the removal of white space leaders up to the left-most character of the first line of the string.

EXAMPLE 2

```
a &isin; A                  -- prints as: a ∈ A
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.1.4.2.3 Integer data

Integers are represented simply as numbers. Commas or periods for breaking long numbers are not permitted (see 8.1.4.4).

EXAMPLE

```
25
300000
29e6
```

8.1.4.2.4 Real data

Real numbers are assumed whenever a decimal is included within a number. Commas or periods for breaking long numbers are not permitted. Only periods may be used to separate the decimal part of a number (see 8.1.4.4).

EXAMPLE

```
25.0
3.1415926
6.023e23
```

8.1.4.2.5 Boolean data

Boolean values may be indicated by the following values (case-insensitive):

```
True
False
```

8.1.4.2.6 Dates and times

Complete date/time

In dADL, full and partial dates, times and durations can be expressed. All full dates, times and durations are expressed using a subset of ISO 8601. In dADL, the use of ISO 8601 allows extended form only (i.e. ":" and "-" shall be used). The ISO 8601 method of representing partial dates consisting of a single year number, and partial times consisting of hours only are not supported. See below for partial forms. Patterns for complete dates and times in dADL include the following:

```
yyyy-MM-dd                                      -- a date
hh:mm[:ss[.sss][Z]]                             -- a time
yyyy-MM-dd hh:mm:ss[.sss][Z]                    -- a date/time
```

where:

```
yyyy      = four-digit year
MM        = month in year
dd        = day in month
hh        = hour in 24 hour clock
mm        = minutes
ss.sss    = seconds, including fractional part
Z         = the timezone in the form of a + or - followed by four digits
            indicating the hour offset, e.g. +0930, or else the literal Z
            indicating +0000 (the Greenwich meridian).
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Durations are expressed using a string which starts with “P”, and is followed by a list of periods, each appended by a single letter designator: “Y” for years, “M” for months, “W” for weeks, “D” for days, “H” for hours, “M” for minutes, and “S” for seconds. The literal “T” separates the YMWD part from the HMS part, ensuring that months and minutes can be distinguished.

EXAMPLE

```
1919-01-23                -- birthdate of Django Reinhardt
16:35  .04                -- rise of Venus in Sydney on 24 Jul 2003
2001-05-12 07:35:20+1000  -- timestamp on an email received from Australia
P22D4H15M0S               -- period of 22 days, 4 hours, 15 minutes
```

Partial date/time

Two ways of expressing partial (i.e. incomplete) date/times are supported in dADL. The ISO 8601 incomplete formats are supported in extended form only (i.e. with “-” and “:” separators) for all patterns that are unambiguous on their own. Dates consisting of only the year, and times consisting of only the hour are not supported. The supported ISO 8601 patterns are as follows:

```
yyyy-MM                   -- a date with no days
hh:mm                     -- a time with no seconds
yyyy-MM-ddThh:mm          -- a date/time with no seconds
yyyy-MM-ddThh             -- a date/time with no minutes or seconds
```

To deal with the limitations of ISO 8601 partial patterns in a context-free parsing environment, a second form of pattern is supported in dADL, based on ISO 8601. In this form, “?” characters are substituted for missing digits.

Valid partial dates follow the patterns:

```
yyyy-MM-??                -- date with unknown day in month
yyyy-??-??                -- date with unknown month and day
```

Valid partial times follow the patterns:

```
hh:mm:??                  -- time with unknown seconds
hh:??:??                  -- time with unknown minutes and seconds
```

Valid date/times follow the patterns:

```
yyyy-MM-ddThh:mm:??       -- date/time with unknown seconds
yyyy-MM-ddThh:??:??       -- date/time with unknown minutes and seconds
yyyy-MM-ddT??:??:??       -- date/time with unknown time
yyyy-MM-??T??:??:??       -- date/time with unknown day and time
yyyy-??-??T??:??:??       -- date/time with unknown month, day and time
```

8.1.4.3 Intervals of ordered primitive types

Intervals of any ordered primitive type, i.e. Integer, Real, Date, Time, Date_Time and Duration, can be stated using the following uniform syntax, where N and M are instances of any of the ordered types:

`|N..M|` the two-sided range N <= x <= M;

`|N<..M|` the two-sided range N < x <= M;

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

`|N..<M|`	the two-sided range N <= x <M;
`|N<..<M|`	the two-sided range N < x <M;
`|<N|`	the one-sided range x < N;
`|>N|`	the one-sided range x > N;
`|>=N|`	the one-sided range x >= N;
`|<=N|`	the one-sided range x <= N;
`|N +/-M|`	interval of N ± M.

The allowable values for N and M include any value in the range of the relevant type as well as:

```
infinity
-infinity
*      equivalent to infinity
```

8.1.4.4 Other built-in types

8.1.4.4.1 URIs

URIs follow the standard syntax from http://www.ietf.org/rfc/rfc3986.txt. No quotes or inverted commas are needed; neither spaces nor angle brackets are allowed; both have to be quoted.

EXAMPLE

```
http://archetypes.are.us/home.html
ftp://get.this.file.com#section_5
http://www.mozilla.org/products/firefox/upgrade/?application=thunderbird
```

8.1.4.4.2 Coded terms

The logical structure of a coded term consists of an identifier of a terminology, and an identifier of a code within that terminology. (The rubric associated with the code forms part of the ontology package, documented later in this Section.) The dADL string representation is as follows:

```
[terminology_id::code]
```

EXAMPLE

```
[icd10AM::F60.1]                  -- from ICD10AM
[snomed-ct::2004950]              -- from snomed-ct
[snomed-ct(3.1)::2004950]         -- from snomed-ct v 3.1
```

8.1.4.5 Lists of built-in types

Data of any primitive type may occur singly or in lists, which are shown as comma-separated lists of items, all of the same type.

EXAMPLE 1

```
cyan, magenta, yellow, black      -- printers colours
1, 1, 2, 3, 5                     -- first 5 fibonacci numbers
08:02, 08:35, 09:10               -- set of train times
```

No assumption is made in the syntax about whether a list represents a set, a list or some other kind of sequence; such semantics shall be taken from an underlying information model.

Lists which have only one datum are indicated by using a comma followed by a list continuation marker of three dots, i.e. ...

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

EXAMPLE 2

```
en, ...              -- languages
icd10, ...           -- terminologies
[at0200], ...
```

White space may be optionally used within lists.

EXAMPLE 3 The following two lists are identical:

```
1,1,2,3
1, 1, 2,3
```

8.1.5 dADL syntax

8.1.5.1 Grammar

This section specifies the dADL grammar.

```
input:
    attr_vals
|   complex_object_block
    | error
    ;

---------------------- body ---------------------

attr_vals: attr_val
    | attr_vals attr_val
    | attr_vals ';' attr_val
    ;

attr_val: attr_id SYM_EQ object_block -- could be a single or multiple attr
    ;

attr_id:
  V_ATTRIBUTE_IDENTIFIER
| V_ATTRIBUTE_IDENTIFIER error

object_block:
complex_object_block
| primitive_object_block
| plugin_object_block
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
plugin_object_block:
  V_PLUGIN_SYNTAX_TYPE V_PLUGIN_BLOCK


complex_object_block:
    single_attr_object_block
    | multiple_attr_object_block
    ;


multiple_attr_object_block: untyped_multiple_attr_object_block
    | TYPE_IDENTIFIER untyped_multiple_attr_object_block
    ;


untyped_multiple_attr_object_block: multiple_attr_object_block_head
                                     keyed_objects SYM_END_DBLOCK
    ;


multiple_attr_object_block_head: SYM_START_DBLOCK
    ;


keyed_objects: keyed_object
    | keyed_objects keyed_object
    ;


keyed_object: object_key SYM_EQ object_block
    ;


attr_id: V_ATTRIBUTE_IDENTIFIER
    | V_ATTRIBUTE_IDENTIFIER error
    ;


object_key: '[' simple_value ']'
    ;


single_attr_object_block: untyped_single_attr_object_block
    | TYPE_IDENTIFIER untyped_single_attr_object_block
    ;


untyped_single_attr_object_block:
    single_attr_object_complex_head SYM_END_DBLOCK
    single_attr_object_complex_head attr_vals SYM_END_DBLOCK
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
single_attr_object_complex_head: SYM_START_DBLOCK
    ;
primitive_object_block:
  untyped_primitive_object_block
| type_identifier untyped_primitive_object_block

untyped_primitive_object_block:
single_attr_object_primitive:
    SYM_START_DBLOCK primitive_object_value SYM_END_DBLOCK
    ;

primitive_object_value: simple_value
    | simple_list_value
    | simple_interval_value
    | term_code
    | term_code_list_value
    | query
    ;

simple_value: string_value
    | integer_value
    | real_value
    | boolean_value
    | character_value
    | date_value
    | time_value
    | date_time_value
    | duration_value
    | uri_value
    ;

simple_list_value: string_list_value
    | integer_list_value
    | real_list_value
    | boolean_list_value
    | character_list_value
    | date_list_value
    | time_list_value
    | date_time_list_value
    | duration_list_value
    ;
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
simple_interval_value: integer_interval_value
    | real_interval_value
    | date_interval_value
    | time_interval_value
    | date_time_interval_value
    | duration_interval_value
    ;

type_identifier:
  V_TYPE_IDENTIFIER
| V_GENERIC_TYPE_IDENTIFIER

---------------------- BASIC DATA VALUES -----------------------

string_value: V_STRING
    ;

string_list_value: V_STRING ',' V_STRING
    | string_list_value ',' V_STRING
    | V_STRING ',' SYM_LIST_CONTINUE
    ;

integer_value: V_INTEGER
    | '+' V_INTEGER
    | '-' V_INTEGER
    ;

integer_list_value: integer_value ',' integer_value
    | integer_list_value ',' integer_value
    | integer_value ',' SYM_LIST_CONTINUE
    ;

integer_interval_value:
  SYM_INTERVAL_DELIM integer_value SYM_ELLIPSIS integer_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT integer_value SYM_ELLIPSIS integer_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM integer_value SYM_ELLIPSIS SYM_LT integer_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT integer_value SYM_ELLIPSIS SYM_LT integer_value
SYM_INTERVAL_DELIM
```

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
| SYM_INTERVAL_DELIM SYM_LT integer_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LE integer_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT integer_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GE integer_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM integer_value SYM_INTERVAL_DELIM
    ;

real_value: V_REAL
    | '+' V_REAL
    | '-' V_REAL
    ;

real_list_value: real_value ',' real_value
    | real_list_value ',' real_value
    | real_value ',' SYM_LIST_CONTINUE
    ;

real_interval_value:
SYM_INTERVAL_DELIM real_value SYM_ELLIPSIS real_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT real_value SYM_ELLIPSIS real_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM real_value SYM_ELLIPSIS SYM_LT real_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT real_value SYM_ELLIPSIS SYM_LT real_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LT real_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LE real_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT real_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GE real_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM real_value SYM_INTERVAL_DELIM
    ;

boolean_value: SYM_TRUE
    | SYM_FALSE
    ;

boolean_list_value: boolean_value ',' boolean_value
    | boolean_list_value ',' boolean_value
    | boolean_value ',' SYM_LIST_CONTINUE
    ;
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
character_value: V_CHARACTER
     ;

character_list_value: character_value ',' character_value
     | character_list_value ',' character_value
     | character_value ',' SYM_LIST_CONTINUE
     ;

date_value: V_ISO8601_EXTENDED_DATE

date_list_value: date_value ',' date_value
     | date_list_value ',' date_value
     | date_value ',' SYM_LIST_CONTINUE
     ;

date_interval_value:
SYM_INTERVAL_DELIM date_value SYM_ELLIPSIS date_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT date_value SYM_ELLIPSIS date_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM date_value SYM_ELLIPSIS SYM_LT date_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT date_value SYM_ELLIPSIS SYM_LT date_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LT date_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LE date_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT date_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GE date_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM date_value SYM_INTERVAL_DELIM
     ;

time_value: V_ISO8601_EXTENDED_TIME

time_list_value: time_value ',' time_value
     | time_list_value ',' time_value
     | time_value ',' SYM_LIST_CONTINUE
     ;

time_interval_value:
SYM_INTERVAL_DELIM time_value SYM_ELLIPSIS time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT time_value SYM_ELLIPSIS time_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM time_value SYM_ELLIPSIS SYM_LT time_value
SYM_INTERVAL_DELIM
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
| SYM_INTERVAL_DELIM SYM_GT time_value SYM_ELLIPSIS SYM_LT time_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LT time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LE time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GE time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM time_value SYM_INTERVAL_DELIM
    ;

date_time_value: V_ISO8601_EXTENDED_DATE_TIME
    ;

date_time_list_value: date_time_value ',' date_time_value
    | date_time_list_value ',' date_time_value
    | date_time_value ',' SYM_LIST_CONTINUE
    ;

date_time_interval_value:
SYM_INTERVAL_DELIM date_time_value SYM_ELLIPSIS date_time_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT date_time_value SYM_ELLIPSIS date_time_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM date_time_value SYM_ELLIPSIS SYM_LT date_time_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT date_time_value SYM_ELLIPSIS SYM_LT

date_time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LT date_time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LE date_time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT date_time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GE date_time_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM date_time_value SYM_INTERVAL_DELIM
    ;

duration_value: V_ISO8601_DURATION
    | - V_ISO8601_DURATION

duration_list_value: duration_value ',' duration_value
    | duration_list_value ',' duration_value
    | duration_value ',' SYM_LIST_CONTINUE
    ;
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
duration_interval_value:
SYM_INTERVAL_DELIM duration_value SYM_ELLIPSIS duration_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT duration_value SYM_ELLIPSIS duration_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM duration_value SYM_ELLIPSIS SYM_LT duration_value
SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT duration_value SYM_ELLIPSIS SYM_LT
     ;
duration_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LT duration_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_LE duration_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GT duration_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM SYM_GE duration_value SYM_INTERVAL_DELIM
| SYM_INTERVAL_DELIM duration_value SYM_INTERVAL_DELIM


term_code: V_QUALIFIED_TERM_CODE_REF
     ;

term_code_list_value: term_code ',' term_code
     | term_code_list_value ',' term_code
     | term_code ',' SYM_LIST_CONTINUE
     ;

uri_value: V_URI
```

8.1.5.2 Symbols

The following specifies the symbols and lexical patterns used in the above grammar.

```
----------/* definitions */ -----------------------------------------------
ALPHANUM [a-zA-Z0-9]
IDCHAR [a-zA-Z0-9_]
NAMECHAR [a-zA-Z0-9._\-]
NAMECHAR_SPACE [a-zA-Z0-9._\- ]
NAMECHAR_PAREN [a-zA-Z0-9._\-()]
UTF8CHAR (([\xC2-\xDF][\x80-\xBF])|(\xE0[\xA0-\xBF][\x80-\xBF])|([\xE1-
\xEF][\x80-\xBF][\x80-\xBF])|(\xF0[\x90-\xBF][\x80-\xBF][\x80-\xBF])|([\xF1-
\xF7][\x80-\xBF][\x80-\xBF][\x80-\xBF]))
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
----------/** Separators **/---------------------------------------------
[ \t\r]+ -- Ignore separators
\n+ -- (increment line count)

----------/* comments */ ------------------------------------------------
"--".*                                          -- Ignore comments
"--".*\n[ \t\r]*

----------/* symbols */ -------------------------------------------------
-         Minus_code
+         Plus_code
*         Star_code
/         Slash_code
^         Caret_code
.         Dot_code
;         Semicolon_code
,         Comma_code
:         Colon_code
!         Exclamation_code
(         Left_parenthesis_code
)         Right_parenthesis_code
$         Dollar_code
“??”      SYM_DT_UNKNOWN
?         Question_mark_code

|         SYM_INTERVAL_DELIM

[         Left_bracket_code
]         Right_bracket_code

=         SYM_EQ

>=        SYM_GE
<=        SYM_LE

<         SYM_LT / SYM_START_DBLOCK

>         SYM_GT / SYM_END_DBLOCK

..        SYM_ELLIPSIS
...       SYM_LIST_CONTINUE
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
----------/* keywords */ ---------------------------------------------
[Tt][Rr][Uu][Ee]                                   SYM_TRUE
[Ff][Aa][Ll][Ss][Ee]                               SYM_FALSE
[Ii][Nn][Ff][Ii][Nn][Ii][Tt][Yy]                   SYM_INFINITY
[Qq][Uu][Ee][Rr][Yy]                               SYM_QUERY_FUNC

----------/* URI */ --------------------------------------------------
[a-z]+:\/\/[ ^<>|\\{}^~"\[\]]*                                   V_URI

----------/* term code reference of form [ICD10AM(1998)::F23 */ ----------
\[{NAMECHAR_PAREN}+::{NAMECHAR_SPACE}+\]   V_QUALIFIED_TERM_CODE_REF
\[{ALPHANUM}{NAMECHAR}*\]       V_LOCAL_TERM_CODE_REF

----------/* local code definition */ --------------------------------------
a[ct][0-9.]+                                  V_LOCAL_CODE

----------/* V_ISO8601_EXTENDED_DATE_TIME YYYY-MM-DDThh:mm:ss[,sss][Z|+/-
nnnn] */ ---
[0-9]{4}-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-6][0-9]:[0-6][0-9](,[0-
9]+)?(Z|[+-][0-9]{4})? |
[0-9]{4}-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-6][0-9](Z|[+-][0-9]{4})? |
[0-9]{4}-[0-1][0-9]-[0-3][0-9]T[0-2][0-9](Z|[+-][0-9]{4})?

----------/* V_ISO8601_EXTENDED_TIME hh:mm:ss[,sss][Z|+/-nnnn] */ --------
[0-2][0-9]:[0-6][0-9]:[0-6][0-9](,[0-9]+)?(Z|[+-][0-9]{4})? |
[0-2][0-9]:[0-6][0-9](Z|[+-][0-9]{4})?

----------/* V_ISO8601_EXTENDED_DATE YYYY-MM-DD */ ------------------------
[0-9]{4}-[0-1][0-9]-[0-3][0-9] |
[0-9]{4}-[0-1][0-9]

----------/* V_ISO8601_DURATION PnYnMnWnDTnnHnnMnnS */ -------------
P([0-9]+[yY])?([0-9]+[mM])?([0-9]+[wW])?([0-9]+[dD])?T([0-9]+[hH])?([0-
9]+[mM])?([0-9]+[sS])? |
P([0-9]+[yY])?([0-9]+[mM])?([0-9]+[wW])?([0-9]+[dD])?

----------/* V_TYPE_IDENTIFIER */ --------------------------------------
[A-Z]{IDCHAR}*

----------/* V_GENERIC_TYPE_IDENTIFIER */ -------------------------------
[A-Z]{IDCHAR}*<[a-zA-Z0-9,_<>]+>

----------/* V_ATTRIBUTE_IDENTIFIER */ ----------------------------------
[a-z]{IDCHAR}*
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
----------/* CADL Blocks */ ---------------------------------------------
\{[^{}]*
<IN_CADL_BLOCK>\{[^{}]*                                -- got an open brace
<IN_CADL_BLOCK>[^{}]*\}                                -- got a close brace

----------/* numbers */ ---------------------------------------------
[0-9]+\.[0-9]+                                         V_INTEGER
[0-9]+\.[0-9]+[eE][+-]?[0-9]+                          V_REAL

----------/* Strings */ ---------------------------------------------
\"[^\\\n"]*\"                                          V_STRING

---- strings containing quotes, special characters etc
\"[^\\\n"]*                        -- beginning of a string
<IN_STR>\\\\                       -- match escaped backslash
<IN_STR>\\\"                       -- match escaped double quote
{UTF8CHAR}+                        -- match UTF8 chars
<IN_STR>[^\\\n"]+                  -- match any other characters
<IN_STR>\\\n[ \t\r]*               -- match LF in line
<IN_STR>[^\n"]*\"                  -- match final end of string

<IN_STR>.|\n       |               -- Error
<IN_STR><<EOF>>                    -- unclosed String

----------/* V_CHARACTER */ -------------------------------------------
\'[^\\\n']\' -- normal character in 0-127
\'\\n\ -- \n
\'\\r\ -- \r
\'\\t\ -- \t
\'\\'\ -- \′
\'\\\\ -- \\
\'{UTF8CHAR}\' -- UTF8 char
\'.{1,2} |
\'\\[0-9]+(\/)? -- invalid character -> ERR_CHARACTER
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2 cADL — Constraint ADL

8.2.1 Overview (informative)

cADL is a syntax which enables constraints on data defined by object-oriented information models to be expressed in archetypes or other knowledge definition formalisms. It is most useful for defining the specific allowable constructions of data whose instances conform to very general object models. cADL is used both at design time, by authors and/or tools, and at runtime, by computational systems which validate data by comparing it to the appropriate sections of cADL in an archetype. The general appearance of cADL is illustrated by the following example:

```
PERSON[at0000] matches {                -- constraint on PERSON instance
    name matches {                      -- constraint on PERSON.name
        TEXT matches {/.+/}             -- any non-empty string
    }
    addresses cardinality matches {0..*} matches {    -- constraint on
        ADDRESS matches {                             -- PERSON.addresses
            -- etc --
        }
    }
}
```

Some of the textual keywords in this example can be more efficiently rendered using common mathematical logic symbols. In the following example, the **matches**, **exists** and **implies** keywords have been replaced by appropriate symbols:

```
PERSON[at0000] ∈ {                      -- constraint on PERSON instance
    name ∈ {                            -- constraint on PERSON.name
        TEXT ∈ {/..*/}                  -- any non-empty string
    }
    addresses cardinality ∈ {0..*} ∈ {  -- constraint on
        ADDRESS ∈ {                     -- PERSON.addresses
            -- etc --
        }
    }
}
```

The full set of equivalences appears below. Raw cADL is stored in the text-based form, to remove any difficulties with representation of symbols, to avoid difficulties of authoring cADL text in text editors which do not supply symbols, and to aid reading in English. However, the symbolic form might be more widely used due to the use of tools, and formatting in HTML and other documentary formats, and may be more comfortable for non-English speakers and those with formal mathematical backgrounds. cADL supports both conventions: the use of symbols or text is completely a matter of personal preference.

Literal leaf values (such as the regular expression `/..*/` in the above example) are always constraints on a set of standard primitive types. Other more sophisticated constraint syntax types are described under cADL — Constraint ADL on page 79.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.2 Basics

8.2.2.1 Keywords

The following keywords are recognised in cADL:

- **matches**, **~matches**, **is_in**, **~is_in**
- **occurrences**, **existence**, **cardinality**
- **ordered**, **unordered**, **unique**
- **infinity**
- **use_node**, **allow_archetype**[1)]
- **include**, **exclude**

Symbol equivalents for some of the above are given in the following table.

Textual rendering	Symbolic rendering	Meaning
matches, is_in	∈	Set membership, p is in P
not, ~	∼	Negation, not p

The **matches** or **is_in** operator is a key operator in cADL; it corresponds mathematically to set membership. When it occurs between a name and a block delimited by braces, the meaning is: the set of values allowed for the entity referred to by the name (either an object, or parts of an object — attributes) is specified between the braces. What appears between any matching pair of braces can be thought of as a specification for a set of values. Since blocks can be nested, this approach to specifying values can be understood in terms of nested sets, or in terms of a value space for objects of a set of defined types.

NOTE 1 In the following example, the **matches** operator links the name of an entity to a linear value space (i.e. a list), consisting of all words ending in ion.

```
aaa matches {/.*ion[^\s\n\t]/} -- the set of english words ending in ion
```

NOTE 2 The following example links the name of a type XXX with a complex multidimensional value space.

```
XXX matches {
    aaa matches {                  --
       YYY matches {0..3}          --
    }                              -- the value space of the
    bbb matches {                  -- and instance of XXX
       ZZZ matches {>1992-12-01}   --
    }                              --
}
```

1) The keyword was once use_archetype, which is now deprecated.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NOTE 3 Occasionally, the **matches** operator needs to be used in the negative, usually at a leaf block. Any of the following can be used to constrain the value space of XXX to any number except 5.

```
XXX ~matches {5}
XXX ~is_in {5}
XXX ∉{5}
```

8.2.2.2 Comments

In cADL, comments are indicated by the -- characters. Multiline comments are achieved using the -- leader on each line where the comment continues.

8.2.2.3 Information model identifiers

A type name is any identifier with an initial upper case letter, followed by any combination of letters, digits and underscores. A generic type name (including nested forms) may additionally include commas and angle brackets, but no spaces, and shall be syntactically correct as per the UML. An attribute name is any identifier with an initial lower case letter, followed by any combination of letters, digits and underscores.

8.2.2.4 Node identifiers

In cADL, an entity in brackets such as `[xxxx]` is used to identify object nodes, i.e. nodes expressing constraints on instances of some type. Object nodes always commence with a type name. Any string may appear within the brackets, depending on how it is used.

8.2.2.5 Natural language

cADL is independent of natural language. The only potential exception is where constraints include literal values from some language, which may be avoided by the use of separate language and terminology definitions. However, for the purposes of readability, comments in English have been included in this part of ISO 13606 in order to aid the reader.

8.2.3 Structure

8.2.3.1 General

cADL constraints are written in a block-structured style. The general structure is a nesting of constraints on types, followed by constraints on properties (of that type), followed by types (being the types of the attribute under which it appears), and so on. The term *object block* or *object node* refers to any block introduced by a type name (all in upper case), while an *attribute block* or *attribute node* is any block introduced by an attribute identifier (all in lower case).

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NOTE 1 A typical block resembles the following (the recurring pattern `/.+/` is a regular expression meaning non-empty string):

```
PERSON[at0001] ∈ {
    name ∈ {
        PERSON_NAME[at0002] ∈ {
            forenames cardinality ∈ {1..*} ∈ {/.+/}
            family_name ∈ {/.+/}
            title ∈ {Dr, Miss, Mrs, Mr, ...}
        }
    }
    addresses cardinality ∈ {1..*} ∈ {
        LOCATION_ADDRESS[at0003] ∈ {
            street_number existence ∈ {0..1} ∈ {/.+/}
            street_name ∈ {/.+/}
            locality ∈ {/.+/}
            post_code ∈ {/.+/}
            state ∈ {/.+/}
            country ∈ {/.+/}
        }
    }
}
```

NOTE 2 In the above, any identifier (shown in green) followed by the ∈ operator (equivalent text keyword: **matches** or **is_in**), followed by an open brace, is the start of a block, which continues until the closing matching brace (normally visually indented to come under the start of the line at the beginning of the block). The example expresses a constraint on an instance of the type **PERSON**; the constraint is expressed by everything inside the **PERSON** block. The two blocks at the next level define constraints on properties of **PERSON**, in this case *names* and *addresses*.

8.2.3.2 Complex objects

Constraints expressed in cADL cannot be stronger than those from the underlying information model being constrained by the archetype. Furthermore, a cADL text includes constraints *only for those parts of a model that are useful or meaningful to constrain*.

NOTE An example showing how to express a constraint on the *value* property of an **ELEMENT** class to be a **QUANTITY** with a suitable range for expressing blood pressure is as follows:

```
ELEMENT[at0010] matches {                    -- diastolic blood pressure
      value matches {
           QUANTITY matches {
              magnitude matches {0..1000}
              property matches {"pressure"}
              units matches {"mm[Hg]"}
           }
      }
}
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.3.3 Attribute constraints

8.2.3.3.1 General

In any underlying information model, attributes are either single-valued or multiple-valued, i.e. of a generic container type such as **`List<Contact>`**.

8.2.3.3.2 Existence

An existence constraint may be used directly after any attribute identifier, and indicates whether the object to which the attribute value refers is mandatory or optional in the data. The meaning of an existence constraint is to indicate whether the corresponding object or attribute is mandatory or optional in the instance data. The same logic applies whether the attribute is of single or multiple cardinality, i.e. whether it is a container or not. For container attributes, the existence constraint indicates whether the whole container (usually a list or set) is mandatory or not; a further *cardinality* constraint (described below) indicates how many members in the container are allowed. Existence is shown using the same constraint language as the rest of the archetype definition. Existence constraints may take the values **`{0}`**, **`{0..0}`**, **`{0..1}`**, **`{1}`**, or **`{1..1}`**. The default existence constraint, if none is shown, is **`{1..1}`**.

NOTE Existence constraints are expressed in cADL as in the following example:

```
QUANTITY matches {
    units existence matches {0..1} matches {mm[Hg]}
}
```

8.2.3.4 Single-valued attributes

Repeated blocks of object constraints of the same class (or its subtypes) may have two possible meanings in cADL, depending on whether the cardinality is present or not in the containing attribute block. Two or more object blocks introduced by type names appearing after an attribute that is not a container (i.e. for which there is no cardinality constraint) are taken to be alternative constraints, only one of which needs to be matched by the data.

EXAMPLE

```
ELEMENT[at0004] matches {                           -- speed limit
    value matches {
        QUANTITY matches {
            magnitude matches {|0..55|}
            property matches {"velocity"}
            units matches {"mph"}                   -- miles per hour
        }
        QUANTITY matches {
            magnitude matches {|0..100|}
            property matches {"velocity"}
            units matches {"km/h"}                  -- km per hour
        }
    }
}
```

NOTE Here, the cardinality of the *value* attribute is 1..1 (the default), while the occurrences of both **`QUANTITY`** constraints is optional, leading to the result that only one **`QUANTITY`** instance may appear in runtime data, and it may match either of the sets of constraints.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.3.5 Container attributes

8.2.3.5.1 Cardinality

Container attributes are indicated in cADL with the *cardinality* constraint. Cardinalities indicate limits for the number of members of container types such as lists and sets.

EXAMPLE 1

```
HISTORY[at0001] occurrences ∈ {1} ∈ {
    periodic ∈ {False}
    events cardinality ∈ {*} ∈ {
        EVENT[at0002] occurrences ∈ {0..1} ∈ {    }
               -- 1 min sample
        EVENT[at0003] occurrences ∈ {0..1} ∈ {    }
               -- 2 min sample
        EVENT[at0004] occurrences ∈ {0..1} ∈ {    }
               -- 3 min sample
    }
}
```

A cardinality constraint may be used after any attribute name (or after its existence constraint, if there is one) in order to indicate that the attribute refers to a container type, what number of member items it shall have in the data, and optionally, whether it has “list”, “set”, or “bag” semantics, via the use of the keywords ordered, unordered, unique and non-unique. An integer range is used to specify the valid membership of the container; a single * means the range 0..*, i.e. 0 to many.

EXAMPLE 2

```
events cardinality ∈ {*; ordered} ∈ {                -- logical list
events cardinality ∈ {*; unordered; unique} ∈ {      -- logical set
events cardinality ∈ {*; unordered} ∈ {              -- logical bag
```

Cardinality and existence constraints can co-occur in order to indicate various combinations on a container type property.

EXAMPLE 3 To specify that it is optional but, if present, is a container that may be empty:

```
events existence ∈ {0..1} cardinality ∈ {0..*} ∈ {-- etc --}
```

8.2.3.5.2 Occurrences

A constraint on occurrences may be used only with cADL object nodes (not attribute nodes), to indicate how many times in runtime data an instance of a given class conforming to a particular constraint can occur. It only has significance for objects which are children of a container attribute, since by definition, the occurrences of an object which is the value of a single-valued attribute may only be 0..1 or 1..1, and this is already defined by the attribute existence. However, it is not illegal. The default `occurrences`, if none is mentioned, is `{1..1}`.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

EXAMPLE 1 Below, three **EVENT** constraints are shown; the first one (1 min sample) is shown as mandatory, while the other two are optional.

```
events cardinality ∈ {*} ∈ {
   EVENT[at0002] occurrences ∈ {1..1} ∈ {    }      -- 1 min sample
   EVENT[at0003] occurrences ∈ {0..1} ∈ {    }      -- 2 min sample
   EVENT[at0004] occurrences ∈ {0..1} ∈ {    }      -- 3 min sample
  }
```

EXAMPLE 2 Expressed below is a constraint on instances of **GROUP** such that for **GROUPs** representing tribes, clubs and families, there shall only be one head, but there may be many members.

```
GROUP[at0103] ∈ {
    kind ∈ {/tribe|family|club/}
    members cardinality ∈ {*} ∈ {
       PERSON[at0104] occurrences ∈ {1} matches {
          title ∈ {head}
          -- etc --
       }
       PERSON[at0105] occurrences ∈ {0..*} matches {
          title ∈ {member}
          -- etc --
       }
    }
}
```

8.2.3.6 “Any” constraints

The “any” constraint is shown by a single asterisk (*) in accolades. It may be used to specify explicitly that some property may have any value.

EXAMPLE 1 Below, the “any” constraint on *name* means that any value permitted by the underlying information model is also permitted by the archetype; however, it also provides an opportunity to specify an existence constraint which might be narrower than that in the information model.

```
PERSON[at0001] matches {
    name existence matches {0..1} matches {*}
    -- etc --
}
```

The “any” constraint may also be used to specify that the value property of **ELEMENT** shall be of a particular data value type, but may have any compatible value.

EXAMPLE 2

```
ELEMENT[at0004] matches {                                -- speed limit
    value matches {
       QUANTITY matches {*}
    }
}
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.3.7 Object node identification and paths

Node identifiers are required for any object node that is intended to be addressable elsewhere in the cADL text, or in the runtime system and which would otherwise be ambiguous (i.e. has sibling nodes).

EXAMPLE 1

```
members cardinality ∈ {*} ∈ {
  PERSON[at0104] ∈ {
     title ∈ {"head"}
  }
  PERSON[at0105] matches {
     title ∈ {"member"}
  }
}
```

All nodes in a cADL text, which correspond to nodes in data that might be referred to from elsewhere in the archetype, or might be used for querying at runtime, require a node identifier. The node identifier might also be used to apply a design-time *meaning* to the node by equating the node identifier to some description.

Paths are used in cADL to refer to cADL nodes, and are expressed in the standard ADL path syntax, described in detail in 8.4. ADL paths have the same alternating object/attribute structure implied in the general hierarchical structure of cADL, which follows the pattern **TYPE/attribute/TYPE/attribute/**... Paths in cADL always refer to object nodes, and may only be constructed to nodes having node ids, or nodes which are the only child object of a single-cardinality attribute. The slash (/) separator shall always terminate a path.

Unusually for a path syntax, a trailing object identifier may be required, even if the property corresponds to a single relationship (as might be expected with the "name" property of an object), because in cADL it is legal to define multiple alternative object constraints – each identified by a unique node id – for a relationship node which has single cardinality.

EXAMPLE 2

```
HISTORY occurrences ∈ {1} ∈ {
    periodic ∈ {FALSE}
    events cardinality ∈ {*} ∈ {
        EVENT[at0002] occurrences ∈ {1..1} ∈ { }     -- 1 min sample
        EVENT[at0003] occurrences ∈ {0..1} ∈ { }     -- 2 min sample
        EVENT[at0004] occurrences ∈ {0..1} ∈ { }     -- 3 min sample
        }
    }
```

the following paths can be constructed:

```
/                           -- the HISTORY object
/periodic                   -- the HISTORY.periodic attribute
/events[at0002]             -- the 1 minute event object
/events[at0003]             -- the 2 minute event object
/events[at0004]             -- the 3 minute event object
```

It is valid to add attribute references to the end of a path, if the underlying information model permits.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

EXAMPLE 3

```
/events/count                -- count attribute of the items property
```

Physical paths may be converted to logical paths using descriptive meanings for node identifiers, if defined.

EXAMPLE 4 The following two paths are equivalent:

```
/events[at0004]                  -- the 3 minute event object
/events[3 minute event]          -- the 3 minute event object
```

To reference a cADL node in an archetype from elsewhere (e.g. another archetype of a template) requires that the identifier of the source archetype be prefixed to the path.

EXAMPLE 5

```
[openehr-ehr-entry.apgar-result.v1]/events[at0002]
```

8.2.3.8 Archetype internal references

It occurs reasonably often that one needs to include a constraint that is essentially a repeat of an earlier complex constraint, but within a different block. This is achieved using an archetype internal reference according to the following rule.

An archetype internal reference, to repeat the use of a previously defined complex constraint within the same archetype, is specified through the use_node keyword, in a line of the following form:

```
use_node TYPE object_path
```

EXAMPLE

```
PERSON [at0001] ∈ {
    identities ∈ {
        -- etc --
    }
    contacts cardinality ∈ {0..*} ∈ {
        CONTACT [at0002] ∈ {                        -- home address
            purpose ∈ {-- etc --}
            addresses ∈ {-- etc --}
        }
        CONTACT [at0003] ∈ {                        -- postal address
            purpose ∈ {-- etc --}
            addresses ∈ {-- etc --}
        }
        CONTACT [at0004] ∈ {                        -- home contact
            purpose ∈ {-- etc --}
            addresses cardinality ∈ {0..*} ∈ {
                ADDRESS [at0005] ∈ {                -- phone
                    type ∈ {-- etc --}
                    details ∈ {-- etc --}
                ADDRESS [at0006] ∈ {                -- fax
                    type ∈ {-- etc --}
                    details ∈ {-- etc --}
                }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                ADDRESS [at0007] ∈ {                    -- email
                    type ∈ {-- etc --}
                    details ∈ {-- etc --}
                }
            }
        }
        CONTACT [at0008] ∈ {                            -- work contact
            purpose ∈ {-- etc --}
            addresses cardinality ∈ {0..*} ∈ {
                use_node ADDRESS /[at0001]/contacts[at0004]/addresses[at0005]/ -- phone
                use_node ADDRESS /[at0001]/contacts[at0004]/addresses[at0006]/ -- fax
                use_node ADDRESS /[at0001]/contacts[at0004]/addresses[at0007]/ -- email
            }
        }
    }
```

8.2.3.9 Archetype slots

An archetype slot is introduced with the keyword allow_archetype, and is expressed using two lists of assertions, introduced with the keywords include and exclude, respectively. This allows other pre-existing archetypes to be used, rather than defining the desired constraints inline, and defines two lists of assertion statements defining which archetypes are allowed and/or which are excluded from filling that slot.

The slot might be wide, meaning it allows numerous other archetypes, or narrow, where it allows only a few or just one archetype. The point at where the slot occurs in the archetype is a *chaining point*.

EXAMPLE The following shows how the objective **SECTION** in a problem headings archetype defines two slots, indicating which **ENTRY** and **SECTION** archetypes are allowed and excluded under the *items* property.

```
SECTION [at2000] occurrences ∈ {0..1} ∈ {
                        -- objective
    items ∈ {
        allow_archetype ENTRY occurrences ∈ {0..1} ∈ {
            include
                concept_short_name ∈ {/.+/}
        }
        allow_archetype SECTION occurrences ∈ {0..*} ∈ {
            include
                id ∈ {/.*\.iso-ehr\.section\..*\..*/}
            exclude
                id ∈ {/.*\.iso-ehr\.section\.patient_details\..*/}
        }
    }
}
```

An archetype slot constraint may specify that the allowed archetype(s) shall contain a certain keyword or a certain path.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.3.10 Mixed structures

Three types of structure which represent constraints on complex objects have been presented so far:

- complex object structures: any node introduced by a type name and followed by { } containing constraints on attributes, invariants, etc;
- internal references: any node introduced by the keyword use_node, followed by a type name; such nodes stand for a complex object constraint that has already been expressed elsewhere in the archetype;
- archetype slots: any node introduced by the keyword allow_archetype, followed by a type name; such nodes stand for a complex object constraint that is expressed in some other archetype.

At any given node, all three types may co-exist.

EXAMPLE

```
SECTION[at2000] ∈ {
    items cardinality ∈ {0..*; ordered} ∈ {
       ENTRY[at2001] ∈ {-- etc --}
       allow_archetype ENTRY ∈ {-- etc --}
       use_node ENTRY [at0001]/some_path[at0004]/
       ENTRY[at2002] ∈ {-- etc --}
       use_node ENTRY /[at1002]/some_path[at1012]/
       use_node ENTRY /[at1005]/some_path[at1052]/
       ENTRY[at2003] ∈ {-- etc --}
    }
}
```

8.2.4 Constraints on primitive types

8.2.4.1 General

Constraints on attributes of primitive types in cADL may optionally be expressed without type names and omitting one level of braces.

EXAMPLE

```
some_attr matches {some_pattern}
```

rather than:

```
some_attr matches {
    PRIMITIVE_TYPE matches {
       some_pattern
    }
}
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.4.2 Constraints on string

8.2.4.2.1 General

Strings can be constrained in two ways: using a fixed string, and using a regular expression. All constraints on strings are case-sensitive.

8.2.4.2.2 List of strings

A string-valued attribute can be constrained by a list of strings (using the dADL syntax for string lists), including the simple case of a single string.

EXAMPLE

```
species matches {"platypus"}
species matches {"platypus", "kangaroo"}
species matches {"platypus", "kangaroo", "wombat"}
```

NOTE The first example constrains the runtime value of the species attribute of some object to take the value “platypus”; the second constrains it be either “platypus” or “kangaroo”, and so on. In almost all cases, this kind of string constraint should be avoided, since it renders the body of the archetype language-dependent, except for proper names, which are usually standardized internationally.

8.2.4.2.3 Regular expression

The second way of constraining strings is with regular expressions. The regular expression syntax used in cADL is a proper subset of that used in the Perl language. Three uses of it are accepted in cADL:

```
string_attr matches {/regular expression/}
string_attr matches {=~ /regular expression/}
string_attr matches {!~ /regular expression/}
```

The first two are identical, indicating that the attribute value shall match the supplied regular expression. The last indicates that the value shall *not* match the expression.

If the delimiter character is required in the pattern, it shall be quoted with the backslash (\) character, or else alternative delimiters may be used, enabling more comprehensible patterns.

NOTE A typical example is regular expressions including units; the following two patterns are equivalent:

```
units matches {/km\/h|mi\/h/}
units matches {^km/h|mi/h^}
```

The regular expression patterns supported in cADL are as follows.

Atomic items

match any single character.

E.g. `/ ... /` matches any three characters that occur with a space before and after;

`[xyz]` match any of the characters in the set **`xyz`** (case-sensitive).

E.g. `/[0-9]/` matches any string containing a single-decimal digit;

`[a-m]` match any of the characters in the set of characters formed by the continuous range from **`a`** to **`m`** (case-sensitive).

E.g. `/[0-9]/` matches any single character string containing a single-decimal digit, /[S-Z]/ matches any single character in the range `S - Z`;

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

`[^a-m]` match any character except those in the set of characters formed by the continuous range from `a` to `m`.

E.g. `/[^0-9]/` matches any single character string as long as it does not contain a single-decimal digit.

Grouping

`(pattern)` parentheses are used to group items; any pattern appearing within parentheses is treated as an atomic item for the purposes of the occurrences operators.

E.g. `/([0-9][0-9])/` matches any two-digit number.

Occurrences

`*` match 0 or more of the preceding atomic item.

E.g. `/.*/` matches any string; `/[a-z]*/` matches any non-empty lower-case alphabetic string;

`+` match 1 or more occurrences of the preceding atomic item.

E.g. `/a.+/` matches any string starting with "a", followed by at least one further character;

`?` match 0 or 1 occurrences of the preceding atomic item.

E.g. `/ab?/` matches the strings `a` and `ab`;

`{m,n}` match m to n occurrences of the preceding atomic item.

E.g. `/ab{1,3}/` matches the strings `ab` and `abb` and `abbb`; `/[a-z]{1,3}/` matches all lower-case alphabetic strings of one to three characters in length;

`{m,}` match at least m occurrences of the preceding atomic item;

`{,n}` match at most n occurrences of the preceding atomic item;

`{m}` match exactly m occurrences of the preceding atomic item.

Special character classes

`\d`, `\D` match a decimal digit character; match a non-digit character;

`\s`, `\S` match a whitespace character; match a non-whitespace character.

Alternatives

`pattern1|pattern2` match either pattern1 or pattern2.

E.g. `/lying|sitting|standing/` matches any of the words `lying`, `sitting` and `standing`.

8.2.4.3 Constraints on integer

Integers may be constrained with a single integer value, an integer interval, or a list of integers.

EXAMPLE

```
length matches {1000}          -- point interval of 1000 (=fixed value)
length matches {|950..1050|}   -- allow 950 - 1050
length matches {|0..1000|}     -- allow 0 - 1000
length matches {|0..<1000|}    -- allow 0 <= x < 1000
length matches {|0<..<1000|}   -- allow 0 < x < 1000
length matches {|<=10|}        -- allow up to 10
length matches {|>=10|}        -- allow 10 or more
length matches {|100+/-5|}     -- allow 100 +/- 5, i.e. 95 - 105
rate matches {|0..infinity|}   -- allow 0 - infinity, i.e. same as >= 0
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.4.4 Constraints on real

Constraints on real follow exactly the same syntax as for integers, except that all real numbers are indicated by the use of the decimal point and at least one succeeding digit, which may be 0.

EXAMPLE

```
magnitude matches {5.5}                -- fixed value
magnitude matches {|5.5|}              -- point interval (=fixed value)
magnitude matches {|5.5..6.0|}         -- interval
magnitude matches {5.5, 6.0, 6.5}      -- list
magnitude matches {|0.0..<1000.0|}     -- allow 0>= x <1000.0
magnitude matches {|>10.0|}            -- allow greater than 10.0
magnitude matches {|<=10.0|}           -- allow up to 10.0
magnitude matches {|>=10.0|}           -- allow 10.0 or more
magnitude matches {|80.0+/-12.0|}      -- allow 80 +/- 12
```

8.2.4.5 Constraints on Boolean

Boolean runtime values may be constrained to be true, false, or either, as follows:

```
some_flag matches {True}
some_flag matches {False}
some_flag matches {True, False}
```

8.2.4.6 Constraints on character

8.2.4.6.1 General

Characters may be constrained in two ways: using a list of characters, and using a regular expression.

8.2.4.6.2 Lists of characters

A character value may be constrained using a list of fixed character values. Each character is enclosed in single quotes.

EXAMPLE

```
color_name matches {‘r’}
color_name matches {‘r’, ‘g’, ‘b’}
```

8.2.4.6.3 Regular expressions

Character values may also be constrained using single-character regular expression elements, also enclosed in single quotes.

EXAMPLE

```
color_name matches {‘[rgbcmyk]’}
color_name matches {‘[^\s\t\n]’}
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The only allowed elements of the regular expression syntax in character expressions are the following:

— any item from the Atomic items list in 8.2.4.2.3;

— any item from the Special character classes list in 8.2.4.2.3;

— the character, standing for any character;

— an alternative expression whose parts are any item types, e.g. **a|b|[m-z]**

8.2.4.7 Constraints on dates, times and durations

8.2.4.7.1 General

Dates, times, date/times and durations may all be constrained in three ways: using a list of values; using intervals; using patterns.

8.2.4.7.2 Date, time and date/time

Patterns

Dates, times, and date/times (i.e. timestamps) may be constrained using patterns based on the ISO 8601 date/time syntax, which indicate which parts of the date or time shall be supplied. A constraint pattern is formed from the abstract pattern **yyyy-mm-ddThh:mm:ss** (itself formed by translating each field of an ISO 8601 date/time into a letter representing its type), with either "?" (meaning optional) or "X" (not allowed) characters substituted in appropriate places. A simplified grammar of the pattern is as follows (EBNF; all tokens shown are literals):

```
date_constraint:            yyyy - mm|??|XX - dd|??|XX
time_constraint:            hh : mm|??|XX : ss|??|XX
time_in_date_constraint:    T hh|??|XX : mm|??|XX : ss|??|XX
date_time_constraint:       date_constraint time_in_date_constraint
```

All expressions generated by this grammar shall also satisfy the validity rules:

— where "??" appears in a field, only "??" or "XX" may appear in fields to the right;

— where "XX" appears in a field, only "XX" may appear in fields to the right.

A fuller grammar can be defined to implement both the simplified grammar and validity rules.

The following table shows the valid patterns that may be used, and the types implied by each pattern.

Implied type	Pattern	Explanation
Date	yyyy-mm-dd	full date shall be specified
Date, partial date	yyyy-mm-??	optional day; e.g. day in month forgotten
Date, partial Date	yyyy-??-??	optional month, day; i.e. any date allowed; e.g. mental health questionnaires which include well-known historical dates
Partial date	yyyy-??-XX	optional month, no day; (any examples?)

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Time	Thh:mm:ss	full time shall be specified
Partial time	Thh:mm:XX	no seconds; e.g. appointment time
Partial time	Thh:??:XX	optional minutes, no seconds; e.g. normal clock times
Time, partial time	Thh:??:??	optional minutes, seconds; i.e. any time allowed
Date/time	yyyy-mm-ddThh:mm:ss	full date/time shall be specified
Date/time, Partial date/time	yyyy-mm-ddThh:mm:??	optional seconds; e.g. appointment date/time
Partial date/time	yyyy-mm-ddThh:mm:XX	no seconds; e.g. appointment date/time
Partial date/time	yyyy-mm-ddThh:??:XX	no seconds, minutes optional; e.g. in patient-recollected date/times
Date/time, Partial date/time Partial date/partial time	yyyy-??-??T??:??:??	minimum valid date/time constraint

Intervals

Dates, times and date/times may also be constrained using intervals. Each date, time, etc., in an interval may be a literal date, time, etc., value, or a value based on a pattern. In the latter case, the limit values are specified using the patterns from the above table, but with numbers in the positions where “X” and “?” do not appear.

EXAMPLE

```
|1995-??-XX|                                 -- any partial date in 1995
|09:30:00|                                   -- exactly 9:30 am
|< 09:30:00|                                 -- any time before 9:30 am
|<= 09:30:00|                                -- any time at or before 9:30 am
|> 09:30:00|                                 -- any time after 9:30 am
|>= 09:30:00|                                -- any time at or after 9:30 am
|2004-05-20..2004-06-02                      -- a date range
|2004-05-20T00:00:00..2005-05-19T23:59:59|   -- a date/time range
```

8.2.4.7.3 Duration constraints

Patterns

Patterns based on ISO 8601 may be used to constrain durations in the same way as for date/time types. The general form of a pattern is (EBNF; all tokens are literals):

```
P[Y|y][M|m][W|w][D|d][T[H|h][M|m][S|s]]
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Note that allowing the “W” designator to be used with the other designators corresponds to a deviation from the published ISO 8601. The “W” (week) designator may be used with the other designators, since it is very common to state durations of conditions or treatments as some combination of weeks and days. The use of this pattern indicates which “slots” in an ISO duration string may be filled. Where multiple letters are supplied in a given pattern, the meaning is “or”, i.e. any one or more of the slots may be supplied in the data.

EXAMPLE

```
:
Pd      -- a duration containing days only, e.g. P5d
Pm      -- a duration containing months only, e.g. P5m
PTm     -- a duration containing minutes only, e.g. PT5m
Pwd     -- a duration containing weeks and/or days only, e.g. P4w
PThm    -- a duration containing hours and/or minutes only, e.g. PT2h30m
```

Lists and intervals

Durations may also be constrained using absolute ISO 8601 values, or ranges of the same.

EXAMPLE

```
PT1m                 -- 1 minute
P1dT8h               -- 1 day, 8 hrs
|PT0m..PT1m30s|      -- Reasonable time offset of first Apgar sample
```

8.2.4.8 Constraints on lists of primitive types

In many cases, the type in the information model of an attribute to be constrained is a list or set of primitive types. This shall be indicated in cADL using the **cardinality** keyword (as for complex types), as follows:

```
some_attr cardinality matches {0..*} matches {some_pattern}
```

The pattern to match in the final accolades will have the meaning of a list or set of value constraints, rather than a single value constraint. Any constraint described above for single-valued attributes, which is commensurate with the type of the attribute in question, may be used. However, as with complex objects, the meaning is that every item in the list is constrained to be any one of the values implied by the constraint expression.

NOTE The following example constrains each value in the list corresponding to the value of the attribute *speed_limits* (of type `List<Integer>`) to be any one of the values 50, 60, 70, etc.

```
speed_limits cardinality matches {0..*; ordered} matches {50, 60, 70, 80,
100, 130}
```

8.2.4.9 Assumed values

Archetypes allow assumed values to be explicitly stated so that all users/systems know what value to assume when optional items are not included in the data. Assumed values are optionally definable on primitive types only, and are commenced with a semicolon followed by a value of the same type as that implied by the preceding part of the constraint.

EXAMPLE

```
length matches {|0..1000|; 200}               -- allow 0 - 1000, assume 200
some_flag matches {True, False; True }        -- allow T or F, assume T
some_date matches {yyyy-mm-dd hh:mm:XX; 1800-01-01 00:00:00}
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

If no assumed value is specified, no reliable assumption can be made by the receiver of archetyped data about what the values of removed optional parts might be, from inspecting the archetype.

8.2.5 cADL syntax

8.2.5.1 Grammar

This subclause defines the cADL grammar.

```
input:
  c_complex_object
| error

c_complex_object:
  c_complex_object_head SYM_MATCHES SYM_START_CBLOCK c_complex_object_body
c_invariants SYM_END_CBLOCK


c_complex_object_head:
  c_complex_object_id c_occurrences


c_complex_object_id:
  TYPE_IDENTIFIER
| TYPE_IDENTIFIER V_LOCAL_TERM_CODE_REF

c_complex_object_body:
  c_any
| c_attributes

c_object:
  c_complex_object
     | archetype_internal_ref
     | archetype_slot
     | constraint_ref
     | c_coded_term
     | c_ordinal
     | c_primitive_object
     | V_C_DOMAIN_TYPE
     | ERR_C_DOMAIN_TYPE
     | error

     archetype_internal_ref:
       SYM_USE_NODE TYPE_IDENTIFIER_C_OCCURRENCES object_path
     | SYM_USE_NODE TYPE_IDENTIFIER error

     archetype_slot:
       c_archetype_slot_head SYM_MATCHES SYM_START_CBLOCK c_includes c_excludes
     SYM_END_CBLOCK
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
c_archetype_slot_head:
  c_archetype_slot_id c_occurrences

c_archetype_slot_id:
  SYM_ALLOW_ARCHETYPE TYPE_IDENTIFIER
| SYM_ALLOW_ARCHETYPE TYPE_IDENTIFIER V_LOCAL_TERM_CODE_REF
| SYM_ALLOW_ARCHETYPE error

c_primitive_object:
  c_primitive

c_primitive:
  c_integer
| c_real
| c_date
| c_time
| c_date_time
| c_duration
| c_string
| c_boolean
| error

c_any:
  *


c_attributes:
  c_attribute
| c_attributes c_attribute

c_attribute:
  c_attr_head SYM_MATCHES SYM_START_CBLOCK c_attr_values SYM_END_CBLOCK

c_attr_head:
  V_ATTRIBUTE_IDENTIFIER c_existence
| V_ATTRIBUTE_IDENTIFIER c_existence c_cardinality

c_attr_values:
  c_object
| c_attr_values c_object
| c_any
| error

c_includes:
  -/-
```

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
| SYM_INCLUDE invariants

c_excludes:
  -/-
| SYM_EXCLUDE invariants

c_existence:
  -/-
| SYM_EXISTENCE SYM_MATCHES SYM_START_CBLOCK existence_spec SYM_END_CBLOCK

existence_spec:
  V_INTEGER
| V_INTEGER SYM_ELLIPSIS V_INTEGER

c_cardinality:
  SYM_CARDINALITY SYM_MATCHES SYM_START_CBLOCK cardinality_spec
SYM_END_CBLOCK

cardinality_spec:
  occurrence_spec
| occurrence_spec ; SYM_ORDERED
| occurrence_spec ; SYM_UNORDERED
| occurrence_spec ; SYM_UNIQUE
| occurrence_spec ; SYM_ORDERED ; SYM_UNIQUE
| occurrence_spec ; SYM_UNORDERED ; SYM_UNIQUE
| occurrence_spec ; SYM_UNIQUE ; SYM_ORDERED
| occurrence_spec ; SYM_UNIQUE ; SYM_UNORDERED

cardinality_limit_value:
  integer_value
| *

c_occurrences:
  -/-
| SYM_OCCURRENCES SYM_MATCHES SYM_START_CBLOCK occurrence_spec
SYM_END_CBLOCK
| SYM_OCCURRENCES error

occurrence_spec:
  cardinality_limit_value
| V_INTEGER SYM_ELLIPSIS cardinality_limit_value

c_integer_spec:
  integer_value
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
| integer_list_value
| integer_interval_value
| occurrence_spec

c_integer:
  c_integer_spec
| c_integer_spec ; integer_value
| c_integer_spec ; error

c_real_spec:
  real_value
| real_list_value
| real_interval_value

c_real:
  c_real_spec
| c_real_spec ; real_value
| c_real_spec ; error

c_date_constraint:
  V_ISO8601_DATE_CONSTRAINT_PATTERN
| date_value
| date_interval_value

c_date:
  c_date_constraint
| c_date_constraint ; date_value
| c_date_constraint ; error
c_time_constraint:
  V_ISO8601_TIME_CONSTRAINT_PATTERN
| time_value
| time_interval_value

c_time:
  c_time_constraint
| c_time_constraint ; time_value
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
| c_time_constraint ; error

c_date_time_constraint:
  V_ISO8601_DATE_TIME_CONSTRAINT_PATTERN
| date_time_value
| date_time_interval_value

c_date_time:
  c_date_time_constraint
| c_date_time_constraint ; date_time_value
| c_date_time_constraint ; error

c_duration_constraint:
V_ISO8601_DURATION_CONSTRAINT_PATTERN
| duration_value
| duration_interval_value

c_duration:
  c_duration_constraint
| c_duration_constraint ; duration_value
| c_duration_constraint ; error

c_string_spec:
  V_STRING
| string_list_value
| string_list_value , SYM_LIST_CONTINUE
| V_REGEXP

c_string:
  c_string_spec
| c_string_spec ; string_value
| c_string_spec ; error

c_boolean_spec:
  SYM_TRUE
| SYM_FALSE
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
| SYM_TRUE , SYM_FALSE
| SYM_FALSE , SYM_TRUE

c_boolean:
  c_boolean_spec
| c_boolean_spec ; boolean_value
| c_boolean_spec ; error

constraint_ref:
  V_LOCAL_TERM_CODE_REF

any_identifier:
 TYPE_IDENTIFIER
| V_ATTRIBUTE_IDENTIFIER
```

8.2.5.2 Symbols

This subclause defines the lexical specification for the cADL grammar.

```
----------/* definitions */ ---------------------------------------------
ALPHANUM [a-zA-Z0-9]
IDCHAR [a-zA-Z0-9_]
NAMECHAR [a-zA-Z0-9._\-]
NAMECHAR_SPACE [a-zA-Z0-9._\- ]
NAMECHAR_PAREN [a-zA-Z0-9._\-()]
UTF8CHAR (([\xC2-\xDF][\x80-\xBF])|(\xE0[\xA0-\xBF][\x80-\xBF])|([\xE1-
\xEF][\x80-\xBF][\x80-\xBF])|(\xF0[\x90-\xBF][\x80-\xBF][\x80-\xBF])|([\xF1-
\xF7][\x80-\xBF][\x80-\xBF][\x80-\xBF]))

----------/* comments */ -----------------------------------------------
"--".*                                          -- Ignore comments
"--".*\n[ \t\r]*

----------/* symbols */ ------------------------------------------------
"-"       Minus_code
"+"       Plus_code
"*"       Star_code
"/"       Slash_code
"^"       Caret_code
"="       Equal_code
"."       Dot_code
";"       Semicolon_code
","       Comma_code
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
":"        Colon_code
"!"        Exclamation_code
"("        Left_parenthesis_code
")"        Right_parenthesis_code
"$"        Dollar_code
"??"       SYM_DT_UNKNOWN
"?"        Question_mark_code
"|"        SYM_INTERVAL_DELIM
"["        Left_bracket_code
"]"        Right_bracket_code

"{"        SYM_START_CBLOCK
"}"        SYM_END_CBLOCK
".."       SYM_ELLIPSIS
"..."      SYM_LIST_CONTINUE

----------/* common keywords */ ------------------------------------------
[Mm][Aa][Tt][Cc][Hh][Ee][Ss]          SYM_MATCHES
[Ii][Ss]_[Ii][Nn]                     SYM_MATCHES

----------/* assertion keywords */ ------------------------------------------
--------

[Tt][Hh][Ee][Nn]                                        SYM_THEN

[Ee][Ll][Ss][Ee]                                        SYM_ELSE

[Aa][Nn][Dd]                                            SYM_AND

[Oo][Rr]                                                SYM_OR

[Xx][Oo][Rr]                                            SYM_XOR

[Nn][Oo][Tt]                                            SYM_NOT

[Ii][Mm][Pp][Ll][Ii][Ee][Ss]                            SYM_IMPLIES

[Tt][Rr][Uu][Ee]                                        SYM_TRUE

[Ff][Aa][Ll][Ss][Ee]                                    SYM_FALSE

[Ff][Oo][Rr][_][Aa][Ll][Ll]                             SYM_FORALL
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
[Ee][Xx][Ii][Ss][Tt][Ss]                          SYM_EXISTS

[Ee][Xx][Ii][Ss][Tt][Ee][Nn][Cc][Ee]              SYM_EXISTENCE

[Oo][Cc][Cc][Uu][Rr][Rr][Ee][Nn][Cc][Ee][Ss]      SYM_OCCURRENCES

[Cc][Aa][Rr][Dd][Ii][Nn][Aa][Ll][Ii][Tt][Yy]      SYM_CARDINALITY

[Oo][Rr][Dd][Ee][Rr][Ee][Dd]                      SYM_ORDERED

[Uu][Nn][Oo][Rr][Dd][Ee][Rr][Ee][Dd]              SYM_UNORDERED

[Uu][Nn][Ii][Qq][Uu][Ee]                          SYM_UNIQUE

[Ii][Nn][Ff][Ii][Nn][Ii][Tt][Yy]                  SYM_INFINITY

[Uu][Ss][Ee][_][Nn][Oo][Dd][Ee]                   SYM_USE_NODE

[Uu][Ss][Ee][_][Aa][Rr][Cc][Hh][Ee][Tt][Yy][Pp][Ee]     SYM_ALLOW_ARCHETYPE

[Aa][Ll][Ll][Oo][Ww][_][Aa][Rr][Cc][Hh][Ee][Tt][Yy][Pp][Ee] SYM_ALLOW_ARCHETYPE

[Ii][Nn][Cc][Ll][Uu][Dd][Ee]                      SYM_INCLUDE

[Ee][Xx][Cc][Ll][Uu][Dd][Ee]                      SYM_EXCLUDE

---------/* V_URI */ -------------------------------------------------
[a-z]+:\/\/[^<>|\\{}^~"\[\] ]*{


---------/* V_QUALIFIED_TERM_CODE_REF */ -----------------------------
-any qualified code, e.g. [local::at0001], [local::ac0001], [loinc::700-0]-
\[{NAMECHAR_PAREN}+::{NAMECHAR}+\]

\[{NAMECHAR_PAREN}+::{NAMECHAR_SPACE}+\]        -- error
---------/* V_TERM_CODE_CONSTRAINT of form */ -----------------------
-- [terminology_id::code, -- comment
--        code, -- comment
--   code] -- comment
--
-- Form with assumed value
-- [terminology_id::code, -- comment
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
--   code; -- comment
--   code] -- an optional assumed value
--
\[[a-zA-Z0-9()._\-]+::[ \t\n]* -- start IN_TERM_CONSTRAINT

<IN_TERM_CONSTRAINT> {

        [ \t]*[a-zA-Z0-9._\-]+[ \t]*;[ \t\n]*

              -- match second last line with ';' termination (assumed value)

        [ \t]*[a-zA-Z0-9._\-]+[ \t]*,[ \t\n]*

              -- match any line, with ',' termination

        \-\-[^\n]*\n     -- ignore comments

        [ \t]*[a-zA-Z0-9._\-]*[ \t\n]*\] -- match final line, terminating in ']'


---------/* V_LOCAL_TERM_CODE_REF */ --------------------------------
-- any unqualified code, e.g. [at0001], [ac0001], [700-0] --
\[{ALPHANUM}{NAMECHAR}*\]
----------/* V_LOCAL_CODE */ -----------------------------------------
a[ct][0-9.]+
---------/* V_QUALIFIED_TERM_CODE_REF */ ---------------------
-any qualified code, e.g. [local::at0001], [local::ac0001], [loinc::700-0]-
\[{NAMECHAR_PAREN}+::{NAMECHAR}+\]

\[{NAMECHAR_PAREN}+::{NAMECHAR_SPACE}+\]       -- error

\[[a-zA-Z------/* V_ISO8601_EXTENDED_DATE_TIME */ ---

-- YYYY-MM-DDThh:mm:ss[,sss][Z|+/-nnnn]

--

[0-9][a-zA-Z]{4}-[0-1][0-9._\-]*\]
----------/* V_LOCAL_CODE */ -----------------------------------------
a[ct]-[0-3][0-9.]+

---------/* V_QUALIFIED_TERM_CODE_REF */ ---------------------
-any qualified code, e.g. [local::at0001], [local::ac0001], [loinc::700-0]-
\[[a-zA-Z]T[0-2][0-9()._\-]+::[a-zA-Z]:[0-6][0-9._\-]+\] ]:[0-6][0-9](,[0-

9]+)?(Z|[+-][0-9]{4})? |

[0-9]{4}-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:[0-6][0-9](Z|[+-][0-9]{4})? |

[0-9]{4}-[0-1][0-9]-[0-3][0-9]T[0-2][0-9](Z|[+-][0-9]{4})?

----------/* V_ISO8601_EXTENDED_TIME */ --------
-- hh:mm:ss[,sss][Z|+/-nnnn]
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
--
[0-2][0-9]:[0-6][0-9]:[0-6][0-9](,[0-9]+)?(Z|[+-][0-9]{4})? |
[0-2][0-9]:[0-6][0-9](Z|[+-][0-9]{4})?
----------/* V_ISO8601_DATE YYYY-MM-DD */ -------------------
[0-9]{4}-[0-1][0-9]-[0-3][0-9] |
[0-9]{4}-[0-1][0-9]

----------/* V_ISO8601_DURATION */ -------------------------
P([0-9]+[yY])?([0-9]+[mM])?([0-9]+[wW])?([0-9]+[dD])?T([0-9]+[hH])?([0-9]+[mM])?([0-
9]+[sS])? |
P([0-9]+[yY])?([0-9]+[mM])?([0-9]+[wW])?([0-9]+[dD])?
----------/* V_ISO8601_DATE_CONSTRAINT_PATTERN */ -----------------
[yY][yY][yY][yY]-[mM?X][mM?X]-[dD?X][dD?X]
----------/* V_ISO8601_TIME_CONSTRAINT_PATTERN */ ------------------
 [hH][hH]:[mM?X][mM?X]:[sS?X][sS?X]
----------/* V_ISO8601_DATE_TIME_CONSTRAINT_PATTERN */ -------------
[yY][yY][yY][yY]-[mM?][mM?]-
[dD?X][dD?X][T][hH?X][hH?X]:[mM?X][mM?X]:[sS?X][sS?X]
----------/* V_ISO8601_DURATION_CONSTRAINT_PATTERN */ --------------
P[yY]?[mM]?[wW]?[dD]?T[hH]?[mM]?[sS]? |
P[yY]?[mM]?[wW]?[dD]?
----------/* V_TYPE_IDENTIFIER */ ---------------------------------------
[A-Z]{IDCHAR}*
Health informatics -- Vocabulary for terminological systems [A-Z]{IDCHAR}*<[a-
zA-Z0-9,_<>]+>
----------/* V_FEATURE_CALL_IDENTIFIER */ --------------------------
[a-z]{IDCHAR}*[ ]*\(\) ----------/* V_ATTRIBUTE_IDENTIFIER */ ----------------
------------------
[a-z]{IDCHAR}*
----------/* V_GENERIC_TYPE_IDENTIFIER */ -----------------------------
[A-Z]{IDCHAR}*<[a-zA-Z0-9,_<>]+>
----------/* V_ATTRIBUTE_IDENTIFIER */ --------------------------------
[a-z]{IDCHAR}*
----------/* V_C_DOMAIN_TYPE - sections of dADL syntax */ ------------------
{mini-parser specification}
[A-Z]{IDCHAR}*[ \n]*<        -- match a pattern like
                                  -- 'Type_Identifier whitespace <'
<IN_C_DOMAIN_TYPE>[^}>]*>[ \n]*[^>}A-Z]     -- match up to next > not
                                            -- followed by a '}' or '>'
<IN_C_DOMAIN_TYPE>[^}>]*>+[ \n]*[}A-Z]      -- final section - '...>
                                            -- whitespace } or beginning of
                                            -- a type identifier'
<IN_C_DOMAIN_TYPE>[^}>]*[ \n]*}             -- match up to next '}' not
                                            -- preceded by a '>'
----------/* V_REGEXP */ ------------------------------------
{mini-parser specification}
"{/"                       -- start of regexp
<IN_REGEXP1>[^/]*\\\/      -- match any segments with quoted slashes
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<IN_REGEXP1>[^/}]*\/      -- match final segment
\^[^^\n]*\^{              -- regexp formed using '^' delimiters
----------/* V_INTEGER */ ---------------------------------------------
[0-9]+
----------/* V_REAL */ ---------------------------------------------
[0-9]+\.[0-9]+
[0-9]+\.[0-9]+[eE][+-]?[0-9]+
----------/* V_STRING */ ---------------------------------------------
\"[^\\\n"]*\"
\"[^\\\n"]*{              -- beginning of a multiline string
<IN_STR> {
\\\\                      -- match escaped backslash, i.e. \\ -> \
\\\"                      -- match escaped double quote, i.e. \″ -> “
{UTF8CHAR}+               -- match UTF8 chars
[^\\\n"]+                 -- match any other characters
\\\n[ \t\r]*              -- match LF in line
[^\\\n"]*\"               -- match final end of string
.|\n |
<<EOF>>                   -- unclosed String -> ERR_STRING
}
```

8.3 Assertions

8.3.1 Overview

This subclause describes the assertion sublanguage of ADL archetypes. Assertions are used in archetype “slot” clauses in the cADL `definition` section, and in the `invariant` section.

8.3.2 Keywords

The syntax of the invariant section is a subset of first-order predicate logic, in which the following keywords may be used:

- `exists`, `for_all`,
- `and`, `or`, `xor`, `not`, `implies`
- `true`, `false`

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Symbol equivalents for some of the above are given in the following table.

Textual rendering	Symbolic rendering	Meaning
matches, is_in	∈	Set membership, p is in P
exists	∃	Existence quantifier, there exists ...
for_all	∀	Universal quantifier, for all x...
implies	⊃	Material implication, p implies q, or if p then q
and	∧	Logical conjunction, p and q
or	∨	Logical disjunction, p or q
xor	⊻	Exclusive or, only one of p or q
not, ~	∼	Negation, not p

The not operator may be applied as a prefix operator to all other operators except `for_all`; either textual rendering “not” or “~” may be used.

8.3.3 Operators

8.3.3.1 General

Assertion expressions may include arithmetic, relational and boolean operators, plus the existential and universal quantifiers.

8.3.3.2 Arithmetic operators

The supported arithmetic operators are as follows:

addition: +

subtraction: -

multiplication: *

division: /

exponent: ^

modulo division: % - remainder after integer division

8.3.3.3 Equality operators

The supported equality operators are as follows:

equality: =

inequality: <>

The semantics of these operators are of value comparison.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.3.3.4 Relational operators

The supported relational operators are as follows:

less than: <

less than or equal: <=

greater than: >

greater than or equal: >=

The semantics of these operators are of value comparison. Their domain is limited to values of comparable types.

8.3.3.5 Boolean operators

The supported Boolean operators are as follows:

not: **not**

and: **and**

xor: **xor**

implies: **implies**

set membership: **matches**, **is_in**

The Boolean operators also have the symbolic equivalents shown earlier.

8.3.3.6 Quantifiers

The two standard logical quantifier operators are supported:

existential quantifier: **exists**

universal quantifier: **for_all**

These operators also have the usual symbolic equivalents shown earlier.

8.3.4 Operands

Operands in an assertion expression may be any of the following:

manifest constant: any constant of any primitive type, expressed according to the dADL syntax for values

variable reference: any name starting with $, e.g. $body_weight;

property reference: a path referring to a property, i.e. any path ending in .property_name

object reference: a path referring to an object node, i.e. any path ending in a node identifier

If an assertion is used in an archetype slot definition, its operands refer to the archetype filling the slot, not the one containing the slot.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.3.5 Variables

Predefined variables

A number of predefined variables may be referenced in ADL assertion expressions, without prior definition, including

- **`$current_date`**: **`Date`**; returns the date whenever the archetype is evaluated
- **`$current_time`**: **`Time`**; returns time whenever the archetype is evaluated
- **`$current_datetime`**: **`Date_Time`**; returns date/time whenever the archetype is evaluated

Archetype-defined variables

Variables may also be defined inside an archetype, as part of the assertion statements in an invariant. The syntax of variable definition is as follows:

```
let $var_name = reference
```

Here, a reference may be any of the operand types listed above. “Let statements” may come anywhere in an invariant block, but, for readability, should generally come first.

NOTE The following example illustrates the use of variables in an invariant block:

```
invariant
    let $sys_bp =
       /data[at9001]/events[at9002]/data[at1000]/items[at1100]
    let $dia_bp =
       /data[at9001]/events[at9002]/data[at1000]/items[at1200]
    $sys_bp >= $dia_bp
```

8.3.6 Grammar

```
assertions:
  assertion
| assertions assertion

assertion:
  any_identifier : boolean_expression
| boolean_expression
| any_identifier : error

boolean_expression:
  boolean_leaf
| boolean_node

boolean_node:
  SYM_EXISTS absolute_path
| SYM_EXISTS error
| relative_path SYM_MATCHES SYM_START_CBLOCK c_primitive SYM_END_CBLOCK
| SYM_NOT boolean_leaf
| arithmetic_expression = arithmetic_expression
| arithmetic_expression SYM_NE arithmetic_expression
| arithmetic_expression SYM_LT arithmetic_expression
| arithmetic_expression SYM_GT arithmetic_expression
| arithmetic_expression SYM_LE arithmetic_expression
| arithmetic_expression SYM_GE arithmetic_expression
| boolean_expression SYM_AND boolean_expression
| boolean_expression SYM_OR boolean_expression
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
  | boolean_expression SYM_XOR boolean_expression
  | boolean_expression SYM_IMPLIES boolean_expression

boolean_leaf:
    ( boolean_expression )
  | SYM_TRUE
  | SYM_FALSE

arithmetic_expression:
  arithmetic_leaf
  | arithmetic_node

arithmetic_node:
    arithmetic_expression + arithmetic_leaf
  | arithmetic_expression - arithmetic_leaf
  | arithmetic_expression * arithmetic_leaf
  | arithmetic_expression / arithmetic_leaf
  | arithmetic_expression ^ arithmetic_leaf

arithmetic_leaf:
    ( arithmetic_expression )
  | integer_value
  | real_value
  | absolute_path
```

8.4 ADL paths

8.4.1 Overview

The notion of paths is integral to ADL, and a common path syntax is used to reference nodes in both dADL and cADL sections of an archetype. The same path syntax works for both, because both dADL and cADL have an alternating object/attribute structure. However, the interpretation of path expressions in dADL and cADL differs slightly; the differences are explained in the relevant subclauses of this part of ISO 13606. This subclause describes only the common syntax and semantics.

The general form of the path syntax is as follows:

[/][object_id/]{attr_name[object_id]/}*

ADL paths are formed from an alternation of segments made up of an attribute name and optional object node identifier predicate, separated by slash ("/") characters. Node identifiers are delimited by brackets ([ ]). A path either finishes in a slash, and identifies an object node, or finishes in an attribute name, and identifies an attribute node.

Paths are either absolute or relative to the node in which they are mentioned. Absolute paths always commence with an initial slash character.

8.4.2 Path syntax

8.4.2.1 Grammar

```
input:
    movable_path
  | absolute_path
  | relative_path
  | error
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
movable_path:
  SYM_MOVABLE_LEADER relative_path

absolute_path:
  / relative_path
| absolute_path / relative_path

relative_path:
  path_segment
| relative_path / path_segment

path_segment:
  V_ATTRIBUTE_IDENTIFIER V_LOCAL_TERM_CODE_REF
| V_ATTRIBUTE_IDENTIFIER
```

8.4.2.2 Symbols

```
"."          Dot_code
"/"          Slash_code

"["          Left_bracket_code
"]"          Right_bracket_code
"//"                                          SYM_MOVABLE_LEADER

\[[a-zA-Z0-9][a-zA-Z0-9._\-]*\]               V_LOCAL_TERM_CODE_REF

[A-Z][a-zA-Z0-9_]*                            V_TYPE_IDENTIFIER

[a-z][a-zA-Z0-9_]*[ ]*\(\)                    V_FEATURE_CALL_IDENTIFIER

[a-z][a-zA-Z0-9_]*                            V_ATTRIBUTE_IDENTIFIER
```

8.5 ADL — Archetype definition language

8.5.1 General

This subclause describes ADL archetypes as a whole, adding a small amount of detail to the descriptions of dADL and cADL already given. The important topic of the relationship of the cADL-encoded **definition** section and the dADL-encoded **ontology** section is discussed in detail.

An ADL archetype follows the structure shown below:

```
archetype
    archetype_id
[specialize
    parent_archetype_id]
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
concept
    coded_concept_name
language
    dADL language description section
description
    dADL metadata section
definition
    cADL structural section
invariant
    assertions
ontology
    dADL definitions section
[revision_history
    dADL section]
```

8.5.2 Basics

8.5.2.1 Keywords

ADL has a small number of keywords that are reserved for use in archetype declarations as follows:

- **archetype**, **specialise**/**specialize**, **concept**,
- **description**, **definition**, **ontology**

All of these words may safely appear as identifiers in the **definition** and **ontology** sections.

8.5.2.2 Node identification

In the **definition** section of an ADL archetype, a particular scheme of codes is used for node identifiers as well as for denoting constraints on textual (i.e. language-dependent) items. Codes are either local to the archetype, or from an external lexicon. This means that the archetype description is the same in all languages, and is available in any language into which the codes have been translated. All term codes are shown in brackets (`[]`). Codes used as node identifiers and defined within the same archetype are prefixed with at, and by convention have four digits, e.g. `[at0010]`. Codes of any length are acceptable in ADL archetypes. Specializations of locally coded concepts have the same root, followed by dot extensions, e.g. `[at0010.2]`. From a terminology point of view, these codes have no implied semantics — the dot structuring is used as an optimization on node identification.

8.5.2.3 Local constraint codes

A second kind of local code is used to stand for constraints on textual items in the body of the archetype. Although these could be included in the main archetype body, because they are language- and/or terminology-sensitive, they are defined in the ontology section, and referenced by codes prefixed by “ac”, e.g. `[ac0009]`. As for “at” codes, the convention used in this part of ISO 13606 is to use four-digit “ac” codes, even though any number of digits is acceptable. The use of these codes is described in 8.5.6.4.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.5.3 Header sections

8.5.3.1 Archetype section

This section introduces the archetype and shall include an identifier. A multi-axial identifier identifies archetypes in a global space.

NOTE A typical **archetype** section is as follows:

```
archetype (adl_version=1.4)
    mayo.openehr-ehr-entry.haematology.v1
```

8.5.3.2 Controlled indicator

A flag indicating whether the archetype is change-controlled or not may be included after the version; for example,

```
archetype (adl_version=1.4; controlled)
    mayo.openehr-ehr-entry.haematology.v1
```

This flag may have the two values “controlled” and “uncontrolled” only, and is an aid to software. Archetypes that include the “controlled” flag should have the revision history section included, while those with the “uncontrolled” flag, or no flag at all, may omit the revision history. This enables archetypes to be privately edited in an early development phase without generating large revision histories of little or no value.

8.5.3.3 Specialize section

This optional section indicates that the archetype is a specialization of some other archetype, whose identity shall be given. Only one specialization parent is allowed, i.e. an archetype cannot multiply-inherit from other archetypes.

NOTE An example of declaring specialization is as follows, in which the identifier of the new archetype is derived from that of the parent by adding a new section to its domain concept section:

```
archetype (adl_version=1.4)
    mayo.openehr-ehr-entry.haematology-cbc.v1
specialise
    mayo.openehr-ehr-entry.haematology.v1
```

Both the United States English and British English versions of the word specialize/specialise are valid in ADL.

8.5.3.4 Concept section

All archetypes represent a real-world concept, such as a “patient”, “blood pressure”, or an “antenatal examination”. The concept is always coded, ensuring that it can be displayed in any language to which the archetype has been translated.

NOTE In this example, the term definition of **[at0010]** is the proper description corresponding to the **haematology-cbc** section of the archetype id above:

```
concept
    [at0010]            -- haematology result
```

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.5.3.5 Language section and language translation

The **language** section includes data describing the original language in which the archetype was authored (essential for evaluating natural language quality), and the total list of languages available in the archetype. There may be only one **original_language**. The **translations** list shall be updated every time a translation of the archetype is incorporated.

EXAMPLE

```
language
    original_language = <“en”>
    translations = <
       [“de”] = <
          provenance = <"freddy@something.somewhere.co.uk">
          quality_control = <"British Medical Translator id 00400595”>
       >
       [“ru”] = <
          provenance = <"vladimir@something.somewhere.ru">
          quality_control = <"Russion Translator id 892230A”>
    >
```

Archetypes shall always be translated completely, or not at all. This means that when a new translation is made, every language-dependent section of the **description** and **ontology** sections shall be translated into the new language, and an appropriate addition made to the **translations** list in the language section.

8.5.3.6 Description section

The **description** section of an archetype contains descriptive information (sometimes called document metadata) such as items that can be used in repository indexes and for searching. The dADL syntax is used for the description.

EXAMPLE

```
description
    original_author = <
       [“name”] = <"Dr J Joyce">
       [“organisation”] = <"NT Health Service">
       [“date”] = <2003-08-03>
    >
    lifecycle_state =                       <"initial">
    archetype_package_uri =
       <"www.aihw.org.au/data_sets/diabetic_archetypes.html">

    details = <
       [“en”] = <
          purpose = <"archetype for diabetic patient review">
          use = <"used for all hospital or clinic-based diabetic reviews,
             including first time. Optional sections are removed according
             to the particular review">
          misuse = <"not appropriate for pre-diagnosis use">
          original_resource_uri =
             <"www.healthdata.org.au/data_sets/
                                        diabetic_review_data_set_1.html">
          other_details =                 <...>
       >
       [“de”] = <
          purpose = <"Archetyp für die Untersuchung von Patienten
                  mit Diabetes">
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                    use = <"wird benutzt für alle Diabetes-Untersuchungen im
                          Krankenhaus, inklusive der ersten Vorstellung. Optionale
                          Abschnitte werden in Abhängigkeit von der speziellen
                          Vorstellung entfernt."
                    >
                    misuse = <"nicht geeignet für Benutzung vor Diagnosestellung">
                    original_resource_uri =
                      <"www.healthdata.org.au/data_sets/
                                           diabetic_review_data_set_1.html">
                    other_details =          <...>
                  >
                >
```

8.5.4 Definition section

The **definition** section contains the main formal definition of the archetype, and is written in the Constraint Definition Language (cADL).

EXAMPLE

```
definition
    ENTRY[at0000] ∈ {                                    -- blood pressure measurement
        name ∈ {                                         -- any synonym of BP
            CODED_TEXT ∈ {
                code ∈ {
                    CODE_PHRASE ∈ {[ac0001]}
                }
            }
        }
        data ∈ {
            HISTORY[at9001] ∈ {                          -- history
                events cardinality ∈ {1..*} ∈ {
                    EVENT[at9002] occurrences ∈ {0..1} ∈ {       -- baseline
                        name ∈ {
                            CODED_TEXT ∈ {
                                code ∈ {
                                    CODE_PHRASE ∈ {[ac0002]}
                                }
                            }
                        }
                        data ∈ {
                            LIST_S[at1000] ∈ {           -- systemic arterial BP
                                items cardinality ∈ {2..*} ∈ {
                                    ELEMENT[at1100] ∈ {  -- systolic BP
                                        name ∈ {         -- any synonym of 'systolic'
                                            CODED_TEXT ∈ {
                                                code ∈ {
                                                  CODE_PHRASE ∈ {[ac0002]}
                                                }
                                            }
                                        }
                                        value ∈ {
                                            QUANTITY ∈ {
                                                magnitude ∈ {0..1000}
                                                property ∈ {[properties::0944]}   -- "pressure"
                                                units ∈ {[units::387]}        -- "mm[Hg]"
                                            }
                                        }
                                    }
                                    ELEMENT[at1200] ∈ {  -- diastolic BP
                                        name ∈ {         -- any synonym of 'diastolic'
                                            CODED_TEXT ∈ {
                                                code ∈ {
                                                  CODE_PHRASE ∈ {[ac0003]}
                                                }
                                        }
                                        }
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                    value ∈ {
                        QUANTITY ∈ {
                            magnitude ∈ {0..1000}
                            property ∈ {[properties::0944]}    -- "pressure"
                            units ∈ {[units::387]}           -- "mm[Hg]"
                        }
                    }
                }
                ELEMENT[at9000] occurrences ∈ {0..*} □ {*}
                                    -- unknown new item
            }
        ...
```

8.5.5 Invariant section

The `invariant` section in an ADL archetype introduces assertions which relate to the entire archetype, and may be used to make statements that are not possible within the block structure of the `definition` section. Any constraint which relates more than one property to another is in this category, as are most constraints containing mathematical or logical formulae. An invariant statement is a first-order predicate logic statement which can be evaluated to a Boolean result at runtime. Objects and properties are referred to using paths.

EXAMPLE

```
invariant
    validity: /[at0001]/speed[at0002]/kilometres/magnitude =
        /[at0003]/speed[at0004]/miles/magnitude * 1.6
```

8.5.6 Ontology section

8.5.6.1 Overview

The **ontology** section of an archetype is expressed in dADL, and is where codes representing node IDs, constraints on text or terms, and bindings to terminologies are defined. Linguistic language translations are added in the form of extra blocks keyed by the relevant language.

EXAMPLE

```
ontology
    terminologies_available = <"snomed_ct", ...>

    term_definitions = <
        ["en"] = <
            items = <...>
        >
        ["de"] = <
            items = <...>
        >
    >

    term_binding = <
        ["snomed_ct"] = <
            items = <...>
        >
        ...
    >

    constraint_definitions = <
        ["en"] = <...>
```


© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        ["de"] = <
            items = <...>
        >
        ...
    >

    constraint_binding = <
        ["snomed_ct"] = <...>
        ...
    >
```

The **term_definitions** section is mandatory, and shall be defined for each translation carried out.

Each of these sections may have its own metadata, which appears within description subsections, such as the one shown above providing translation details.

8.5.6.2 Ontology header statements

The “terminologies_available” statement includes the identifiers of all terminologies for which **term_binding** sections have been written.

8.5.6.3 Term_definition section

This section is where all archetype local terms (that is, terms of the form **[atNNNN]**) are defined. Each term is defined using a structure of name-value pairs, and shall at least include the names “text” and “description”. Each term object is then included in the appropriate language list of term_definitions.

NOTE 1 The following example shows an extract from the English and German term_definitions for the archetype local terms in a problem/SOAP headings archetype.

```
term_definitions = <
    ["en"] = <
        items = <
            ["at0000"] = <
                text = <"problem">
                description = <"The problem experienced by the subject of care
                                to which the contained information relates">
            >
            ["at0001"] = <
                text = <"problem/SOAP headings">
                description = <"SOAP heading structure for multiple problems">
            >
            ...
            ["at4000"] = <
                text = <"plan">
                description = <"The clinician's professional advice">
            >
        >
    >
    ["de"] = <
        items = <
            ["at0000"] = <
                text = <"klinisches Problem">
                description = <"Das Problem des Patienten worauf sich diese \
                                                  Informationen beziehen">
            >
            ["at0001"] = <
                text = <"Problem/SOAP Schema">
                description = <"SOAP-Schlagwort-Gruppierungsschema fuer
                                                  mehrfache Probleme">
            >
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                ["at4000"] = <
                   text = <"Plan">
                   description = <"Klinisch-professionelle Beratung des
                                                         Pflegenden">
                >
             >
          >
        >
```

A provenance tag may be used to indicate the source of term definitions.

EXAMPLE

```
                [at4000] = <
                   text = <"plan">;
                   description = <"The clinician's professional advice">;
                   provenance = <"ACME_terminology(v3.9a)">
                >
```

NOTE 2 This example does not indicate a *binding* to any term, only its origin. Bindings are described in 8.5.6.5 and 8.5.6.6.

8.5.6.4 Constraint_definition section

The **constraint_definition** section is of exactly the same form as the **term_definition** section, and provides the definitions; i.e. the meanings of the local constraint codes, which are of the form **[acNNNN]**. The constraint definitions do not incorporate the constraints themselves, but define the *meanings* of such constraints. The actual constraints are defined in the **constraint_binding** section.

EXAMPLE

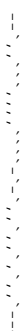
```
     items = <
          [ac1015] = <
             text = <"type of hepatitis">
             description = <"any term which means a kind of viral hepatitis">
          >
     >
```

8.5.6.5 Term_binding section

This section is used to describe the equivalences between archetype local terms and terms found in external terminologies. Each mapping expression indicates which term in an external terminology is equivalent to the archetype internal codes.

EXAMPLE

```
     term_binding(umls) = <
          [umls] = <
             items =<
                [at0000] = <[umls::C124305]> -- apgar result
                [at0002] = <[umls::0000000]> -- 1-minute event
                [at0004] = <[umls::C234305]> -- cardiac score
                [at0005] = <[umls::C232405]> -- respiratory score
                [at0006] = <[umls::C254305]> -- muscle tone score
                [at0007] = <[umls::C987305]> -- reflex response score
```



© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
                [at0008] = <[umls::C189305]> -- color score
                [at0009] = <[umls::C187305]> -- apgar score
                [at0010] = <[umls::C325305]> -- 2-minute apgar
                [at0011] = <[umls::C725354]> -- 5-minute apgar
                [at0012] = <[umls::C224305]> -- 10-minute apgar
            >
        >
    >
```

8.5.6.6 Constraint_binding section

This section formally describes the text constraints within the main archetype body. They are described separately because they are terminology-dependent, and because there may be more than one for a given logical constraint.

EXAMPLE

```
    constraint_binding = <
      [“snomed_ct”]
         items = <
            [“ac0001”] = <http://terminology.org?terminology_id=snomed_ct&&
                           has_relation=[102002];with_target=[128004]>
            [“ac0002”] = <http://terminology.org?terminology_id=snomed_ct&&
                           synonym_of=[128025]>
         >
      >
    >
```

8.5.7 Revision_history section

The revision_history section of an archetype shows the audit history of changes to the archetype, and is expressed in dADL syntax. It is optional, and is included at the end of the archetype.

EXAMPLE

```
  revision_history
      revision_history = <
         ["1.57"] = <
            committer = <"Miriam Hanoosh">
            committer_organisation = <"AIHW.org.au">
            time_committed = <2004-11-02 09:31:04+1000>
            revision = <"1.2">
            reason = <"Added social history section">
            change_type = <"Modification">
         >
         -- etc
         ["1.1"] = <
            committer = <"Enrico Barrios">
            committer_organisation = <"AIHW.org.au">
            time_committed = <2004-09-24 11:57:00+1000>
            revision = <"1.1">
            reason = <"Updated HbA1C test result reference">
            change_type = <"Modification">
         >
         ["1.0"] = <
            committer = <"Enrico Barrios">
            committer_organisation = <"AIHW.org.au">
```

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                time_committed = <2004-09-14 16:05:00+1000>
                revision = <"1.0">
                reason = <"Initial Writing">
                change_type = <"Creation">
            >
        >
```

8.5.8 Validity rules

8.5.8.1 General

This subclause describes the formal (i.e. checkable) semantics of ADL archetypes.

8.5.8.2 Global archetype validity

The following validity constraints apply to an archetype as a whole.

NOTE The term “section” means the same as “attribute” in the following, i.e. a section called “definition” in a dADL text is a serialization of the value for the attribute of the same name.

VARID: archetype identifier validity. The archetype shall have an identifier value for the archetype_id section.

VARCN: archetype concept validity. The archetype shall have an archetype term value in the concept section. The term shall exist in the archetype ontology.

VARDF: archetype definition validity. The archetype shall have a definition section, expressed as a cADL syntax string, or in an equivalent plug-in syntax.

VARON: archetype ontology validity. The archetype shall have an ontology section, expressed as a cADL syntax string, or an equivalent plug-in syntax.

VARDT: archetype definition typename validity. The topmost typename mentioned in the archetype definition section shall match the type mentioned in the type- name-slot of the first segment of the archetype id.

8.5.8.3 Coded term validity

All node identifiers (“at” codes) used in the `definition` section of the archetype shall be defined in the `term_definitions` section of the ontology.

VATDF: archetype term validity. Each archetype term used as a node identifier the archetype definition shall be defined in the term_definitions section of the ontology.

All constraint identifiers (“ac” codes) used in the `definition` section of the archetype shall be defined in the `constraint_definitions` section of the ontology.

VACDF: node identifier validity. Each constraint code used in the archetype definition shall be defined in the constraint_definitions section of the ontology.

8.5.8.4 Definition section

The following constraints apply to the `definition` section of the archetype.

VDFPT: **path validity in definition**. Any path mentioned in the definition section shall be valid syntactically, and have a valid path with respect to the hierarchical structure of the definition section.

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.5.9 Archetype syntax

8.5.9.1 Grammar

```
input:
  archetype
| error

archetype:
  arch_identification arch_specialisation arch_concept arch_description
arch_definition arch_invariant arch_ontology

arch_identification:
  arch_head V_ARCHETYPE_ID
| SYM_ARCHETYPE error

arch_head:
  SYM_ARCHETYPE
| SYM_ARCHETYPE arch_meta_data

arch_meta_data:
  ( arch_meta_data_items )

arch_meta_data_items:
  arch_meta_data_item
| arch_meta_data_items ; arch_meta_data_item

arch_meta_data_item:
  SYM_ADL_VERSION = V_VERSION_STRING
| SYM_IS_CONTROLLED

arch_specialisation:
  -/-
| SYM_SPECIALIZE V_ARCHETYPE_ID
| SYM_SPECIALIZE error

arch_concept:
  SYM_CONCEPT V_LOCAL_TERM_CODE_REF
| SYM_CONCEPT error

arch_description:
  -/-
| SYM_DESCRIPTION V_DADL_TEXT
| SYM_DESCRIPTION error

arch_definition:
  SYM_DEFINITION V_CADL_TEXT
| SYM_DEFINITION error

arch_invariant:
  -/-
| SYM_INVARIANT V_ASSERTION_TEXT
| SYM_INVARIANT error

arch_ontology:
  SYM_ONTOLOGY V_DADL_TEXT
| SYM_ONTOLOGY error
```

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.5.9.2 Symbols

```
----------/* symbols */ ----------------------------------------------
"-"        Minus_code
"+"        Plus_code
"*"        Star_code
"/"        Slash_code
"^"        Caret_code
"="        Equal_code
"."        Dot_code
";"        Semicolon_code
","        Comma_code
":"        Colon_code
"!"        Exclamation_code
"("        Left_parenthesis_code
")"        Right_parenthesis_code
"$"        Dollar_code
"?"        Question_mark_code
"["        Left_bracket_code
"]"        Right_bracket_code

----------/* keywords */ ----------------------------------------------
^[Aa][Rr][Cc][Hh][Ee][Tt][Yy][Pp][Ee][ \t\r]*\n            SYM_ARCHETYPE
^[Ss][Pp][Ee][Cc][Ii][Aa][Ll][Ii][SsZz][Ee][ \t\r]*\n      SYM_SPECIALIZE
^[Cc][Oo][Nn][Cc][Ee][Pp][Tt][ \t\r]*\n                    SYM_CONCEPT
^[Dd][Ee][Ff][Ii][Nn][Ii][Tt][Ii][Oo][Nn][ \t\r]*\n        SYM_DEFINITION
  -- mini-parser to generate V_DADL_TEXT

^[Dd][Ee][Ss][Cc][Rr][Ii][Pp][Tt][Ii][Oo][Nn][ \t\r]*\n   SYM_DESCRIPTION
  -- mini-parser to generate V_CADL_TEXT

^[Ii][Nn][Vv][Aa][Rr][Ii][Aa][Nn][Tt][ \t\r]*\n             SYM_INVARIANT
  -- mini-parser to generate V_ASSERTION_TEXT

^[Oo][Nn][Tt][Oo][Ll][Oo][Gg][Yy][ \t\r]*\n                 SYM_ONTOLOGY
  -- mini-parser to generate V_DADL_TEXT

---------/* term code reference */ -------------------------------------
\[[a-zA-Z0-9][a-zA-Z0-9.-]*\]                         V_LOCAL_TERM_CODE_REF

----------/* archetype id */ ---------------------------------------------
[a-zA-Z][a-zA-Z0-9_-]+\.[a-zA-Z][a-zA-Z0-9_-]+\.[a-zA-Z0-9]+
                                                      V_ARCHETYPE_ID

----------/* identifiers */ ----------------------------------------------
[a-zA-Z][a-zA-Z0-9_]*                                 V_IDENTIFIER
```


© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Bibliography

[1] EN 13940-1, *Health informatics — System of concepts to support continuity of care — Part 1: Basic concepts*

[2] CEN/TS 14796, *Health informatics — Data types*

[3] ISO 3166 (all parts), *Codes for the representation of names of countries and their subdivisions*

[4] ISO/IEC 10746-1:1998, *Information technology — Open distributed processing — Reference model: Overview — Part 1*

[5] EN 14822-2:2005, *Health informatics — General purpose information components — Part 2: Non-clinical*

[6] ISO 1087-1:2000, *Terminology work — Vocabulary — Part 1: Theory and application*

[7] ISO/IEC 11179-3:2003, *Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes*

[8] ISO/IEC 11404, *Information technology — General-Purpose Datatypes (GPD)*

[9] ISO/TS 18308:2004, *Health informatics — Requirements for an electronic health record architecture*

[10] ISO/TR 20514:2005, *Health informatics — Electronic health record — Definition, scope and context*

[11] RFC 1738:2004, *Uniform Resource Locators (URL)*

[12] RFC 2045:1996, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*

[13] RFC 2046:1996, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*

[14] RFC 2806:2000, *URLs for Telephone Calls*

[15] RFC 2936:2000, *HTTP MIME Type Handler Detection*

[16] RFC 2978:2000, *IANA Charset Registration Procedures*

Example background R&D projects that informed the archetype approach:

[17] MOORMAN, P.W., VAN GINNEKEN, A.M., VAN DER LEI, J. and VAN BEMMEL, J.H., *A model for structured data entry based on explicit descriptional knowledge*, Methods of Information in Medicine, **33**(5), pp 454-63, December 1994

[18] DORE, L., LAVRIL, M., JEAN, F.C. and DEGOULET P.A., *Development environment to create medical applications*, GREENES, R.A. et al., eds., Medinfo, **8**, pp 185-189, 1995

[19] KALRA D., ed., *Synapses ODP Information Viewpoint*, EU Telematics Application Programme, Brussels, 1998; The Synapses Project: Final Deliverable, 10 chapters, 64 pages

[20] BEALE T. *The GEHR Archetype System*, The Good Electronic Health Record Project, Australia, August 2000
http://www.openehr.org/downloads/usage/gehr_australia/gehr_archetypes.pdf

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Publications about the archetype approach:

[21] KALRA, D., *Clinical Foundations and Information Architecture for the Implementation of a Federated Health Record Service,* PhD Thesis, University of London, 2002. Available from: http://www.chime.ucl.ac.uk/~rmhidxk/Thesis/Kalra_Dipak_PhD_2002.pdf

[22] BEALE, T., *Archetypes — An Interoperable Knowledge Methodology for Future-proof Information Systems*. 2001, 69 pages. Available from: BEALE, T. *Archetypes: Constraint-Based Domain Models for Future-proof Information Systems*, in OOPSLA-2002, Workshop on behavioural semantics, 2002

[23] The openEHR Foundation: Archetypes FAQ. Available from: http://www.openehr.org/shared-resources/faqs/archetypes.html

Other contemporary work in this area is being undertaken by the HL7 Templates SIG and the HL7 Care Provision Technical Committee.

Research on the conversion of ADL-expressed archetype constraints to OWL is taking place, an OWL representation of archetypes might in the future play a complementary role to an ADL representation.

© ISO 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

ICS 35.240.80

Price based on 124 pages

© ISO 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale