



REFERENCE MODEL

The *openEHR* Data Structures Information Model

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

Revision: 1.6rc2

Pages: 39

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

© 2003-2005 The *openEHR* Foundation

The *openEHR* foundation

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

Founding Chairman

David Ingram, Professor of Health Informatics, CHIME, University College London

Founding Members

Dr P Schloeffel, Dr S Heard, Dr D Kalra, D Lloyd, T Beale

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

Copyright Notice

© Copyright openEHR Foundation 2001 - 2005
All Rights Reserved

1. This document is protected by copyright and/or database right throughout the world and is owned by the openEHR Foundation.
2. You may read and print the document for private, non-commercial use.
3. You may use this document (in whole or in part) for the purposes of making presentations and education, so long as such purposes are non-commercial and are designed to comment on, further the goals of, or inform third parties about, openEHR.
4. You must not alter, modify, add to or delete anything from the document you use (except as is permitted in paragraphs 2 and 3 above).
5. You shall, in any use of this document, include an acknowledgement in the form: "© Copyright openEHR Foundation 2001-2005. All rights reserved. www.openEHR.org"
6. This document is being provided as a service to the academic community and on a non-commercial basis. Accordingly, to the fullest extent permitted under applicable law, the openEHR Foundation accepts no liability and offers no warranties in relation to the materials and documentation and their content.
7. If you wish to commercialise, license, sell, distribute, use or otherwise copy the materials and documents on this site other than as provided for in paragraphs 1 to 6 above, you must comply with the terms and conditions of the openEHR Free Commercial Use Licence, or enter into a separate written agreement with openEHR Foundation covering such activities. The terms and conditions of the openEHR Free Commercial Use Licence can be found at http://www.openehr.org/free_commercial_use.htm

Amendment Record

Issue	Details	Who	Completed
1.6rc2	CR-000014. Adjust History. Major simplification to package; make Events absolute in time. CR-000155: Summary data. CR-000183. Remove root node from ITEM_TREE. CR-000185. Improved EVENT model	S Heard T Beale S Heard G Grieve S Heard S Heard G Grieve	15 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 <i>meaning</i> to STRING and rename as <i>archetype_node_id</i> . 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 <i>ITEM.displayed</i> . 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 <i>EVENT.duration</i> and <i>SINGLE_EVENT.duration</i> to <i>width</i> . 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 <i>HISTORY.is_periodic</i> 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.....	7
1.1	Purpose.....	7
1.2	Related Documents	7
1.3	Status.....	7
1.4	Peer review	7
1.5	Conformance.....	8
2	Background.....	9
2.1	Requirements	9
2.2	Design Principles	9
3	Overview	11
3.1	Instance Structures	11
3.2	Class Descriptions.....	11
3.2.1	DATA_STRUCTURE Class	12
4	Item Structure Package	13
4.1	Overview	13
4.2	Path Semantics	14
4.2.1	ITEM_SINGLE Paths	14
4.2.2	ITEM_LIST Paths.....	14
4.2.3	ITEM_TABLE Paths	14
4.2.4	ITEM_TREE Paths	14
4.3	CEN EN13606 Encoding Rules.....	14
4.3.1	ITEM_TABLE	14
4.3.2	ITEM_TREE.....	15
4.4	Class Descriptions.....	15
4.4.1	ITEM_STRUCTURE Class.....	15
4.4.2	ITEM_SINGLE Class.....	15
4.4.3	ITEM_LIST Class.....	16
4.4.4	ITEM_TABLE Class	16
4.4.5	ITEM_TREE Class	18
4.5	Instance Structures	19
4.5.1	ITEM_SINGLE Instance Structure.....	19
4.5.2	ITEM_LIST Instance Structure	19
4.5.3	ITEM_TABLE Instance Structure	21
4.5.4	ITEM_TREE Instance Structure.....	22
5	Representation Package	23
5.1	Overview	23
5.2	Class Descriptions.....	23
5.2.1	ITEM Class	23
5.2.2	CLUSTER Class	24
5.2.3	ELEMENT Class	25
6	History Package.....	27
6.1	Overview	27
6.1.1	Semantics	27
6.2	Path Semantics	32

6.3.1	HISTORY<T: ITEM_STRUCTURE> Class	32
6.3.2	EVENT <T: ITEM_STRUCTURE> Class	33
6.3.3	POINT_EVENT <T: ITEM_STRUCTURE> Class.....	34
6.3.4	INTERVAL_EVENT <T: ITEM_STRUCTURE> Class	34
6.4	History Instance Structures.....	35
6.4.1	Single Sample.....	35
6.4.2	5-minute Blood Pressure Averages	35
A	References	37
A.1	General	37
A.2	European Projects	37
A.3	CEN	37
A.4	OMG.....	37
A.5	Software Engineering	37
A.6	Resources.....	38

1 Introduction

1.1 Purpose

This document describes the common data structures used in *openEHR* 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 *openEHR*;
- Academic groups using *openEHR*;
- The open source healthcare community;
- Medical informaticians and clinicians interested in health information;
- Health data managers.

1.2 Related Documents

Prerequisite documents for reading this document include:

- The *openEHR* Modelling Guide
- The *openEHR* Support Information Model
- The *openEHR* 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 *openEHR* Reference Model specification is determined by a formal test of that artifact against the relevant *openEHR* 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.

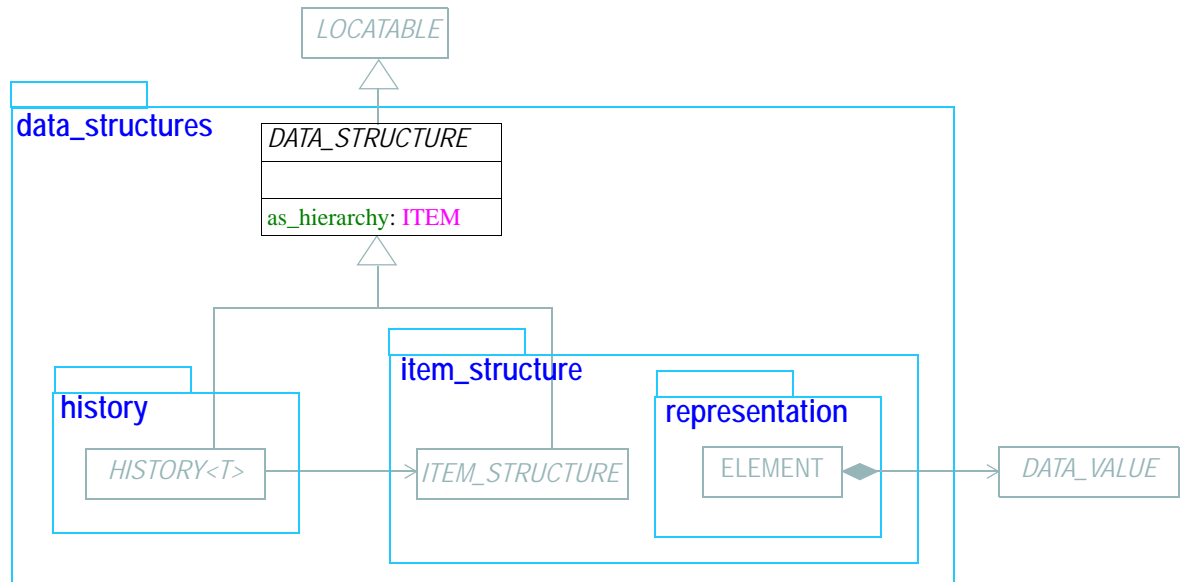


FIGURE 1 `rm.data_structures` Package

The `data_structures` package itself contains a single class, `DATA_STRUCTURE`, which is the ancestor of all *openEHR* 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 *openEHR* 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 *openEHR* 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 <i>as_hierarchy</i> 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 <i>as_hierarchy</i> will generate a CLUSTER containing the set of ELEMENT nodes from the list.	
Inherit	LOCATABLE	
Function	Signature	Meaning
	<i>as_hierarchy</i> : ITEM	Hierarchical equivalent of the physical representation of each subtype, compatible with CEN EN 13606 structures.
Invariants	<i>As_hierarchy_exists</i> : <i>as_hierarchy</i> /= 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 *openEHR* 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 *openEHR* Support Information Model.

The *openEHR* 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.

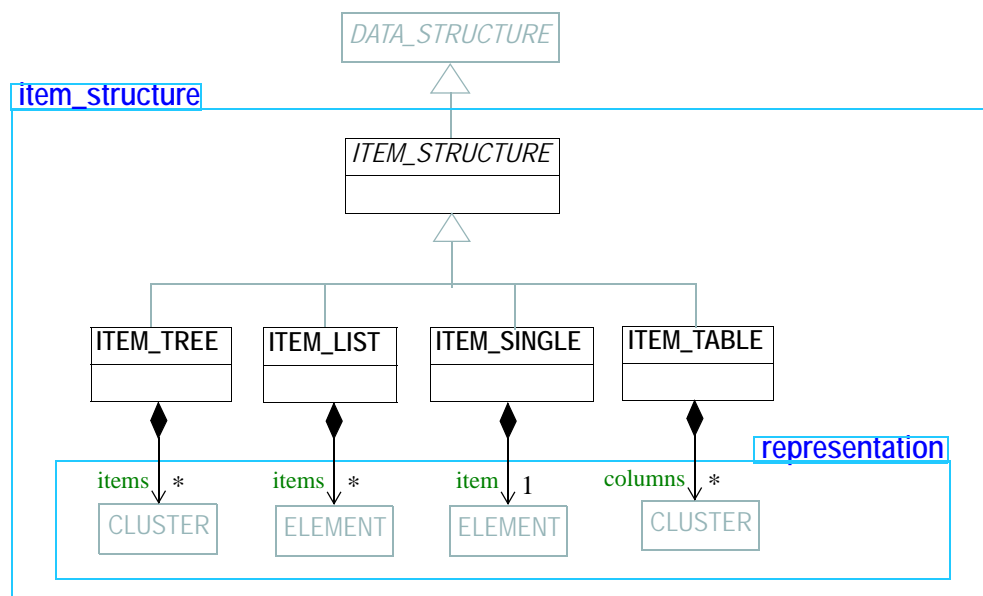


FIGURE 2 `rm.data_structures.item_structure` Package

4.2 Path Semantics

4.2.1 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 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 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 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]`

4.3 CEN EN13606 Encoding Rules

4.3.1 ITEM_TABLE

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.

4.3.2 ITEM_TREE

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

4.4 Class Descriptions

4.4.1 ITEM_STRUCTURE Class

CLASS	<i>ITEM_STRUCTURE (abstract)</i>	
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		

4.4.2 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
1..1	item: ELEMENT	
Functions	Signature	Meaning
	as_hierarchy: ELEMENT	Generate a CEN EN13606-compatible hierarchy consisting of a single ELEMENT.
Invariants	<i>Item_valid</i> : item /= Void	

4.4.3 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
	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'
	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 ELEMENTS of this list.
Invariants	<i>Valid_structure:</i> items.forall({ITEM}.type = "ELEMENT")	

4.4.4 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. 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	columns: List<CLUSTER>	Physical representation of the table as a list of CLUSTERS, 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> <i>require</i> i > 0 <i>and</i> i < row_count	Return the i-th row
	has_row_with_name (a_key:String): Boolean <i>require</i> a_key /= Void <i>and then not</i> 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 <i>require</i> a_key /= Void <i>and then not</i> a_key.empty	True if there is a column with name 'a_key'
	named_row (a_key:String): List<ELEMENT> <i>require</i> 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> <i>require</i> 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 <i>require</i> i >= 1 <i>and</i> i <= column_count j >= 1 <i>and</i> j <= row_count	Return the element at the column i, row j.
	element_at_named_cell (row_key, col_key:String): ELEMENT <i>require</i> 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	<i>Valid_structure</i> : items.forall({ITEM}.type = "CLUSTER" <i>and then</i> {ITEM}.items.forall({ITEM}.type = "ELEMENT"))	

4.4.5 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

CLASS	ITEM_TREE	
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 <i>require</i> 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		

4.5 Instance Structures

4.5.1 ITEM_SINGLE Instance Structure

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

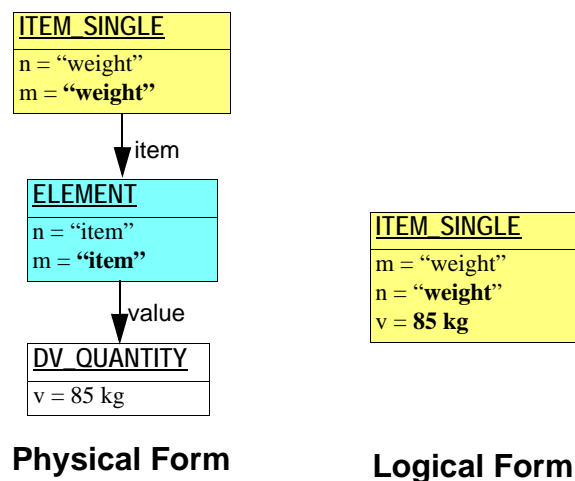


FIGURE 3 Instance Structure of ITEM_SINGLE

4.5.2 ITEM_LIST Instance Structure

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

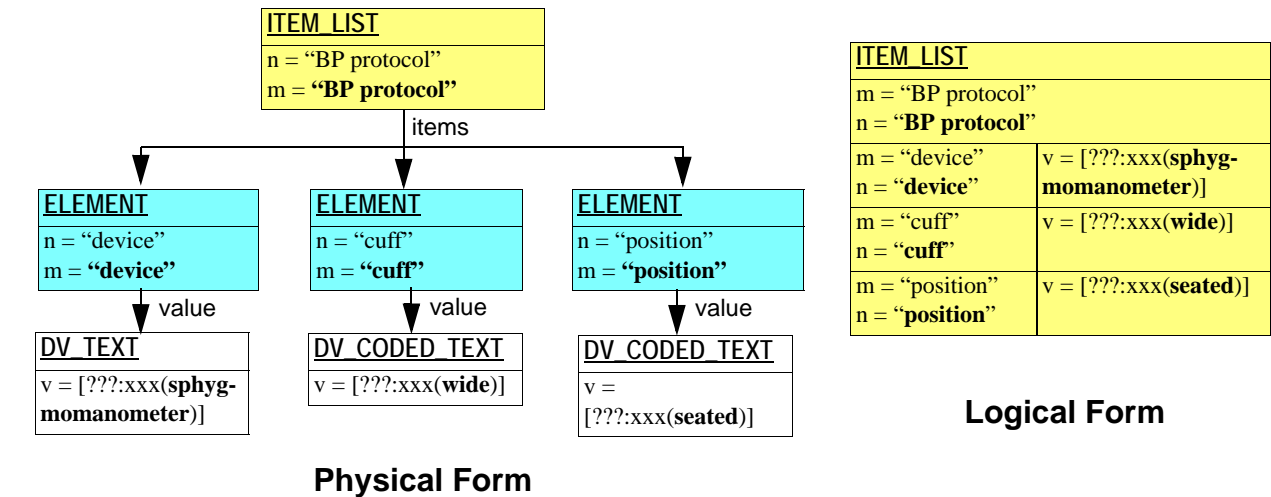
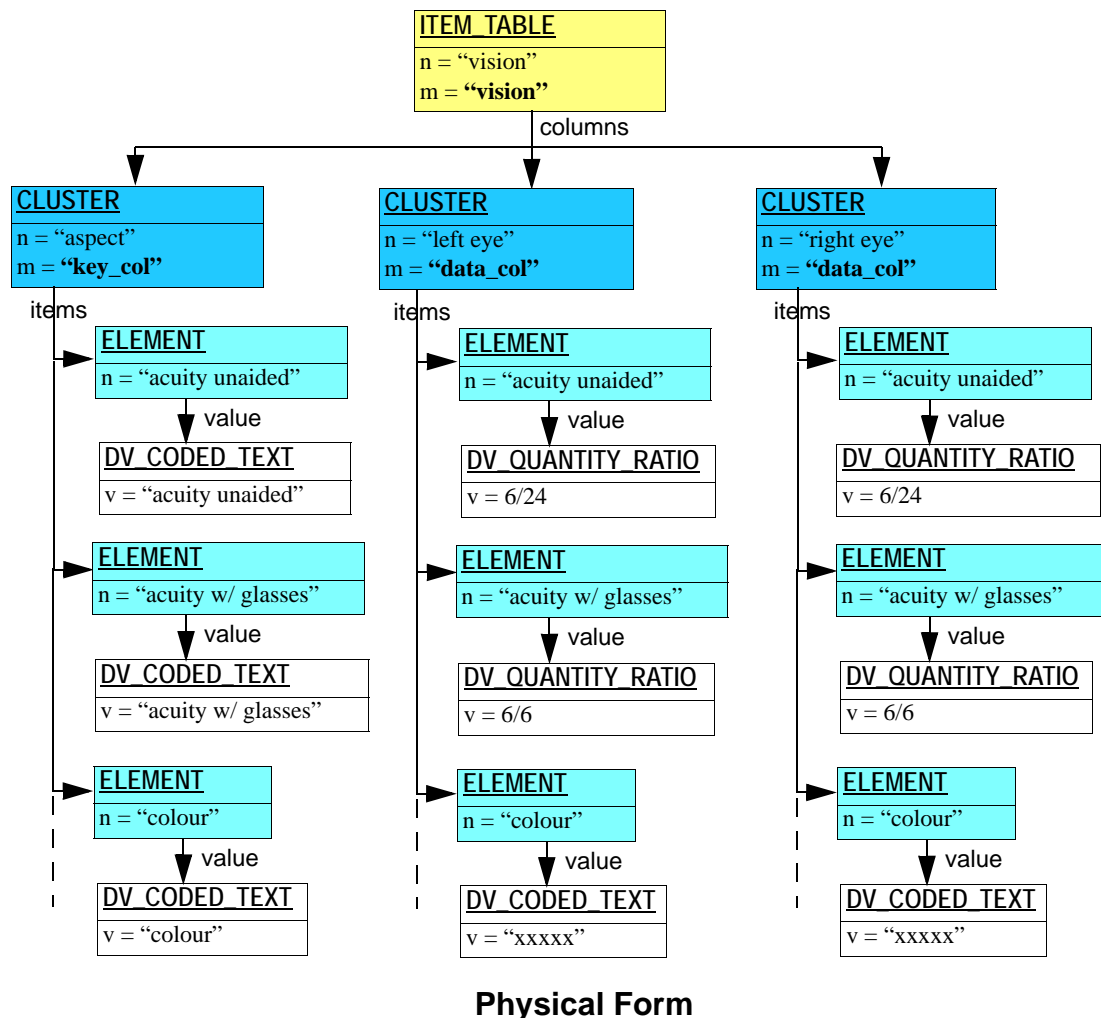


FIGURE 4 ITEM_LIST Instance Structure

4.5.3 ITEM_TABLE Instance Structure

FIGURE 5 illustrates a table of visual acuity test results.



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

Logical Form

FIGURE 5 ITEM_TABLE Instance Structure

4.5.4 ITEM_TREE Instance Structure

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

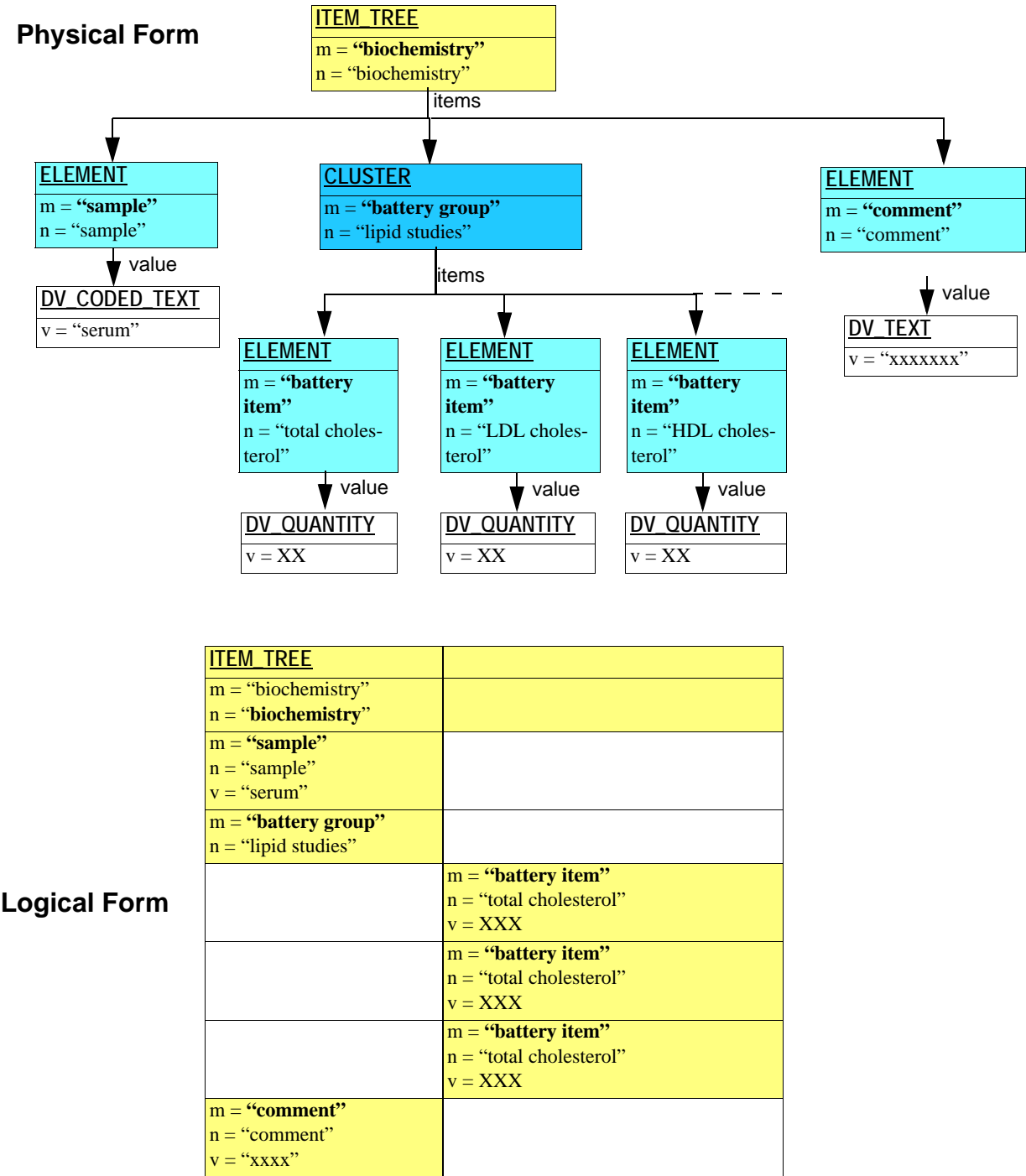


FIGURE 6 ITEM_TREE Instance Structure

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.

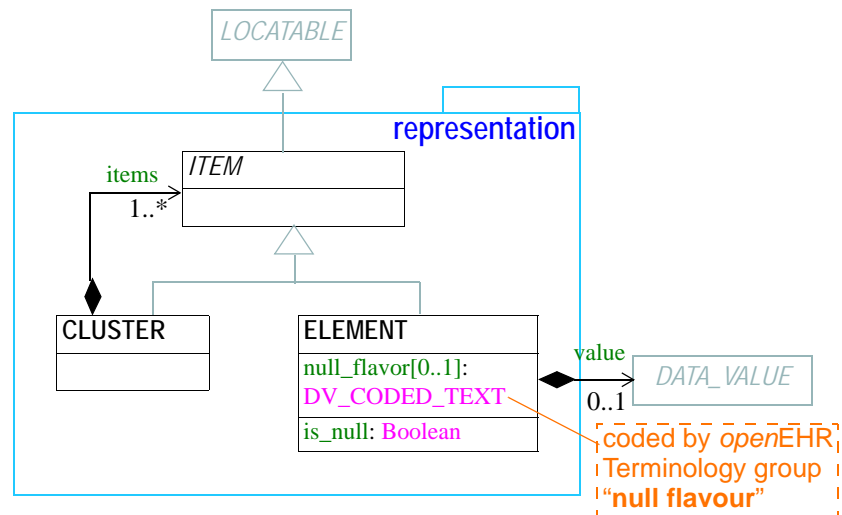


FIGURE 7 rm.data_structures.representation Package

5.2 Class Descriptions

5.2.1 ITEM Class

CLASS	<i>ITEM (abstract)</i>	
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	<i>Items_non_empty</i> : 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
	is_null: Boolean	True if value logically not known, e.g. if indeterminate, not asked etc.
Invariants	<i>Null_flavor_indicated:</i> is_null <i>xor</i> null_flavour /= Void <i>Null_flavour_valid:</i> is_null <i>implies</i> 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 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 *openEHR* “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 *openEHR* 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 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 archotyping (typically within *openEHR* 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 instances of the *EVENT* subtype, each containing a *time*: *DV_DATE_TIME* attribute representing the absolute time of the event. The relative off-

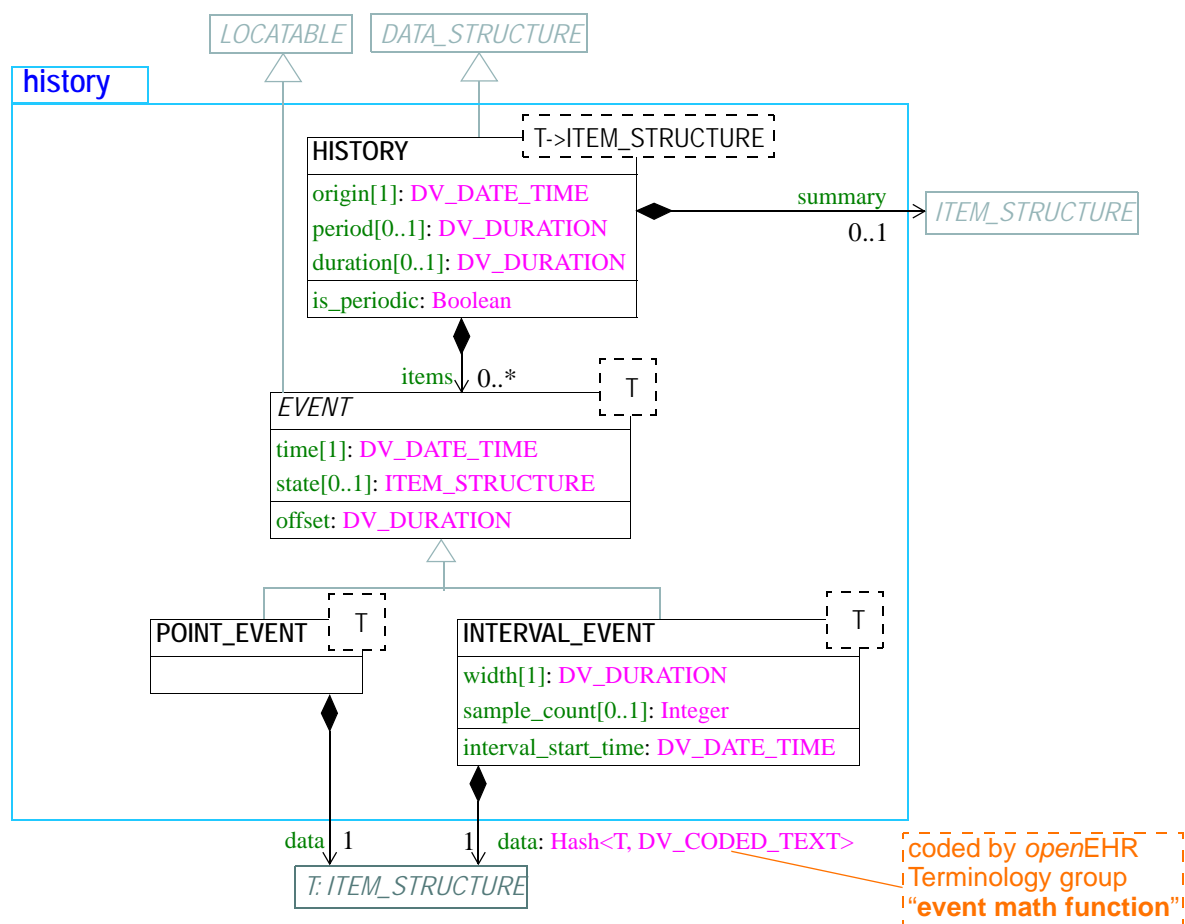


FIGURE 8 rm.data_structures.history Package

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. Apgar¹ 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.

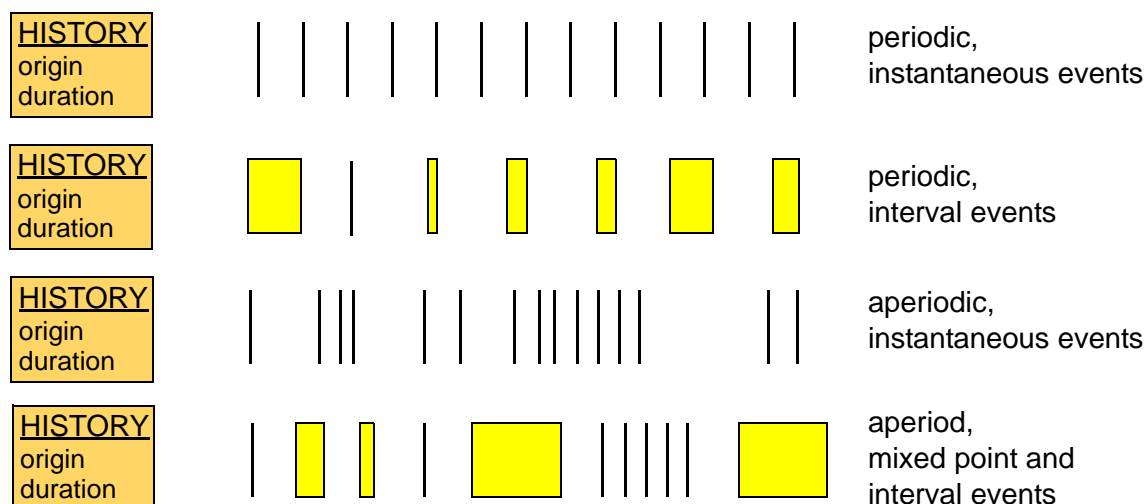
The total duration of the History is given by the `HISTORY.duration` function.

FIGURE 9 illustrates a number of variations in History periodicity and Event type.

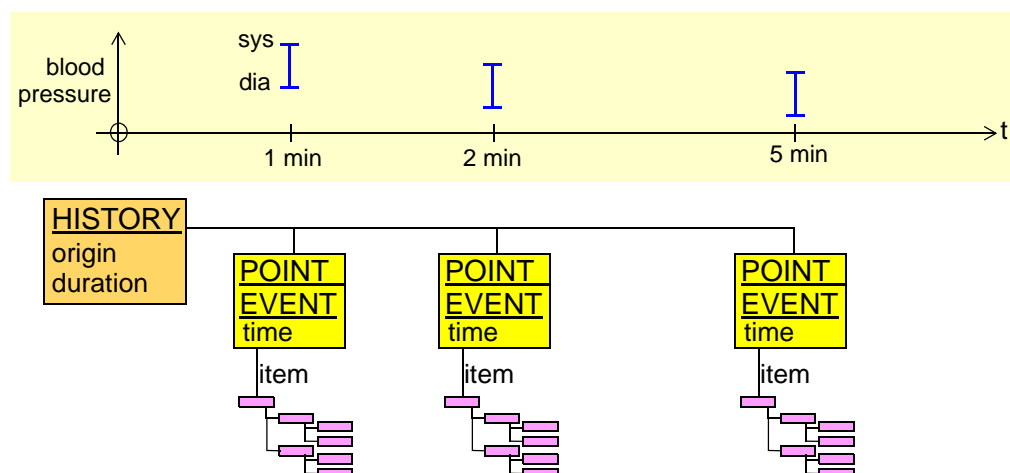
Point Events

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 may 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 consisting of 2 or 3 events, each containing 5 values and a 6th value representing the Apgar score for that

1. A simple 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

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. More than one value structure can be recorded for each `INTERVAL_EVENT`, such as average, minimum and maximum. Each value structure in the event is tagged with a `DV_CODED_TEXT` which codes for the “math function” (i.e. statistical function) for that particular structure. The full set of functions is coded by the openEHR Terminology group “event math functions”. 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 an average only, while a following event might contain maximum and minimum val-

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

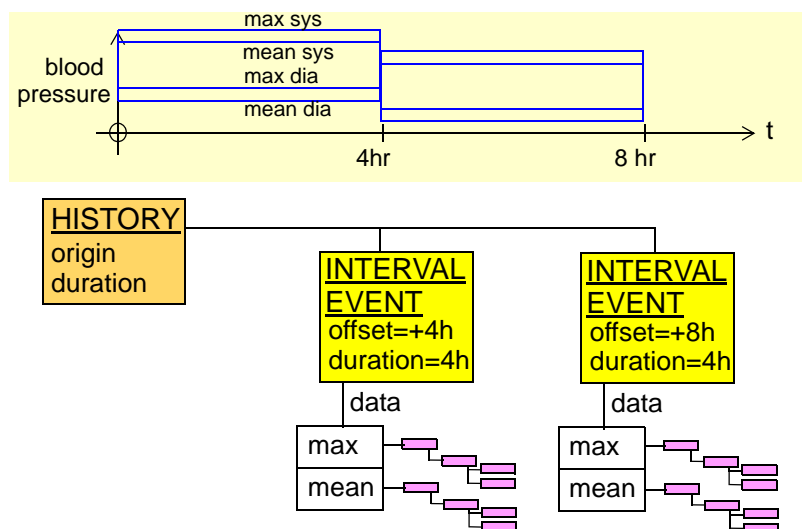


FIGURE 11 Structure of HISTORY<T> of INTERVAL_EVENTS

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.

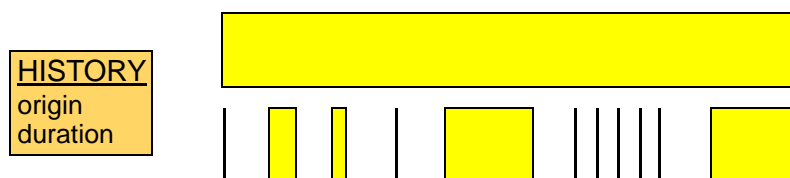


FIGURE 12 Overlapping Events

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.

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

Efficient Representation of Fine-grained Device Data

A useful practical consequence of the of interval Events is that it allows long periods of unchanging data to be represented with a single Interval event, while interesting perturbations will be represented with a number of fine-grained events, as shown in FIGURE 13, where only 5 Event instances represent 4 hours of data. The optional `INTERVAL_EVENT.sample_count` attribute can be used to record the number of original samples summarised in the event. 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 transformed into equivalent *openEHR* History form first. Usually,

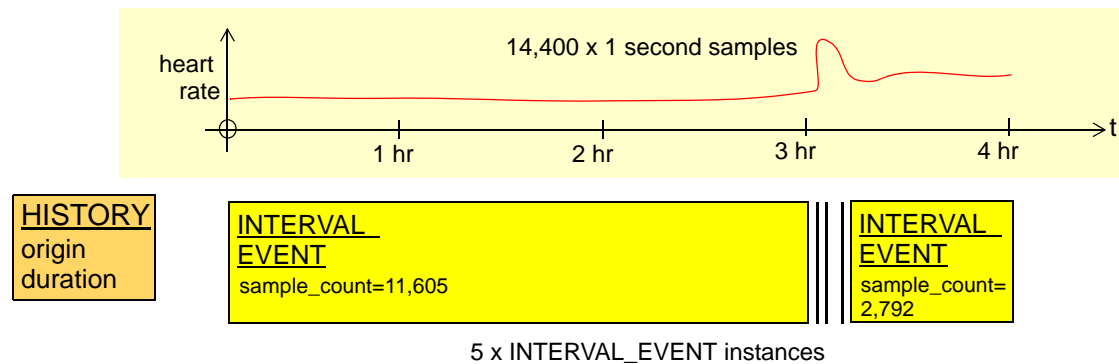


FIGURE 13 Data compression effect of History with Interval Events

State

A feature particular to a model of recording historical data for scientific and clinical use is the ability to record 'state'. In *openEHR*, '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 *openEHR* 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 *openEHR* 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.

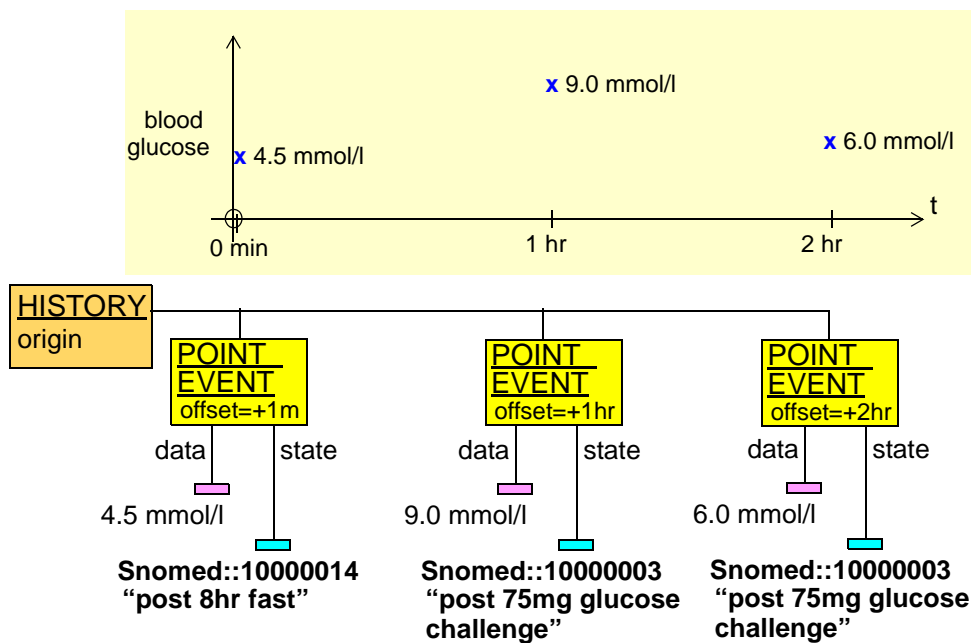


FIGURE 14 State in HISTORY

6.2 Path Semantics

Paths within History objects include the following possibilities:

- the origin of the History: `/origin`
- the third event in the History: `/items[event_3]`
- 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:
`/items[5 minute]/data/items[heart rate]`

6.3 Class Descriptions

6.3.1 HISTORY<T: ITEM_STRUCTURE> Class

CLASS	HISTORY<T: ITEM_STRUCTURE>
Purpose	Root object of a linear history, i.e. time series 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

CLASS	HISTORY<T: ITEM_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.
1..1	items: List <EVENT<T>>	The events in the series.
0..1	period: DV_DURATION	period between samples in this segment if periodic
0..1	summary: ITEM_STRUCTURE	Optional summary data expressing e.g. text or image which summarises entire History.
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	<i>origin_exists:</i> origin /= Void <i>items_exists:</i> (items /= Void and then not items.is_empty) <i>xor</i> summary /= Void <i>periodic_validity:</i> is_periodic <i>xor</i> period = Void	

6.3.2 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 interval 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.
0..1	state: ITEM_STRUCTURE	Optional state data for this event.
Functions	Signature	Meaning
	parent: HISTORY	Redefinition of LOCATABLE. <i>parent</i> to be of type HISTORY.

CLASS	<i>EVENT</i> <T: ITEM_STRUCTURE> (abstract)	
	offset: DV_DURATION	Offset of this event from origin.
Invariants	<i>Time_exists</i> : time /= Void <i>Offset_validity</i> : offset = time - parent.origin	

6.3.3 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
1..1	item: T	The data of this event.
Functions	Signature	Meaning
Invariants	<i>item_exists</i> : item /= Void	

6.3.4 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	data: Hash<T, DV_CODED_TEXT>	Data items for this event, keyed by mathematical function, e.g. “maximum”, “mean” etc. Coded using <i>openEHR</i> Terminology group “event math function”.
0..1	sample_count: Integer	Optional count of original samples to which this event corresponds.
Functions	Signature	Meaning
	interval_start_time: DV_DATE_TIME	Start time of the interval of this event.
	is_point_in_time: Boolean	True if the width is zero.

CLASS	INTERVAL_EVENT <T: ITEM_STRUCTURE>
Invariants	<i>data_exists</i> : data \neq Void <i>math_function_validity</i> : for_all k in data.keys: terminology("openehr").codes_for_group_name("event math function", "en").has(k.defining_code)

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 15 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).

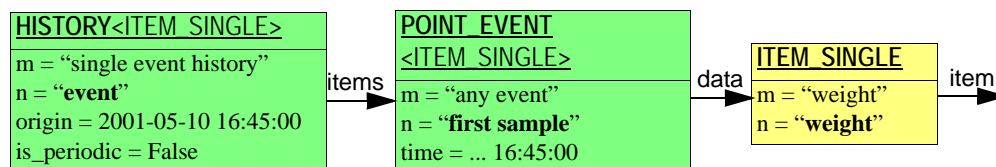


FIGURE 15 Single sample Instance Structure

6.4.2 5-minute Blood Pressure Averages

FIGURE 16 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.

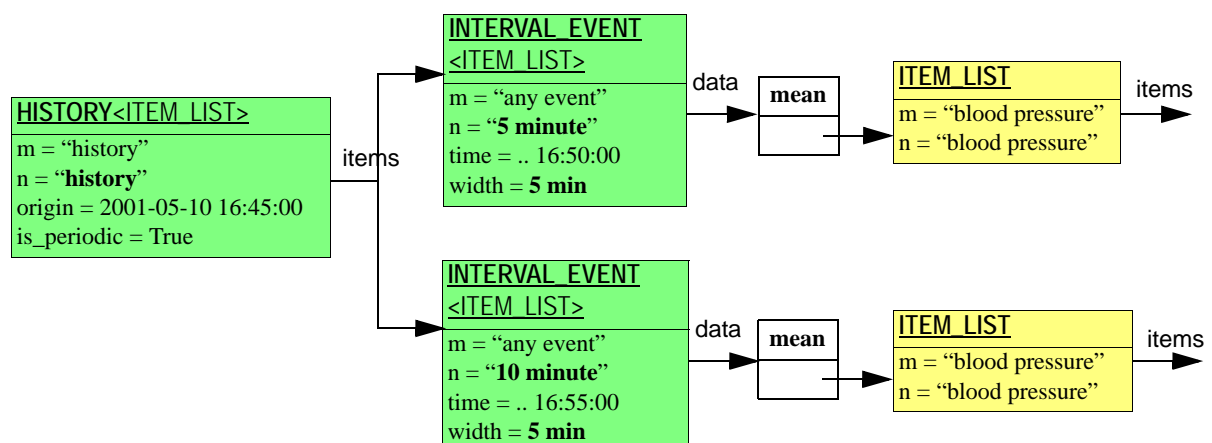


FIGURE 16 Blood Pressure Series History Instance Structure

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 2001 59pp 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