ARCHETYPE MODEL

The *open*EHR Common Archetype Model

Editors:{T Beale, S Heard}[1], *{D Kalra, D Lloyd}*[2]

Revision: 0.8

Pages: 23

1. Ocean Informatics Australia
2. Centre for Health Informatics and Multi-professional Education, University College London

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
0.8	CR-000071. Allow version ids to be optional in TERMINOLOGY_ID. CR-000046. Rename COORDINATED_TERM and DV_CODED_TEXT.*definition*.	T Beale	25 Feb 2004
0.7	CR-000013. Rename key classes according to CEN ENV13606. CR-000041. Visually differentiate primitive types in openEHR documents. CR-000043. Move External package to Common RM and rename to Identification (incorporates CR-000036 - Add HIER_OBJECT_ID class, make OBJECT_ID class abstract.)	S Heard, D Kalra, T Beale, D Lloyd	09 Oct 2003
0.6	CR-000007. Added GENERIC package.	S Heard, D Kalra	11 Apr 2003
0.5	CR-000003, CR-000004 changes. Changed package naming, improved heading structures. Occurrences added to ARCHETYPE_NODE class (Formally validated).	T Beale	21 Mar 2003
0.4	Added C_ARCHETYPE.*c_classifiers*. Renamed *parent_version* to *previous_version*, *parent_id* to *specialisation_parent_id*. Added ARCHETYPE_PROXY class. Renamed LIFECYCLE to ARCHETYPE_DESCRIPTION and moved descriptive attributes from ARCHETYPE. Removed ORGANISER_TREE.	T Beale	10 Mar 2003
0.3	Removed Support and C_Relationship packages to Data Types AM document. Changed LIFECYCLE.*submitter* type to PARTY. **Formally validated using ISE Eiffel 5.2**.	T Beale	25 Feb 2003
0.2	Continued development	T Beale	15 Nov 2002
0.1	Initial Writing	T Beale	14 Oct 2002

Acknowledgements

The work reported in this paper has been funded by a number of organisations, including The University College, London; Ocean Informatics Pty Ltd, Australia.

Table of Contents

1 Introduction

1.1 Purpose

This document describes the *open*EHR Common Archetype Model, which describes the semantics of basic archetype concepts, common to all other archetype models. The intended audience includes:

- Standards bodies producing health informatics standards
- Software development organisations using *open*EHR
- Academic groups using *open*EHR
- The open source healthcare community

1.2 Related Documents

Prerequisite documents for reading this document include:

- The *open*EHR Modelling Guide

Other documents describing related models, include:

- The *open*EHR Common Reference Model

1.3 Status

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

Currently the UML diagrams are hand-produced. Various tool versions exist (Rose, Objecteering), but the visual quality is still being improved; when this is complete, the tool-generated images will be used.

Also in the future, specific design principles will be referred to throughout the model text, so that readers can easily find the theoretical discussion on which any part of the model is based.

The latest version of this document can be found in PDF and HTML formats at http://www.openEHR.org/doculist.htm. New versions are announced on openehr-announce@openehr.org.

1.4 Peer review

Areas where more analysis or explanation is required are indicated with “to be continued” paragraphs like the following:

To Be Continued: more work required

Reviewers are encouraged to comment on and/or advise on these paragraphs as well as the main content. Please send requests for information to info@openEHR.org. Feedback should preferably be discussed on one of the appropriate mailing lists, openehr-technical@openehr.org or openehr-clinical@openehr.org.

2 Overview

FIGURE 1 illustrates the AM.COMMON package structure.

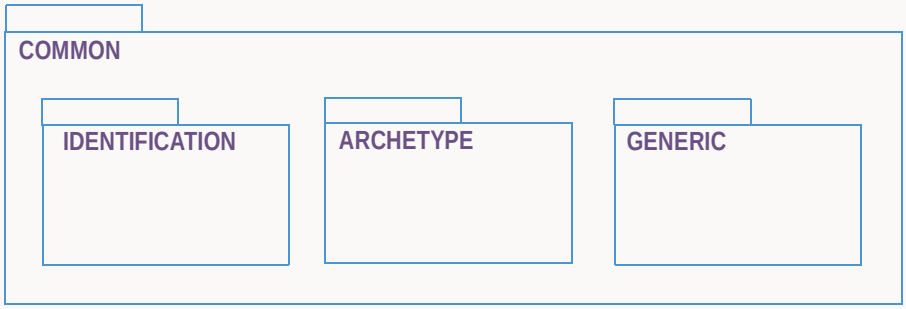

FIGURE 1 AM.COMMON Package

3 AM.COMMON.IDENTIFICATION Package

3.1 Overview

This package describes semantics for constraints on identifiers of entities in other reference models and is illustrated in FIGURE 2.

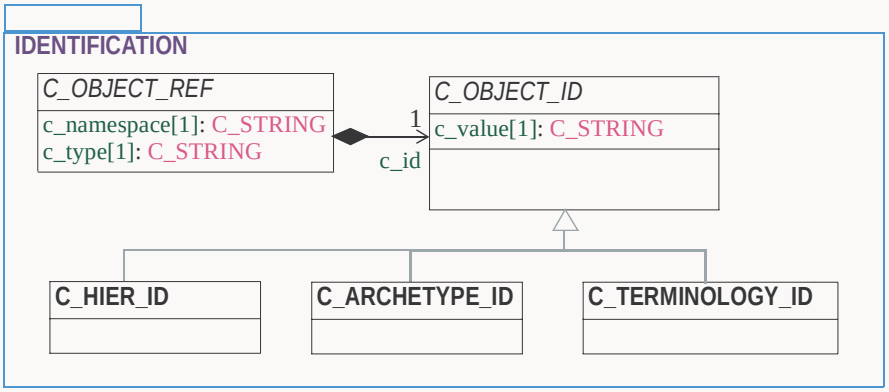

FIGURE 2 AM.COMMON.IDENTIFICATION Package

3.2 Class Definitions

3.2.1 C_OBJECT_REF Class

CLASS	C_OBJECT_REF	
Purpose	Constrain instances of `OBJECT_ID`	
Attributes	**Signature**	**Meaning**
	c_id: `C_OBJECT_ID`	Constraint on the value of the ID
	c_namespace: `C_STRING`	Constraint on the namespace.
	c_type: `C_STRING`	Constraint on the class name.
Invariant	***c_id_exists***: c_id /= Void ***c_namespace_valid***: c_namespace /= Void ***c_type_valid***: c_type /= Void	

3.2.2 C_OBJECT_ID Class

CLASS	C_OBJECT_ID	
Purpose	Constrain instances of `OBJECT_ID`	
Attributes	**Signature**	**Meaning**

CLASS	C_OBJECT_ID	
	c_value: C_STRING	Constraint on the value of the ID
Invariant	***c_value_exists***: c_value /= Void	

3.2.3 C_HIER_ID Class

CLASS	C_HIER_ID	
Purpose	Constraint on archetype identifiers.	
Inherit	C_OBJECT_ID	
Attributes	**Signature**	**Meaning**
Invariant		

3.2.4 C_ARCHETYPE_ID Class

CLASS	C_ARCHETYPE_ID	
Purpose	Constraint on archetype identifiers.	
Inherit	C_OBJECT_ID	
Attributes	**Signature**	**Meaning**
Invariant		

An instance of C_ARCHETYPE_ID takes the form of a regular expression which fits the form of instances of ARCHETYPE_ID, i.e.:

```
archetype_id: archetype_originator '.' qualified_rm_entity '.'
              domain_concept '.' version_id
```

The constraint can therefore be any regular expression which satisfies the template of the identifier, i.e.:

- `*\.*\.*\.*`

Examples include

- `*\.*\.*\.*` -- any archetype at all
- `*\.openehr:any:any\.*\.*` -- the openEHR “any” archetype
- `*\.openehr:ehr_rm:organiser\.soap\.*` -- a problem/SOAP organiser archetype for the openEHR EHR

- `*\.openehr:demographic_rm:agent\.software\.*` -- an archetype for software agents for the openEHR demographic model Agent class.
- `mayo\.hl7:cda:section\.*_exam\.*` -- any "exam" archetype for HL7 CDA section headings
- `racgp.openehr:*:*\.*\.* -- any RACGP openEHR archetype`

3.2.5 C_TERMINOLOGY_ID Class

CLASS	C_TERMINOLOGY_ID
Purpose	Constraint on terminology identifiers.
Invariant	

An instance of `C_TERMINOLOGY_ID` takes the form of a regular expression which fits the basic pattern of instances of `TERMINOLOGY_ID`, which is of the form:

```
terminology_name [ "(" version ")" ]
```

The constraint can therefore be any regular expression which satisfies the template of the identifier, i.e.:

- `*`

or

- `*(*)`

Examples include:

- `"SNOMED-CT"` -- any revision of SNOMED-CT
- `"*(*)"` -- any terminology
- `"ICD10(*)"` -- any revision of ICD10
- `"ICD*(*)"` -- any revision of any version of ICD
- `"ICD10|ICD10CM(*)"` -- any revision of ICD10 or ICD10CM
- `"ICPC+(1.*)"` -- any 1.* revision of ICPC+

4 AM.COMMON.ARCHETYPE Package

4.1 Overview

The ARCHETYPE package is illustrated in FIGURE 3. It contains the two principle classes on which all other archetype structures are based - ARCHETYPE and ARCHETYPE_FRAGMENT.

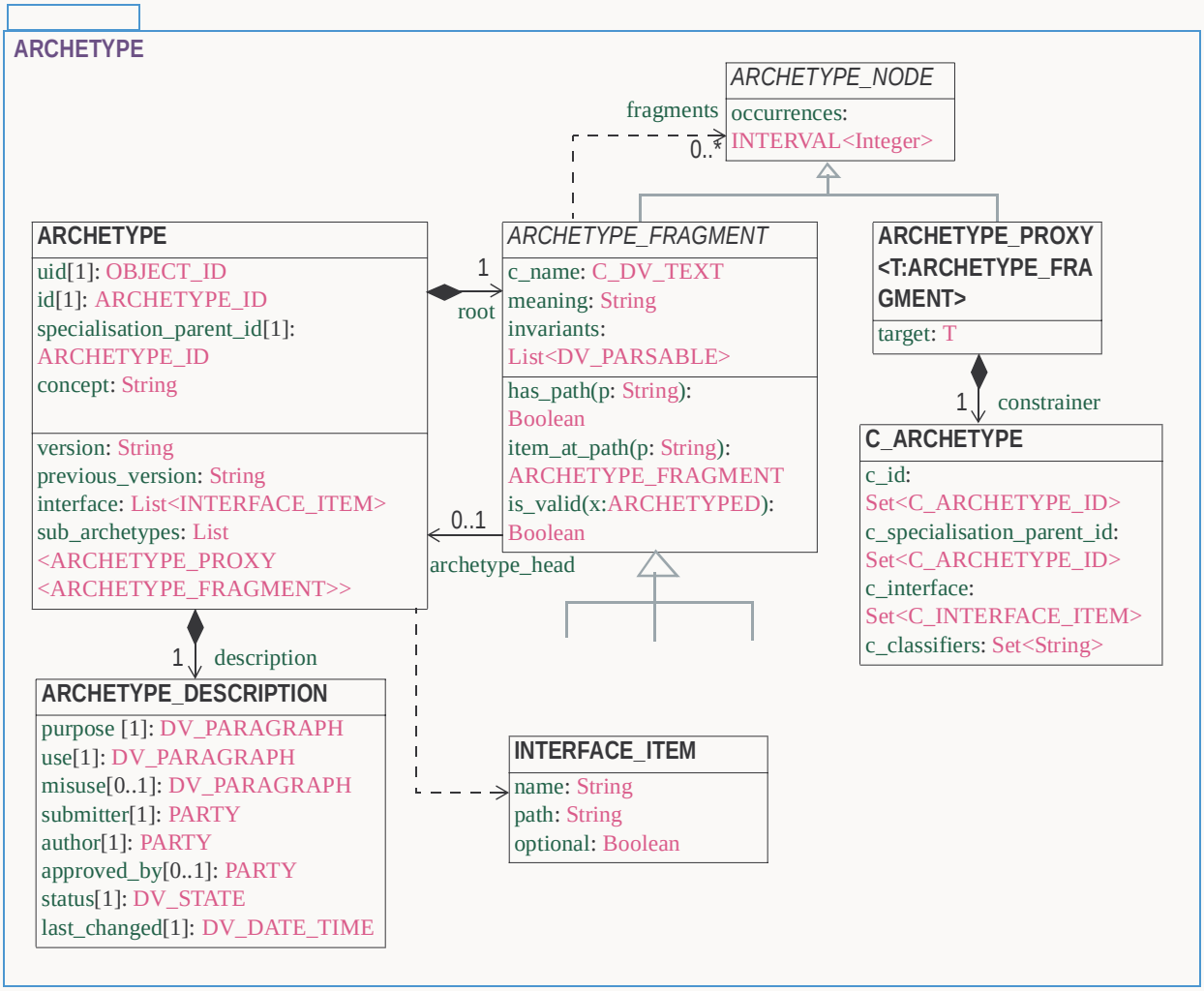

FIGURE 3 AM.COMMON.ARCHETYPE Package

An archetype consists of an ARCHETYPE instance, The former is the root class of any archetype structure. The archetype classes for the key reference classes such as COMPOSITION, SECTION, ENTRY - C_COMPOSITION, C_SECTION, C_ENTRY - inherit from the ARCHETYPE class. This class contains all the meta-data relating to the archetype as well as its structural relationships.

4.2 Class Definitions

4.2.1 ARCHETYPE_NODE Class

CLASS	*ARCHETYPE_NODE (abstract)*	
Purpose	Abstract class defining common features of all node types in an archetype structure.	
Attributes	**Signature**	**Meaning**
	occurrences: `INTERVAL <Integer>`	Cardinality of this object in the context of its parent object.
Invariant	***occurrences_exists***: occurrences /= Void	

4.2.2 ARCHETYPE_FRAGMENT Class

CLASS	*ARCHETYPE_FRAGMENT (abstract)*	
Purpose	Archetype equivalent to `LOCATABLE` class in Common reference model. Defines common constraints for any inheritor of `LOCATABLE` in any reference model.	
Inherit	`ARCHETYPE_NODE`	
Attributes	**Signature**	**Meaning**
	c_name: `C_DV_TEXT`	Constraint on the name of the fragment.
	meaning: `DV_TEXT`	The normative meaning of the fragment.
	archetype_head: `ARCHETYPE`	The head of the archetype, containing all relevant definitional information. Non-Void only is this is the root node.
	invariants: `List<DV_PARSABLE>`	List of invariant expressions for this archetype as a whole.
Functions	**Signature**	**Meaning**
	is_archetype_root: `Boolean`	This fragment is the root node of this archetype tree.
	has_path (a_path: `String`): `Boolean`	Report whether this archetype has a particular path.
	item_at_path (a_path: `String`): `ARCHETYPE_FRAGMENT` ***require*** has_path(a_path)	Return the item at a path.

CLASS	*ARCHETYPE_FRAGMENT (abstract)*
Invariant	***c_name_exists***: c_name /= Void ***meaning_valid***: meaning /= Void ***Archetype_root_valid***: is_archetype_root ***xor*** archetype_head = Void ***Archetype_invariants_valid***: invariants /= Void ***implies*** is_archetype_root

4.2.3 ARCHETYPE Class

CLASS	ARCHETYPE	
Purpose	Parent of all archetype root classes. Archetype equivalent to `ARCHETYPED` class in Common reference model. Defines semantics of identfication, lifecycle, versioning, composition and specialisation.	
Attributes	**Signature**	**Meaning**
	id: `ARCHETYPE_ID`	Multi-axial identifier of this archetype in archetype space.
	uid: `OBJECT_ID`	OID identifier of this archetype.
	concept: `DV_TEXT`	The normative meaning of the archetype as a whole.
	specialisation_parent_id: `ARCHETYPE_ID`	Identifier of the parent of this archetype. All archetypes have the “any” archetype as parent if there is no other parent.
	description: `ARCHETYPE_DESCRIPTION`	Description and lifecycle information of the archetype - all archetype information which is not required at runtime.
	root: `ARCHETYPE_FRAGMENT`	Root node of this archetype
Functions	**Signature**	**Meaning**
	status: `DV_STATE`	Lifecycle status, derived from lifecycle attribute.
	version: `String`	Version of this archetype, extracted from id.
	previous_version: `String`	Version of parent archetype of this archetype, extracted from id.
	interface: `List<INTERFACE_ITEM>`	Derived interface of the whole archetype as a list of leaf names with their associated paths.
	sub_archetypes: `List <ARCHETYPE_PROXY <ARCHETYPE_FRAGMENT>>`	Sub-archetypes of this archetype, i.e. other archetypes connected by composition to this one.

CLASS	ARCHETYPE	
	is_valid (x: ARCHETYPED): Boolean	Validate runtime data against this archetype.
Invariant	***id_exists***: id /= Void ***specialisation_parent_id_exists***: specialisation_parent_id /= Void ***description_exists***: description /= Void ***root_exists***: root /= Void	

4.2.4 ARCHETYPE_DESCRIPTION Class

CLASS	ARCHETYPE_DESCRIPTION	
Purpose	Defines description and lifecycle of archetype.	
Attributes	**Signature**	**Meaning**
	purpose: DV_PARAGRAPH	Purpose of the archetype.
	use: DV_PARAGRAPH	Description of the uses of the archetype, i.e. contexts in which it could be used.
	misuse: DV_PARAGRAPH	Description of any misuses of the archetype, i.e. contexts in which it should not be used.
	submitter: PARTY	Identification of the organisation submitting this archetype.
	author: PARTY	Identification of the author of the main content of this archetype.
	approved_by: PARTY	Identification of the approving organisation of this archetype.
	last_changed: DV_DATE_TIME	Description of any misuses of the archetype, i.e. contexts in which it should not be used.
	status: DV_STATE	Lifecycle state of the archetype. Includes at least the states: submitted, experimental, awaiting_approval, approved, superseded, obsolete.
Invariant	***submitter_exists***: submitter /= Void ***author_exists***: author /= Void ***approved_by_valid***: status.value.is_equal(“approved”) ***implies*** approved_by /= Void ***last_changed_exists***: last_changed /= Void	

4.2.5 INTERFACE_ITEM Class

CLASS	INTERFACE_ITEM	
Purpose	Describes an item of the interface to the archetype. Instances of `INTERFACE_ITEM` are derived from the internal structure of the archetype.	
Attributes	**Signature**	**Meaning**
	name: `String`	Name of the interface item - the meaning of a leaf item.
	path: `String`	Path at which the leaf item is found.
	optional: `Boolean`	True if this item is an optional part of the interface; derivable from the item.
Invariant		

4.2.6 ARCHETYPE_PROXY Class

CLASS	ARCHETYPE_PROXY<T:ARCHETYPE_FRAGMENT>	
Purpose	Proxy for further archetypes. Contains a constraint on what the next archetypes can be, and a target reference to a chosen archetype.	
Inherit	`ARCHETYPE_NODE`	
Attributes	**Signature**	**Meaning**
	constrainer: `C_ARCHETYPE_ID`	Constrainer for allowed archetypes
	target: `T`	Installed target
Invariant	***constrainer_exists***: constrainer /= Void ***target_valid***: target.is_archetype_root	

4.2.7 C_ARCHETYPE Class

CLASS	C_ARCHETYPE
Purpose	Constraint on composition of archetypes. Used to constrain which archetypes are allowed to occur in a given “slot” in another archetype, i.e. to express the compositional structure of archetypes.
Status	EXPERIMENTAL. This attributes of this class probably have to be replaced by a nested boolean expression including these attributes. The current class definition can be used for simple systems in which an “or” relationship can be assumed between the separate conditions.

CLASS	C_ARCHETYPE	
Attributes	**Signature**	**Meaning**
	c_id: `Set<C_ARCHETYPE_ID>`	Ids of allowed archetypes.
	c_specialisation_parent_id: `Set<C_ARCHETYPE_ID>`	Ids of allowed parent archetypes. Allows any archetype with a certain parent to be used.
	c_interface: `Set<INTERFACE_ITEM>`	Constrain via allowed interface, e.g. certain required paths or leaf items.
	c_classifiers: `Set<String>`	Constrain via classification keywords. This allows new archetypes to be considered as valid after the fact.
Invariant		

5 AM.COMMON.GENERIC Package

5.1 Overview

FIGURE 4 illustrates the Generic archetype package.

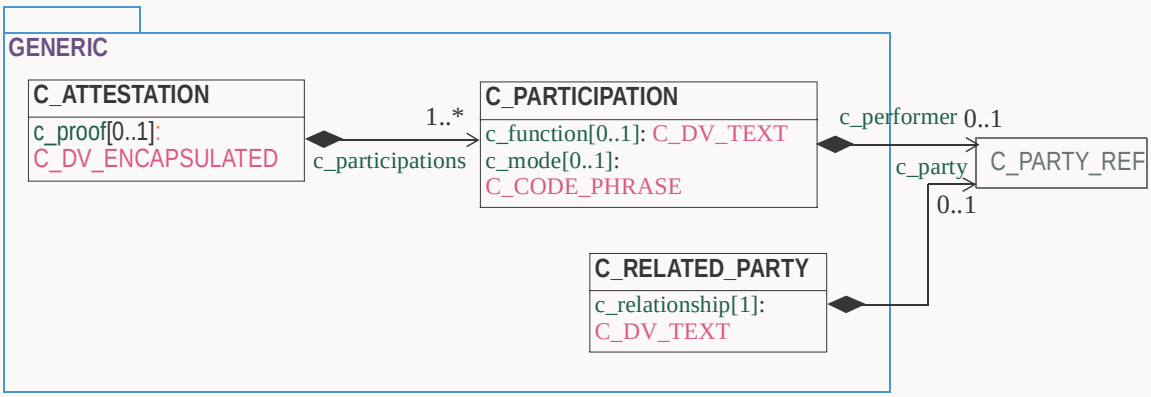

FIGURE 4 RM.COMMON.GENERIC Package

5.2 Class Descriptions

5.2.1 C_PARTICIPATION Class

CLASS	C_PARTICIPATION	
Purpose	Constrainer class for instances of type `PARTICIPATION`.	
Attributes	**Signature**	**Meaning**
	c_performer: `C_PARTY_REF`	Constraint on party participating in the activity.
	c_function: `C_DV_TEXT`	Constraint on function of party.
	c_mode: `C_CODE_PHRASE`	Constraint on mode of participation.
Invariant		

5.2.2 C_ATTESTATION Class

CLASS	C_ATTESTATION	
Purpose	Constrainer class for instances of type `ATTESTATION`.	
Attributes	**Signature**	**Meaning**

CLASS	C_ATTESTATION	
	c_participations: C_LIST <C_PARTICIPATION>	Constraint on participations in attestation.
	c_proof: C_DV_ENCAPSULATED	Constrainer on proof of attestation.
Invariants		

5.2.3 C_RELATED_PARTY Class

CLASS	C_RELATED_PARTY	
Purpose	Constrainer class for instances of type RELATED_PARTY.	
Attributes	**Signature**	**Meaning**
	c_party: C_PARTY_REF	Constraint on id of party.
	c_relationship: C_DV_TEXT	Constraint on relationship.
Invariants		

A References

A.1 General

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

2 Beale T *et al. Design Principles for the EHR*. See http://www.openEHR.org.

A.2 CEN

3 ENV 13606-1 - *Electronic healthcare record communication - Part 1: Extended architecture*. CEN/ TC 251 Health Informatics Technical Committee.

4 ENV 13606-4 - *Electronic Healthcare Record Communication standard Part 4: Messages for the exchange of information*. CEN/ TC 251 Health Informatics Technical Committee.

END OF DOCUMENT