The GEHR Archetype System

Authors: Thomas Beale

Revision: 3.1 draft B

Pages: 21

Amendment Record

Issue	Details	Who	Date
1.1 draft A	Initial Writing. Content taken from architecture document; archtype domain diagram from requirements.	T Beale	24 Jan 2000
1.1 draft B	Added termset requirements.	T Beale	24 Jan 2000
1.1 draft C	Added Linda Bird's (DSTC) archetype examples.	T Beale	12 Feb 2000
1.1 draft D	Improved architectural overview and added scenarios with interatcion diagrams. [GP EHR project].	T Beale	20 Mar 2000
2.1 draft A	Restructure of document, better elucidation of requirements.	T Beale	1 Apr 2000
2.1 draft B	Reworked architectural diagram, detailed archetype formal definitions.	T Beale	10 May 2000
2.1 draft C	Updated diagrams and text for subtyped archetype classes (now 1:1 with GOM).	T Beale	15 May 2000
2.1 draft D	Added ORGANISER_ROOT class to clinical model.	T Beale	17 May 2000
2.1 draft E	Added EHR_CONTENT class to clinical model.	T Beale	20 July 2000
2.1 draft F	Added unit archetype classes.	T Beale	2 Aug 2000
3.1 draft A	Removed software architecture information to kernel architecture document	T Beale	12 Aug 2000
3.1 draft B	Corrected archetype tables.	T Beale	22 Aug 2000

Table of Contents

1 Introduction

1.1 Purpose

This document describes the GEHR archetype system concept, its software requirements and the *archetype domain* system.

1.2 Audience

The primary users of this document are:

GEHR kernel developers: this document provides a basis for developing kernel archetype classes.

GEHR-based application and system developers: the archetype syntax and examples described here are a guide to implementing parts of applications which need to deal with archetypes.

Archetype specifiers: people or organisations creating archetypes for use with GEHR-based systems and applications.

1.3 Status

This document is under construction. Known omissions or questions are indicated in the text with paragraphs like the following:

To Be Determined: indicating not yet resolved

To Be Continued: indicating more work required

Reviewers are invitied to comment on these paragraphs as well as the main content.

2 Background

2.1 Motivation

The GEHR object model described in The GEHR Object Model Architecture is essentially a generic knowledge representation model, driven by external archetypes to create valid clinical information structures. Archetypes express particular knowledge models, which configure the use of concrete structures in the EHR. They provide a key solution to a number of problems:

- *Disengaging the standardisation* of clinical structures from the standardisation of the concrete structures. This is the key to enabling the GEHR Object Model to be quickly completed, allowing software development to commence
- Enabling *software to be future-proof*, since no "hard-coded" clinical concepts are used in the concrete model on which implementations are based. In particular, older GEHR software will be able to read EHRs created by newer software (forward compatibility).
- Ensuring that *EHRs are future-proofed*, i.e. that all EHRs created by GEHR-based software remain viable regardless of changes to clinical structures or concepts.
- Retaining the knowledge of clinical information structures *somewhere* in the system; cf the original GOM model described in Deliverable 19, and also the CEN ENV 13606 pre-standard, which provide for very generic content structures, but no governing models, which would result in software being unable to reason about clinical structures.
- Allowing the *free development of clinical concept definitions at international, national and local levels*, via a hierarchical domain-based management system.
- Ensuring that *exactly one GEHR representation is used for a given clinical concept*, guaranteeing that transferred parts of EHRs will be understood in the same way, and also that software systems (particularly decision support) can reason about the structures.

This document describes the nature of archetypes, and how they relate to the GEHR concrete model (the GOM). By way of background, it takes an excursion into health record systems, so as to see how these artefacts are introduced into the system, and how they result in concrete object structures being created.

2.2 Technical Function

Consider a health record system, constructed using GEHR kernel software. The purpose of the system, in information terms, is to create object structures according to the classes in the GOM, and to store them using a persistence mechanism (i.e. database) of some kind. Each object structure needs to be not only "formally valid", i.e. it must correspond to the semantics of the GOM, but also "clinically valid", i.e. it must be *meaningful clinical information*. It is the latter kind of validity that archetypes are concerned with.

Clearly, an infinite variety of formally valid structures could be created, but only a few of these (nevertheless, still a very large number!) are likely to be valid or desirable structures in the clinical sense. A simple example of an invalid structure would be a prescription minus the "generic name" field, assuming this field was regarded as mandatory by the users of the information.

In a GEHR record, clinically valid structures exhibit a range of variability. It appears that most content items (structures defined by the GOM `XXX_CONTENT` classes) are variable only to the extent that their structures need be predefined with allowance for mandatory or optional elements. For example, blood pressure, audiogram, and prescription, this is certainly true: each structure could be created by

The Lego-brick Analogy

The classes of the GEHR object model are like the *specifications* for Lego® bricks. Instances of these classes - "objects" - are like *actual* Lego bricks, from which real structures can be built. The GEHR Object Model can thus be thought of as a "concrete model" since it is like the engineering specification for the set of construction elements.

Archetypes are like the *designs* for Lego structures, which come printed on paper in a Lego box. Each Lego design expresses a meaningful structure, such as a house, a tractor or a dog. The designs constrain the use of the Lego pieces to a meaningful structure space, which although vast, is not nearly as vast as the total possible structure space, i.e. all possible (but mostly meaningless) combinations of Lego bricks.

In the same way, GEHR Archetypes constrain the use of GOM objects to valid, agreed upon structures, such as for "blood pressure" or "audiology results".

And in the same way that Lego bricks were standardised and manufactured prior to the development of many (or any) designs, the GOM can be standardised independently of its archetypes. Conversely, the development of both Lego designs and GEHR Archetypes may continue forever, while still using the same basic building blocks.

Old blocks can be used for new archetypes; new blocks can be used with old archetypes.

an application retrieving a template structure, cloning it, and obtaining the values from the user. In the case of prescription, some items could be removed, and extra items could be added.

At the organiser level however, the structure is more variable. Weed's "problem/SOAP" organiser structure for organising clinical information on a problem basis is an example (here, "SOAP" stands for Subjective/Objective/Asessment/Plan, and has nothing to do with Microsoft's "simple object access protocol"). In a given transaction, there may be any number of "problems", each distinctly named, preventing the use of a simple template. More complex structures are certainly possible, and although clinicians are likely to generally follow standard models of organising information, there is no reason why they should not use site-specific navigation structures.

Archetypes provide a solution to the problem of expressing allowable clinical information, by defining structure, optionality, constraints on types, values, names and so on.

As mentioned above, they need to satisfy two basic requirements. Firstly, to be the object of clinical standardisation processes, and secondly to exist in a runtime system, so as to perform their intended configuration function. For the first purpose, we will assume that archetypes exist initially in the form of documents, while for the latter, archetypes are ultimately expressed in terms of instances of archetype classes in the kernel. (The design of these classes is somewhat special, in that they describe the valid semantics of other objects.)

Three types of archetype entity can thus be identified in the GEHR archetype system:

- Archetype documents (the original entities being authored)
- Archetype objects (for runtime use)
- Archetype kernel classes, which define archetype objects.

A mechanism is required for conversion from document form to object form, providing a way for archetype definitions to enter the EHR system.

Let us consider an example. For the "problem/SOAP" organiser structure often used in general practice, the archetype object structure illustrated in FIGURE 1 might be envisaged. This structure is a

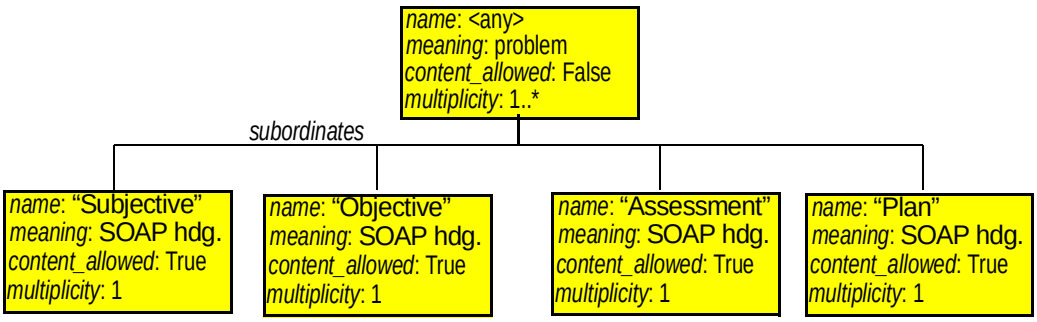

FIGURE 1 Organiser Archetype Instance

group of objects (i.e. instances, in computing terms) of kernel archetype classes. One set of archetype classes is needed to express the valid structure for organisers, but many instances may be possible, one of which is the "problem/SOAP" archetype structure. With such a structure, the GEHR kernel would be able to determine how to ensure that the organiser structure created by application calls conforms to the archetype.

Modification of EHR Content

Archetype objects play a *constraining* role not only in the *creation* of new EHR content, but in the *modification* of existing content. Although structures are never actually modified in a GEHR record, the effect is attained by the retrieval of an existing version of a structure, and committal of a modified copy of the structure, as a new version. One approach to modification would be to simply allow any kind of alteration to occur, i.e. only to require clinical correctness to be enforced at creation time. This might even be a reasonable strategy for some structures, since they will never modified - recall that the only reason to modify event transaction content is to correct errors. This same is not true however for persistent transactions - modification of content structures may be frequent, and significant, such as in the case of a family history or a summary. In general, therefore, archetypes need to be available during modification of structures.

A second aspect of modification needs to be addressed, namely that it may occur at an HCF other than the original HCF of creation of a structure. This implies that at least archetypes need to be available at both ends, enabling modification at the receiver's end. During transfer, a negotiation stage might be required, during which both parties determine whether the receiver needs to obtain a particular archetype or not.

The following box serves as a reminder as to the purposes of archetypes:

What is the purpose of Archetypes?

Definition of **clinical knowledge** concepts in a system- and technology-independent way, by users in the clinical domain.

Ensuring EHR **information quality**, by guaranteeing content always conforms to the constraints expressed in archetypes.

Enabling knowledge-level system **interoperability**, by the global use of standard and specialised archetypes.

3 Requirements

3.1 Overview

This section describes the formal requirements of archetypes, which fall into two broad categories. The first set of requirements deals with archetypes as technical artefacts: when new archetypes are created, what their semantics are, and the language of archetype documents. The second is concerned with the management of archetypes within local, national and global systems of healthcare, for which requirements for identification, authorisation, and evolution are stated.

3.2 Scope

Let us first restate the primary purpose of GEHR archetypes: *to define clinical concepts in a way amenable to offline authoring by health professionals, and online use for configuring EHR content*. The following subsections describe the dimensions of variability which need to be supported by GEHR archetypes.

3.2.1 Levels of Abstraction

The RECORD cluster of the GEHR Object Model is split into five levels: EHR, TRANSACTION, ORGANISER, CONTENT and DATA. The first two may be considered information management levels in that they provide the semantics of creation, containment, and transfer of EHR information. The `EHR_CONTENT` class, found in the TRANSACTION cluster is the starting point for what can be considered record "content", whose structure is then defined by the ORGANISER and CONTENT clusters. Archetypes are required for the root classes at each level at which content structure is defined, i.e. for the classes `EHR_CONTENT`, `ORGANISER_ROOT` and `DEFINITION_CONTENT`. We can thus talk (somewhat loosely) of "transaction" archetypes, "organiser" archetypes, and "content" archetypes, to mean the archetypes for these classes.

3.2.2 Composition

As a corollory to the notion of separate archetypes at each GOM level of abstraction, valid combinations, or *compositions* of archetypes need to be taken into account. For example, it may be decided by a national health body that in general practice, a "contact" transaction should always use a "SOAP" organiser, and that the content under the SOAP headings should be further constrained in some way. In other words, the archetypes "contact transaction", "SOAP organiser" and certain content archetypes may not be legal on their own, but only in specific compositions. Compositions might be combinations of archetypes at two levels, i.e. transaction + organiser, organiser + content, or all three, i.e. transaction + organiser + content.

Another circumstance in which particular combinations might be mandated is that of legacy software applications which have predefined visual interfaces, in which structure is more or less invariable. For instance, a diabetic summary report in a particular application may be rigidly structured like a database report, in which certain fields are filled in to produce a completed report. In addition, legacy screens will not normally have been designed with a GEHR-like mentality of transaction/organiser/content/data. In any case, archetypes supporting legacy interfaces may need to be completely specified, so that the application developer is not forced to introduce unnecessary choices in the user interface, requesting which archetypes to put together to create the report.

Completely- or partially-specified *composite archetypes are therefore needed to define legal combinations*, for both agreed standards at various levels as well as vendor-specific legacy software.

To clarify the language of archetypes, we will describe a clinical concept at a single GOM level as a *primitive clinical concept*, and one which is a combination of primitive concepts as a *composite clinical concept.* Accordingly, a *primitive archetype* will refer to the archetype for a concept at one GOM level, while a *composite archetype* will refer to one specifying a combination.

3.2.3 Specialisation

It is a stated aim of GEHR that it should not presuppose or constrain how clinical medicine is practised. GEHR models and systems therefore must be amenable to different cultural (in both its medical and ethnic meanings) and regional norms. One way this aim can be satisfied is with flexibility in the archetype system. It can *allow for culturally specific concepts* as might occur in chinese medicine, naturopathy, tropical medicine in developing countries, and so on, simply by accepting archetypes for the relevant concepts. Thus the total GEHR archetype space might include whole categories of archetype particular to disparate medical cultures in the world. It is quite likely that specific archetypes at the organiser level will be defined for the way information is organised during carer/patient contacts in, for example, refugee camps, malaria treatment centres, and ayurvedic medicine.

Another kind of required flexibility is to allow for variations on previously existing archetypes, corresponding to national or regional versions of a concept, or more precise versions of a general concept as might be required for expressing test results from new hi-tech machines.

For example, if a standard GEHR archetype exists for the SOAP organiser, there may well be regional variations, in which other headings are added (such as the “E”/education heading used in some places). Examples of national specialisation might be a prescriber number on a prescription in Australia or the RFA requirements on cervical cytology recording in the United Kingdom.

Standard archetypes for basic concepts such as prescription, weight and blood pressure might be customised by some countries to include more elements, to make “optional” settings in the standard version “mandatory”, to add protocols and so on. Archetypes for more complex concepts such as care plans and diabetic summaries are very likely to require national or regional customisation due to differing current practices and clinical trends.

Local specialisation may be used to ensure standard recording where this is required, or by professional groups setting standards for quality record keeping.

3.2.4 Evolution in Time

The example of “current clinical practice” also reminds us that clinical norms change in time, and it is to be expected that no concept is “true” for all time. Thus *archetypes need to be able to evolve in time* as well. However, information created by a system using a previous archetype for the same concept must remain sensible, indicating the need for time (or version) to be included in archetype identification.

3.3 Archetype Document Management

The use of archetypes introduces an element of complexity into the GEHR system, since archetypes themselves have to managed. Management problems can be characterised as follows:

Identification: how are archetypes identified, in particular to ensure that names are globally unique?

Sourcing: where do HCFs obtain archetypes from?

Verification: how can an archetype instance be verified with respect to its name? Is there a standard repository of instances?

3.3.1 Identification

Archetypes need to be globally uniquely identified, mandating the need for a single overseeing body to manage the identification scheme.

An identification scheme must taken into account the archetype space described in the previous section, which may be summarised as a primitive space and a composite space, each characterised by the dimensions:

- Authoring/accrediting authority.
- Distinct clinical concepts.
- Cultural specialisations.
- Time-related versions.

Hierarchy should be assumed in both the clinical and specialisation dimensions.

The naming of composite archetypes should probably be derived unambiguously from the clinical name parts of the primitive constituents; note that specialisation and versioning is (most likely) also required for composites.

Parent Archetype Derivability

The naming scheme should also ensure that for any archetype, its immediate *clinical* parent archetype name can be determined. This requirement ensures that the recipient of record content based on a local (to the sender) or otherwise unknown archetype can process the content with the *nearest appropriate clinical archetype*, without having to guess what this is. For example, if an EHR transaction was received in the UK containing blood pressures constructed using the archetype “au-nt.cont.bp-4th-sound.v1” (a Northern Territory, Australia specialisation of the content archetype “gehr.cont.bp.v1”) the receiver may not have access to the special archetype, but would at least (via some rule) be able to derive the name of the parent archetype from which it was derived. This ensures that the receiving system can process the blood pressure items sensibly, by using the assumption that they were extensions to the parent blood pressure archetype.

It may be the case that the name of the parent archetype name is also not known; therefore the derivability requirement must be met in a recursive fashion, so that a recipient can eventually discover the parent archetype closest in clinical meaning.

Possible design approaches to this problem include:

- Parent archetype names are derivable directly and recursively from an initial archetype name. This approach mandates a hierarchically structured namespace for the clinical name section of an archetype name.
- The whole namespace, including relationships between names is available for inspection by all EHR sites at any time. In other words, an EHR source can interrogate an archetype namespace server for the clinical parent of a specified archetype name.

Versioning

It is particularly important that version identifiers should be included in names, to ensure that even slight changes to the meaning of archetypes causes an explicit name change.

Subsequent versions of archetypes must be backwardly compatible for querying, that is to say, any later version must define EHR structures which respond to the same queries as earlier versions.

3.3.2 Sourcing

To Be Determined:

3.3.3 Verification

To Be Continued:

3.4 Archetype Documents

As implied in the previous section, two styles of archetype document are required: one to express primitive archetypes whose content will refer to GEHR kernel concepts, and one to express legal compositions, whose content will contain references to primitive archetypes.

A syntax used to express primitive archetype document instances must satisfy the following broad aims:

- Powerful enough to express GOM content structures, and their variability.
- Can be parsed in order to create kernel archetype objects, capable of creating new content structures.
- Can be created and edited with relatively simple tools, independently of EHR systems.

3.4.1 Semantics

Primitive archetypes express constraints at three levels of the GOM, as described in the following sections. Eiffel-style types are used in the following to indicate the logical types of the fields *in the archetype*; where *types* in GOM structures are indicated, this is done with `LIST[STRING]` or similar. Chained archetypes are expressed as match-patterns for archetype names, and appear in green in the table.

All Archetypes

The following characteristics are required for all archetypes:

- *gehr_archetype_id*:`STRING` -- the official GEHR id of the archetype, e.g. `"gehr.cont.bp.v1"`
- *concept*:`PLAIN_TEXT` -- the clinical concept name of this archetype, e.g. `"arterial blood pressure"`, `"contact transaction"`, `"care plan"`, or a `TERM_TEXT` equivalent, e.g. a UMLS CUI.

EHR_CONTENT

The following constraints can be expressed for the `EHR_CONTENT` class in the Transaction cluster:

Archetype constraint	Constrains GOM element	Explanation
persistent:`BOOLEAN`	`EHR_CONTENT.`*is_persistent*`:BOOLEAN`	indicates whether transaction is of persistent or temporal event type
context_archetype_id_pattern: `STRING`	`EHR_CONTENT.`*context*`: DEFINITION_CONTENT`	pattern defining allowed `DEFINITION_CONTENT` archetype names for the context. Regular expressions are allowed. See Composite Archetypes below.
content_archetype_id_pattern: `STRING`	`EHR_CONTENT.`*content*`: ORGANISER_ROOT`	pattern defining allowed `ORGANISER_ROOT` archetype names for this transaction. Regular expressions are allowed. See Composite Archetypes below.

ORGANISER_ROOT

For each organiser the following constraint definition is required:

Archetype constraint	Constrains GOM element	Explanation
valid_names:(text constraint type)	ORGANISER. *name:*PLAIN_TEXT	set of allowable names (including wildcards) and/or terms or CUIs as well as the *meaning* of the name attribute.
minimum_occurrences:INTEGER	ORGANISER.*organisers:* LIST[ORGANISER]	cardinality of this organiser under its owner. Use "*" for infinity. Mandatory is expressed by "1,1", optional by "0,1"
maximum_occurrences:INTEGER		
organisers: LIST[??]		list of organiser constraint definitions of allowed subordinate organisers
organiser_archetype_id_pattern: STRING		allowed organiser root archetype names, in the form of a regular expression string. See Composite Archetypes below.
organiser_roots_allowed: BOOLEAN		are separately archetyped organiser trees allowed at this level, or only more organisers?
content_allowed: BOOLEAN	ORGANISER.*content_items:* LIST[DEFINITION_CONTENT]	is content allowed at this level, or only more organisers?
content_archetype_id_pattern: STRING		allowed content archetype names, in the form of a regular expression string. See Composite Archetypes below.

DEFINITION_CONTENT

The archetypes for content must indicate characteristics for the content root as per the table below. In the table, the second column gives the name of the GOM element constrained; entries correspond to DEFINITION_CONTENT or one of its subtypes. When defining the values in this table for a given archetype, the correct subtype must be taken into account. For example, an archetype for SUBJECTIVE_CONTENT (e.g. a differential diagnosis) must provide values for rows in the table below where the second column refers to DEFINITION_CONTENT, PREDICATE_CONTENT, or SUBJECTIVE_CONTENT.

Archetype constraint	Constrains GOM element	Explanation
context_required: BOOLEAN	DEFINITION_CONTENT. *context*	may be False for content_type = DEFINITION_CONTENT (note that context now exists on DEFINITION_CONTENT, allowing *recorder* and *comment* to be recorded)
protocol_required: BOOLEAN	PREDICATE_CONTENT. *protocol*	must be False for content_type = DEFINITION_CONTENT
subject: (text constraint type)	PREDICATE_CONTENT. *subject:*TERM_TEXT	for all types except DEFINITION_CONTENT. Values e.g. "self", "foetus", "mother".

Archetype constraint	Constrains GOM element	Explanation
provider: ???	SUBJECTIVE_CONTENT.*context.provider:* PERSON_IMPL	values = "HCP", "self", "mother", "automated" etc
protocol	PREDICATE_CONTENT.*protocol:* HIERARCHICAL_PROPOSITION	Constraints are expressed according to the Hierarchical Propositions section below.
content	DEFINITION_CONTENT.content: HIERARCHICAL_PROPOSITION	Constraints are expressed according to the Hierarchical Propositions section below.

Hierarchical Propositions

A number of places in the GOM have free-form content in the form of a HIERARCHICAL_PROPOSITION, including EHR_CONTENT.*context*. For both the main content, and the protocol content (which is also of type HIERARCHICAL_PROPOSITION), constraints are expressed in the following table.

To Be Continued: subtypes of HP

For HIERARCHICAL_PROPOSITION:

Archetype constraint	Constrains GOM element	Explanation
For the root HIERARCHICAL_PROPOSITION object		
form:STRING	HIERARCHICAL_PROPOSITION.*form*	the structural form of the information. A value from the list "hierarchy" | "simple" | "list" | "time_series" | "regular_time_series" | "tree" | "table" | "matrix" | "xy_plot" | ...
For each HIERARCHICAL_ITEM object (i.e. HIERARCHICAL_GROUP and HIERARCHICAL_VALUE)		
default_name: PLAIN_TEXT	HIERARCHICAL_ITEM.*name*: PLAIN_TEXT	name of this node after default creation has occurred
valid_names:(text constraint type)		set of valid name patterns for this node. Specify TERM_TEXTs or CUIs here if necessary, or just strings, including wildcards, e.g. "diastolic *". Also specify meaning.
context_required: BOOLEAN	HIERARCHICAL_ITEM.*context*: ANY_CONTEXT	is context required at this node?
minimum_occurrences, *maximum_occurrences*: INTEGER	HIERARCHICAL_GROUP.*children, values*	Cardinality of this item under its parent. Use "*" to mean infinity. Mandatory is expressed by "1,1", optional by "0,1"
For each HIERARCHICAL_GROUP object:		

Archetype constraint	Constrains GOM element	Explanation
groups	HIERARCHICAL_GROUP.*children*: LIST[HIERARCHICAL_GROUP]	the list of predefined HIERARCHICAL_GROUP items under this one
new_group		optionally, a HIERARCHICAL_GROUP specifying the constraints on new group nodes added, if this is allowed
values	HIERARCHICAL_GROUP.*values*: LIST[HIERARCHICAL_VALUE]	the list of predefined HIERARCHICAL_VALUE items under this one
new_value		optionally, a HIERARCHICAL_VALUE specifying the constraints on new value nodes added, if this is allowed. Probably Void for a blood pressure archetype, but for an address archetype, could allow other fields, with any name and type
For each HIERARCHICAL_VALUE object:		
default_value: DATA_VALUE	HIERARCHICAL_VALUE.*values*: DATA_VALUE	a value of a subtype e.g. "QUANTITY: 13 ml". Can just state "Void" for no default
type_value_constraints	HIERARCHICAL_VALUE.*values*: DATA_VALUE	a set of type/value constraints, of the form of a table {"type_name", {value constraint}+}. Value constraints. See example below.

Example type/value constraints for HIERARCHICAL_VALUE.*value*:DATA_VALUE.

Type	Values
"PLAIN_TEXT", "TERM_TEXT"	value matches pattern: "diabetes mellitis|diabetes insipidus" value matches "diabetes.*" termset matches "ICD.*|UMLS" term matches {UMLS:cui=nnnn, UMLS:cui=mmm, ...}
"QUANTITY"	range: 80 mmHg - 110 mmHg range: >= 80 mmHg value: 110mmHg ANY
"DATE", "TIME, "DATE_TIME", etc	from 1/1/2000 to 2/5/2000 on or after 4/5/2000 at 10:18am
etc	

For data types not shown above, new constraint models can be designed as appropriate.

3.4.2 Composite Archetypes

As a logical concept, the purpose of composite archetypes is easily understood: to constrain the possible combinations, of "chains" of archetypes in the vertical sense, to those which are clinically valid. As a design concept, things are less clear, since there are at least two obvious ways to go.

The first possibility is for composite archetypes to exist as separate, special documents in their own right, expressing only the valid combinations in a certain context.

The second possibility is for existing archetype documents to include an indication of legal sub-archetypes. Initially, this approach apears limiting, since in some cicumstances, very specific sub-archetypes may be required in some health care contexts, while in other places, very general ones may be required - both coming under the same parent archetype. This would seem to lead to a situation where an organiser archetype (constraint definition for an `ORGANISER_ROOT`) might need different lists of allowable archetypes for content (ids of `DEFINITION_CONTENT` level archetypes), depending on the HCF, or some other factor (e.g. subject type = foetus rather than adult).

However, early analysis shows that in fact, in most (if not all) cases, higher-level archetypes will always have a sensible, meaningful default set of sub-archetypes. Where this set needs to be arbitrarily restricted, it can be done by archetype specialisation. It could be argued that this is a misuse of specialisation, but from the clinical point of view, it must in fact be the case - specialising the definition of legal sub-archetypes must have a clinical meaning, or else it does not correspond to clinical facts in the real world.

In GEHR, the current approach to archetype-chaining is the second approach, since it is easier to implement, and appears to be valid for all cases studied. It may be that in the future a more flexible method is needed, but this is unlikely to be known without an initial implementation and a number of clinical trials.

Terms

see: www.nlm.nih.gov for UMLS

Wherever `PLAIN_TEXT` or `TERM_TEXT` appears in the GOM, the expansion of a termset code may appear. GEHR uses the CUI (Concept Unique Identifier) of the UMLS (US National Library of Medicine coding scheme) to specify concept codes for any attribute in the model (apart from `HIERARCHICAL_VALUE`.*value*, i.e. the use of `PLAIN_TEXT` or `TERM_TEXT` as data, where any term set including UMLS may be used). Archetypes need to include both the term expansion, i.e. the actual string required (e.g. “Atrial Fibrillation”), and the CUI (e.g. C0004238).

Where terms are used as data, there is often an implied enumeration of possible values, such as the values for “sex”, which are typically “male”, “female”, “not determined”, “not disclosed”. Each of these terms can be found in an appropriate term set (presumably UMLS). The set comprising the allowable enumeration needs to appear in relevant value fields in archetypes.

3.4.3 Default Structure

In most cases, a default structure, or template, needs to be provided by an archetype, primarily for the purposes of initial creation for GUI applications. For example, the default structure for the SOAP organiser archetype might be a single problem organiser called “<problem #1>”, with the four required sub-organisers beneath it. This would enable any GUI application to display a sensible default which further modifications can be done to as necessary. Default structures are important for most content items, such as blood pressure and prescription, since the final result will typically be very much like the default, but with actual values set.

To Be Determined: Alternative #2: default structure is deducible from archetype content, and should be built on the fly during parsing.

4 Archetype Management

4.1 Archetype Domains

A suitable basis for managing archetypes is the *archetype domain*: a context in which archetypes are *identified* and *issued*. Each domain needs to be an archetype naming (identifying) authority, ensuring unique archetype identification. The namespace used by each domain needs to be globally assigned, e.g. by the *open*EHR Foundation. FIGURE 2 illustrates a possible archetype domain hierarchy.

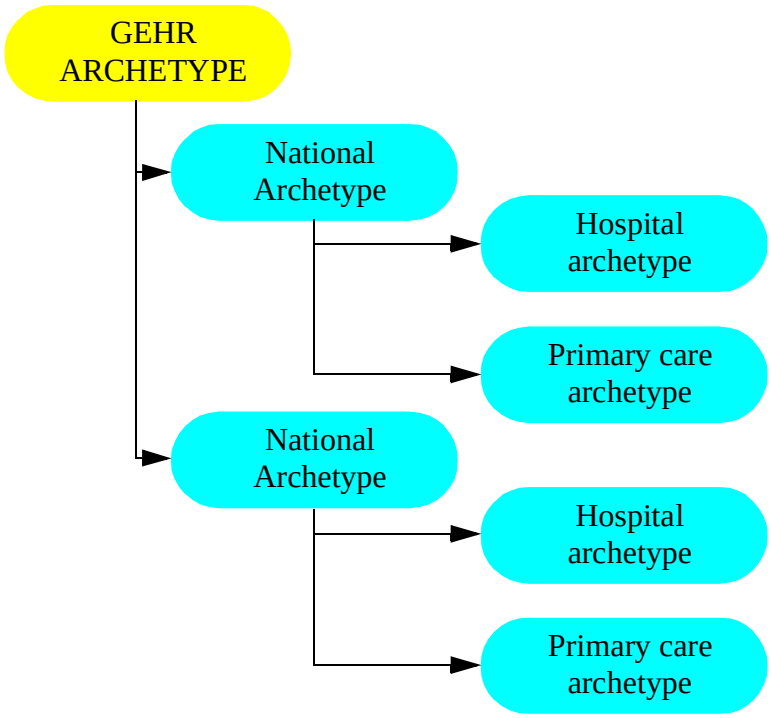

FIGURE 2 Archetype Specialisation Hierarchy

Each domain should maintain an *archetype domain server*, providing an online reference repository of its archetypes, from which any HCF could download archetypes on request. This approach means that EHR sources do not need to worry about whether the receiver of transferred record extracts have the required archetypes; the receiver will always be able to read archetype identifiers from the extract, and source archetypes as needed from a domain archetype server.

Every domain server should know how to retrieve archetypes from at least one alternate server, and servers must know the identity of an ultimate server, in a similar fashion as the internet DNS.

Archetype domain servers should support the following queries (in pseudo-Eiffel syntax):

- `get_archetype(id:STRING): ARCHETYPE`
- `is_valid_archetype_id(id:STRING):BOOLEAN`
- `is_latest_archetype(id:STRING):BOOLEAN`
- `get_latest_archetype(partial_id:STRING):ARCHETYPE`
- `get_matching_archetypes(partial_id:STRING):SET[ARCHETYPE]`
- `get_matching_archetype_ids(partial_id:STRING):SET[STRING]`

4.2 Identification

To Be Continued:

5 Archetype Documents

5.1 Formalism

Since the archetype instance structures described above are likely to be the object of significant authoring and review processes in their own right, XML has been chosen an appropriate formalism, for a number of reasons:

- It is capable of expressing object semantics, in its XML-schema guise.
- It is becoming widely accepted in the IT industry, and is likely to become a *defacto* document format for the web and simple text editors. Its standards are openly available from `w3.org`.
- It is a parseable text format, ensuring that it is readable even with the simplest of tools.
- Parsers between XML and object oriented languages are becoming available.

Standardisation processes for XML templates and archetypes are likely at the international level (e.g. CEN, ISO, GEHR Foundation), the national level (peak clinical practice bodies), and quite possibly at sites as well. As archetypes and templates become available, they are added to the database at each site, enabling that site to being creating new types of standardised clinical information. For this information to be available to GEHR applications, it must first be converted from XML to GEHR internal format.

5.1.1 Example

The following example is are initial archetypes, by the Titanium team at DSTC, Qld, Australia (`http://www.dstc.org.au`), written using XML-schema. These archetypes can be parsed by the archetype_initialiser application, which creates instances of A_* classes as a result.

The first shows a standard blood pressure structure, incorporating a protocol

```
<!--  Blood Pressure Archetype - XML SCHEMA DEFINITIONS   -->
<!--                                                      -->
<!--      GEHR identifier: gehr.cont.observe.bloodpressure.v2 -->
<!--      Example of how extension might work         -->

<schema
 targetNamespace="http://www.gehr.org/namespace/gehr.cont.observe.bloodpressure.v2/"
 xmlns="http://www.w3.org/1999/XMLSchema"
 xmlns:gom="http://www.gehr.org/namespace/GOM/"
 xmlns:bp="http://www.gehr.org/namespace/gehr.cont.observe.bloodpressure/">
 xmlns:bp2="http://www.gehr.org/namespace/gehr.cont.observe.bloodpressure.v2/">

  <import namespace="http://www.gehr.org/namespace/GOM/"
   schemaLocation="http://www.gehr.org/archetypes/gom.xsd"/>

  <import namespace="http://www.gehr.org/namespace/gehr.cont.observe.bloodpressure/"
   schemaLocation="http://www.gehr.org/archetypes/gehr.cont.observe.bloodpressure.xsd"/>

  <!-- Top-level Archetype -->

  <complexType name="gehr.cont.observe.bloodpressure.v2"
   base="bp:gehr.cont.observe.bloodpressure" derivedBy="restriction">
    <element name="proposition" type="bp:BPProposition2"/>
    <attribute name="gehr_archetype_id" use="fixed"
     value="gehr.cont.observe.bloodpressure.v2"/>
  </complexType>

  <!-- Proposition -->

  <complexType name="BPProposition2" base="bp:BPProposition" derivedBy="restriction">
    <element name="root" type="bp2:BPPropositionRoot2">
  </complexType>

  <complexType name="BPPropositionRoot2" base="bp:BPPropositionRoot" derivedBy="extension">
    <element name="additional_field" type="gom:G1_HIERARCHICAL_VALUE"/>
  </complexType>
</schema>
```

The following archetypes express the SOAP organiser example.

(To be replaced with latest update of SOAP organiser archetype)

END OF DOCUMENT