The *open*EHR Reference Model

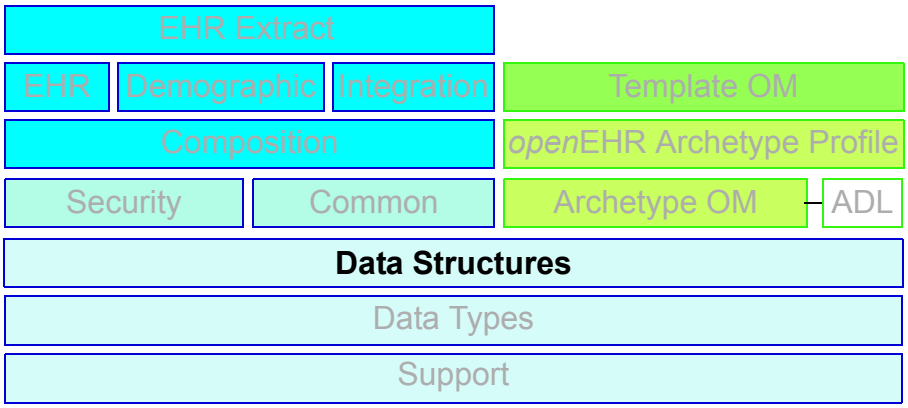
Data Structures Information Model

Editors: {T Beale, S Heard}[a], {D Kalra, D Lloyd}[b]		
Revision: 1.7	*Pages:* 39	*Date of issue:* 26 Sep 2006

a. Ocean Informatics

b. Centre for Health Informatics and Multi-professional Education, University College London

Keywords: EHR, reference model, openehr, data structures



Copyright Notice

Amendment Record

Issue	Details	Who	Completed
RELEASE 1.0.1			
1.7	**CR-000200**. Correct Release 1.0 typographical errors. Minor cosmetic changes to diagrams. Correct return types of ITEM_TABLE functions to CLUSTER rather than List<ELEMENT>. **CR-000207**: Change ITEM_TABLE *columns* to *rows*. **CR-000219**: Use constants instead of literals to refer to terminology in RM. **CR-000220**: Tighten semantics of HISTORY.*period* and EVENT.*time*.	D Lloyd, T Beale S Heard R Chen A Patterson	26 Sep 2006
RELEASE 1.0			
1.6	**CR-000014**. Adjust History. Major simplifcation to package; make Events absolute in time. **CR-000155**: Summary data. **CR-000183**. Remove root node from ITEM_TREE. **CR-000185**. Improved EVENT model. **CR-000155**: Summary data. **CR-000192**: Add display-as-absolute facility to delta Events in History (added explanation only). **CR-000193**: Simplify INTERVAL_EVENT for archetyping and paths. Revert to one math function per INTERVAL_EVENT. **CR-000196**: Rename HISTORY.*items* to *events*. **CR-000192**. Support change, increase and decrease Events in History.	S Heard T Beale S Heard G Grieve S Heard S Heard S Heard G Grieve S Heard S Heard S Heard S Heard D Lloyd	16 Dec 2005
RELEASE 0.95			
1.5.1	**CR-000048**. Pre-release review of documents. Fixed HISTORY UML diagram - remove superfluous T:XXX (no semantic change). Converted parameter types to UML box form.	D Lloyd	22 Feb 2005
1.5	**CR-000101**. Improve modelling of Structure classes. **CR-000100**. Correct inheritance error in ITEM_STRUCTURE package. **CR-000024**. Revert *meaning* to STRING and rename as *archetype_node_id*. **CR-000118**. Make package names lower case. **CR-000123**. EVENT should inherit from LOCATABLE. **CR-000124**. Fix path syntax in data structures IM document.	DSTC T Beale S Heard, T Beale T Beale Rong Chen T Beale	10 Dec 2004
RELEASE 0.9			

Issue	Details	Who	Completed
1.4	**CR-000019**. Add HISTORY & STRUCTURE supertype. **CR-000028**. Change name of STRUCTURE class to avoid clashes. **CR-000089**. Remove ITEM.*displayed*. **CR-000091**. Correct anomalies in use of CODE_PHRASE and DV_CODED_TEXT. **CR-000067**. Change EVENT class to enable math function interval measurements. Renamed EVENT.*duration* and SINGLE_EVENT.*duration* to *width*. **Formally validated using ISE Eiffel 5.4.**	T Beale H Frankel DSTC T Beale S Heard	09 Mar 2004
1.3.3	**CR-000041**. Visually differentiate primitive types in openEHR documents.	D Lloyd	04 Sep 2003
1.3.2	**CR-000013** Rename key classes - rename COMPOUND to CLUSTER to conform with CEN 13606.	D Kalra, T Beale	20 Jun 2003
1.3.1	Improved heading layout, package naming. Made HISTORY.*is_periodic* a function.	T Beale, Z Tun	18 Mar 2003
1.3	**Formally validated using ISE Eiffel 5.2.** No changes.	T Beale	20 Feb 2003
1.2.1	Minor corrections to terminology_id invariants.	Z Tun	08 Jan 2003
1.2	Defined packages properly and moved HISTORY classes from EHR RM. No change to semantics.	T Beale	18 Dec 2002
1.1.1	Minor corrections: SINGLE_S new function.	T Beale	10 Nov 2002
1.1	Minor adjustments due to change in DV_CODED_TEXT.	T Beale	1 Nov 2002
1.0	Taken from Common RM.	T Beale	11 Oct 2002

Acknowledgements

The work reported in this paper has been funded in by a number of organisations, including Ocean Informatics; The University College, London; Australia; The Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

Thanks to Grahame Grieve of Kestral Computing for general input and examples relating to History data.

Table of Contents

1 Introduction

1.1 Purpose

This document describes the common data structures used in *open*EHR reference model, including lists, tables, trees, and history, along with one possible data representation (hierarchical) which is compatible with the CEN 13606 EHCR standard.

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;
- Medical informaticians and clinicians intersted in health information;
- Health data managers.

1.2 Related Documents

Prerequisite documents for reading this document include:

- The *open*EHR Architecture Overview
- The *open*EHR Modelling Guide
- The *open*EHR Support Information Model
- The *open*EHR Data Types Information Model

1.3 Status

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

This document is available at http://svn.openehr.org/specification/TAGS/Release-1.0.1/publishing/architecture/rm/data_structures_im.pdf.

The latest version of this document can be found at http://svn.openehr.org/specification/TRUNK/publishing/architecture/rm/data_structures_im.pdf.

Blue text indicates sections under active development.

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 provided on the mailing list openehr-technical@openehr.org, or by private email.

1.5 Conformance

Conformance of a data or software artifact to an *open*EHR Reference Model specification is determined by a formal test of that artifact against the relevant *open*EHR Implementation Technology Specification(s) (ITSs), such as an IDL interface or an XML-schema. Since ITSs are formal, automated derivations from the Reference Model, ITS conformance indicates RM conformance.

2 Background

2.1 Requirements

The requirements for structured data in the EHR and other systems are essentially that low-level data can be expressed in standard structures. The structures which are commonly required are as follows:

- single values, e.g. weight, height, blood sugar;
- lists of named/numbered elements, e.g. blood test results;
- tables of values with named columns and/or named rows, e.g. visual acuity results;
- trees of values, e.g. biochemistry, microbiology results;
- histories of values, each of which takes any of the above forms, e.g. a time series of blood pressures, glucose levels, or imaging data.

2.2 Design Principles

The design principle which particularly applies to the data structure models described here is the need to provide explicit specifications for logical structures using the same generic representation, such as hierarchy. The logical structures include tables, lists, trees, and the concept of history.

Regardless of whether such structures are treated as pure presentation or as having semantic significance, there are at various reasons for explicitly including the semantics of logical structures which are represented in a generic way such as hierarchy, including:

- it is essential for interoperability that a structure such as a logical table, list or linear history be encoded into the generic representation in the same way by all senders and receivers of information, otherwise there is no guarantee that any communicating party's software processes the structures in the intended fashion;
- software implementors can develop software which explicitly captures the logical structures as functional interfaces which are used as the only way of building such structures. Such interfaces (assuming they are bug-free) guarantee that all application software always creates correct structures - there is no need to rely on caller software each time making low level calls to create a table or list out of hierarchy elements;
- the use of a functional interface for such types means that application software at the receiver's end can always process incoming information in its intended form, enabling correct presentation of data on the screen.

One of the motivations for defining logical data structures explicitly is to remove the ambiguity in recording structure and time in previous EHR specifications and standards, such as CEN 13606, GEHR, GEHR Australia, and HL7v3 CDA specifications. The alternative in the past was to simply use generic hierarchical structures; there was no agreement in the standard about how a table might be represented, similarly, time had no standard representation. Where single values were recorded, an attribute meaning 'time of recording' was set appropriately; if a time series was required, there was no clear guideline as to how to model it. One way would have been to build a double list which is logically a two column table, whose first column was time-point data, but many other approaches are possible. The standardised approach removes all such ambiguity, and improves the quality of data and software.

3 Overview

The rm.data_structures package contains two packages: the item_structure package and the history package. The first describes generic, path-addressable data structures, while the latter describes a generic notion of linear history, for recording events in past time. The data_structures package is illustrated in FIGURE 1.

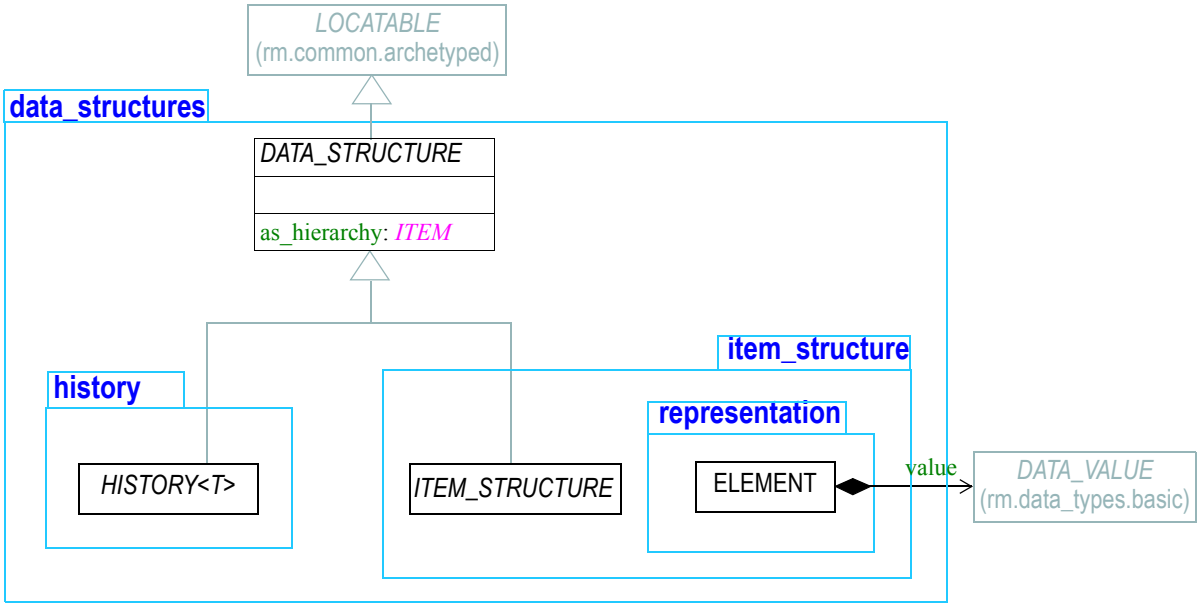


FIGURE 1 rm.data_structures Package

The data_structures package itself contains a single class, DATA_STRUCTURE, which is the ancestor of all *open*EHR data structures. Its only feature is the function *as_hierarchy*, which is implemented by each subtype of DATA_STRUCTURE, in order to generate a physical representation of the structure in CEN EN13606 form. The 13606 form is usually less optimal than the *open*EHR form, but is compatible with the less semantically rich standard, and is guaranteed (in theory) to be comprehensible to other systems which support CEN EN13606 as an interoperability standard.

3.1 Instance Structures

Diagrams of typical instances of the structures are included throughout this document. Each instance of shown in both physical and logical form. The physical form shows the instances which will occur in data if the structure is implemented using the representation package. The logical form shows the same instance in a logical form only - i.e. hiding the physical implementation. Only the latter form is used in other *open*EHR documents. In all instance diagrams, the following shorthand is used for well-known attribute names:

- "m = xxxx" - means "meaning = xxxx", i.e. the meaning of the *archetype_node_id* attribute inherited from the LOCATABLE class.
- "n = xxxx" - means "name = xxxx", i.e. the value of the *name* attribute inherited from the LOCATABLE class.
- "v = xxxx" - means "value = xxxx", i.e. the value of the *value* attribute from the ELEMENT class.

3.2 Class Descriptions

3.2.1 DATA_STRUCTURE Class

CLASS	*DATA_STRUCTURE (abstract)*	
Purpose	Abstract parent class of all data structure types. Includes the *as_hierarchy* function which can generate the equivalent CEN EN13606 single hierarchy for each subtype's physical representation. For example, the physical representation of an `ITEM_LIST` is `List<ELEMENT>`; its implementation of *as_hierarchy* will generate a `CLUSTER` containing the set of `ELEMENT` nodes from the list.	
Inherit	`LOCATABLE`	
Function	**Signature**	**Meaning**
	as_hierarchy: `ITEM`	Hierarchical equivalent of the physical representation of each subtype, compatible with CEN EN 13606 structures.
Invariants	***As_hierarchy_exists***: as_hierarchy /= Void	

4 Item Structure Package

4.1 Overview

The Item_Structure classes presented here are a formalisation of the need for generic, archetypable data structures, and are used by all *open*EHR reference models.

The subtypes of the ITEM_STRUCTURE class explicitly model the logical data structure types which typically occur in health record data, and include ITEM_SINGLE (for single values such as a patient weight), ITEM_LIST (for lists such as parts of an address), ITEM_TREE (for hierarchically structured data such as a microbiology report) and ITEM_TABLE (for tabular data such as visual acuity or reflex test results). Each of these classes defines a functional interface, has an optimal physical representation using the basic types CLUSTER and ELEMENT from the representation package, and can generate a CEN EN13606-compliant hierarchical representation of its data. Any system implementing these types is guaranteed to create data which represents the logical structures of lists, tables and trees identically.

Data values are connected to spatial structures via the *value* attribute of the ELEMENT class of the Representation cluster. This class also carries an important attribute *null_flavor*, whose value indicates how to read the value. A small domain termlist containing values such as “unknown”, “not disclosed”, “undetermined”, etc, as described in the Flavours of Null vocabulary in the *open*EHR Support Information Model.

The *open*EHR class model for spatial structures is illustrated in FIGURE 2. It should be noted that these classes (ITEM_LIST etc) are not equivalents of similarly named classes (such as List<T>) in most data structure libraries - they also include per-node *name*, *archetype_node_id* and leaf node value and null flavour, and path capabilities.

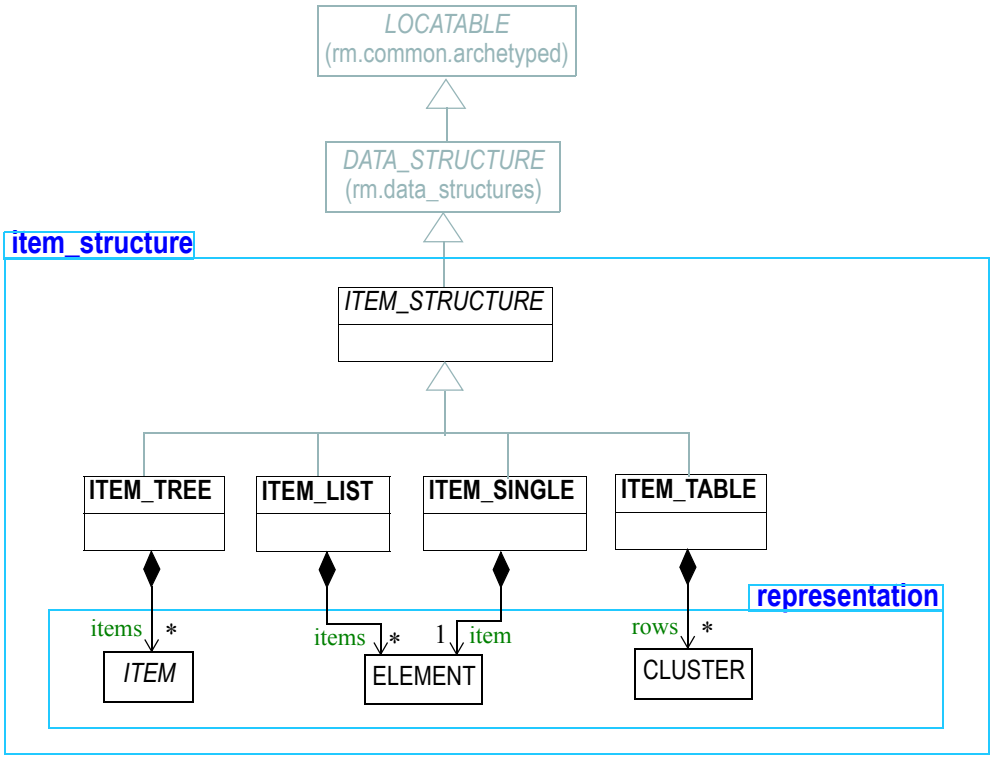


FIGURE 2 rm.data_structures.item_structure Package

4.2 CEN EN13606 Encoding Rules

4.2.1 ITEM_SINGLE

An ITEM_SINGLE object is encoded in EN13606 as a single ELEMENT object.

4.2.2 ITEM_LIST

An ITEM_LIST object is encoded in EN13606 as a CLUSTER object containing the set of ELEMENTs from the *open*EHR list.

4.2.3 ITEM_TABLE

The ITEM_TABLE encoding rules are as follows:

- Each row is encoded as a Cluster containing a number of Elements, each corresponding to the value of a column in that row.
- An empty/void column value for a row is represented by an Element containing no value, and with *null_flavour* set.
- The names of the Element in a row are the column names.
- The names of the containing Cluster of each row is the stringified number of the row in the overall table.

4.2.4 ITEM_TREE

Data of an ITEM_TREE instance are simply replicated as is to produce the correct EN13606 hierarchical form.

4.3 Class Descriptions

4.3.1 ITEM_STRUCTURE Class

CLASS	*ITEM_STRUCTURE (abstract)*	
Purpose	Abstract parent class of all spatial data types.	
GEHR	G1_HIERARCHICAL_PROPOSITION	
HL7v3	CDA Structure abstract type.	
Inherit	DATA_STRUCTURE	
Abstract	**Signature**	**Meaning**
Invariants		

4.3.2 ITEM_SINGLE Class

CLASS	ITEM_SINGLE
Purpose	Logical single value data structure.

CLASS	ITEM_SINGLE	
Use	Used to represent any data which is logically a single value, such as a person's height or weight.	
GEHR	`G1_SIMPLE_PROPOSITION`	
HL7v3	CDA `Item` type.	
Inherit	`ITEM_STRUCTURE`	
Attributes	**Signature**	**Meaning**
1..1	**item**: `ELEMENT`	
Functions	**Signature**	**Meaning**
	as_hierarchy: `ELEMENT`	Generate a CEN EN13606-compatible hierarchy consisting of a single `ELEMENT`.
Invariants	***Item_valid***: item /= Void	

4.3.3 ITEM_LIST Class

CLASS	ITEM_LIST	
Purpose	Logical list data structure, where each item has a value and can be referred to by a name and a positional index in the list. The list may be empty.	
Use	Used to represent any data which is logically a list of values, such as blood pressure, most protocols, many blood tests etc.	
MisUse	Not to be used for time-based lists, which should be represented with the proper temporal class, i.e. `HISTORY`.	
GEHR	`G1_LIST_PROPOSITION`	
HL7v3	CDA 1.0 `List Entry` type.	
Inherit	`ITEM_STRUCTURE`	
Attributes	**Signature**	**Meaning**
0..1	**items**: `List<ELEMENT>`	Physical representation of the list.
Functions	**Signature**	**Meaning**
1..1	**item_count**: `Integer`	Count of all items
1..1	**names**: `List<DV_TEXT>`	Retrieve the names of all items

CLASS	ITEM_LIST	
	named_item(a_name:`String`**)**: `ELEMENT`	Retrieve the item with name ‘a_name’
	ith_item(i:`Integer`**)**: `ELEMENT`	Retrieve the i-th item with name
	as_hierarchy: `CLUSTER`	Generate a CEN EN13606-compatible hierarchy consisting of a single `CLUSTER` containing the `ELEMENT`s of this list.
Invariants	***Valid_structure***: items.forall({ITEM}.type = “ELEMENT”) ***Names_valid***: names /= Void	

4.3.4 ITEM_TABLE Class

CLASS	ITEM_TABLE	
Purpose	Logical relational database style table data structure, in which columns are named and ordered with respect to each other. Implemented using Cluster-per-row encoding. Each row Cluster must have an identical number of Elements, each of which in turn must have identical names and value types in the corresponding postions in each row. Some columns may be designated ‘key’ columns, containing key data for each row, in the manner of relational tables. This allows row-naming, where each row represents a body site, a blood antigen etc. All values in a column have the same data type.	
Use	Used to represent any data which is logically a table of values, such as blood pressure, most protocols, many blood tests etc.	
MisUse	Not used for time-based data, which should be represented with the temporal class `HISTORY`.. The table may be empty.	
CEN	n/a	
GEHR	`G1_TABLE_PROPOSITION, G1_MATRIX_PROPOSITION`	
HL7v3	RIM structured types Table_structure, Table_cell, Table etc ; CDA 1.0 Table Entry type.	
Inherit	`ITEM_STRUCTURE`	
Attributes	**Signature**	**Meaning**
0..1	**rows**: `List<CLUSTER>`	Physical representation of the table as a list of `CLUSTER`s, each containing the data of one row of the table.
Functions	**Signature**	**Meaning**

CLASS	ITEM_TABLE	
1..1	**row_count**: `Integer`	Return the number of rows
1..1	**column_count**: `Integer`	Return the number of columns
1..1	**row_names**: `List<DV_TEXT>`	Return the row names
1..1	**column_names**: `List<DV_TEXT>`	Return the column names
	ith_row(i:`Integer`**)**: `CLUSTER` ***require*** i >0 ***and*** i < row_count	Return the i-th row
	has_row_with_name(a_key: `String`**)**: `Boolean` ***require*** a_key /= Void **and then not** a_key.empty	True if there is a row whose first column has the name ‘a_key’
	has_column_with_name(a_key: `String`): `Boolean` ***require*** a_key /= Void **and then not** a_key.empty	True if there is a column with name ‘a_key’
	named_row(a_key: `String`): `CLUSTER` ***require*** has_row_with_name(a_key)	Return the row whose first column has the name ‘a_key’
	has_row_with_key (keys: `Set<String>`): `Boolean`	True if there is a row whose first n columns have the names in ‘keys’
	row_with_key (keys: `Set<String>`): `CLUSTER` ***require*** has_row_with_key(keys)	Return the row whose first n columns have names equal to the values in ‘keys’
	element_at_cell_ij (i, j:`Integer`): `ELEMENT` ***require*** i >= 1 **and** i <= column_count j >= 1 **and** j <= row_count	Return the element at the column i, row j.
	element_at_named_cell (row_key, col_key:`String`): `ELEMENT` ***require*** has_row_with_name(row_key) has_column_with_name(column_key)	Return the element at the row whose first column has the name ‘row_key’ and column has the name ‘col_key’

CLASS	ITEM_TABLE	
	as_hierarchy: CLUSTER	Generate a CEN EN13606-compatible hierarchy consisting of a single CLUSTER containing the CLUSTERs representing the columns of this table.
Invariants	***Valid_structure***: rows.forall(items.forall(instance_of("ELEMENT"))) ***Column_names_valid***: column_names /= Void	

4.3.5 ITEM_TREE Class

CLASS	ITEM_TREE	
Purpose	Logical tree data structure. The tree may be empty.	
Use	Used to represent data which are logically a tree such as audiology results, microbiology results, biochemistry results.	
MisUse		
CEN	The CEN cluster is effectively the only data structure available in CEN, and is equivalent to the ITEM_TREE type.	
GEHR	G1_TREE_PROPOSITION	
HL7v3	This can be constructed with CDA 1.0 Lists. Act and Act_relationship are the closest correspondents in the HL7v3 RIM.	
Inherit	ITEM_STRUCTURE	
Attributes	**Signature**	**Meaning**
0..1	**items**: LIST<*ITEM*>	Physical representation of the tree.
Functions	**Signature**	**Meaning**
	has_element_path(a_path:String): Boolean	True if path 'a_path' is a valid leaf path
	element_at_path(a_path:String): ELEMENT ***require*** has_element_path(a_path)	Return the leaf element at the path 'a_path'
	as_hierarchy: CLUSTER	Generate a CEN EN13606-compatible hierarchy, which is the same as the tree's physical representation.
Invariants		

4.4 Instance Structures

4.4.1 ITEM_SINGLE Instance Structure

FIGURE 3 illustrates a `ITEM_SINGLE` instance, in both physical and logical forms.

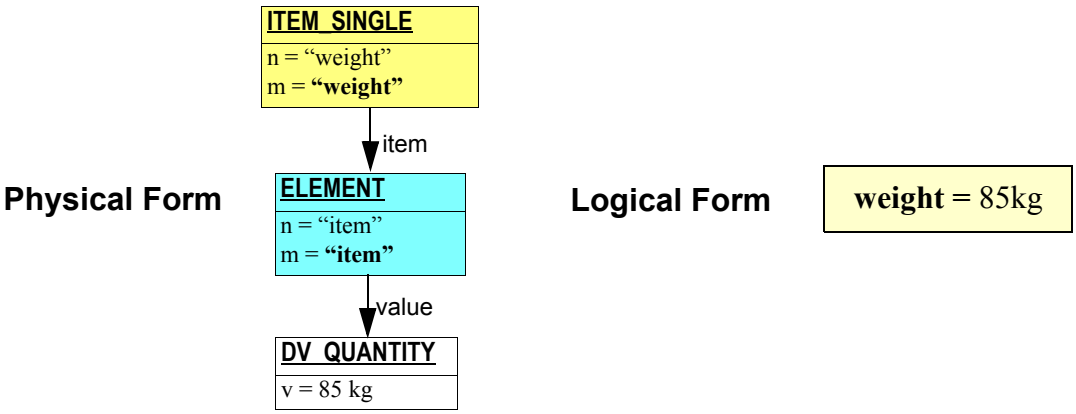


FIGURE 3 Instance Structure of ITEM_SINGLE

4.4.2 ITEM_LIST Instance Structure

FIGURE 4 illustrates a typical `ITEM_LIST` structure, in this case for a BP protocol.

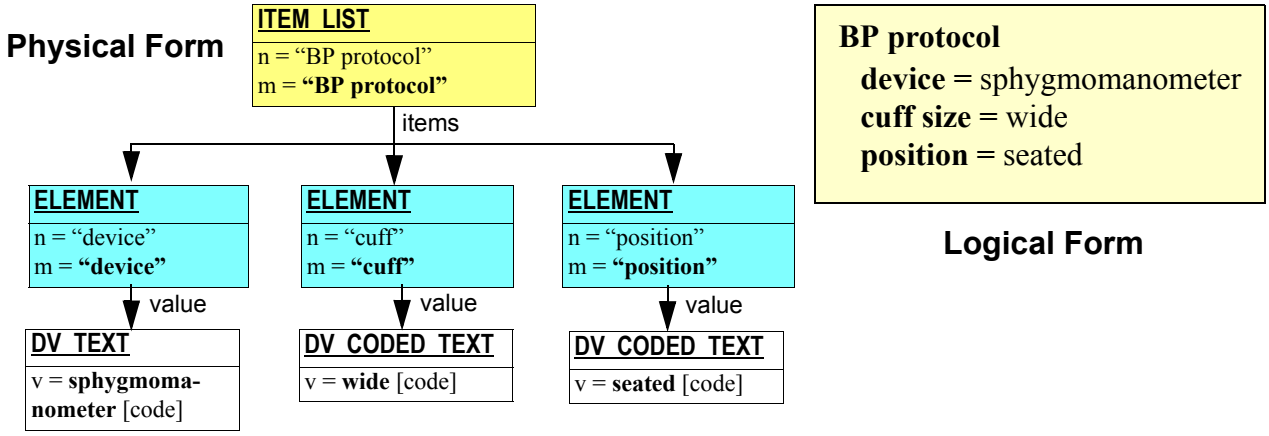
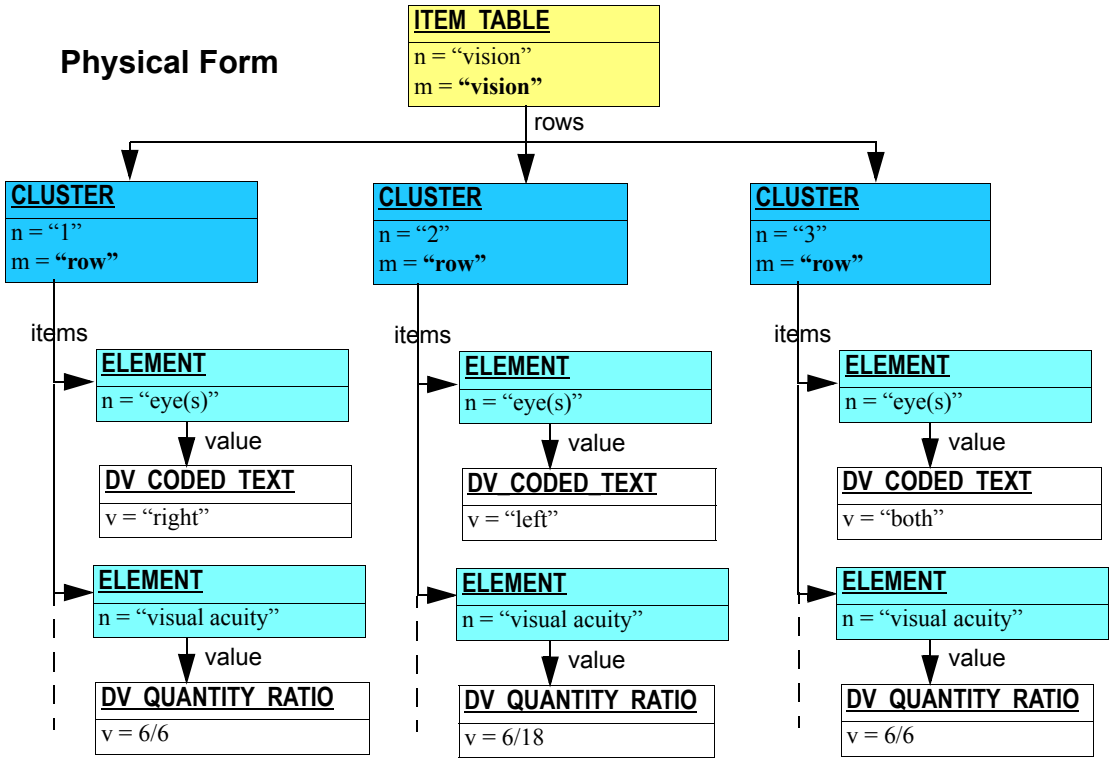


FIGURE 4 ITEM_LIST Instance Structure

4.4.3 ITEM_TABLE Instance Structure

FIGURE 5 illustrates a table of visual acuity test results.



Logical Form

Vision	
Eye(s)	**visual acuity**
Right	6/6
Left	6/18
Both	6/6

FIGURE 5 ITEM_TABLE Instance Structure

4.4.4 ITEM_TREE Instance Structure

FIGURE 6 illustrates te logical and physical form of an example `ITEM_TREE` instance, representing a biochemistry result.

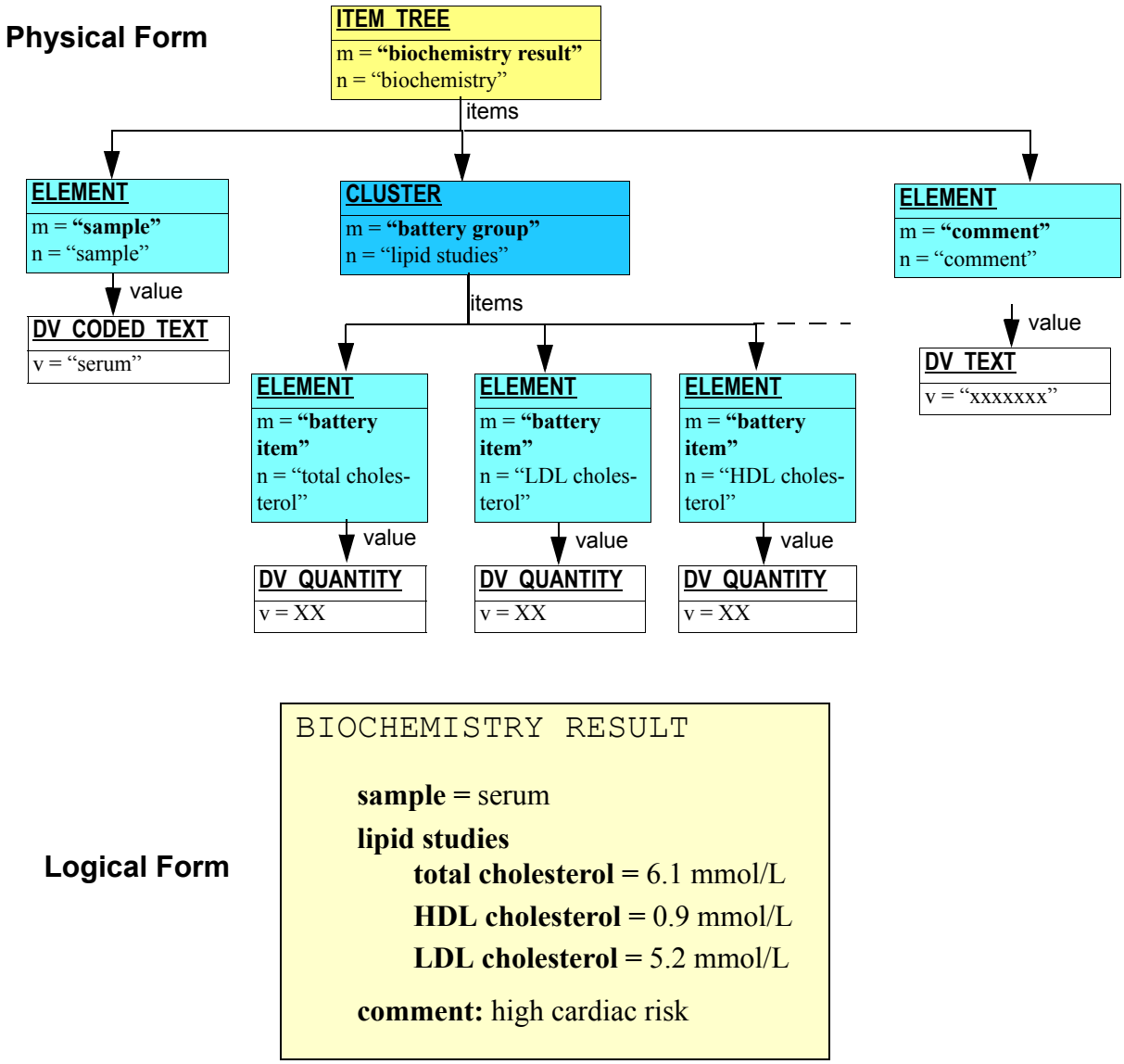


FIGURE 6 ITEM_TREE Instance Structure

5 Representation Package

5.1 Overview

This package contains classes for a simple hierarchical representation of any data structure, as shown in FIGURE 7. These classes are compatible with the CEN EN13606 classes of the same names, and instances can be losslessly generated to and from EN13606 instances structures.

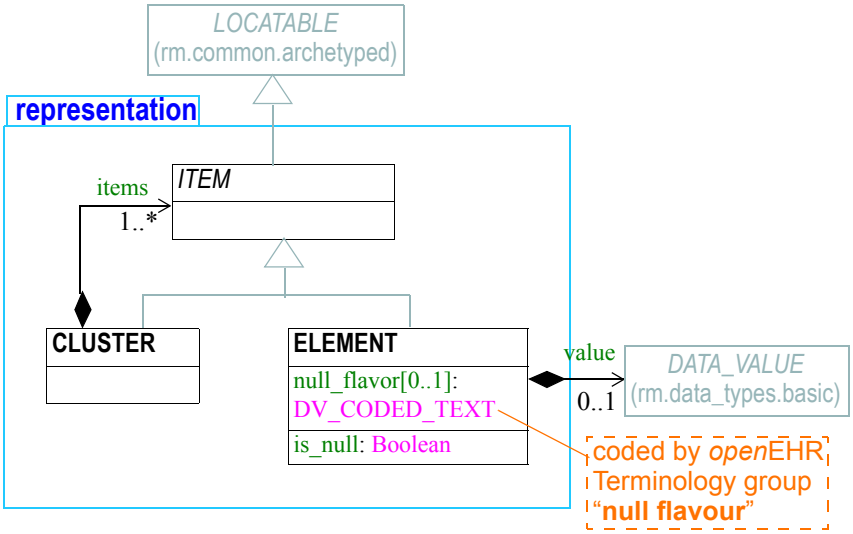


FIGURE 7 rm.data_structures.representation Package

5.2 Class Descriptions

5.2.1 ITEM Class

CLASS	*ITEM (abstract)*	
Purpose	The abstract parent of `CLUSTER` and `ELEMENT` representation classes.	
CEN	`ITEM` class	
OMG HDTF	COAS::Observation	
Synapses	Item class	
GEHR	`G1_HIERARCHICAL_ITEM`	
HL7v3	n/a	
Inherit	`LOCATABLE`	
Attributes	**Signature**	**Meaning**

5.2.2 CLUSTER Class

CLASS	CLUSTER	
Purpose	The grouping variant of `ITEM`, which may contain further instances of `ITEM`, in an ordered list.	
CEN	(ENV 13606:2000) ClusterOCC class; (prEN 13606:2006)	
OMG HDTF	COAS::CompositeObservation	
Synapses	Compound class	
GEHR	`G1_HIERARCHICAL_GROUP`	
HL7v3	Act_context	
Inherit	`ITEM`	
Attributes	**Signature**	**Meaning**
	items: `List<ITEM>`	Ordered list of items - `CLUSTER` or `ELEMENT` objects - under this `CLUSTER`.
Invariants	***Items_non_empty***: items /= Void ***and then not*** items.empty	

5.2.3 ELEMENT Class

CLASS	ELEMENT	
Purpose	The leaf variant of `ITEM`, to which a `DATA_VALUE` instance is attached.	
CEN	DataItem class	
OMG HDTF	COAS::AtomicObservation	
Synapses	Element class	
GEHR	`G1_HIERARCHICAL_VALUE`	
HL7v3	Act	
Inherit	`ITEM`	
Attributes	**Signature**	**Meaning**
0..1 (cond)	**value**: `DATA_VALUE`	data value of this leaf
0..1 (cond)	**null_flavor**: `DV_CODED_TEXT`	flavour of null value, e.g. indeterminate, not asked etc
Functions	**Signature**	**Meaning**
1..1	**is_null**: `Boolean`	True if value logically not known, e.g. if indeterminate, not asked etc.
Invariants	***Is_null_valid***: is_null = (value = Void) ***Null_flavor_indicated***: is_null **xor** null_flavour = Void ***Null_flavour_valid***: is_null **implies** terminology(Terminology_id_openehr).has_code_for_group_id (Group_id_null_flavour, null_flavor.defining_code)	

6 History Package

6.1 Overview

The `history` package defines classes which formalise the concept of past, linear time, via which historical data of any structural complexity can be recorded. It supports both instantaneous and interval event samples within periodic and aperiodic series. Data recorded in interval events are tagged with a mathematical function, including 'point-in-time', 'mean', 'delta' and so on, as defined by the *open*EHR "event math function" vocabulary. It also supports the inclusion of "summary" data, i.e. a textual or image item which summarises in some way the entire history.

Regardless of whether the actual data consist of a single sample or many, they are represented in the same way: as a history of events, i.e. as a time series, allowing all software to access data in a uniform way, whether it is a single measurement of weight, a long series of three- or four-dimensional images, or even a series of encapsulated multimedia items.

The model defines the constrained generic (otherwise known as 'template' or 'parameterised') types `HISTORY<T>`, `EVENT<T>`, `POINT_EVENT<T>`, and `INTERVAL_EVENT<T>` where the type parameter is constrained to the `ITEM_STRUCTURE` type, and defines the type of the data recorded in an instance of `HISTORY`. The effect is that repeated instances of spatially complex data can recur in time, corresponding to the way data are actually measured. An aperiodic series of `POINT_EVENT` instances would typically be used to represent manual measurements repeated in time. Periodic histories of `INTERVAL_EVENT` instances would typically be used to represent vital signs monitor output (which is usually delivered in averaged form potentially with additional minimum and maximum values).

As with all other parts of the *open*EHR reference model, the History package is designed for archetyping; archetypes define the domain semantics of `HISTORY` instances. The `history` package is shown in FIGURE 8.

6.1.1 Semantics

Basic Semantics

The intention of the History model is to represent time-based data for which every sample in the series is a measurement of the same phenomenon (e.g. patient heartrate) and is obtained using the same measurement method (e.g. pulse oximeter). Samples taken in this way can be reliably treated as representing changes in a phenomenon over time, and accordingly can be safely used for time-based computation, such as graphing, statistical analysis and so on. A History can contain any mixture of `POINT_EVENT` and `INTERVAL_EVENT` instances. Clearly it is impossible for the model to guarantee completely correct usage on its own, however there two major safeguards.

Firstly, the use of generic types forces the type of the data in each Event to be the same. A History of type `HISTORY<ITEM_LIST>` therefore constrains the type of the data at each Event (`EVENT`.*item*) to be of type `ITEM_LIST` and nothing else.

Secondly, the use of archetyping (typically within *open*EHR Observation archetypes) ensures the actual structure of the `ITEM_STRUCTURE` subtype is defined in the same way for every sample - e.g. a two-item list representing systolic and diastolic blood pressure.

Timing

An instance of the `HISTORY` class contains the *origin*: `DV_DATE_TIME` attribute, indicating what is considered the '0-point' of the time series, and a series of instances of the `EVENT` subtype, each containing a *time*: `DV_DATE_TIME` attribute representing the absolute time of the event. The relative off-

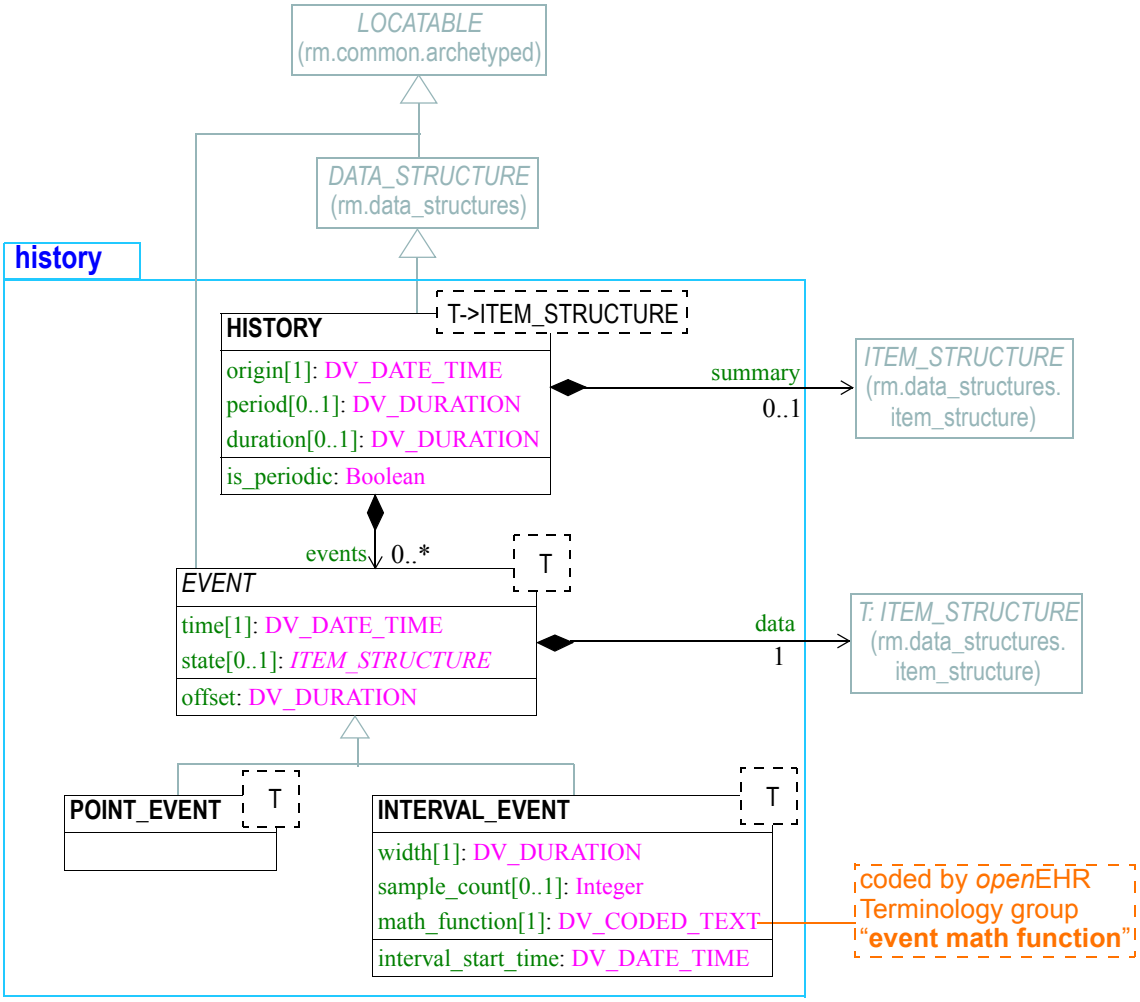


FIGURE 8 rm.data_structures.history Package

set of any Event is computed as the difference between the EVENT.*time* and HISTORY.*origin* by the EVENT.*offset* function. For Interval events (i.e. instances of INTERVAL_EVENT), the *time* attribute always refers to the *end time* of the event, since this is the time at which the data (e.g. average) are true.

The origin time of a History does not have to be the time of the first sample - it might be the time of an event such as child-birth with respect to which the samples are recorded, e.g. Apgar[1] scores at 1 and 3 minute offsets. Periodicity and aperiodicity are expressed via the *is_periodic* and *period* attributes. For a *periodic* time-series, *period* is set to the period duration value and *is_periodic* returns True. The total duration of the History is given by the HISTORY.*duration* function. FIGURE 9 illustrates a number of variations in History periodicity and Event type.

Point Events

The simplest kind of Event in a History is a "point" event, expressed by instances of the class POINT_EVENT, representing an instantaneous value. A History instance many be composed solely of Point events, as would be the case with a number of blood pressure values measured over time as the patient changes position. An Apgar result is a typical example of aperiodic point data, typically con-

1. A 0-10 score indicating the health of a newborn based on breathing, heartrate, colour, muscle-tone and reflexes

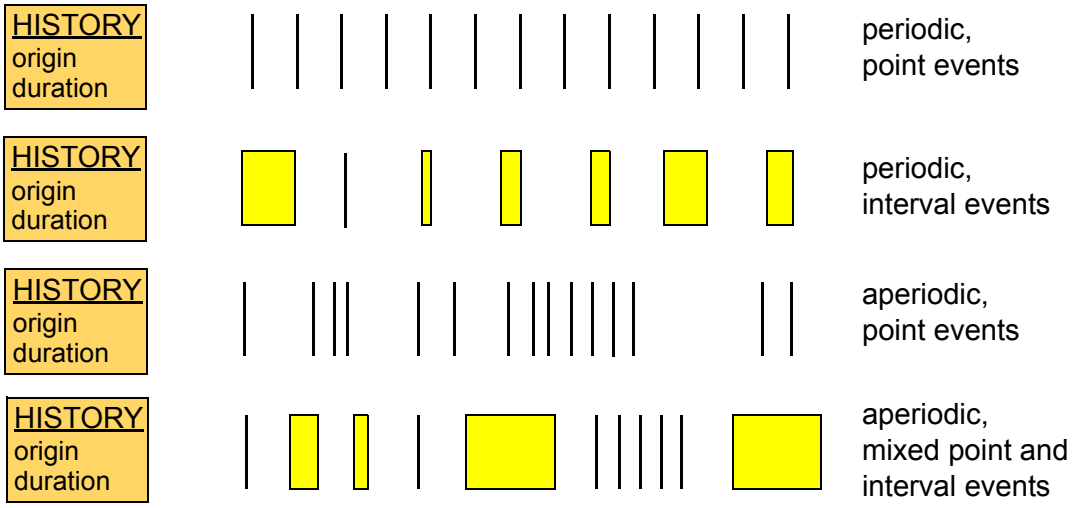


FIGURE 9 Variation in History periodicity and Event width

sisting of 2 or 3 events, each containing 5 values and a 6th value repersenting the Apgar score for that time point. Point data may also be available from monitoring devices. For fine-granularity (e.g. 1 second) data, the number of samples may be too voluminous for the health record, and more efficient recording in the form of summary Interval events (see below) might be desired. FIGURE 10 illustrates the structure of a `HISTORY` containing `POINT_EVENTs`.

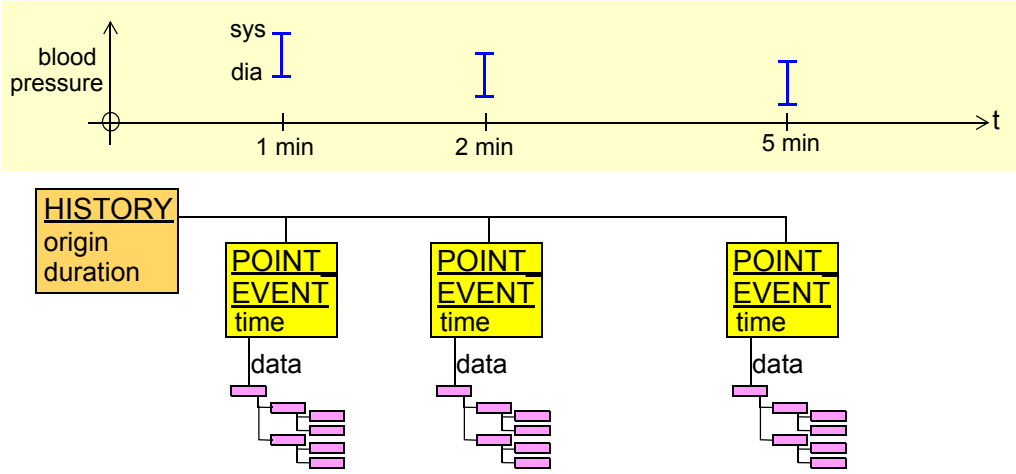


FIGURE 10 Structure of HISTORY<T> of POINT_EVENTs

Interval Events

Instances of the `INTERVAL_EVENT` class are used to express values corresponding to an interval in time. The `INTERVAL_EVENT`.*width* attribute defines the duration of the interval; and the inherited *time* value corresponds to the trailing edge of the event.

The meaning of an Interval event in this model is that the values effectively summarise actual instantaneous values of a datum that have occurred during the period of the event interval. The mathematical meaning of the data of any particular interval event is given by the *math_function* attribute. This is coded by the *open*EHR Terminology group "event math function", and takes values such as "minimum", "maximum", "average", "delta" and so on. The particular math functions used in each Interval event in a History may vary throughout the History; for example, one 4-hour Interval event might contain data representing average values, while a following event might contain data representing

maximum values. Such data can be conveniently used for generating sophisticated graphs of the underlying datum over time. FIGURE 11 illustrates a History containing 2 pairs of 4-hour blood pressure Interval events, with each pair containing maximum and mean blood pressure value structures for +4h and +8h timepoints (each of which consist of a systolic and diastolic value).

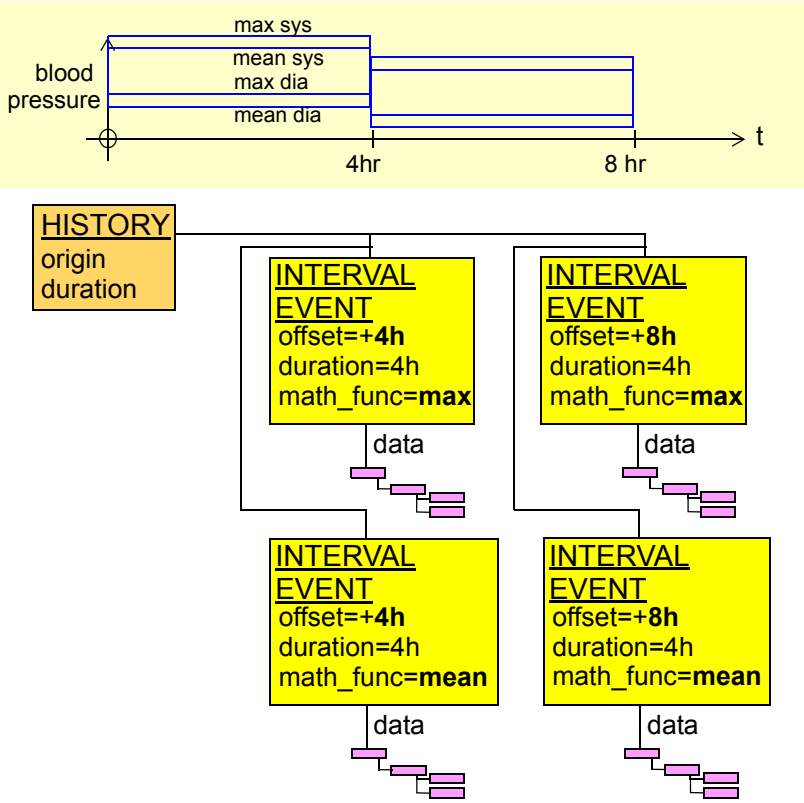


FIGURE 11 Structure of HISTORY<T> of INTERVAL_EVENTs

Interval events can overlap other interval or point events within the same History. A common situation where this occurs is with measurement of different periods of vital signs, such as 4-hourly blood pressure events, overlapped by a 24-hour event which contains the values over a period of 6 x 4 hour periods. In general a long Interval event can overlap any combination of Point or Interval events, as shown in FIGURE 12.

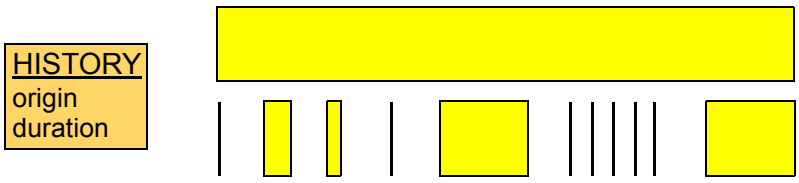


FIGURE 12 Overlapping Events

Change Data

One subcategory of interval data that deserves mention is change data. There are three event math function terms used for indicating changes in data values as follows:

- "change": this means that the value recorded is the difference between the value now and the value some time previously. It can be positive or negative;

- “increase”: the value recorded for the change is positive. The name (i.e. ELEMENT.*name*) chosen for the item in an archetype carries the semantic of positivity e.g. "increase of ....; rise of....; ....gain" etc;
- “decrease”: the value recorded for the change is positive. But the name chosen for the item carries the semantic of negativity e.g. "decrease of ....; fall of ....; .... loss".

The following examples show how the data and these math functions are coordinated.

- weight last week was 76 kg. Wait this week = 74 kg. Possible instances:

Item Name in Archetype	Value stored	Type	Math Function
“weight change”	+ 2kg	DV_QUANTITY	“change”
“weight loss”	(+)2kg	DV_QUANTITY	“decrease”
“weight loss”	True	Boolean	“decrease”

- weight last week was 80 kg. Weight this week = 83 kg. Possible instances:

Item Name in Archetype	Value stored	Type	Math Function
“weight change”	(+)3kg	DV_QUANTITY	“change”
“weight in-crease”	(+)3kg	DV_QUANTITY	“increase”
“weight gain”	True	Boolean	“decrease”

The use of these math function indicators allows the correct representation of change values, no matter how they were captured, in a computable form.

Summary Event Data

A relatively common situation particularly in laboratory testing is the existence of a “summarising” event which accompanies more detailed historical data. Examples where this arises include:[1]

- a series of exams with a single radiologist report for all of them (the report might include one or more key images);
- graphical summary of a dynamic challenge test such as Glucose tolerance test;
- some comment about the pattern of values on a set of observed values in series.

Such data are accommodated within the model via the optional HISTORY.*summary* attribute, which is itself a structure, archetypable separately from the structure of the main data. In the first example above, the summary data might consist of an ITEM_SINGLE object containing a textual report; in the second, an ITEM_SINGLE object containing a image within a DV_MULTIMEDIA instance.

Efficient Representation of Fine-grained Device Data

A useful practical consequence of the of Interval Events is that it allows long periods of relatively stable data to be represented with a single Interval event, while interesting perturbations will be represented with a number of fine-grained Interval or Point Events. In the example in FIGURE 13 5 Event instances are used represent 4 hours of data consisting to 14,400 x 1 second samples from a blood pressure monitor. The optional INTERVAL_EVENT.*sample_count* attribute can be used to record the number of original samples summarised in the event. In the illustration, the *math_function* is shown as mean; clearly in the first long period, the monitored datum was not absolutely flat. The implication

1. Examples provided by G Grieve of Kestral Computing, Australia

is that the recording software was configured to regard variations in a small band (e.g. 5mm Hg) as insignificant, and only to create new Event objects when the underlying values moved outside the band. Another approach woould have been to create two Interval Event objects for each long period, one giving minimum value, the other maxium value, still based on the principle of generating one such pair for periods when the underlying data remained within specified limits. Regardless of the details, this general approach provides a way to include hours of fine-grained data from devices like vital signs monitors in very little space; the data simply need to be transfomed into equivalent *open*EHR History form first.

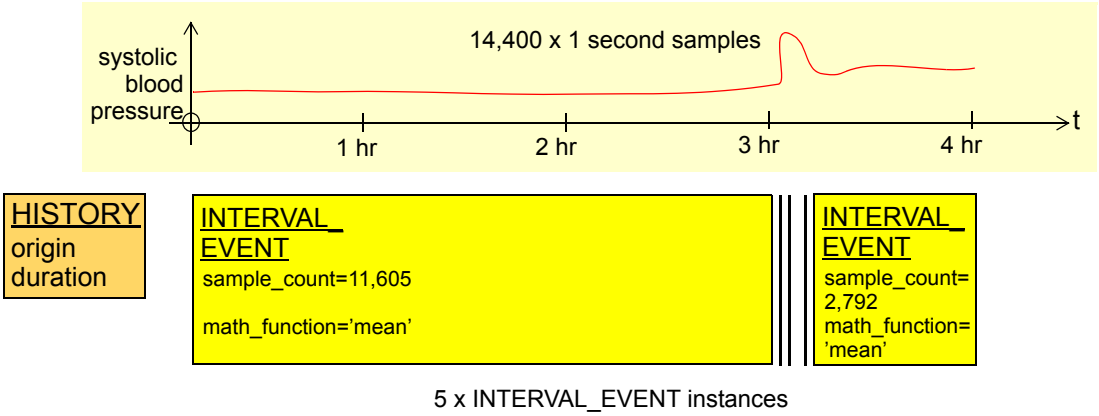


FIGURE 13 Data compression effect of History with Interval Events

State

A feature particular to a model of recording historical data for scientific and clinical use is the ability to record 'state'. In *open*EHR, 'state' is understood as information pertinent to the correct interpretation of the primary data. A simple example is where the primary datum is heartrate; useful state data would be the level of exertion of the subject (resting, after 3 minutes cycling etc). In clinical recording situations, the state data is often crucial to the safe use of the primary data, since the latter might be normal or abnormal depending on the patient state. In *open*EHR there are two ways of recording state. One is via the use of a separate `HISTORY` structure within the `OBSERVATION` class (see `ehr.composition.content.entry` package). The other is via the use of the *state* attribute of type `ITEM_STRUCTURE` defined in the class `EVENT` itself. Experience with *open*EHR archetypes and systems has shown that the latter method corresponds to the most common clinical need, which is to be able to record the state at the time of the event (the other method allows for the recording of independent state events). A simple example is the recording of 3 glucose levels during a glucose tolerance test. The state information for each event is, respectively (in a typical test):

- 0-minute sample: "post 8-hour fast";
- 1-hour sample: "post 75g oral glucose challenge";
- 2-hour sample: "post 75g oral glucose challenge".

The History structure for this example is illustrated in FIGURE 14.

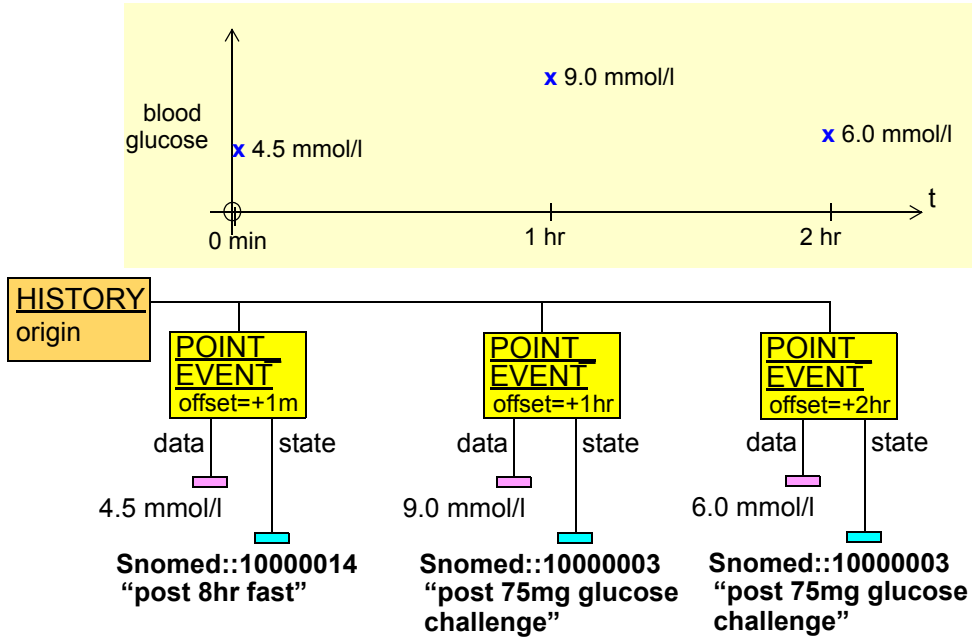


FIGURE 14 State in HISTORY

6.2 Class Descriptions

6.2.1 HISTORY<T: ITEM_STRUCTURE> Class

CLASS	HISTORY<T: ITEM_STRUCTURE>	
Purpose	Root object of a linear history, i.e. time series structure. For a periodic series of events, *period* will be set, and the time of each Event in the History must correspond; i.e. the `EVENT`.*offset* must be a multiple of *period* for each Event. Missing events in a period History are however allowed.	
CEN	Time was encoded as part of the `Item` structure.	
GEHR	Time was encoded as part of the `G1_HIERARCHICAL_PROPOSITION` structure.	
HL7v3	The data type `HIST<T>` is equivalent in intention to `HISTORY<T>` for histories of primitive values; for histories of complex data, Act/Act-relationship structures have to be used.	
Inherit	`DATA_STRUCTURE`	
Attributes	**Signature**	**Meaning**
1..1	**origin**: `DV_DATE_TIME`	Time origin of this event history. The first event is not necessarily at the origin point.
0..1	**events**: `List <EVENT<T>>`	The events in the series.

CLASS	HISTORY<T: ITEM_STRUCTURE>	
0..1	**period** : DV_DURATION	Period between samples in this segment if periodic.
0..1	**duration** : DV_DURATION	Duration of the entire History; either corresponds to the duration of all the events, and/or the duration represented by the summary, if it exists.
0..1	**summary** : ITEM_STRUCTURE	Optional summary data expressing e.g. text or image which summarises entire History.
Functions	**Signature**	**Meaning**
1..1	**is_periodic** : Boolean	Indicates whether history is periodic.
	as_hierarchy: CLUSTER	Generate a CEN EN13606-compatible hierarchy of the physical representation.
Invariants	***origin_exists***: origin /= Void ***events_exists***: (events /= Void **and then not** events.is_empty) ***or*** summary /= Void ***periodic_validity***: is_periodic **xor** period = Void ***period_consistency***: is_periodic **implies** events.for_all (e: EVENT | e.offset.to_seconds.mod(period.to_seconds) = 0)	

6.2.2 EVENT <T: ITEM_STRUCTURE> Class

CLASS	*EVENT <T: ITEM_STRUCTURE> (abstract)*	
Purpose	Defines the abstract notion of a single event in a series. This class is generic, allowing types to be generated which are locked to particular spatial types, such as EVENT<ITEM_LIST>. Subtypes express point or intveral data.	
HL7v3	The data type HistoryItem HXIT<T> is close to EVENT<T> in its intent.	
Inherit	LOCATABLE	
Attributes	**Signature**	**Meaning**
1..1	**time** : DV_DATE_TIME	Time of this event. If the width is non-zero, it is the time point of the trailing edge of the event.
1..1	**data** : T	The data of this event.
0..1	**state** : ITEM_STRUCTURE	Optional state data for this event.
Functions	**Signature**	**Meaning**

CLASS	*EVENT <T: ITEM_STRUCTURE> (abstract)*	
(redefined)	**parent**: HISTORY<T>	Redefinition of LOCATABLE.*parent* to be of type HISTORY.
1..1	**offset**: DV_DURATION	Offset of this event from origin, computed as *time*.diff(*parent.origin*)
Invariants	***Time_exists***: time /= Void ***Data_exists***: data /= Void ***Offset_validity***: offset /= Void **and then** offset = time.diff(parent.origin)	

6.2.3 POINT_EVENT <T: ITEM_STRUCTURE> Class

CLASS	POINT_EVENT <T: ITEM_STRUCTURE>	
Purpose	Defines a single point event in a series.	
Inherit	EVENT	
Attributes	**Signature**	**Meaning**
Invariants		

6.2.4 INTERVAL_EVENT <T: ITEM_STRUCTURE> Class

CLASS	INTERVAL_EVENT <T: ITEM_STRUCTURE>	
Purpose	Defines a single interval event in a series.	
Inherit	EVENT	
Attributes	**Signature**	**Meaning**
1..1	**width**: DV_DURATION	Length of the interval during which the state was true. Void if an instantaneous event.
1..1	**math_function**: DV_CODED_TEXT	Mathematical function of the data of this event, e.g. “maximum”, “mean” etc. Coded using *open*EHR Terminology group “event math function”.
0..1	**sample_count**: Integer	Optional count of original samples to which this event corresponds.
Functions	**Signature**	**Meaning**
1..1	**interval_start_time**: DV_DATE_TIME	Start time of the interval of this event.

CLASS	INTERVAL_EVENT <T: ITEM_STRUCTURE>
Invariants	***Width_valid***: width /= Void ***Math_function_validity***: terminology(Terminology_id_openehr).has_code_for_group_id(Group_id_event_math_function, math_function.defining_code) ***Interval_start_time_valid***: interval_start_time /= Void **and** interval_start_time = time - width

6.3 History Instance Structures

6.3.1 Single Sample

FIGURE 15 illustrates a single weight measurement. The Event objects contain the timing information, which in this case is simply the time of measurement (the origin).

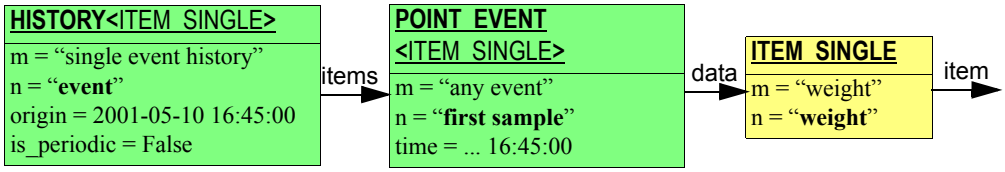


FIGURE 15 Single sample Instance Structure

6.3.2 5-minute Blood Pressure Averages

FIGURE 16 illustrates two Interval events representing 5 minute blood pressure averages, the first at 5 minutes' offset from an initial event and lasting 5 minutes, the second 5 minutes later.

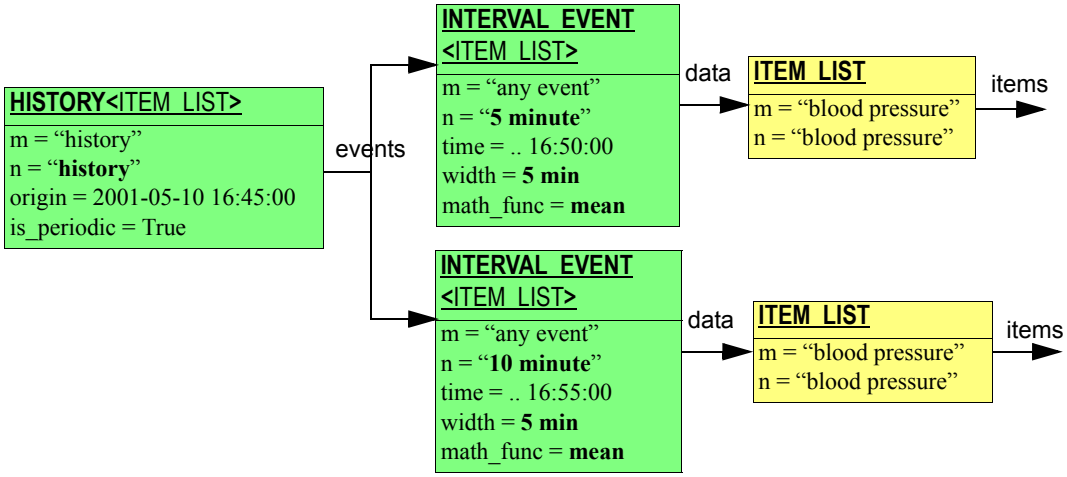


FIGURE 16 Blood Pressure Series History Instance Structure

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.deepthought.com.au/openEHR.

3 Schloeffel P. (Editor). Requirements for an Electronic Health Record Reference Architecture. International Standards Organisation, Australia; Feb 2002; ISO TC 215/SC N; ISO/WD 18308.

A.2 European Projects

4 Dixon R., Grubb P.A., Lloyd D., and Kalra D. *Consolidated List of Requirements. EHCR Support Action Deliverable 1.4*. European Commission DGXIII, Brussels; May 200159pp Available from http://www.chime.ucl.ac.uk/HealthI/EHCR-SupA/del1-4v1_3.PDF.

5 Dixon R, Grubb P, Lloyd D. *EHCR Support Action Deliverable 3.5: "Final Recommendations to CEN for future work"*. Oct 2000. Available at http://www.chime.ucl.ac.uk/HealthI/EHCR-SupA/documents.htm.

6 Dixon R, Grubb P, Lloyd D. *EHCR Support Action Deliverable 2.4 "Guidelines on Interpretation and implementation of CEN EHCRA"*. Oct 2000. Available at http://www.chime.ucl.ac.uk/HealthI/EHCR-SupA/documents.htm.

7 Ingram D. *The Good European Health Record Project*. Laires, Laderia Christensen, Eds. Health in the New Communications Age. Amsterdam: IOS Press; 1995; pp. 66-74.

8 *Deliverable 19,20,24: GEHR Architecture.* GEHR Project 30/6/1995

A.3 CEN

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

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

A.4 OMG

11 CORBAmed document: *Person Identification Service*. (March 1999). (Authors?)

12 CORBAmed document: *Lexicon Query Service*. (March 1999). (Authors?)

A.5 Software Engineering

13 Meyer B. *Object-oriented Software Construction*, 2nd Ed.
Prentice Hall 1997

14 Fowler M. *Analysis Patterns: Reusable Object Models.* Addison Wesley 1997

15 Fowler M, Scott K. *UML Distilled (2nd Ed.).* Addison Wesley Longman 2000

A.6 Resources

16 IANA - http://www.iana.org/.

END OF DOCUMENT