Archetype Definitions and Principles

Editors:{T Beale, S Heard}[1]

Revision: 0.5

Pages: 13

Keywords: EHR, archetypes, constraints

1. Ocean Informatics Australia

The *open*EHR foundation

is an independent, non-profit community, facilitating the creation and sharing of health records by consumers and clinicians via open-source, standards-based implementations.

Founding Chairman	David Ingram, Professor of Health Informatics, CHIME, University College London
Founding Members	Dr P Schloeffel, Dr S Heard, Dr D Kalra, D Lloyd, T Beale
Patrons	To Be Announced

email: info@openEHR.org **web**: http://www.openEHR.org

Copyright Notice

Amendment Record

Issue	Details	Who	Completed
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 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 level (clinical) concept in the form of structured constraint statements, based on some reference model. Archetypes are 1:1 with domain concepts, which may themselves have interior complexity. Archetypes all have the same formalism but may be:

- part of a standardised/shared ontology, i.e. definitional
- only used locally or regionally, and not considered definitional

template: a directly, locally usable data creation / validation artefact which is semantically a constraint / choice on archetypes, and which will often correspond to a whole form or screen. Templates in general have a 1:N relationship with underlying concepts, each of which is described by an archetype.

1.2 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.

1.3 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.

2 Principles

2.1 Overview

In this section we describe the principles of archetypes. From the point of view of principles, we consider the definition of archetype to be as follows:

> an archetype is formal expression of a distinct, domain-level concept, expressed in the form of constraints on data whose instances conform to some class model, known as a reference model.

Examples of domain-level concepts include "blood pressure", "physical examination (headings)", "biochemistry results" and so on. Here, the term reference model refers to any model which can have data instances in a computational system. This includes models like CEN 13606, openEHR, the HL7 CDA schema, R-MIMs and HMDs.

The following figure illustrates the relationships of archetypes with data.

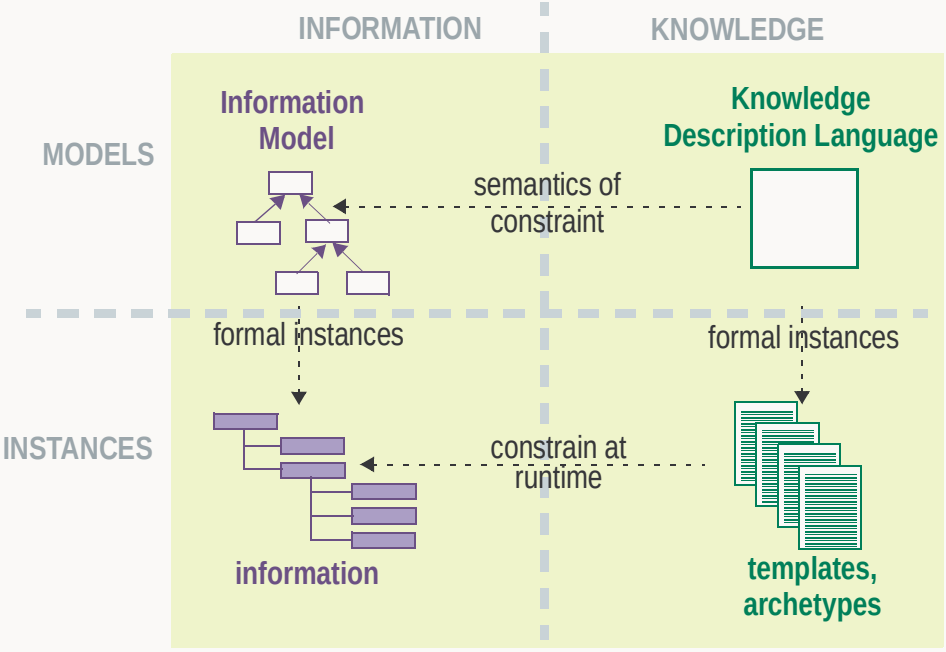

FIGURE 1 Archetype Model Meta-architecture

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.

2.2 Formal Principles

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

1. An archetype defines a whole, distinct, domain-level concept.

   Archetypes must define coherent, whole concepts from the domain, in order to be useful. Archetypes enable distinct concepts to be recognised regardless of context. Accordingly, 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 outside the current context. Similarly, we would not consider the heading "systolic" to be a meaningful archetype on its own, while it could be part of a "blood pressure" or "intravascular pressure" archetype.

2. an archetype defines constraints on reference model instances which express valid structure (i.e. composition, cardinality).

   An archetype may be used to describe the general structuring of data instances to form a logical instance of a domain concept. For example, the hierarchical structure of headings used in problem-oriented recording would be visible in an archetype. In particular, this principle means that archetypes are not reliant on domain-specific semantics in the reference model (indeed, this is the whole point of archetypes - to avoid building such semantics into the reference model).

3. an archetype defines constraints on instances of a reference model which express valid types and values.

   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.

4. the granularity of an archetype corresponds to the granularity of a business concept in the reference model.

   Archetypes are meaningfully defined at the same level of granularity as the "business" entities in the reference model. For example, if a reference model includes the business concepts "section headings" (a model of recursive headings in a document) and "entries" (a model of data taken from observations etc), then there will be archetypes for section heading trees, and for entries.

5. since each business concept (in the reference model) describes a particular ontological level found in the domain, so to do archetypes all belong to one or other ontological level.

   We can think of the archetypes at each of the ontological levels "thematic" (Documents, Composition), "organisational" (Sections, Organisers), "descriptive" (Entries) and so on as describing these levels of the domain. If there were 50 GP heading level archetypes, we could say that the organisational ontological level (from the point of view of GPs) was described by the archetypes.

6. compositional constraint can occur between archetypes within an ontological level, or between adjacent levels.

   Archetypes can be composed to express valid possibilities for larger structures of data from different levels of the ontological hierarchy. For example, Organiser and Entry archetypes can be linked in a compositional way to define valid structures for the headings and data of "physical examination".

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 results" archetype would define the general shape and constraints for all biochemistry results, while a "cholesterol results" 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.

   Four other corollaries which are not strictly mandated by the notion of archetypes, but which are likely to be almost universally true in real applications are as follows:

8. archetypes are hierarchical in structure; that is to say, an archetype has an internal hierarchical compositional structure.
9. archetype nodes, including the root and all leaves, are identified by standardised names, or meanings, which are unique at any given node.

   These design-time names have been designated as "meanings" in this document, to distinguish them from "runtime names" (see next point). Any node can be identified from the top by concatenating the meanings to form an archetype path, which can be thought of as a design-time path. (The use of meanings does not preclude other kinds of node identification).

10. data generated from an archetype will also have a compositional structure, in which nodes must be uniquely named, in order to be able to refer to them.

    Since there can be repetitions of archetype structures in real data, the runtime name of a node is distinguished from the archetype name of the same node. Put another way, each node in runtime data has a name (due to the application or user at runtime) and a meaning (from the archetype). Additionally, each node from any given root point in data can be identified by its runtime path, formed by concatenating the runtime node names. An example might be multiple blood pressures taken over a period of time but each entered as single instances might be called 'resting blood pressure', 'standing blood pressure' and 'resting blood pressure @ 5 minutes'.

11. Different languages are dealt with via the usual means of translations through coded terminologies - this enables both archetypes at design time and data at runtime to appear totally in the local user's language

END OF DOCUMENT