The Purpose of Archetypes

This document describes the raison d’etre of the openEHR Archetype formalism, consisting of the [Archetype Definition Language (ADL)](http://openehr.oceaninformatics.com/releases/trunk/architecture/am/adl2.pdf), the [Archetype Object Model (AOM)](http://openehr.oceaninformatics.com/releases/trunk/architecture/am/aom2.pdf) and the [Archetype Query Language (AQL)](https://openehr.atlassian.net/wiki/display/spec/Archetype+Query+Language+Description).

# Micro-history

The Archetype formalism, comprising the Archetype Definition Language (ADL) and its sibling specification Archetype Object Model (AOM) were devised by the [openEHR Foundation](http://www.openehr.org/) as part of an approach to account for the need to accommodate ‘domain semantics’ and ‘domain models’, which are numerous and highly variable, while preserving existing ‘information models’, where the latter are understood as the definition of data / instances, in the orthodox object-oriented and relational manner. The same need was recognized by the CEN and ISO committees in health with the result that AOM became an ISO standard (13606-2) in 2008. The same need was identified since 2011 by the [Clinical Information Modeling Initiative](http://opencimi.org) (CIMI), which chose the latest version of ADL/AOM as its modeling formalism. Independently of this lineage of development, Intermountain Healthcare developed over many years a system of domain content modeling known as [Clinical Element Models](http://clinicalelement.com/) (CEMs) which in its technical form and tooling approach is very close to the Archetype approach, so much so that inter-conversion from CEMs to Archetypes are available today, and Archetype 🡪 CEM convertibility is imminent.

In the following, the term ‘Archetype’ can be assumed to also stand for Intermountain CEMs.

# Business Purpose

To make the distinction between domain and information models concrete, information models in openEHR, ISO 13606, CIMI, HL7 and more generally in e-health typically define things like ‘clinical data types’, such as Quantity (with units, accuracy etc), Coded text, Ordinal (an Integer/symbol conjunction), and fairly generic clinical structures, such as ‘clinical statement’ (often denoted by the type Entry), clinical document, report, and so on. Such a class model may contain 50-100 classes, including 20+ classes for the clinical data types. This enables the construction of instance structures corresponding to the various parts and sections of e.g. a clinical encounter note or a hospital discharge summary. However, neither a class model of this size, nor the capabilities of standard UML can naturally accommodate the explosion of diversity of possible values of instances which can make up a clinical document created in any particular situation (e.g. a specific kind of patient visiting a specialist), for example the tens of thousands of clinical observations (e.g. ‘systolic blood pressure’, ‘visual acuity’, etc, many of them consisting of multiple data points in specific structures), or the O(1E5) laboratory analyte result types. Further, the size of terminology needed to annotate data items, both ‘names’ and ‘values’ in a name/value understanding of the data is in the O(1E5) – O(1E6) concepts range, as exemplified by the SNOMED CT and ICD11 terminologies.

The above situation applies across most information-rich industries, with varying but generally very large numbers.

Although technically these numerous possible values could just be understood as the specific values that ‘happen to occur’ in a situation of data creation, it is widely understood within IT in general that domain data value ‘complexes’ (co-occurring structures of data) correspond to meaningful patterns that constitute a relatively small fraction of the astronomical number of *possible* combinations of values within structures. Thus, while some tens to hundreds of thousands of ‘clinical statement’ patterns would adequately cover nearly all of general medical data recording (i.e. leaving the terminal real world values such as actual blood pressure open, within their respective sanity ranges), the information models in typical use would permit possible instance structures in the O(1E10) and much higher ranges. In other words, most *possible* data constructions are garbage.

This is akin to the situation in natural language, where meaningful sentences constitute a tiny fraction of possible, grammatically correct sentences.

It is also widely recognized that mechanisms are needed to enable some sort of domain level ‘modeling’ or ‘templating’, to enable the common patterns to be defined, and thus to allow the creation of software or other mechanisms (e.g. pre-built UI forms) to limit the possible instance structures to those that actually make sense. The general need was identified in Martin Fowler’s 1991 book [Analysis Patterns](http://www.amazon.co.uk/Analysis-Patterns-Reusable-Object-Models/dp/0201895420), in which ‘patterns’ are illustrated in ‘above the line’ parts of UML diagrams, but has been known for some decades. It is generally understood that this kind of modeling cannot simply be an extension of the existing software or database schemata; if it is, it implies endless maintenance and updating of deployed software, and worse, frequent database migration. In systems operating 24x365, and routinely creating Terabytes of data per year per hospital, this is not an acceptable approach.

Consequently, most large system software products in the health and other domains have some kind of configuration or template building tool(s) that enable modeling of typical domain content patterns, usually conceived of as screen forms.

The problem to date has been that no such capability is available independent of particular software products (specific vendors), concrete forms (UI forms, XML Schemas etc) or domains (e.g. process and control systems engineering have domain specific languages) – i.e. even tools that may be technically powerful enough are buried inside specific products, and are usually targeted to the database schemas of the product.

An important economic factor is that the creation of good quality domain models is time-consuming and expensive, relying as it does on domain experts – typically experienced clinicians, engineers etc – rather than IT staff. If models are created inside a specific product (e.g. a particular hospital information system), and that product is replaced, there is often little appetite or availability of the staff to recreate the work done to create the models/templates created in the first product. Multiplied across products, sites, and whole industry verticals, **the lack of standard ways of representing models of domain content has become a significant blockage to the production of high quality information systems**. Instead, as each solution is replaced, its domain models usually die with it.

The need for an efficient, formal, and product- and format-independent domain modeling capability is therefore clear. The sheer numbers of content patterns / models in health have led to the creation of an approach, centered around the Archetype formalism, used in conjunction with available terminologies (i.e. SNOMED CT, LOINC, ICDx and many others).

The archetype formalism primarily addresses the expression of models of possible *data instance structures*, rather than higher level concepts such as workflows, clinical guidelines (which are decision graphs) and so on, although its general approach can be applied to any of these, i.e. the use of a model of ‘what can be said’ and a formalism or mechanism for *constraining* possibilities to the meaningful subset.

The openEHR ADL/AOM formalism is designed to be independent of any specific information model (known as a ‘reference model’), product, technical format, or industry vertical. It is designed so that instances of the formalism, known as Archetypes, can be computationally processed into desired output forms corresponding to specific technology environments. This is routinely performed in openEHR (and also Intermountain) tooling environments.

It also supports two distinct types of domain content models, relating to a universal need, which is to be able to represent both use-independent definitions of ‘data points’, and use-case dependent definition of ‘data sets’. Consider the case of recording patient vital signs. Assume that a content model can be defined for ‘blood pressure’, ‘heart rate’ and ‘blood oxygen’. These definitions need to be independent of specific uses such as patient home measurement, GP encounter, and hospital bedside measurement, since in all these cases, the blood pressure etc. are recorded in exactly the same way. However in each case, these vital signs data points are recorded *within* a larger data set of items that correspond to the health system event occurring, such as a GP patient health checkup.

Thus there are two related needs: to be able to model domain data items and structures, and secondly, to be able to model larger structures in which they may occur. The alternative would be to create a domain model for every data set and within many of these models, to repeatedly define the same sub-model of recurring content, such as ‘blood pressure’. The former approach results in two layers of domain models: reusable data point models (Archetypes), and use-case specific data-set models (Templates, in ADL parlance).

# Technical Aims of ADL / AOM

The [ADL/AOM specifications published by openEHR](http://www.openehr.org/programs/specification/releases/currentbaseline#ADL2), and later adopted in various forms by ISO and CIMI, take the following technical approach to domain content modeling:

* Domain content models are separated into two layers – re-usable Archetypes and use-case specific data-set models, known as Templates;
* A single formalism is used for all models: ADL syntax and its parse-tree equivalent AOM; a Template is understood as a specific kind of Archetype, constructed of elements chosen from specific Archetypes;
* The formalism is designed on the basis of constraints on a Reference Model i.e. any standard UML information model, such that instances of the domain models (i.e. actual Archetypes or Templates) are guaranteed to be legal technical instances of the underlying reference model;
* The ADL and AOM expressions of the formalism structurally follow the graph nature of instance networks resulting from class model instantiation, that is to say, ADL is a block-structured syntax, and the AOM defines equivalent in-memory graph structures that relate to corresponding structures from the underlying Reference Model;
* The formalism is independent of natural language, and can accommodate domain models in any language, as well as translation into other languages;
* The formalism accommodates bindings to any terminology, enabling the relationship between archetypes and ontological entities (terminology concepts and ontology entities) to be formally expressed (in the ontology domain, bindings express the ‘is-about’ relationship);
* Specialization and Composition between models are supported, in similar ways to inheritance and association in UML;
* Every individual element in an Archetype or Template is identified by a path that can be used to create statements in a query language for data retrieval;
* Various structured, multi-lingual meta-data are supported, including language, translation details, purpose, use, misuse, keywords, IP-related meta-data, and annotations.

The specifications of ADL and AOM can be referred to for details, but one key feature of the formalism is worth pointing out here: it relies systematically on a simple conjunction of reference model class names with codes, representing domain entities. The following fragment of ADL illustrates this. The names **CLUSTER**, **ELEMENT** and **DV\_QUANTITY** are type-names from the openEHR Reference Model, while the codes [id3], [id22], etc stand for domain semantic definitions such as ‘Blood pressure measurement’ and so on, as shown in the comments (the actual code definitions are in the lower part of the archetype definition, not shown here). This simple device allows, for example, two **ELEMENT** objects to be marked as representing two types of blood pressure. In its general form, it can be understood as a way of marking standard building block instances as being different parts of a domain instance structure, such as a medical result or complex document.

**CLUSTER**[id3] matches { -- Blood pressure measurement

items matches {

**ELEMENT**[id22] matches { -- systolic blood pressure

value matches {

**DV\_QUANTITY**[id35]

}

}

**ELEMENT**[id23] matches { -- diastolic blood pressure

value matches {

**DV\_QUANTITY** [id37]

}

}

}

This ‘concept-marking’ of nodes is applied universally throughout an Archetype, and where nodes have siblings, the codes are defined in an Archetype-local terminology.

An [additional specification](http://openehr.oceaninformatics.com/releases/trunk/architecture/am/knowledge_id_system.pdf) defines the structure and semantics of Archetype identifiers, versioning and lifecycle.

Functionally, archetypes and templates are used at design time to define domain content models, and at runtime for two purposes:

* Creating initial instance structures (from Templates); these must be by definition correct domain content structures, assuming the Archetypes are correct and complete;
* Validating previously created data retrieved from a data source or message channel, including data not originally created using Archetypes.

In openEHR, a third key function, querying, is performed using queries in the [Archetype Query Language (AQL)](https://openehr.atlassian.net/wiki/display/spec/Archetype+Query+Language+Description), written solely based on Archetype paths and Reference Model relations, but independent of physical data storage schema.

These uses of Archetypes and Templates provide a basis for lifting data processing to a domain semantic level, from what would otherwise be a syntactic level; it enables higher level functionality such as decision support and business intelligence to reliably refer to domain semantic entities rather than trying to match in *ad hoc* ways.

To provide a practical idea of use to date, there are nearly 500 international openEHR archetypes, with an average of 15 data points per archetype published – a total of around 7,500 substantive clinical data point definitions - in the openEHR.org [Clinical Knowledge Manager (CKM) repository](http://www.openehr.org/ckm/). Additionally some thousands of archetypes have been created in national repositories in various countries, and also within vendor products. Many thousands of Templates and AQL queries are constructed from this base of archetypes, and are operating in deployed openEHR systems around the world.

There is also a number of ISO 13606 archetypes in use in various countries and projects.

The [Intermountain internal CEM repository](http://clinicalelement.com/) has around 6,500 CEMs, each with one substantive data point (the granularity is finer). The CEMs bind directly to Intermountain’s own controlled terminology, and are used to build Template equivalents, known as CE-Types.