openEHR
Release 1
(in development)

# The *open*EHR Archetype System

| | | |
|---|---|---|
| *Editors:* {T Beale, S Heard}[a] | | |
| *Revision:* 0.5 | *Pages:* 19 | *Date of issue:* 12 Mar 2007 |

a. Ocean Informatics

*Keywords:* EHR, health records, repository, archetype

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

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

## Copyright Notice

## Amendment Record

| Issue | Details | Who | Completed |
|---|---|---|---|
| RELEASE 1.0.1 | | | |
| 0.5 | Restructured and rewritten | T Beale | 12 Mar 2007 |
| RELEASE 1.0 | | | |
| RELEASE 0.95 | | | |
| RELEASE 0.9 | | | |
| 0.3.1 | Updated diagram; added notes on dissemination network. | S Heard | 20 Dec 2003 |
| 0.3 | Initial Writing | T Beale | 29 Nov 2003 |

## Table of Contents

# 1 Introduction

## 1.1 Purpose

This document provides a description of the *open*EHR Archetype System, an archetype authoring, quality assurance, certification and publishing infrastructure for archetype-enabled information systems. The Archetype System provides a technical underpinning for a governance infrastructure for archetypes and templates. Service interfaces based on the Archetype System archictecture are described informally here, and specified in detail in other *open*EHR documents.

## 1.2 Related Documents

Prerequisite documents for reading this document include:

- The *open*EHR Architecture Overview;
- The Archetype Semantics specification.

## 1.3 Status

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

The latest version of this document can be found in PDF format at http://svn.openehr.org/specification/TRUNK/publishing/architecture/am/archetype_system.pdf. New versions are announced on openehr-announce@openehr.org.

# 2 Overview

The ‘Archetype System’ is a framework for development and use of archetypes, and consists of two parts. The first, the Archetype Authoring Network, includes an online repository, collaborative authoring process, querying and research tools, quality assurance, testing and approval processes, ultimately generating archetypes certified for production use. The second consists of dissemination mechanisms for archetypes to be made available to production systems. Two service models are implied. The archetype repository corresponds to one interface, providing the semantics of querying, browsing requests, life-cycle management etc, while the dissemination of archetypes corresponds to a smaller ‘archetype service’ interface that makes certified archetypes available to production systems.

Archetypes can be authored locally as well as centrally, so for any particular archetype consumer, there may be at least two logical repositories. Each logical repository can be linked to a parent, creating a single logical archetype space from the point of view of any particular group of authors. Logical repositories may be engineered as literally separate repository instances, or more likely (given the required sophistication of the functionality), as separate ‘compartments’ in a centralised repository at a national or regional level.

Currently it appears that templates will mostly be authored locally, with some templates that correspond to re-usable message structures, reports (e.g. discharge referrals) being designed for national or regional use. Reusable templates will therefore need to be accommodated in the archetype repository framework, although templates for local use only would be managed locally.

All approved archetypes and templates are made available through instances of the archetype service. The Archetype System is illustrated in FIGURE 1.

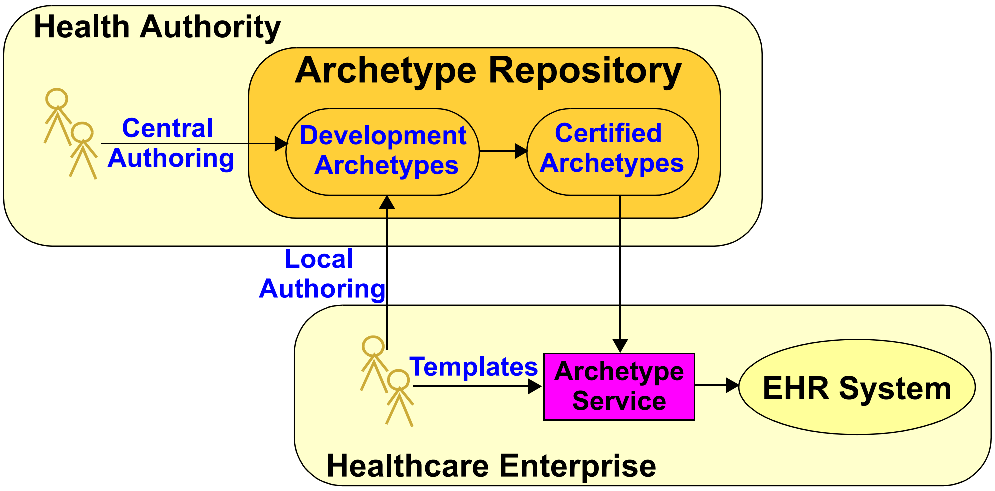

**FIGURE 1** The Archetype System

# 3 The Archetype Authoring Environment

## 3.1 Overview

The archetype and template authoring activity is carried out using tools that interface with the Archetype Repository. Each archetype is elaborated according to a development process over which the archetype will progress from an initial draft to an approved archetype, certified for use. Once an archetype has reached this point, and can no longer be changed, it should be digitally signed. All further changes must be carried out in new *revisions*, new *versions*, *specialisations*, or new archetypes. These categories are explained below. During the development process, the Archetype Repository enables querying and browsing of the existing library of archetypes, as well as access to the classificational ontology, and external resources. The archetype authoring environment is illustrated in FIGURE 2.

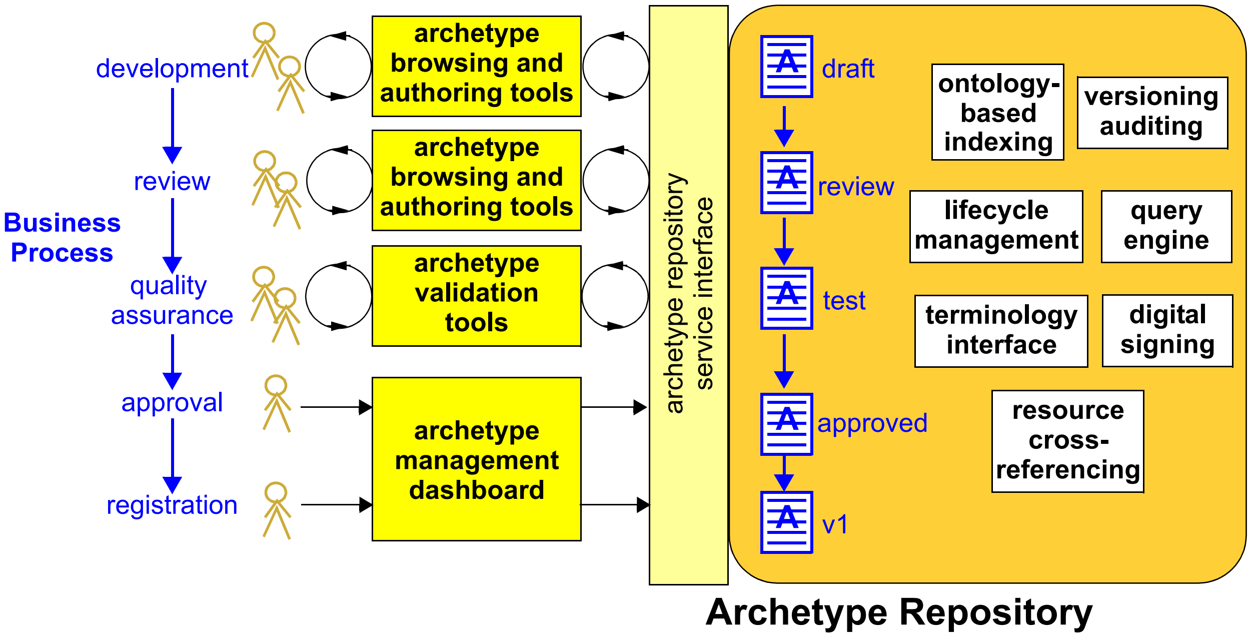

**FIGURE 2** The Archetype Authoring Environment

## 3.2 Versioning and Auditing

A basic function of the Archetype Repository is the versioning and audit-trailing of all committed resources. The versioning functions are the same as available with well-known systems such as Subversion and other commercial tools. Both optimistic and pessimistic locking should be available, with merging tools available for conflicts in the former case.

## 3.3 Querying and Browsing

One of the key parts of the archetype development activity is an initial and ongoing review of work already done in the same area, in terms of previous archetypes, and also related external resources. The Archetype Repository supports querying of archetypes based on identifiers, relationship to other archetypes, theme / clinical details, use, and other attributes. Browsing tools using this querying interface can build graphical or textual renditions of existing archetypes.

## 3.4 Lifecycle Management

The development of an archetype will follow a trajectory similar to documents or other artifacts in a technical authoring space. The archetype proceeds through a defined set of lifecycle states, usually defined as a state machine. At certain points in the lifecycle, group reviews and other similar activities take place. Managing this requires a way of registering users as part of an authoring group, as well as a way of defining the workflows for review, approval and so on. The Archetype Repository provides support for workflow definition and execution.

## 3.5 Ontology-based Indexing

As with software development, the first step to creating an archetype should be to determine whether any archetypes already exist corresponding to the requirements; a basic service is therefore to *find archetypes*. Two types of indexing are required, 'content-based' and 'semantic'. Content-based indexing means that archetypes are indexed by concrete properties of individual archetypes, such as:

- identifier
- other included meta-data author, version, lifecycle state, date of submission etc
- contained terms
- other textual contents, such as term definitions

This kind of indexing can easily be achieved by standard database indexing and content-based retrieval (if indexing is required on the contained text). Querying for archetypes based on combinations of properties can then be supported, such as "find all archetypes created by UK organisations within the last six months, which have 'pain' in the title".

In contrast, semantic indexing allows for much more sophisticated queries, but relies on making inferences on archetypes in the presence of other archetypes, terminologies and ontologies. For example, a user might wish to find all archetypes describing (clinical) examinations or investigations relevant to the heart. Some lab result types are clinically relevant, but their archetypes are unlikely to contain the words "heart" or "cardiac" etc., so content-based indexing won't retrieve them. If however an underlying ontology has links that indicate useful laboratory tests relevant to the heart, the archetypes will be retrieved.

The underlying resources used to achieve this are likely include a ontology defined specifically for indexing archetypes, and one or more published terminologies such as SNOMED-CT and LOINC.

## 3.6 Terminology Access and Use

Once it is determined whether a new archetype needs to be authored or an existing one adapted (usually specailised), the next step is to make the changes. Ontological and terminological resources within the environment can be used to enable the archetype author to understand the meaning of the archetype under construction, in a number of ways.

*Finding internal archetype terms*: internal terms defining node meanings may be derived from terms defined in published terminologies. For example, the author might wish to define an archetype node whose local code means "myocardial infaction", a term which will easily be found in a number of terminologies and ontologies. If searching the available terminologies reveals the exact definition required by the author, she can create the local term, copy the definition, and include a 'provenance' meta-data item in the term definition indicating the source.

*Verifying value sets*. In the 'value' position of an archetype leaf node, constraints are often defined which evaluate to a set of terms, raning from the simple such as "blood groups" to the more complex, such as "any disease which is-a auto-immune disease with-site liver". If these constraints can be tested at authoring time against an appropriate terminology or ontology, the archetype designer will be able to ensure that the intended meaning is achieved.

## 3.7 External Resources

In the authoring of archetypes, recourse to external resources, such as published guidelines, published data-sets, documents from professional bodies and so on is needed. The Archetype Repository provides a way to capture links, resource content, and index and cross-reference this with archetypes created based on the resources. This facilitates maintenance and review by other professionals.

## 3.8 Digital Signing

When an archetype is approved and registered for use, it should be digitally signed using a combination of a hash (e.g. using SHA-1) and the private key of the issuing organisation (e.g. using RSA). The first of these measures ensures that the integrity of the archetype can be trusted, while the second ensures that its origin can be trusted. The decrypted signature should regenerate the hash, which can also be generated from the archetype source in order to check the archetype's integrity. Signatures should be stored within the Archetype Repository as well as in Archetype Service instances.

## 3.9 Service Interface

The service interface to an Archetype Repository includes semantics corresponding to each of the functional areas described above. It is described in a separate specification.

# 4 The Dissemination Network

## 4.1 Overview

Archetypes and templates are disseminated from one or more Archetype Repositories via an Archetype Service network. The topology and dynamics of the network may be similar to the DNS (fully distributed hierarchy), or of a client/server form, where each Archetype Service node communicates only with an Archetype Repository. By either means, archetypes (and centrally defined templates) are made available to client systems such as EHR environments. In the target environment, it is likely although not essential that shareable archetypes (i.e. archetypes in *open*EHR/CEN standard ADL or X-ADL) will be converted locally into a runtime format, typically dependent on the kind of persistence technology in use. One view of these elements is shown in FIGURE 3.

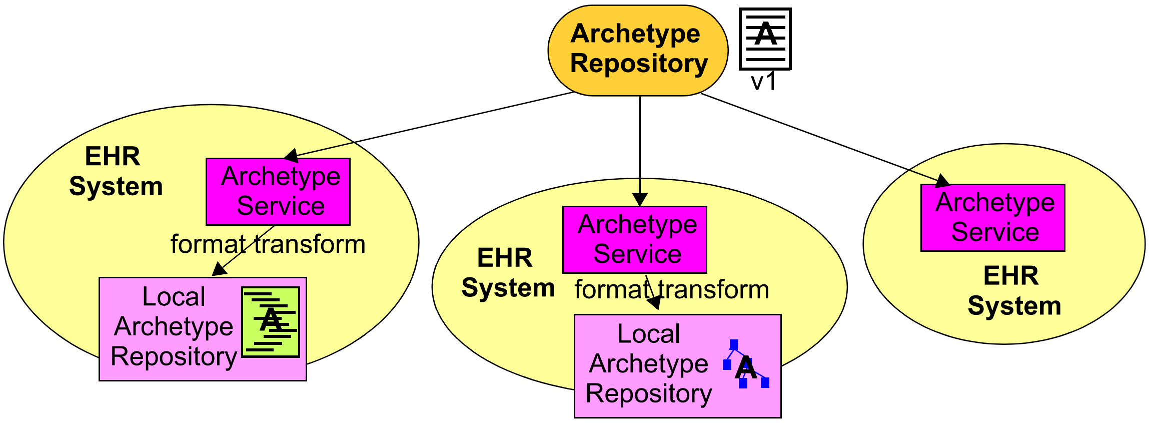

**FIGURE 3** The Archetype Dissemination Network

## 4.2 Service Interface

The service interface of the Archetype Service corresponds to the requirements of archetype consumer systems, rather than human authors. Accordingly it is significantly simpler than that of the Archetype Repository. Most functions are based on archetype identifiers rather than keyword or concept-based searching. The interface is specified in a separete document.

# 5 Archetype Semantics

Much of the functionality of the Archetype System is informed by the semantics of archetypes, particularly the identification system and inter-archetype relationships. The relevant semantics are briefly described here in relation to the Archetype System, and are described in detail in the *open*EHR Archetype Semantics specification.

## 5.1 Archetype Identification

Archetypes can be identified with various kinds of identifiers. Currently, the multi-axial archetype identifier specified by the `rm.support.identification.ARCHETYPE_ID` class in the *open*EHR Support IM is used. Other identifier types such as an ISO Oid could be deployed in the future, most likely in parallel with the multi-axial identifier. A typical example of an archetype identifier is:

```
openEHR-EHR-OBSERVATION.haematology.v1
```

The multi-axial archetype identifier encodes the partitioning of the archetype space in the identifier. Each identifier instance denotes a single archetype within a versioned 2-dimensional space, with the dimensions being:

- reference model entity, i.e. target of archetype;
- domain concept.

As with any multi-axial identifier, the underlying principle of an archetype id is that all parts of the id must be able to be considered immutable. This means that no variable characteristic of an archetype (e.g. accrediting authority, which might change due to later accreditation by another authority, or may be multiple) can be included in its identifier.

ISO Oids can also be used to identify archetypes within storage systems, online repositories etc., regardless of where the archetype sits in the concept space. Currently Oids are not in use in the *open*EHR archetype environment.

## 5.2 Semantic Relationships

Archetypes can relate to each other in the following ways:

*revision*: the revised archetype contains changes that do not invalidate the data created by the previous archetype in any way;

*specialisation*: similar to specialisation in object-oriented models, specialised archetypes can change existing parts of an archetype definition in a conformant way, or introduce new nodes;

*version*: the new version of an archetype will usually contain a small number of changes incompatible with the previous version; a data conversion algorithm should be supplied;

*composition*: archetypes use ‘slots’ to define the set of other archetypes that may be used at a node, rather than defining the same semantics inline.

All of these relationships should be assertable and viewable within the Archetype Repository.

# 6 Change Management

## 6.1 Overview

Archetypes are like any other models in that they reflect changing realities and understandings of reality. For both reasons, changes need to be made to existing archetypes as time goes on. However, archetypes are also used to create data, and to remain technically reliable, the relationship between archetypes and the data must always be known. If an archetype is 'changed' in some way, the relationship of the new variant to the previously created data must be known, in order for the data to be properly queryable and modifiable.

Four types of 'change' are identified below. Only one of them, 'revision' is a change in the literal sense of the word. The other three all create new archetypes that are formally related to existing archetypes. In the following sections, each type of change is described in terms of a 'semantic integrity constraint' that must be preserved, and the entailed logical consequence. The detailed semantics of each type of archetype relationship are found in the Archetype Semantics document.

## 6.2 Revision

Revision is the smallest kind of change that can be done to an archetype, and the only one not causing a change in identifier. The new revision is indicated in the `revision_history` section of an archetype. A new revision can safely replace the previous revision in an Archetype Service. Revisions are made in order to:

- modify the `description` section;
- modify the `ontology` section by changing the text descriptions of codes, or adding or changing terminology bindings;
- add or change translations (affects `language` and/or `ontology` and `description` sections);
- broaden an existing constraint in the `definition` section; this often means the changing of a mandatory constraint to an optional one.

The validity rule for revisions is:

- *Backward compatibility*: data created by precursor of revised archetype is compatible with revised version.

## 6.3 Specialisation

A specialisation of an archetype is similar in concept to a descendant class in an object-oriented model. Specialised archetypes can redefine nodes and add node in a way that respects the conformance relationship between archetypes; essentially this means that redefine nodes must be narrower or the same as the corresponding nodes in the parent archetype in terms of occurrences, reference model type and other constraints.

The identifier of a specialised archetype is a clone of the parent identifier, with an extra item appended to the concept section of the identifier, using a '-' separator. The following exemplify a typical parent and specialised child archetype identifiers:

```
openEHR-EHR-EVALUATION.problem.v1
openEHR-EHR-EVALUATION.problem-diagnosis.v1
openEHR-EHR-EVALUATION.problem-diagnosis-histological.v1
```

The node term code of a redefined node is defined as a ‘dot’ specialisation of the parent, e.g. ‘at0013.1’ is a valid code for the redefined version of the ‘at0013’ node in a top-level archetype. New nodes are given ‘dot’ node identifiers specific to the specialisation level; for example ‘at0.1’ is a code introduced new in a second level specialisation of an archetype.

Specialised archetypes are created when there is a need to customise an existing general purpose archetype, e.g. a ‘laboratory result’ archetype (containing generic ‘panel items’) into a particular subtype, such as ‘cholesterol result’ (containing named ‘LDL’, ‘HDL’, ‘Total’ etc items which are specialisations of the ‘panel item’ nodes in the parent). Sometimes the customisation is just the addition of new nodes.

The validity rule for specialised archetypes is:

- Specialised archetype creates data that conform to the parent archetype in the sense of data/archetype conformance described in the Archetype Semantics document.

## 6.4 New Version

A new version of an archetype is created when the existing version of an archetype is incompatible with the current data collection requirement. Given that well-designed archetypes contain mostly optional parts, this occurs rarely. The usual reason is that a mandatory item in the existing archetype is now regarded as inappropriate; a value range or coded term set may also be incorrect from the point of view of current requirements. Almost all other required changes can be achieved by a new revision or specialisation.

A new version should only be created when the archetype required models the same content as the existing archetype, and is intended to replace it in time. If this is not the case, a new (unrelated) archetype is created. When a new version is created, the same archetype identifier is used, with the version number in the final segment incremented, thus generating a new archetype identifier, as in the following example:

```
openEHR-EHR-EVALUATION.problem.v1
openEHR-EHR-EVALUATION.problem.v2
```

In general, nothing can be formally said about the semantic relationship between different versions of an archetype. However, it is strongly recommended in all archetype environments that:

- changes between versions of an archetype be minimised;
- a conversion algorithm to make data created from previous version(s) compatible with the latest version should be supplied with the latest version.

## 6.5 New Archetype

A new archetype is one whose identifier is different in the reference model or concept sections to that of other archetypes. New archetypes should be created compatibly with existing archetypes. ‘Compatibility’ here means that the scope of the archetype should not straddle concepts found in existing archetypes, but should either re-use them, or be complementary.

# G References

## Publications

1 Beale T. *Archetypes: Constraint-based Domain Models for Future-proof Information Systems*. OOPSLA 2002 workshop on behavioural semantics.
Available at http://www.deepthought.com.au/it/archetypes.html.

2 Beale T. *Archetypes: Constraint-based Domain Models for Future-proof Information Systems*. 2000.
Available at http://www.deepthought.com.au/it/archetypes.html.

3 Beale T, Heard S. *A Shared Language for Archetypes and Templates - Part I*. 2003.
Available at http://www.deepthought.com.au/health/archetypes/archetype_language_2v0.6.2.doc.

4 Beale T, Heard S. *A Shared Language for Archetypes and Templates - Part II*. 2003.
Available at http://www.deepthought.com.au/health/archetypes/archetype_language.doc.

5 Dolin R, Elkin P, Mead C *et al. HL7 Templates Proposal*. 2002.
Available at http://www.hl7.org.

6 Gruber T R. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. in Formal Ontology in Conceptual Analysis and Knowledge Representation. Eds Guarino N, Poli R. Kluwer Academic Publishers. 1993 (Aug revision).

## Resources

7 OWL - Web Ontology Language.
See http://www.w3.org/TR/2003/CR-owl-ref-20030818/.

8 *open*EHR. Knowledge-enabled EHR and related specifications.
See http://www.openEHR.org.

9 *open*EHR. EHR reference model.
See http://www.openEHR.org.

**END OF DOCUMENT**