openEHR
Release 1.0 draft

REFERENCE MODEL

The *open*EHR Data Structures Information Model

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

Revision: 1.6rc2

Pages: 37

1. Ocean Informatics Australia

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

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

Copyright Notice

Amendment Record

Issue	Details	Who	Completed
1.6rc2	**CR-000014**. Adjust History. Major simplifcation to package; make Events absolute in time. **CR-000183**. Remove root node from ITEM_TREE.	S Heard T Beale S Heard	05 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			
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.

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 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.

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

Blue text indicates sections under active development.

NOTE THAT NOT ALL CHANGES MADE TO THIS DOCUMENT HAVE BEEN APPROVED BY THE OPENEHR ARB, AND MAY BE SUBJECT TO FURTHER CHANGE.

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.

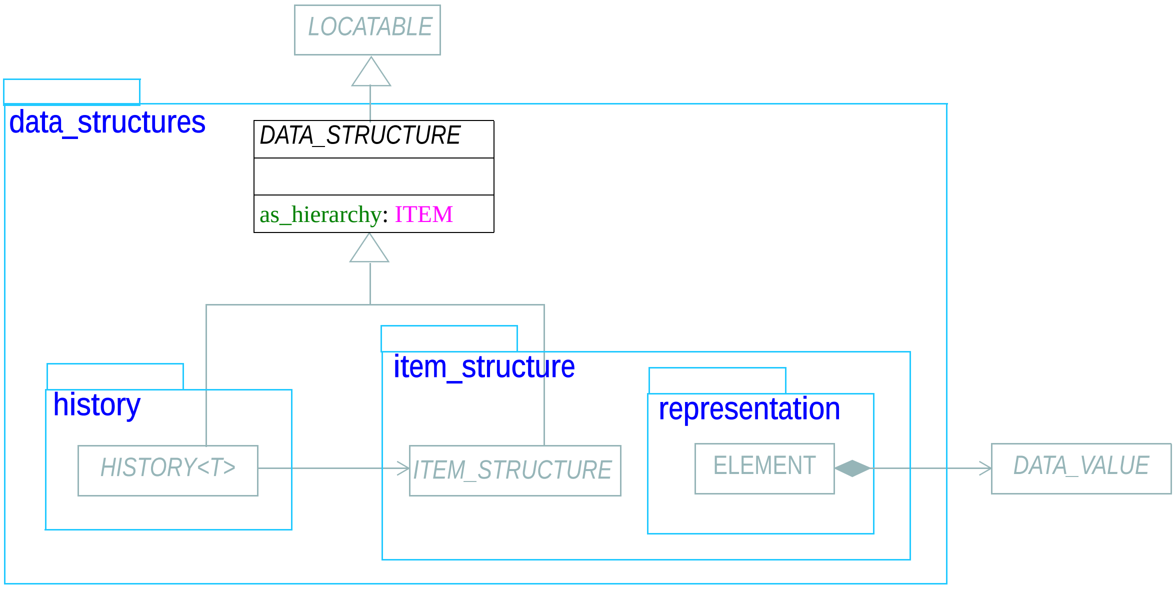


FIGURE 1 rm.data_structures Package

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. The design principle RM-spatial [2] and preceding discussion describe the reasons behind this approach.

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.

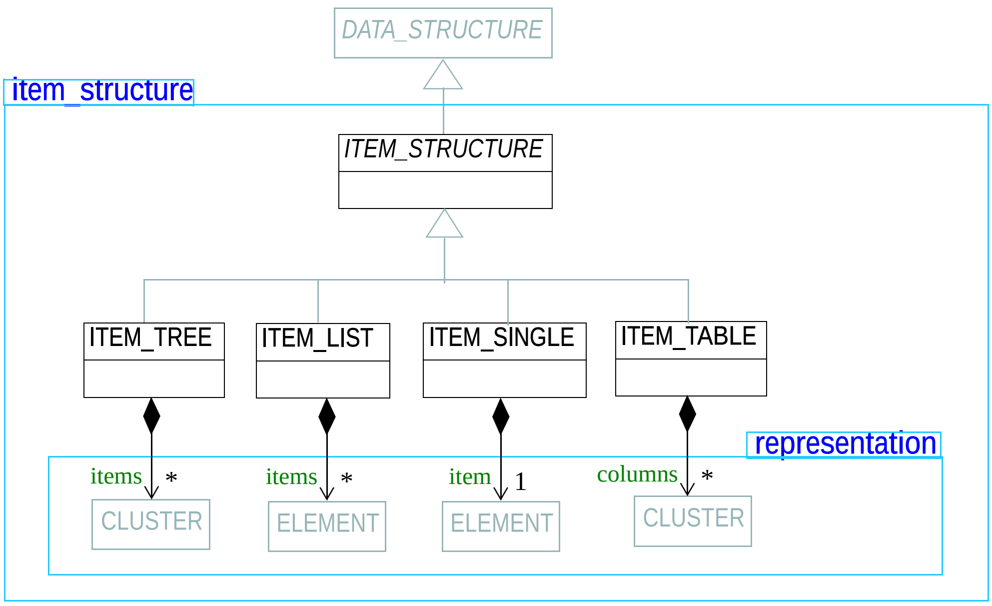


FIGURE 2 rm.data_structures.item_structure Package

4.2 Class Descriptions

4.2.1 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		

4.2.2 ITEM_SINGLE Class

CLASS	ITEM_SINGLE	
Purpose	Logical single value data structure.	
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
	item: ELEMENT	
Functions	Signature	Meaning
	as_hierarchy: ELEMENT	Generate a CEN EN13606-compatible hierarchy consisting of a single ELEMENT.
Invariants		

4.2.2.1 ITEM_SINGLE Paths

In the following path structure, the name of the ITEM_SINGLE object acts as the root-name.

- *the item*: "item" e.g. item Xpath equivalent: item

4.2.2.2 ITEM_SINGLE Instance Structure

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

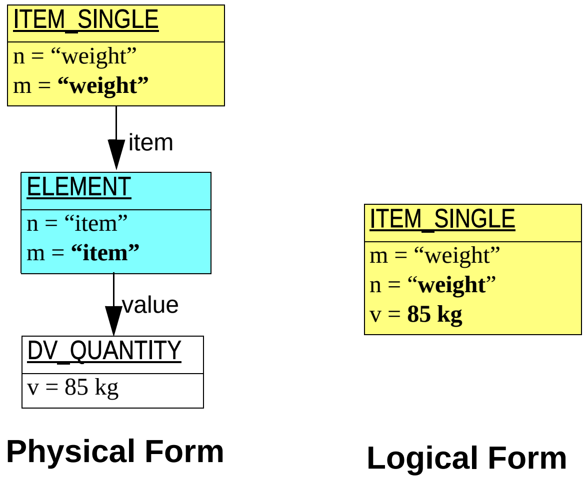


FIGURE 3 Instance Structure of ITEM_SINGLE

4.2.3 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.	
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**
	items: `List<ELEMENT>`	Physical representation of the list.
Functions	**Signature**	**Meaning**
	item_count: `Integer`	Count of all items
	names: `List<DV_TEXT>`	Retrieve the names of all items
	named_item(a_name:`String`**)**: `ELEMENT`	Retrieve the item with name 'a_name'

CLASS	ITEM_LIST	
	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")	

4.2.3.1 ITEM_LIST Paths

In the following path structure for Lists, the *name* attribute of the `ITEM_LIST` object acts as the root-name.

- *nth list item*: `"items[" <n> "]"` e.g. `items[2]`
  Xpath equivalent: `items[2]`
- *named list item*: `"items[" <item_name> "]"` e.g. `items[cuff]`
  Xpath equivalent: `items[meaning() = 'cuff']`

4.2.3.2 ITEM_LIST Instance Structure

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

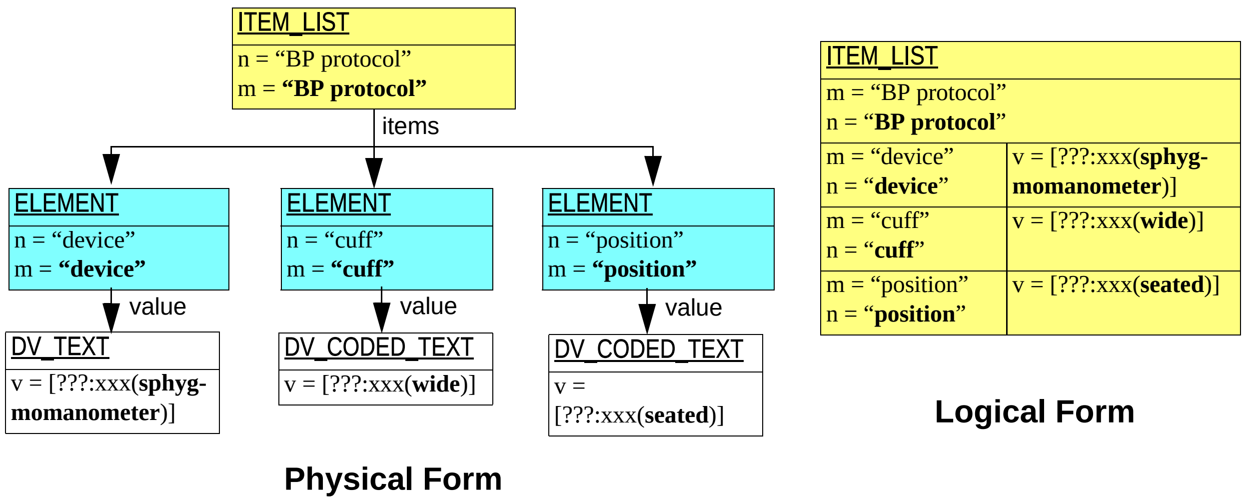


FIGURE 4 ITEM_LIST Instance Structure

4.2.4 ITEM_TABLE Class

CLASS	ITEM_TABLE
Purpose	Logical table data structure, in which columns are named and ordered. 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.

CLASS	**ITEM_TABLE**	
MisUse	Not used for time-based data, which should be represented with the temporal class `HISTORY`.	
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**
	columns: `List<CLUSTER>`	Physical representation of the table as a list of `CLUSTER`s, each containing the data of one column of the table.
Functions	**Signature**	**Meaning**
	row_count:`Integer`	Return the number of rows
	column_count:`Integer`	Return the number of columns
	row_names: `List<DV_TEXT>`	Return the row names
	column_names: `List<DV_TEXT>`	Return the column names
	ith_row(i:`Integer`**)**: `List<ELEMENT>` ***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`): `List<ELEMENT>` ***require*** has_row_with_name(a_key)	Return the row whose first column has the name ‘a_key’

CLASS	ITEM_TABLE	
	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(key_vals:Set<String>): List<ELEMENT> ***require*** has_row_with_key(key_vals)	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’
	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***: items.forall({ITEM}.type = “CLUSTER” ***and then*** {ITEM}.items.forall({ITEM}.type = “ELEMENT”))	

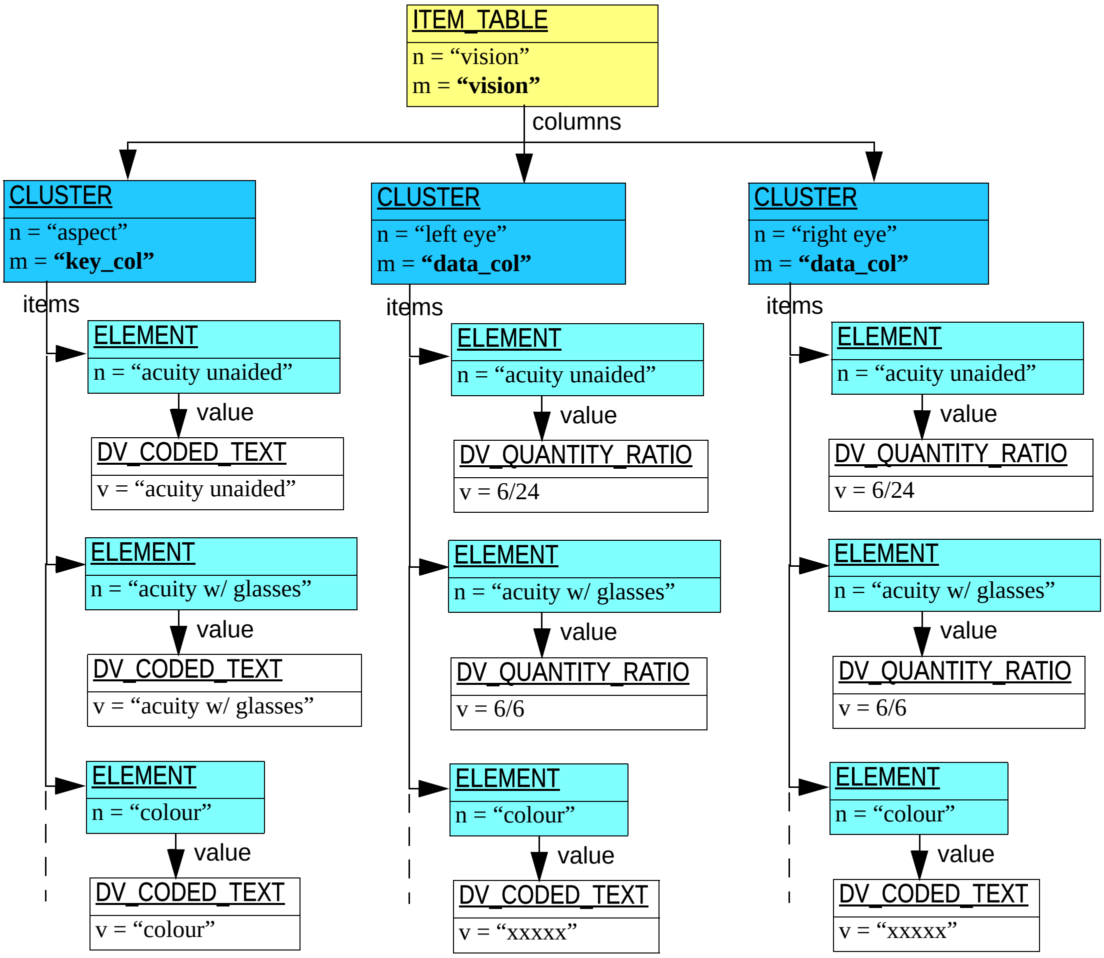
4.2.4.1 ITEM_TABLE Paths

The following path patterns are legal for tables.

- *column*: `“columns[” <column-name> “]”` e.g. `columns[left eye]`
  Xpath equivalent: `columns[meaning() = ‘left eye’]`
- *row*: `“columns/items[” <row-name> “]”` e.g. `columns/items[colour]`
  Xpath equivalent: `columns/items[meaning() = ‘colour’]`
- *cell*: `“columns[” <column-name> “]/items[” <row-name> “]”`
  e.g. `columns[left eye]/items[acuity w\/ glasses]`
  Xpath equivalent: `columns[meaning() = ‘left eye’]/items[meaning() = ‘acuity w/ glasses’]`

4.2.4.2 ITEM_TABLE Instance Structure

FIGURE 5 illustrates a table of visual acuity test results.



Physical Form

ITEM_TABLE		
m = "vision" n = "**vision**"		
m = "key_col" n = "**aspect**"	m = "data_col" n = [???:xxx(**left eye**)]	m = "data_col" n = [???:xxx(**right eye**)]
m = "acuity unaided" n = "**acuity unaided**"	6/24	6/24
m = "acuity with glasses" n = "**acuity with glasses**"	6/6	6/6
m = "colour" n = "**colour**"	normal	normal

Logical Form

FIGURE 5 ITEM_TABLE Instance Structure

4.2.4.3 CEN EN13606 Hierarchy Encoding Rules

The `ITEM_TABLE` encoding rules are as follows:

- A `CLUSTER` is required as the parent of all columns.

- Each column is represented by a CLUSTER, whose name value is the name of the column.
- Each row item in a given column is represented by an ELEMENT under the relevant column CLUSTER.
- The name of each ELEMENT object is the name of its row.

4.2.5 ITEM_TREE Class

CLASS	ITEM_TREE	
Purpose	Logical tree data structure.	
Use	Used to represent data which are logically a tree such as audiology results, micro-biology 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**
	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		

4.2.5.1 ITEM_TREE Paths

Tree paths are of the following form.

- *node*:“items[<node-id>], e.g. items[at0012] or “items[” <node-name>...<node-name>, e.g. items[lipid studies]

- *leaf value*: `"items[" <node-name>...<node-name> "/items[" <leaf-name> "]"`
  e.g. `items[lipid studies]/items[LDL cholesterol]`

4.2.5.2 ITEM_TREE Instance Structure

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

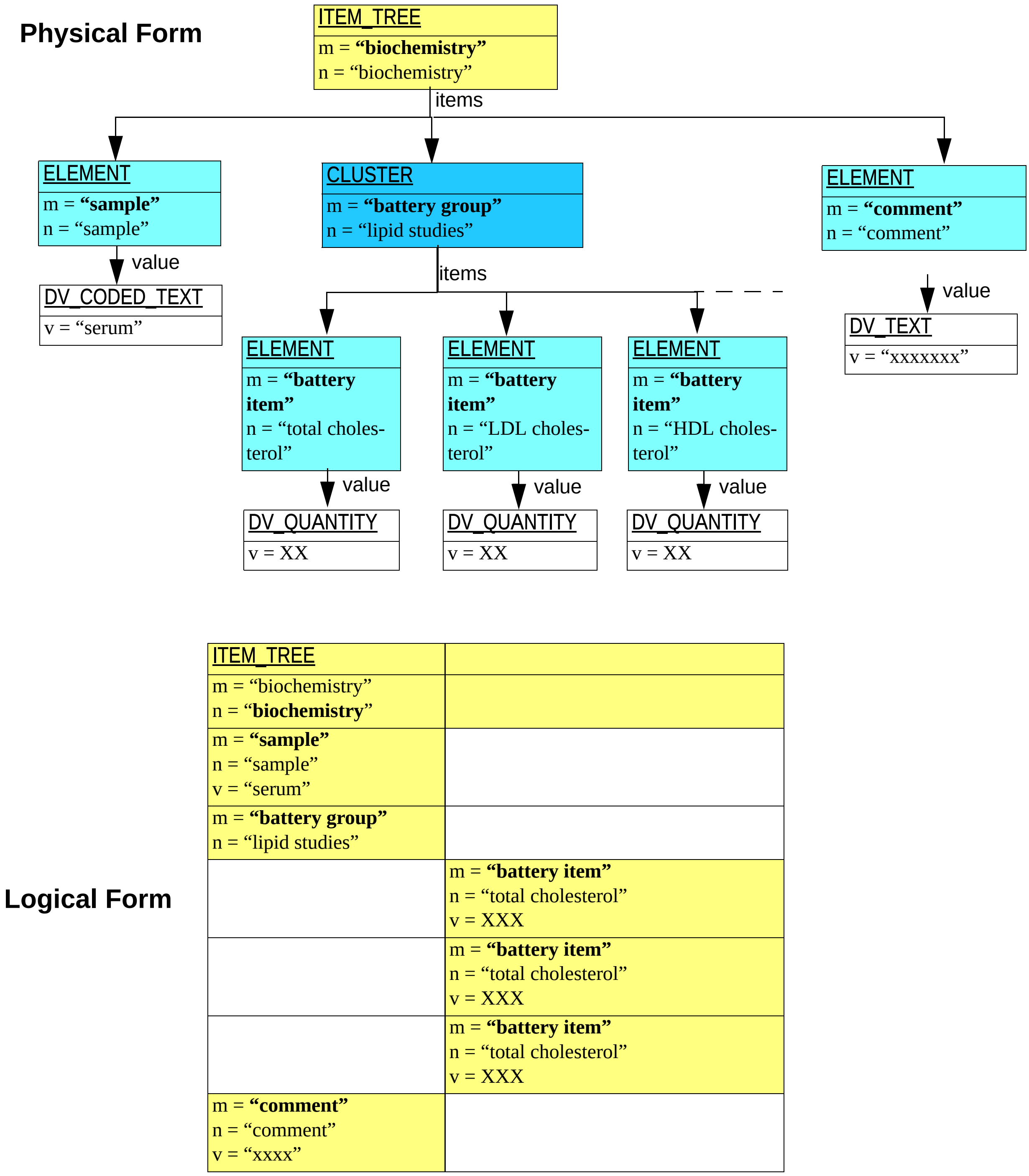


FIGURE 6 ITEM_TREE Instance Structure

4.2.5.3 CEN EN13606 Hierarchy Encoding Rules

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

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.

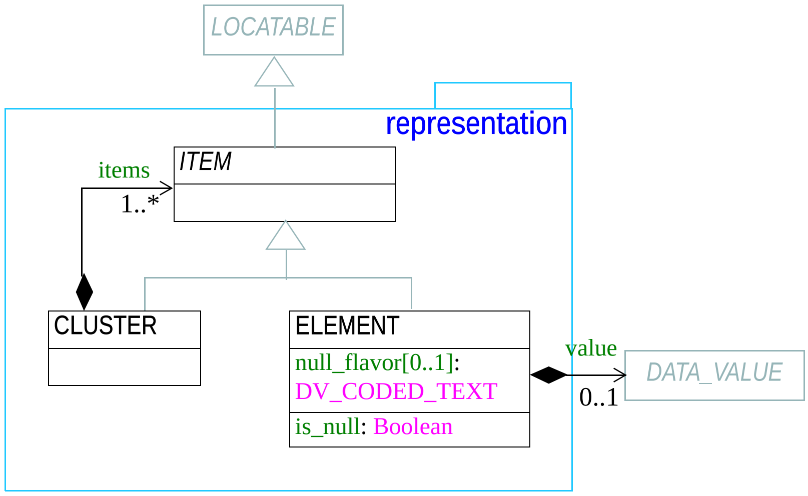


FIGURE 7 rm.data_structures.representation Package

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	ClusterOCC class	
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**
	value: `DATA_VALUE`	data value of this leaf
	null_flavor: `DV_CODED_TEXT`	flavour of null value, e.g. indeterminate, not asked etc
Functions	**Signature**	**Meaning**
	is_null: `Boolean`	True if value logically not known, e.g. if indeterminate, not asked etc.
Invariants	***Null_flavor_indicated***: is_null ***xor*** null_flavour /= Void ***Null_flavour_valid***: is_null ***implies*** terminology(“openehr”).codes_for_group_name(“null flavour”, “en”).has(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 continuous event samples within periodic and aperiodic series. It also allows for the recorded data at each even to be tagged with a mathematical function, including 'point-in-time', 'mean', 'delta' and so on, as defined by the *open*EHR Event Math Function vocabulary.

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, regardless of 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>` and `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. A periodic discrete series would be used to represent most vital signs monitor output, but also manual measurements repeated in time.

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.

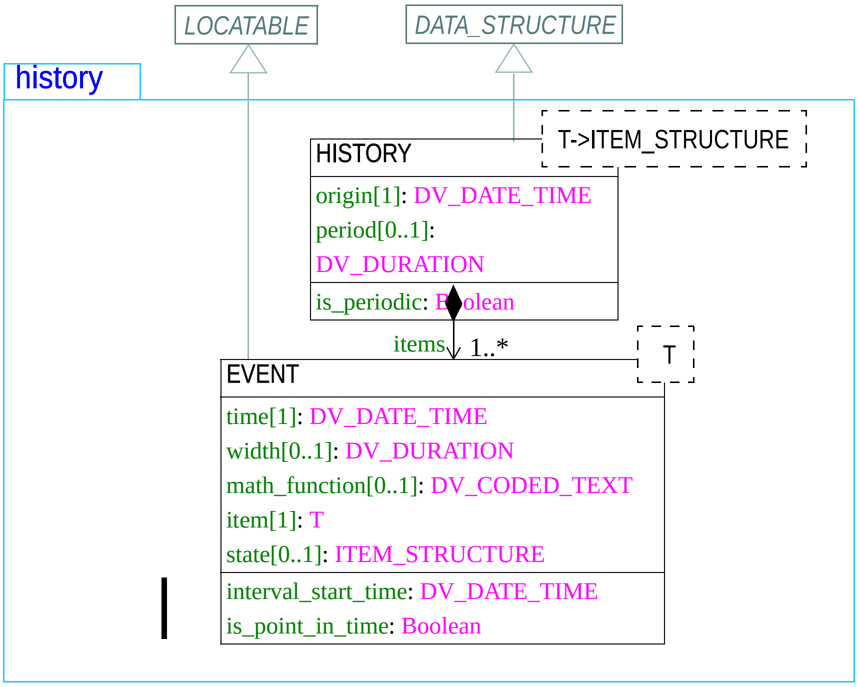


FIGURE 8 rm.data_structures.history Package

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. 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 real 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 (e.g. a two-item list representing systolic and diastolic blood pressure) is defined in the same way for every sample.

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 `EVENT` instances each containing a *time*: `DV_DATE_TIME` attribute representing the absolute time of the event. The relative offset of any Event can be computed as the difference between the `EVENT`.*time* and `HISTORY`.*origin*. Note that 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.

FIGURE 9 illustrates pictorially the general structure of `HISTORY` objects, and their relationship to the timeline.

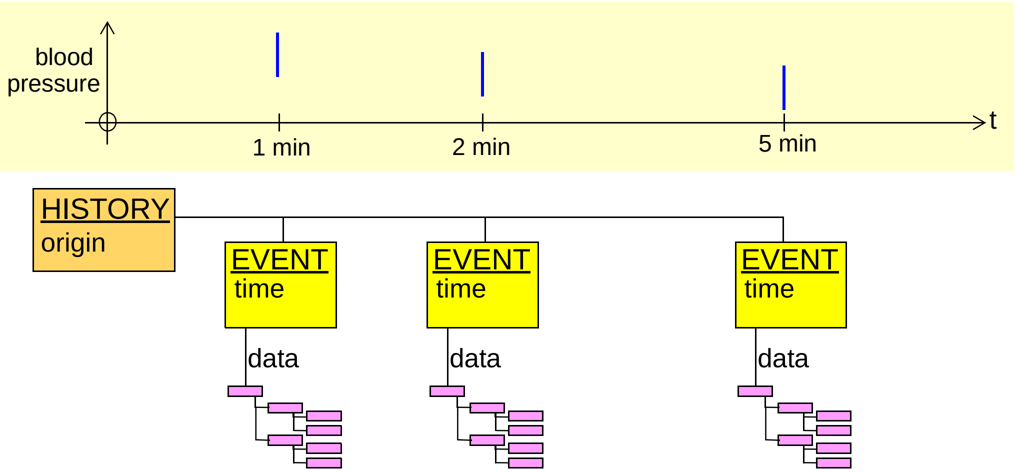


FIGURE 9 Basic Structure of HISTORY<T>

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

Events are not limited to instants in time - they may correspond to an interval in time. If the event is instantaneous, the attribute EVENT.*is_point_in_time* is True; if not, it returns False, and the EVENT.*width* attribute is set to the duration of the interval; in this case, the *time* value corresponds to the trailing edge of the event. A single HISTORY may have a mixture of instantaneous and interval events. FIGURE 10 illustrates a number of variations in History periodicity and Event type.

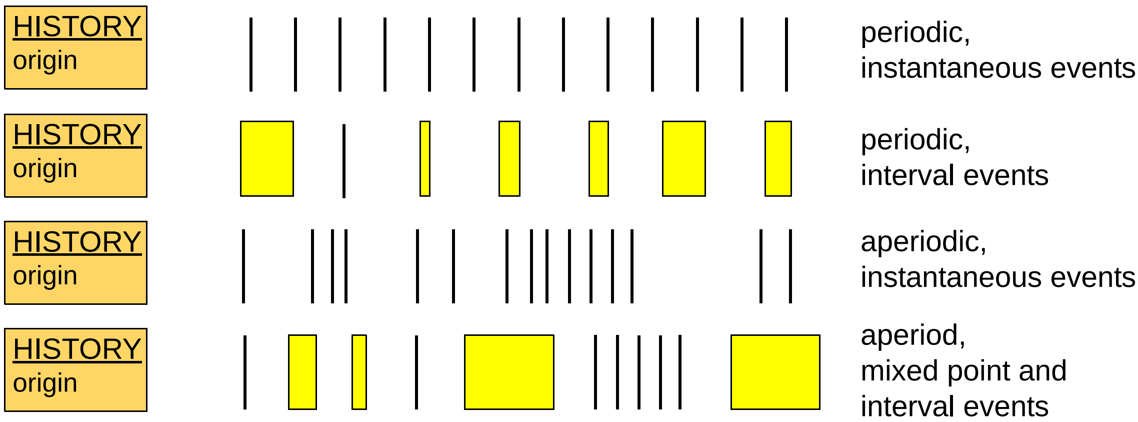


FIGURE 10 Variation in History periodicity and Event width

Interval Events and Math Functions

The meaning of an Interval event in this model is that the value effectively summarises the actual instantaneous values that have occurred during the period of the event interval. It may be that the value was absolutely steady during the period, or that it fluctuated. The interpretation of the event data is indicated in the *math_function* attribute. This expresses the mathematical function of the value (which itself is in general a complex structure) with respect to the actual instantaneous values which existed in the real world, and may have been sampled at a fine rate to generate the interval event. Function types include "actual", "minimum", "maximum", "mean", "mode" and so on. The full set of possibilities is coded by the *open*EHR Terminology group "Event Math Functions". The value of the *math_function* attribute does not have to be the same for all Events in a History, for example a history of maxima and minima. Such data can be conveniently used for generating sophisticated graphs of the underlying datum over time.

Efficient Representation of Fine-grained Device Data

A practical consequence of the of interval Events is that it allows long periods of unchanging data to be represented with a single Event, while interesting perturbations will be represented with a number of fine-grained Events, as shown in FIGURE 11, where only 5 Event instances represent 4 hours of data. This 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.

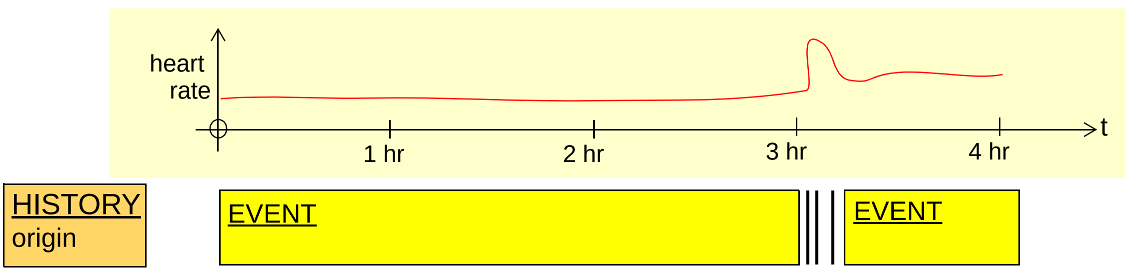


FIGURE 11 Data compression effect of History with Interval Events

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 exersion of the subject (resting, after 3 minutes cycling etc). In clinical recording sitatoins, 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 12.

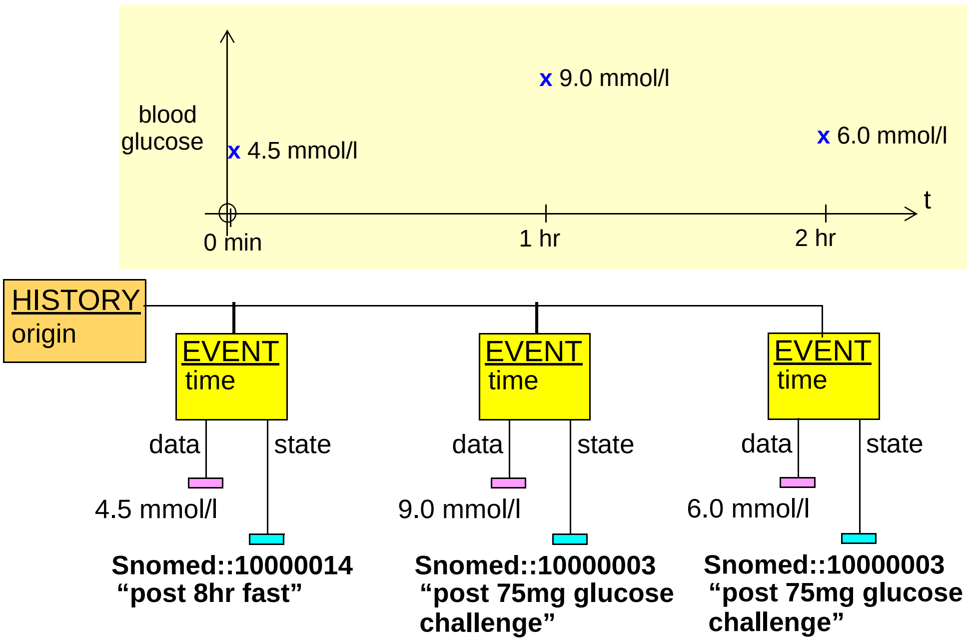


FIGURE 12 State in HISTORY

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..

CLASS	HISTORY<T: ITEM_STRUCTURE>	
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
	origin: `DV_DATE_TIME`	Time origin of this event history. The first event is not necessarily at the origin point.
	items: `List <EVENT<T>>`	The items in the series.
	period: `DV_DURATION`	period between samples in this segment if periodic
Functions	Signature	Meaning
	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 ***items_exists***: items /= Void **and then not** items.empty ***periodic_validity***: is_periodic ***xor*** period = Void	

6.2.2 EVENT <T: ITEM_STRUCTURE> Class

CLASS	EVENT <T: ITEM_STRUCTURE>	
Purpose	Defines 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>`. In cases where samples are missing, there will correspondingly be missing `EVENT` instances. Every `EVENT` instance must have a data item.	
HL7v3	The data type HistoryItem `HXIT<T>` is close to `EVENT<T>` in its intent.	
Inherit	`LOCATABLE`	
Attributes	Signature	Meaning
	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.

CLASS	EVENT <T: ITEM_STRUCTURE>	
	width: `DV_DURATION`	Length of the interval during which the state was true. Void if an instantaneous event.
	math_function: `DV_CODED_TEXT`	Mathematical function for non-instantaneous events - e.g. "maximum", "mean" etc. Coded using *open*EHR Terminology group "event math function".
	item: `T`	The data of this event.
Functions	**Signature**	**Meaning**
	interval_start_time: `DV_DATE_TIME`	Start time of this event. If the width is zero, returns the value of *time*, otherwise it is the time point of the leading edge of the event.
	is_point_in_time: `Boolean`	True if the width is zero.
Invariants	***time_exists***: time /= Void ***time_validity***: ***not*** is_point_in_time ***implies*** (interval_start_time + duration = time) ***item_exists***: item /= Void ***math_function_validity***: width /= Void ***implies*** (math_function /= Void ***and then*** terminology("openehr").codes_for_group_name("event math function", "en").has(math_function.defining_code))	

6.3 History Paths

Paths within History objects include the following possibilities:

- *origin*: `"items" EVENT.name`, e.g. `origin`
- *event*: `"items" EVENT.name`, e.g. `items[event_3]`

Typical paths which refer to a particular item within the spatial data of an event series:

- 16th sample on lead 3 of an ECG: `items[sample 16]/data/items[lead_3]`
- 5 minute sample of apgar breathing datum of a newborn: `events[5 minute]/data/items[heart rate]`

6.4 History Instance Structures

All data corresponding to events in historical time are represented as histories which ultimately contain one or more instances of a spatial data structure representing a particular instance of clinical data. A history consists of segments of time. Each segment is either a periodic discrete series - a series of time points - or a continuous time section of a certain duration. For each timepoint in a discrete series, there is an instance of the data structure, which might be a list, table, tree or other structure; for each continuous segment, there is one instance, representing the state of something which was true during the duration of the segment.

6.4.1 Single Sample

FIGURE 13 illustrates a weight measurement in instance form. The event history objects contain the timing information, which in this case is simply the time of measurement (the origin).

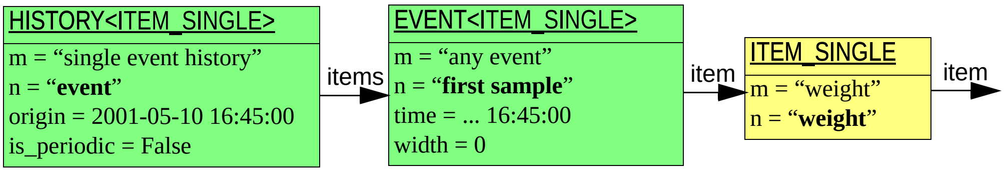


FIGURE 13 Single sample Instance Structure

6.4.2 5-minute Blood Pressure Averages

FIGURE 14 illustrates two time segments representing episodes of chest pain, the first at 5 minutes' offset from an initial event and lasting 5 minutes, the second 15 minutes later, and lasting 15 minutes.

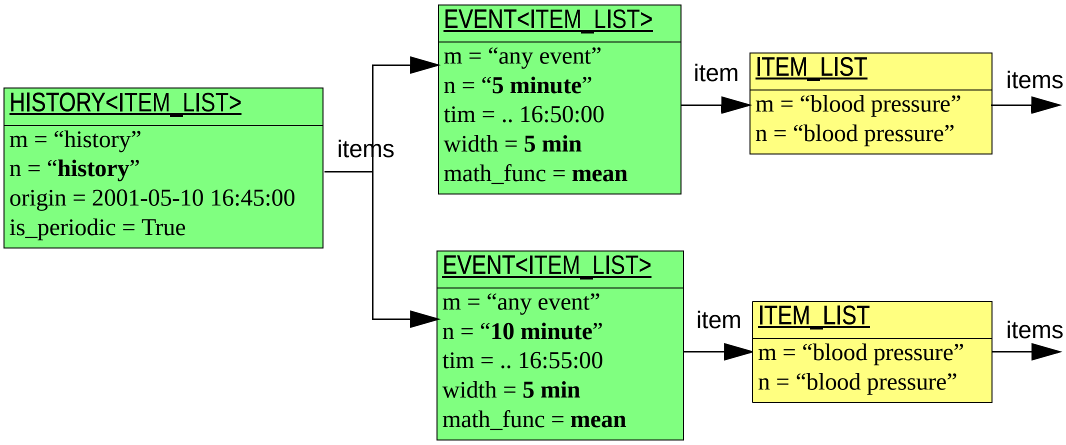


FIGURE 14 Blood Pressure Series History Instance Structure

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

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