Archetype Definitions and Principles

Editors: {T Beale, S Heard}[a]		
Revision: 1.0	*Pages:* 15	*Date of issue:* 14 Mar 2007

a. Ocean Informatics

Keywords: EHR, health records, archetypes, principles

Copyright Notice

Amendment Record

Issue	Details	Who	Completed
RELEASE 1.0.1			
1.0	**CR-000203**: Release 1.0 explanatory text improvements.	T Beale, S Heard	14 Mar 2007
RELEASE 1.0			
RELEASE 0.95			
0.6	**CR-000127**. Restructure archetype specifications. Minor text changes.	T Beale, S Heard	14 Mar 2005
RELEASE 0.9			
0.5	Initial Writing. Based on Material taken from "A Shared Archetype and Template Language, Part I: A Position Paper for HL7, CEN TC 251, openEHR and other organisations".	T Beale, S Heard	28 Dec 2003

Table of Contents

1 Introduction

1.1 Purpose

This document describes the design principles of archetypes and templates. It is recommended that the *open*EHR ADL document [4] be read in conjunction with this document, since it contains a detailed explanation of the semantics of archetypes.

1.2 Related Documents

Prerequisite documents for reading this document include:

- The *open*EHR Architecture Overview

Related documents include:

- The *open*EHR Archetype Definition Language (ADL)
- The *open*EHR Archetype Object Model (AOM)

1.3 Status

This document is under development, and is published as a proposal for input to standards processes and implementation works.

This document is available at http://svn.openehr.org/specification/TAGS/Release-1.0.1/publishing/architecture/am/archetype_principles.pdf.

The latest version of this document can be found at http://svn.openehr.org/specification/TRUNK/publishing/architecture/am/archetype_principles.pdf.

2 Definitions

The definitions of the terms "archetype", "template" and variants as used in this paper are as follows:

archetype: a computable expression of a domain content model in the form of structured constraint statements, based on a reference (information) model. *open*EHR archetypes are based on the *open*EHR reference model. Archetypes are all expressed in the same formalism. In general, they are defined for wide re-use, however, they can be specialized to include local particularities. They can accommodate any number of natural languages and terminologies.

template: a directly locally usable definition which composes archetypes into a larger structures often corresponding to a screen form, document, report or message. A templates may add further local constraints on the archetypes it mentions, including removing or mandating optional sections, and may define default values.

3 Purpose of Archetypes and Templates

3.1 Purpose of Archetypes

Archetypes are created for a number of purposes (described in detail in [1] and [3]), summarised here:

Human Communication: to enable domain concepts to be modelled in a formal way by domain experts;

Specialised Searching: also to compare data to specialised archetypes, or "predicates".

Archetypes can be used directly for the computational purposes described below, but are normally encapsulated by templates for this purpose. The key benefits of archetypes include:

Knowledge-enabled systems: the separation of information and knowledge concerns in software systems, allowing cheap, future-proof software to be built;

Knowledge-level interoperability: the ability of systems to reliably communicate with each other at the level of knowledge concepts;

Domain empowerment: the empowerment of domain specialists to define the informational concepts they work with, and have direct control over their information systems.

Intelligent Querying: to be used at runtime to enable the efficient querying of data based on the structure of archetypes from which the data was created.

3.2 Purpose of Templates

Templates constitute a form of constraint statement model, which is directly usable for:

Data Construction: to be used at runtime to constrain the creation of data in local contexts to conform to data capture requirements;

Data Validation: to be used at runtime to validate data from other sources.

While archetypes are generally broad models, and have very open compositional possibilities, templates are used to narrow the choices of archetypes for local or specific purposes. They can be used to control the following things:

- archetype composition, or *chaining*
- reduction in allowed terms
- restricting optionality
- removing structures defined in the referenced archetypes.

4 Principles

4.1 Overview

Examples of domain-level concepts include "blood pressure", "physical examination (headings)", "biochemistry results" and so on. Here, the term reference model refers to any information model which can have data instances in a computational system. The following figure illustrates the relationships of archetypes with data.

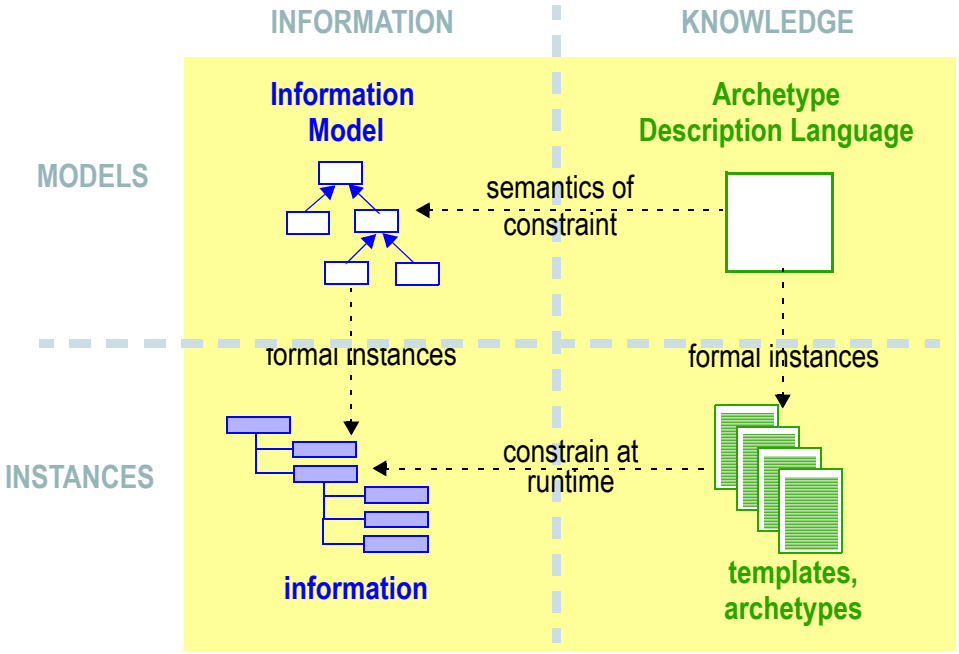

FIGURE 1 Archetype Model Meta-architecture

In this figure, the following relationships hold:

- data are instances of a reference model, such as an model of the EHR, Demographics or other concepts
- archetypes are instances of an "archetype model" which is a common formalism for expressing all archetypes
- the archetype model is formally related to the reference model, such that its semantics are those of constraint on objects of types defined in the reference model. It may also include linguistic elements allowing relationships between elements and invariants to be expressed (e.g. relationships between BMI and height and weight, apgar score and its 5 inputs etc)
- if data are created and modified using archetypes, archetypes constrain the configuration of data instances to be valid according to the archetype. E.g. Section and Entry objects are forced into a structure which is agreed to be correct for an ante-natal examination.

4.2 Archetype Design Principles

These concepts can be stated in more formal terms as the following principles:

Principle 1: An archetype defines a whole, distinct, domain-level model of content.

Archetypes should define coherent, whole informational concepts from the domain, in order to be useful. Archetypes enable self-standing groupings of information to be defined regardless of context. For example, there may be an archetype for "ECG result" since this is understood and used as a whole concept by clinicians, but not "ECG lead 2 result", which would only ever be understood as part of an "ECG result". The heart rate, as determined in an ECG, may be archetyped separately as this is a distinct concept that can be understood on its own. Similarly, we would not consider the heading "systolic" to be a meaningful archetype on its own, rather it would be part of a "blood pressure" or "intravascular pressure" archetype.

Principle 2: An archetype defines constraints on the structure of instances of a reference model.

An archetype can define the valid structuring of data instances to form a logical instance of domain content. For example, the hierarchical structure of “SOAP” headings used in problem-oriented recording is definable in an archetype in terms of a structure of Section instances, assuming Section is a type from the model. The implication of this principle is that reference models do not need to supply domain-specific structures, only generic building blocks suitable for creating the latter.

Principle 3: An archetype defines constraints on types and values of instances of a reference model.

Archetypes also express constraints on allowable constructions of reference model instances, e.g. on allowed types, ordering, cardinality, values and so on. The combination of structure and constraint expression means that numerous variations on a data instance may conform to a single archetype.

Principle 4: The granularity of an archetype corresponds to the granularity of a business concept in an information model.

Archetypes are defined at the same level of granularity as the ‘business’ entities in the reference model, i.e. the key types that in turn may have internal finer-grained structure. For example, since the *open*EHR reference model includes the business concepts Composition and Observation, archetypes of these same concepts can be created, e.g. a “cholesterol result” (a kind of Observation) and an “encounter note” (a kind of Composition).

Principle 5: Since each business concept in the reference model corresponds to a particular ontological level found in the domain, archetypes based on each business concept belong to the same ontological level.

Taking the *open*EHR reference model as an example, there are four ontological levels: the EHR, the Composition, the Section, and the Entry. Each of these defines a different category of artefact, for exmaple, the Entry (including subtypes Observation, Evaluation etc) defines the generic semantics of ‘clinical statements’. All archetypes based on Entry or its subtypes belong to this same ontological category, defining specific kinds of clinical statement.

Principle 6: A *compositional* relationship can exist between archetypes.

Archetypes can be composed to express valid possibilities for larger structures of data from different levels of the ontological hierarchy of the reference model. Such compositional connections are termed ‘slots’. For example, Section and Entry archetypes can be linked in a compositional way to define valid structures for the headings and data of a model of information captured in a "physical examination".

Principle 7: An archetype can be a *specialisation* of another archetype.

Archetypes can be defined at higher or lower levels of detail at a given ontological level. Thus, a "biochemistry result" archetype would define the general shape and constraints for all biochemistry results, while a "cholesterol result" archetype could be defined as a specialisation of this, in order to further constrain data to conform only to the shape of a cholesterol test.

Principle 8: Archetypes are internally hierarchical in structure; that is to say, the constraints in an archetype has an internal hierarchical compositional structure.

This is because object models give rise to data that is inherently hierarchical in structure.

Principle 9: Archetype nodes, including the root and all leaves, are identified by semantic identifiers which act as the basis for human-readable ‘meanings’, and for computational paths.

Archetype node identifiers are defined within the archetype using coded terms. The definition of any such code acts as a standardised ‘design-time meaning’ of the node, e.g. “5 minute Apgar result”.

Any node in an archetype can be referenced by concatenating attribute names and node identifiers from the archetype root to the node, to form an *archetype path*.

Principle 10: Archetype paths form the basis of *reusable semantic queries* on archetyped data.

Archetype paths can be used to construct queries that specify data items at a domain level, rather than being limited to the classes and attributes of the reference model as for a query in standard database theory. For example, paths from a “blood pressure measurement” archetype may identify the systolic blood pressure (baseline), systolic pressures for other time offsets, the patient position, and numerous other data items.

Principle 11: Data generated from an archetype will have a compositional structure, in which nodes must be uniquely named, in order to be able to refer to them.

Since an archetype acts as a kind of ‘template’, there can be repetitions of archetype structures in real data. Each node therefore needs a unique runtime name in order to be uniquely identified within the data. The archetype can be used to constrain this unique name.

Principle 12: Archetypes have no language primacy: they are completely translatable artefacts.

An archetype can be developed in any language, and have translations in other languages added *a posteriori*.

Principle 13: Archetypes are neutral with respect to terminologies.

Archetypes are able to be developed with or without reference to external terminologies. Multiple external terminologies can be bound to an archetype; the definition of terms in an archetype is not predicated on the existence or use of any particular terminology.

Principle 14: There is a means of evolving existing archetypes to accommodate changing requirements, without invalidating data created with earlier versions.

Since archetypes are used to create data, changes to archetypes must be regarded as creating a new archetype; i.e. the identifier of an archetype must incorporate its version. The only

types of change to archetypes that can be made without changing the version are those which do not invalidate previously created data. Formally, such changes must not 'narrow' constraints expressed in the existing version.

4.3 Template Design Principles

While archetypes support the definition of re-usable domain content models, templates provide a means of defining localised use of archetypes, often corresponding to screen forms, reports, documents (such as discharge referrals) and so on.

Templates, like archetypes can be shared. Their use is to express the data collection requirements for specific clinical situations - many will be situation specific and some will express the requirements of individual users. From this we can deduce that just a few archetypes may lead to a plethora of templates.

Principle 1: Templates are used to define aggregates of archetypes suitable for particular local uses.

Archetypes define re-usable content models, and possible ways to connect them compositionally, according to the underlying reference model. Such connections, termed 'slots', are usually defined as openly as possible, i.e. they indicate all allowable compositional relationships from one archetype to others.

Principle 2: Templates cannot create new constraints with respect to the archetypes they reference; they can only further constrain in a compatible way.

While archetypes define the possibilites for compositional relationships with 'slots', templates make the choices of which archetypes will actually be used in particular slots, to form structures directly usable in a local context.

Similarly, templates can further constrain optionality and cardinality constraints in archetypes. Any optional (i.e. 0..1) attribute defined in an archetype can be constrained by a template to be removed (0..0) or to be mandatory (1..1), since both of these constraints are formal subsets of the 0..1 constraint. Similar compatible constraints can be made on container cardinalities defined in archetypes.

Principle 3: Template identifiers do not need to be recorded in the data for the data to be usable.

Since templates do not change the structure of archetypes other than deleting or mandating certain branches, the path structure of the archetypes remains intact. Hence, archetype paths can be used to interrogate the created data, without reference to the particular templates that originally referenced the archetypes.

In summary, the constraints expressible in templates includes:

- Which archetypes must be used (i.e. are mandatory)
- Which archetypes may be used (i.e. are optional)
- Which optional nodes of the archetypes are not utilised
- Which optional nodes of the archetypes are mandatory
- Which 'fillers' for a slot are optional
- Which 'fillers' for a slot are mandatory
- Which language (or languages) are available to the user

- Which terminology (or terminologies) are available to the user at each node
- Which optional values of any element are available
- Which optional value of any element is the default value

END OF DOCUMENT