REFERENCE MODEL

The *open*EHR EHR Information Model

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

Revision: 4.5

Pages: 70

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

The *open*EHR foundation

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

Founding Chairman	David Ingram, Professor of Health Informatics, CHIME, University College London
Founding Members	Dr P Schloeffel, Dr S Heard, Dr D Kalra, D Lloyd, T Beale

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

Copyright Notice

Amendment Record

Issue	Details	Who	Completed
RELEASE 0.95			
4.5	**CR-000108**. Minor changes to change_control package. **CR-000024**. Revert *meaning* to STRING and rename as *archetype_node_id*. **CR-000098**. EVENT_CONTEXT.*time* should allow optional end time. **CR-000109**. Add *act_status* to ENTRY, as in CEN prEN13606. **CR-000116**. Add PARTICIPATION.*function* vocabulary and invariant. **CR-000118**. Make package names lower case. **CR-000064**. Re-evaluate COMPOSITION.*is_persistent* attribute. Converted *is_persistent* to a function; added *category* attribute. **CR-000102**. Make DV_TEXT *language* and *charset* optional.	T Beale S Heard, T Beale S Heard, DSTC A Goodchild T Beale T Beale D Kalra DSTC	10 Dec 2004
RELEASE 0.9			
4.4.1	**CR-000096**. Allow 0..* SECTIONs as COMPOSITION content.	DSTC	11 Mar 2004
4.4	**CR-000019**. Add HISTORY & STRUCTURE supertype. **CR-000028**. Change name of STRUCTURE class to avoid clashes. **CR-000087**. EVENT_CONTEXT.*location* should be optional. **CR-000088**. Move INSTRUCTION.*guideline_id* to ENTRY. **CR-000092**. Improve EVENT_CONTEXT *modelling*. Rename *author* to *composer*. **Formally validated using ISE Eiffel 5.4.**	T Beale H Frankel DSTC T Beale, D Kalra S Heard	06 Mar 2004
4.3.10	**CR-000044**. Add reverse ref from VERSION_REPOSITORY<T> to owner object. Add invariants to DIRECTORY and VERSIONED_COMPOSITION classes. **CR-000046**. Rename COORDINATED_TERM and DV_CODED_TEXT.*definition*.	D Lloyd T Beale	25 Feb 2004
4.3.9	**CR-000021**. Rename CLINICAL_CONTEXT.*practice_setting* to *setting*.	A Goodchild	10 Feb 2004
4.3.8	**CR-000057**. Environmental information needs to be included in the EHR.	T Beale	02 Nov 2003
4.3.7	**CR-000048**. Pre-release review of documents. **CR-000049**. Correct reference types in EHR, DIRECTORY classes. EHR.*contributions*, *all_compositions*, FOLDER.*compositions* attributes and invariants corrected. **CR-000050**. Update Path syntax reference model to ADL specification.	T Beale, D Lloyd	25 Oct 2003
4.3.6	**CR-000041**. Visually differentiate primitive types in openEHR documents.	D Lloyd	04 Oct 2003
4.3.5	**CR-000013**. Rename key classes, according to CEN ENV 13606.	S Heard, D Kalra, T Beale	15 Sep 2003

Issue	Details	Who	Completed
4.3.4	**CR-000011**. Add author attribute to EVENT_CONTEXT. **CR-000027**. Move *feeder_audit* to LOCATABLE to be compatible with CEN 13606 revision.	S Heard, D Kalra	20 Jun 2003
4.3.3	**CR-000020**. Move VERSION.*territory* to TRANSACTION. **CR-000018**. Add DIRECTORY class to RM.EHR Package. **CR-000005**. Rename CLINICAL_CONTEXT to EVENT_CONTEXT.	A Goodchild	10 Jun 2003
4.3.2	**CR-000006**. Make ENTRY.*provider* a PARTICIPATION. **CR-000007**. Replace ENTRY.*subject* and *subject_relationship* with RELATED_PARTY. **CR-000008**. Remove *confidence* and *is_exceptional* attributes from ENTRY. **CR-000009**. Merge ENTRY *protocol* and *reasoning* attributes.	S Heard, T Beale, D Kalra, D Lloyd	11 Apr 2003
4.3.1	DSTC review - typos corrected.	A Goodchild	08 Apr 2003
4.3	**CR-000003**, **CR-000004**. Removed ORGANISER_TREE. CLINICAL_CONTEXT and FEEDER_AUDIT inherit from LOCATABLE. Changes to path syntax. Improved definitions of ENTRY subtypes. Improved instance diagrams. DSTC detailed review. (Formally validated).	T Beale, Z Tun, A Goodchild	18 Mar 2003
4.2	**Formally validated using ISE Eiffel 5.2.** Moved VERSIONED_TRANSACTION class to EHR Package, to correspond better with serialised formalisms like XML.	T Beale, A Goodchild	25 Feb 2003
4.1	**Changes post CEN WG meeting Rome Feb 2003.** Moved TRANSACTION.*version_id* postcondition to an invariant. Moved *feeder_audit* back to TRANSACTION. Added ENTRY.*act_id*. VERSION_AUDIT.*attestations* moved to new ATTESTATIONS class attached to VERSIONED<T>.	T Beale, S Heard, D Kalra, D Lloyd	8 Feb 2003
4.0.2	Various corrections and DSTC change requests. Reverted OBSERVATION.*items*:LIST<HISTORY<T>> to *data*:HISTORY<T> and EVALUATION.*items*:LIST<STRUCTURE<T>> to *data*:STRUCTURE<T>. Changed CLINICAL_CONTEXT.*other_context* to a STRUCTURE. Added ENTRY.*other_participations*; Added CLINICAL_CONTEXT.*participations*; removed *hcp_legally_responsible* (to be archetyped). Replaced EVENT_TRANSACTION and PERSISTENT_TRANSACTION with TRANSACTION and a boolean attribute *is_persistent*.	T Beale	3 Feb 2003
4.0.1	Detailed corrections to diagrams and class text from DSTC.	Z Tun	8 Jan 2003
4.0	Moved HISTORY classes to Data Structures RM. No semantic changes.	T Beale	18 Dec 2002
3.8.2	Corrections on 3.8.1. No semantic changes.	D Lloyd	11 Nov 2002
3.8.1	Removed SUB_FOLDER class. Now folder structure can be nested separately archetyped folder structures, same as for ORGANISERs. Removed AUTHORED_TA and ACQUISITION_TA classes; simplified versioning.	T Beale, D Kalra, D Lloyd A Goodchild	28 Oct 2002

Issue	Details	Who	Completed
3.8	Added *practice_setting* attribute to CLINICAL_CONTEXT, inspired from HL7/ANSI CDA standard Release 2.0. Changed DV_PLAIN_TEXT to DV_TEXT. Removed *hca_coauthorising*; renamed *hca_recording*; adjusted all instances of *_ID; converted CLINICAL_CONTEXT.*start_time*, *end_time* to an interval.	T Beale, S Heard, D Kalra, M Darlison	22 Oct 2002
3.7	Removed Spatial package to Common RM document. Renamed ACTION back to ACTION_SPECIFICATION. Removed the class NAVIGABLE_STRUCTURE. Renamed SPATIAL to STRUCTURE. Removed classes STATE_HISTORY, STATE, SINGLE_STATE. Removed Communication (EHR_EXTRACT) section to ow document.	T Beale	22 Sep 2002
3.6	Removed Common and Demographic packages to their own documents.	T Beale	28 Aug 2002
3.5.1	Altered syntax of EXTERNAL_ID identifiers.	T Beale, Z Tun	20 Aug 2002
3.5	Rewrote Demographic and Ehr_extract packages.	T Beale	18 Aug 2002
3.3.1	Simplified EHR_EXTRACT model, numerous small changes from DSTC review.	T Beale, Z Tun	15 Aug 2002
3.3	Rewrite of contributions, version control semantics.	T Beale, D Lloyd, D Kalra, S Heard	1 Aug 2002
3.2	DSTC comments. Various minor errors/omissions. Changed inheritance of SINGLE_EVENT and SINGLE_STATE. Included STRUCTURE subtype methods from GEHR. *ehr_id* added to VT. Altered EHR/FOLDER attrs. Added EXTERNAL_ID.*version*.	T Beale, Z Tun	25 Jun 2002
3.1.1	Minor corrections.	T Beale	20 May 2002
3.1	Reworking of Structure section, Action class, Instruction class.	T Beale, S Heard	16 May 2002
3.0	Plans, actions updated.	T Beale, S Heard	10 May 2002
2.9	Additions from HL7 coded term model, alterations to quantity model, added explanation sections.	T Beale	5 May 2002
2.8.2a	Interim version with various review modifications	T Beale	28 Apr 2002
2.8.2	Error corrections to EHR_EXTRACT package. P Schloeffel comments on 2.7.	T Beale, P Schloeffel	25 Apr 2002
2.8.1	Further minor changes from UCL on v2.7.	T Beale	24 Apr 2002
2.8	Dipak Kalra (UCL) comments on v2.6 incorporated. Added External Package. Minor changes elsewhere.	T Beale, D Kalra	23 Apr 2002
2.7	Final development of initial draft, including EHR_EXTRACT, related models	T Beale	20 Apr 2002

Issue	Details	Who	Completed
2.6	Further development of path syntax, incorporation of Dipak Kalra's comments	T Beale, D Kalra	15 Apr 2002
2.5	Further development of clinical and record management clusters.	T Beale	10 Apr 2002
2.4	Included David Lloyd's rev 2.3 comments.	T Beale, D Lloyd	4 Apr 2002
2.3	Improved context analysis.	T Beale	4 Mar 2002
2.2	Added path syntax.	T Beale	19 Nov 2001
2.1	Minor organisational changes, some content additions.	T Beale	18 Nov 2001
2.0	Rewrite of large sections post-Eurorec 2001 conference, Aix-en-Provence. Added folder, contribution concepts.	T Beale	15 Nov 2001
1.2	Major additions to introduction, design philosophy	T Beale	1 Nov 2001
1.1	Major changes to diagrams; STILL UNREVIEWED	T Beale	13 Oct 2001
1.0	Based on GEHR Object Model	T Beale	22 Sep 2001

Acknowledgements

Thanks to...

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

Andrew Goodchild, senior research scientist at the Distributed Systems Technology Centre, Brisbane provided valuable in-depth comments and insights on all aspects of the model during its early development.

1 Introduction

1.1 Purpose

This document describes the *open*EHR EHR Information Model, which is a model of an interoperable EHR in the ISO RM/ODP information viewpoint. This model is somewhat different in scope from models such as the CEN ENV 13606 pre-standard and the HL7 Clinical Document Architecture (CDA), in that it describes a logical EHR information architecture rather than just an architecture for communication of EHR extracts or documents between EHR systems. The EHR equivalent of these extract specifications is given in the *open*EHR EHR_EXTRACT Information Model.

The intended audience includes:

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

1.2 Related Documents

Prerequisite documents for reading this document include:

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

Other documents describing related models, include:

- The *open*EHR EHR_EXTRACT Information Model
- The *open*EHR Demographic 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/ehr_im.pdf. New versions are announced on openehr-announce@openehr.org.

1.4 Peer review

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

To Be Continued: more work required

Reviewers are encouraged to comment on and/or advise on these paragraphs as well as the main content. Please send requests for information to info@openEHR.org. Feedback should preferably be 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

This section describes the inputs to the modelling process that created the *open*EHR Information Model.

2.1 Requirements

There are broadly three sets of requirements which inform this model, as described in the following subsections.

2.1.1 Original GEHR Requirements

From the European GEHR project (1992-5; [14]), the following broad requirements areas were identified:

- The life-long EHR
- Priority: Clinician / Patient interaction
- Medico-legal faithfulness, traceability, audit-trailing
- Technology & data format independent
- Facilitate sharing of EHRs
- Suitable for both primary & acute care
- Secondary uses: education, research, population medicine
- Open standard & software deliverables

These can be reviewed in detail at the GEHR page at CHIME, UCL.

2.1.2 GEHR Australia Requirements

The GEHR Australia project (1997-2001; [27], [28]) introduced further requirements, including:

- Support for clinical data structures: lists, tables, time-series etc
- Safer model than the original (European) GEHR: context attributes only in valid places (but still similar style)
- Separate compositions groups for “persistent”, “demographic” and “event” information in EHR, which corresponds closely to real clinical querying patterns.
- Interoperability at the knowledge level, i.e. level of domain definitions of information such as “discharge summary” and “biochemistry result”.
- XML-enabled
- Consider compatibility with CEN 13606, Corbamed, HL7v3

These requirements can be found at [28]. GEHR Australia produced a proof of concept implementation in which clinical archetypes were developed and used. See [2] for the technical description of archetypes.

2.1.3 European Synapses and SynEx Project Requirements

Following the original Good European Health Record project the EU-funded Synapses (1996-8; [20]) and SynEx (1998-2000; [10]) projects extended the original requirements basis of GEHR to include further requirements, as follows:

- the requirements of a federation approach to unifying disparate clinical databases and EPR systems: the federated health record (FHR) [21];

- the need to separate a generic and domain independent high-level model for the federated health record from the (closely related) model of the meta-data which defines the domain specific health record characteristics of any given clinical specialty and any given federation of database schemata;
- a formalism to define and communicate (share) knowledge about the semantic hierarchical organisation of an FHR, the permitted data values associated with each leaf node in a record hierarchy and any constraints on values that leaf nodes may take (the Synapses Object Dictionary) [22];
- the core technical requirements of and interfaces for a federation middleware service [20].

2.1.4 European EHCR Support Action Requirements

This EU Support Action project ("SupA"; [12], [13], [15]) consolidated the requirements published by a wide range of European projects and national health informatics organisations as a Consolidated List of Requirements [11].

2.1.5 ISO EHR Requirements

The above requirements publications and the recent experience of *open*EHR feed into the definition of a set of EHR requirements by ISO Technical Committee 215 (Health Informatics) - ISO TS 18308. The present draft [9] has been reviewed by the authors of this document and *open*EHR will seek to maintain a close mapping between its information models and services and this international requirements work. The *open*EHR mapping to ISO 18308 can be found on the openEHR website.

2.1.6 *open*EHR Requirements

New requirements for the *open*EHR proposal, based on previous experience in the projects mentioned above include the following:

- Better modelling of time and context (temporal/spatial approach)
- Better understanding of legacy system / federation problem (DSTC, UCL)
- Workflow modelling
- Convergence of EHR standards, leading to a future version of CEN ENV 13606.
- Harmonisation with the emerging HL7v3 standard.

2.2 Design Principles

There are numerous considerations outside the requirements which influence the information model, described in "Design Principles for the EHR" [3]. These are summarised as follows:

- System-of-systems understanding of information infrastructures (i.e. a set of collaborating middleware components and services), described in more detail below;
- Design paradigm: two level modelling, archetypes
- Separation of standards according to responsibilities
- Ontological analysis: five levels of concepts above level 0 reference ontologies
- A "context model" of information acquisition which describes contexts for each information fragment from the most detailed information structures to the context of the clinical session, and the enterprise.
- The principle of relegating the vagaries of implementation technologies to separate specifications, rather than compromising the central model in any way.

2.2.1 The System-of-systems EHR Context

As described in [3], health care information systems, both at the level of a single enterprise and regional healthcare networks, increasingly comprise a set of software components collaborating through middleware, enabling distributed operations and data exchange. This "system-of-systems" view embraces the distributed object paradigm (exemplified by CORBA and .net), the message-based paradigm (exemplified by HL7), and in a less formal sense, most of today's organically evolved multi-database hospital IT environments. The distributed systems paradigm is also the generally accepted theoretical and standards view, as exemplified by any modern textbook on information processing, by the ISO RM/ODP standard, by health informatics standards such as the OMG Corbamed [31], [32], [33] specifications and CEN family of health information standards ([23] and many others), and by a wide range of international projects, national health information strategies and demonstrator pilots.

Accordingly, the EHR is understood in *open*EHR as one system (or service) within a distributed health information infrastructure, whose purpose is to manage the longitudinal and comprehensive EHR of individual patients. There is no definitive list of services or systems required within a health care environment. However, it is generally understood that in any deployment scenario an EHR service would be complemented by a range of other services including: terminology, clinical reference data (prescribing, interactions), order management, scheduling, decision support, demographics (both patient and health practitioner), pathology, imaging, and access control. In a distributed system infrastructure, each of these will usually exist as a service within an infrastructure node. Each system has a information model, describing the semantics of the data which can be obtained from it or written to it, as well as its service interface describing the functional interface to the system.

The approach taken in defining the *open*EHR EHR Information Model has been to assume this distributed model, and therefore the existence of several other services, at minimum for demographics, access control, terminology and archetypes. The Information Model therefore contains classes and attributes to facilitate interoperability with such services rather than their duplication inside the EHR service (see design principle RM-ext-ref [3]).

Data from some systems is allowed to be encapsulated as EHR data, leading to the use of encapsulated data types appearing in the EHR information model. In some cases an attribute is defined as a parseable string, and a formal parse specification is supplied for the allowable values of the string. Examples where this occurs include the HL7-based general timing specification, and the *units* attribute of the `DV_QUANTITY` class (Data Types Information Model). Motivations for specifying a syntactical approach are described in [3].

2.3 Relationship to other EHR Information Models

Where relevant, the correspondences to other information models have been documented. Correspondences to the GEHR Australia and EU Synapses/SynEx models are shown, since these are the models on which the *open*EHR EHR information model is primarily based.

2.3.1 CEN TC/251 ENV13606

These models have been influenced by and have also influenced the models in CEN ENV 13606; accordingly, relationships to 13606 have been documented fairly precisely. There are some parts of this pre-standard which are ambiguous, or confusing, particularly where similar semantics appear in both parts I and IV, in which case the best attempt to understand the standard has been made.

Since January 2002, the ENV13606 prestandard has been the subject of significant revision, as part of its transition to a full European Standard ("EN"). This work has been influenced by the *open*EHR

specifications, and has itself been a source of further insights and changes to the *open*EHR specifications. Particular areas of *open*EHR which have been changed due to this process include:

- change of major class names (TRANSACTION -> COMPOSITION etc; see CR-000013);
- improved model of ATTESTATION (see CR-000025);
- improved model of feeder audits (see CR-000027).

2.3.2 HL7 Version 3

Correspondences to some parts of HL7 version 3 (ballot 5, July 2003) are also documented where possible, however, it should be understood that there are a number of difficulties with this. Firstly, while the HL7v3 Reference Information Model (RIM) - the closest HL7 artifact to an information model - provides similar data types and some related semantics, it is not intended to be a model of the EHR. In fact, it differs from the information model presented here (and for that matter most published information models) in two basic aspects: a) it is an amalgam of semantics from many systems which would exist in a distributed health information environment, rather than a model of just one (the EHR); b) it is also not a model of data, but an “analysis pattern” in the sense of Fowler [37] from which further specific models - subschemas - are developed by a process of custom restriction, in order to arrive at message definitions. As a consequence, data in messages are not instances of HL7v3 RIM classes, as would be the case in other systems based on information models of the kind presented here.

Despite the differences, there are also many areas which can be usefully harmonised, specifically, the data types, terminology use, archetypes and HL7 templates, and the correspondence between *open*EHR compositions and the HL7 Clinical Document Architecture (CDA).

2.3.3 OMG HDTF

To Be Continued: relationship to OMG COAS / orders models should be described as well.

3 The EHR Information Model

3.1 Overview

FIGURE 1 illustrates the package structure of the *open*EHR EHR information model.

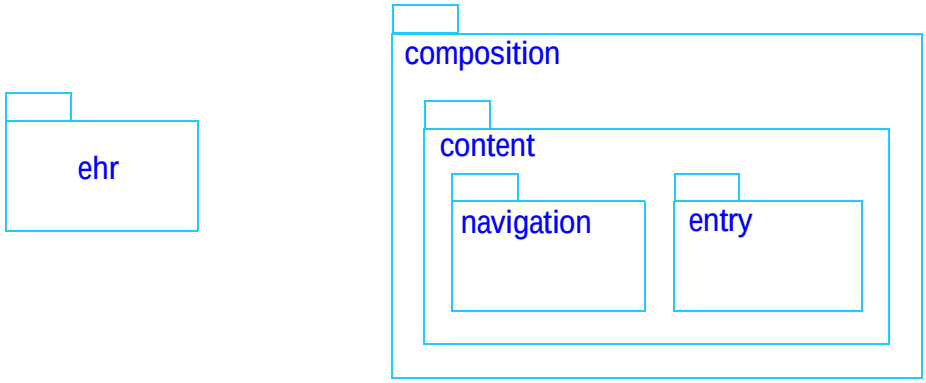

FIGURE 1 ehr Package Structure

The packages are as follows:

`ehr`: This package contains the top level structure, the `EHR`, which consists of a hierarchical structure of `FOLDER`s, containing references to `VERSIONED_COMPOSITION`s, and a collection of `CONTRIBUTION`s which document the changes to the EHR over time.

`composition`: The Composition is the EHR's top level "data container", and is described by the `COMPOSITION` class.

`content`: This package contains the Navigation and Entry packages, whose classes describe the structure and semantics of the contents of Compositions in the health record.

`navigation`: The `SECTION` class provides a navigational structure to the record, similar to "headings" in the paper record. `ENTRY`s and other `SECTION`s can appear under `SECTION`s.

`entry`: This package contains the generic structures for recording clinical statements. Entry types include `OBSERVATION` (for all observed phenomena, including mechanically or manually measured, and responses in interview), `EVALUATION` (for assessments, diagnoses, plans), and `INSTRUCTION` (actionable statements such as medication orders, recalls, monitoring, reviews).

FIGURE 2 illustrates an overview of the class structure of the EHR Information Model, along with the main concepts on which they rely, namely Data Types, Data Structures, Archetyped, and Identification. The EHR Extract core classes are also shown, illustrating the shared content of the EHR and extracts generated from it.

3.2 Archetypes

All compositional nodes in an `EHR` and a `COMPOSITION` are archetypable, with certain nodes being archetype root points. Instances of the types `EHR`, `COMPOSITION` and `ENTRY` are always guaranteed to be archetype root points; the topmost `SECTION` and `FOLDER` instances in any tree are also guaranteed to be archetype root points. Other nodes (e.g. interior `SECTION`s, `ITEM_STRUCTURE` instances) might also be archetype root points, depending on how archetypes are applied at runtime to data. This

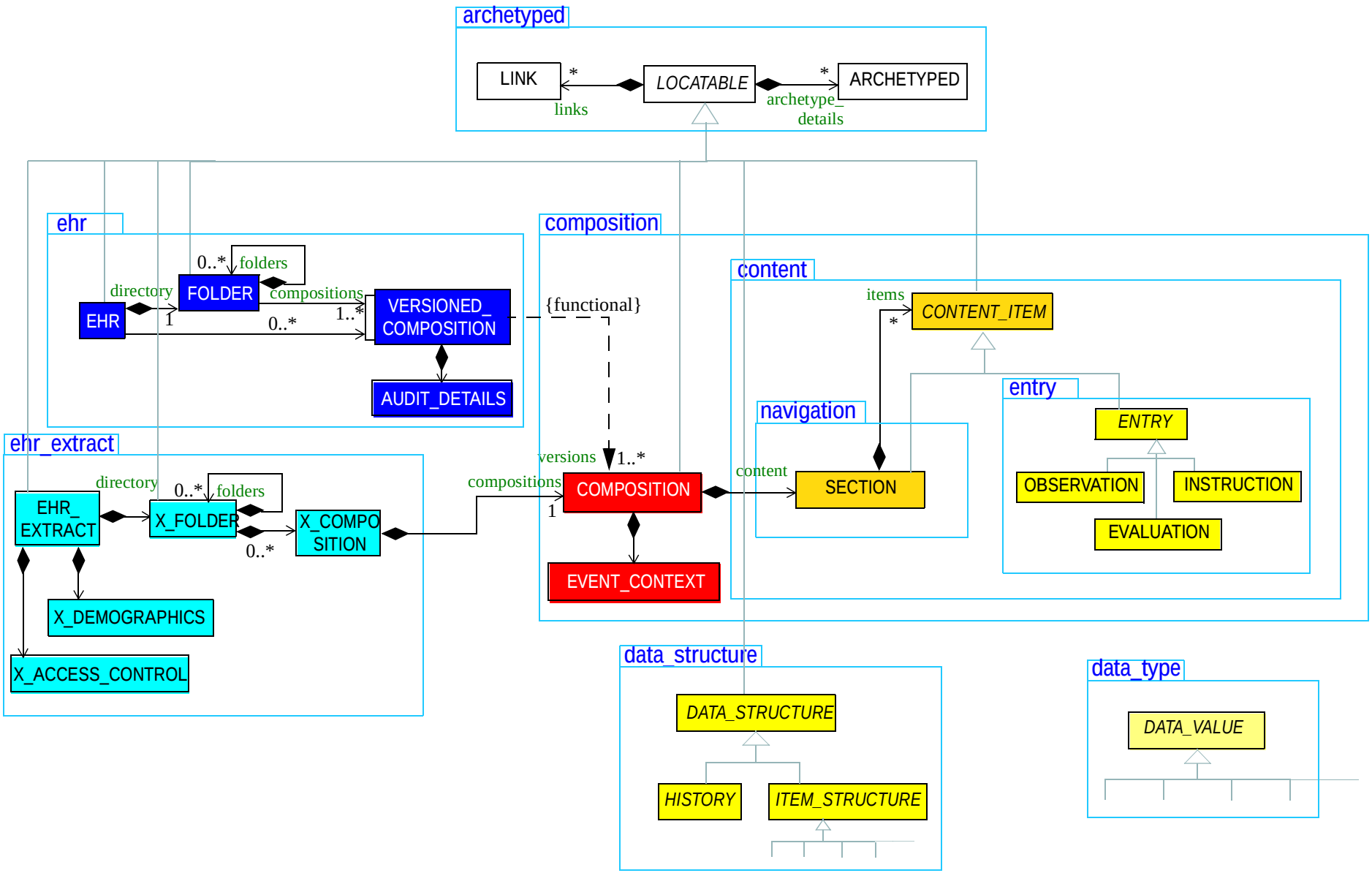

FIGURE 2 *open*EHR and EHR Extract Information Model Overview

is achieved by every node inheriting from the class LOCATABLE. Each archetype root point will have a non-void *archetype_details*, inherited from LOCATABLE; non-root nodes have no *archetype_details*. FIGURE 3 illustrates the application of archetypes to data. In each block of data controlled by a particular archetype the root node (i.e. the top node of the tree inside the block in question) has a non-void *archetype_details* attribute value. This figure shows how archetypes may be applied at any level of the data.

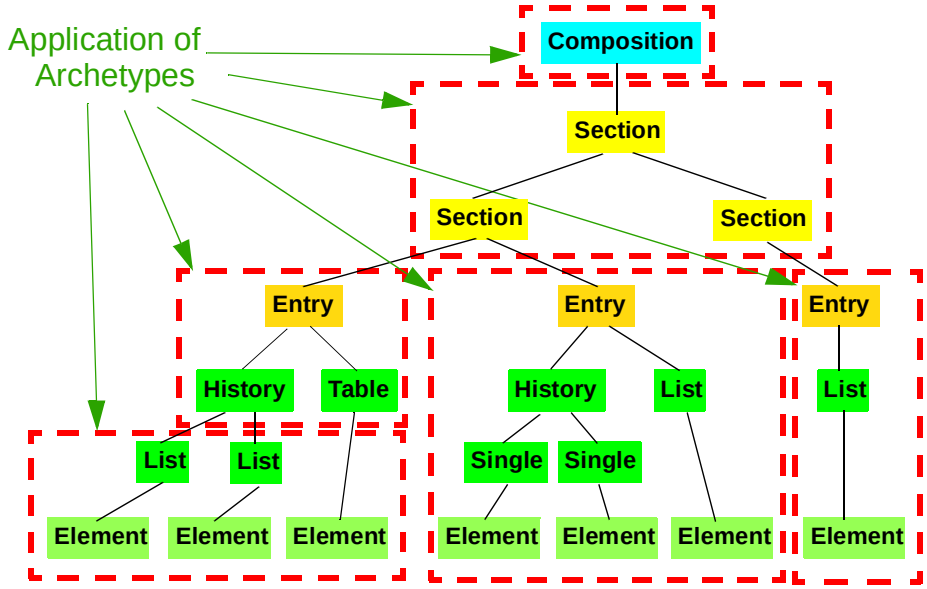

FIGURE 3 How Archetypes apply to Data

In this way, each *archetyped composition* in EHR data has a generating archetype which defines the particular configuration of instances to create the desired composition. An archetype for "biochemistry results" is an ENTRY archetype, and constrains the particular arrangement of instances beneath an ENTRY object; a "problem/SOAP headings" archetype constrains SECTION objects forming a SOAP headings structure. In general, an *archetyped composition* is any composition starting at a root node and continuing to its leaf nodes, at which point lower-level compositions, if they exist begin. Section trees and Entry structures are thus archetype compositions.

The result of the use of archetypes to create data in the EHR is that the structure of data in any particular *open*EHR health record conforms to the constraints defined in a particular composition of archetypes chosen by a user or piece of software during the creation of the data. In particular, it conforms to the path structure of the archetypes, as well as their terminological constraints. Which archetypes were used at data creation time is written into the data, in the form of both archetype identifiers at the relevant root nodes, and values of the *archetype_node_id* attribute (inherited from the class LOCATABLE), the basis for paths. When it comes time to modify or query data, these archetype data enable applications to retrieve and use the original archetypes, ensuring modifications respect the original constraints, and allowing queries to be intelligently constructed.

3.3 Paths

3.3.1 Runtime Path Syntax

The *open*EHR record includes a "runtime" path mechanism which enables any node or leaf item of EHR data to be referred to using a string path. The runtime path to an item obeys the Xpath-like path syntax defined in the Archetype Definition Language (ADL) specification, and is made from the con-

catenation of attribute names and object identifiers. Each object in a hierarchy has as its runtime identifier the value of its *name* attribute. All classes in the model whose instances are accessible by paths inherit the *name* attribute from the LOCATABLE class.

The value of the *name* attribute is chosen at runtime in various ways: either by the software application responsible for building the record data, by requesting a value from a human user, already fixed by an archetype, or via an algorithm. The general syntax model of path expressions to objects in the EHR is exactly the same as the ADL path syntax, i.e.:

['/' | root_object_id] attr_name ['[' object_id ']'] {'/' attr_name ['[' object_id ']' '/']}

This gives rise to paths of the form:

/attr1[object_id]/attr3/attr2[object_id]

/[root_obj_id]/attr1

3.3.2 Path Values and Language

In most cases, the *name* value is related to the value of the *archetype_node_id* attribute of each item, also inherited from the LOCATABLE class. The *archetype_node_id* attribute value in EHR data items is predefined by the archetypes from which the data was generated, and defines the "normative" meaning of a data item, regardless of what name is chosen for it at runtime; it is effectively a clinically meaningful node identifier. The clinical meaning is given in the archetype local ontology. However, an important difference is that *names* are text items, expressed in the language of the locale in which the data were created, whereas *archetype_node_ids* are language-independent codes. Typical items in the EHR will resemble the objects shown in FIGURE 4.

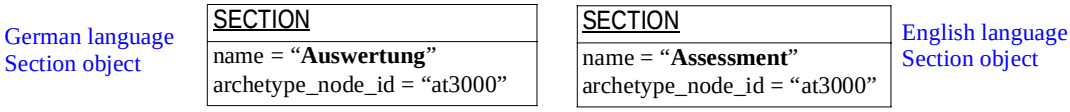

FIGURE 4 Names and archetype_node_ids in EHR Data

To compare a *name* value to a *archetype_node_id*, the logical *archetype_node_id* value in the relevant language has to be obtained from the archetype.

In contrast, all attribute names are drawn from the information models, and are therefore in English. (Making attribute names multi-lingual is more difficult, since object formalisms do not natively support this, and would seem to be of minimal benefit, especially as most reference model attribute names are quite generic, such as "data", "items", "value" and so on). As a consequence, runtime paths for EHR data not created in an English language system will consist of both English words (from the reference model) and clinical words in the language of the locale; this should not cause much of a problem, since all clinical words - the names - will appear within brackets ("[]").

3.3.3 Concrete Path Structure

The syntax applies to the compositional parts of the EHR to create two kinds of paths as follows:

ehr_path [**"/all_compositions"** composition_path [section_path [entry_path]]]

ehr_path [**"/directory"** folder_path composition_path [section_path [entry_path]]]

The first of these enables an EHR node to be referenced bypassing the Folder directory, while the second goes via the Folder directory. The parts are as follows:

ehr_path: identifier of the EHR. See EHR Path on page 33.

folder_path: path to any object in the EHR, via the folder structure. See Folder Paths on page 34.

composition_path: path to a COMPOSITION (a version of a VERSIONED_COMPOSITION) as defined in Composition Path on page 39.

section_path: path to a particular SECTION, as defined in Section Paths on page 44.

entry_path: path to an item in an ENTRY, as defined in ENTRY Paths on page 55.

This syntax allows paths to be partially specified, from the most minimal, referring to the whole EHR, to a fully specified leaf node. Examples of logical paths are as follows.

- Everything under the "subjective" heading of "diabetes" in a Composition committed by Dr Stephen Smith to patient 39403945's EHR at the primary EHR system at Nambour Base Hospital:

  /[39403945@ehr.nambour_bh.health.au]/all_compositions[patient contact (steven_smith @ ehr1.nambour_bh.health.au @ 03-05-1997 23:04:55)] /content[Diabetes] /items[Subjective]
- A complete family history Composition in the summary EHR for patient 959678b09, in the "B" EHR system at Nantes general hospital in France

  /[959678b09]/all_compositions[family history]
- The 1000Hz threshold value of the 3rd sample of an audiogram test, recorded under the headings "Hearing/test results", in a Composition committed to patient op01293's EHR at EHR node 'A' at St Bartholomew's Hospital, London.

  /[op01293@ehr.barts.uk]/all_compositions[Test Results (peter_cole@ehr_a.barts.uk@13-12-1990 09:22:00)] /content[Hearing] /items[test results] /items[audiology] /items[audiology results] /history /events[sample_3] /data[hearing threshold] /items[left ear] /items[1000Hz threshold]

Paths are used to construct instances of the data value type DV_EHR_URI, which are simply paths in the "ehr" scheme-space.

3.3.4 Archetype Paths

Archetype_node_id values are concatenated to form paths in archetypes, known as "archetype" paths using the ADL path syntax. Archetype paths can also be used with runtime data - since each runtime data element contains the relevant *archetype_node_id* value - they act as queries, or patterns. Whereas runtime paths are always unique in data, archetype paths are only unique inside archetypes, but may not be in data.

Runtime paths are thus used to locate data items or trees in the EHR, while archetype paths are used to match sub-compositions to their generating archetype structures, to identify matching sub-compositions during archetype-assisted querying, or to aid GUI display. As an illustration, assume that there is a blood pressure archetype shown here in the ADL abstract syntax:

```
ENTRY[at0000] matches {   -- blood pressure measurement
        name matches {...}
        data matches {
           HISTORY[at9001] matches {        -- history
              count matches {1..*}
              events {1..*} matches {
                 EVENT[at9002] {0..1}  matches {-- baseline
                    name matches {...}
                    data matches {

                       ITEM_LIST[at1000] matches {-- systemic arterial BP
```

```
                        count matches {2..*}
                        ordered matches {True}
                        items matches {
                           ELEMENT[at1100] matches {-- systolic BP
                              name matches {...}
                              value matches {...}
                           }
                           ELEMENT[at1200] matches {-- diastolic BP
                              name matches {...}
                              value matches {...}
                           }
                           ELEMENT[at9000] {0..*} matches {*}
                                                   -- unknown new item
                        }
                     }
                  }
               }
               EVENT[at9003] {0..1}  matches {-- other event
                     name matches {...}
                     data matches {
                        use_node  ITEM_LIST [at0000]/.../data[at1000]/
                                    -- list structure from first sample
                     }
                  }
               }
            }
         }
```

The archetype_node_ids are shown as the codes [atnnnn] at each node; the comments at the end of each of these lines is the english text of the meaning (however, any other language could have been used). The following physical archetype path is visible:

```
/[at1000]/data[at9001]/events[at9002]
```

or in logical form (i.e. with the meaning texts substituted for meaning codes):

```
/[BP measurement]/data[history]/events[baseline]
```

Now consider a data composition, in which a history of two blood pressures has been recorded using this archetype.

```
ENTRY[at0000] = <        -- blood pressure measurement
       name = <xxxx>
       data = <
          HISTORY[at9001] = <              -- history
             count = <xx>
             events = <
                EVENT[at9002] = <          -- baseline
                   name = <"standing">
                   data = <
                      ITEM_LIST[at1000] = <-- systemic arterial BP
                         ...
                      >
                   >
                EVENT[at9003] = <          -- other event
                   name = <"sitting">
                   data = <
                   ITEM_LIST[at1000] = <-- systemic arterial BP
```

```
                        ...
                    >
                >
            >
        >
```

The correspondending runtime paths from this data are as follows:

```
/[BP measurement]/data[history]/events[standing]
/[BP measurement]/data[history]/events[sitting]
```

Thus, where the same archetype path occurs, unique runtime paths are used. The rules governing archetype and runtime paths are as follows:

- The value of the *name* attribute must be unique in runtime data at each node, guaranteeing globally unique paths within the whole structure of data.
- The value of the archetype *archetype_node_id* attribute must be unique in archetypes at each node, guaranteeing globally unique paths within the whole structure of an archetype, and providing reliable “path patterns” in runtime data.

The example above illustrates the need to ensure that all instances of a blood pressure entry in the EHR can be related back to a common archetype governing its logical structure, and also ensure that each instance is uniquely and unambiguously identifiable.

4 The Record

4.1 Overview

The general design of the *open*EHR EHR is a combination of concepts from the GeHR Australia project [27], the Synapses project [20], [21], [22], the CEN ENV13606 standard, the HL7 Clinical Document Architecture (CDA) and the HL7 Reference Information Model (RIM) [29]. The structure consists of Compositions (equivalent to the CEN 13606 Composition and the HL7 CDA Document), organised by a directory of Folders (a CEN 13606 concept). The *open*EHR EHR is also versioned: every Composition is versioned, and so is the folder structure. This means that every previous state of the EHR - i.e. all states of Compositions and the Folder structure - is available if requested. The versioning facilities are provided by the concepts defined in the `rm.common.change_control` package. All changes to the EHR are created by *contributions*, where a contribution might include any of: creation of Compositions, update or correction of compositions, modification of the Folder structure, move of Compositions in the Folder structure, and deletion of Compositions or part of the Folder structure. This approach guarantees that the EHR progresses from one valid state to another, regardless of what changes occur in any particular contribution.

This approach to informational integrity underpins the clinical view of the health record, wherein events, issues, problems, episodes and other clinical arrangements of information are accommodated.

The discussion below commences by describing the general organisation of the EHR, and then how change control applies to the EHR. It then describes the semantics of Compositions - the containers of data in the EHR, and finally describes the role of the Folder structure.

4.2 General Organisation of the EHR

4.2.1 Compositions

The Composition concept in the *open*EHR EHR originated from the the Transaction concept of the GEHR project [16], [17], [18], [19], which was based on the concept of a unit of information corresponding to the interaction of a healthcare agent with the EHR. It was originally designed to satisfy the following needs (which include the well-known ACID characterisation of transactions [4]):

- *durability*: the need for a persistent unit of information committal in the record;
- *atomicity*: the need for a minimal unit of integrity for clinical information, corresponding to a minimal unit for committal, transmission and security;
- *consistency*: the need for contributions to the record to leave the record in a consistent state;
- *isolation*: the need for contributions to the record by simultaneous users not to interfere with each other;
- *indelibility*: the requirement that information committed to the record be indelible in order to support later investigations, for both medico-legal and process improvement purposes, and the consequent requirement to be able to access previous states of the record;
- *modification*: the need for users to be able to modify EHR contents, in order to correct errors or update previously recorded information (e.g. current medications, family history); and
- *traceability*: the need to record adequate auditing information at committal, in order to provide clinical and legal traceability.

The Transaction concept has since been renamed to “Composition”, which is the name of the equivalent concept in the current CEN 13606, and it has been expanded and more formally defined in

*open*EHR in two ways. Firstly, the idea of a unit of committal has been formalised by the *open*EHR model of change control (see the *open*EHR Common Information Model); how this applies to the EHR and compositions is described below. Secondly, the informational purpose of a Composition is no longer just to contain data from a passing clinical event such as a patient contact, but also to capture particular categories of clinical data which have long-lived significance, such as problem and medication lists.

Experience with health information systems, including the GEHR (Australia) project, SynEx, Synapses, and inspection of common commercial systems, has shown that there are basically two types of information at the coarse level which exist in the EHR: *event* items, and longitudinal, or *persistent* items, of which there might be various kinds.

Events record what happens during the *clinical session* context [3] which occur due to billable healthcare system events with or for the patient, such as patient contacts, but also sessions in which the patient is not a participant (e.g. surgery) or not present (e.g. pathology testing). Persistent items capture information which remains valid in the long term, such as the patient's family history, current medications, care plan and so on. Both types of information are contained within Compositions, the top level information container of the *open*EHR EHR. FIGURE 5 illustrates a simple EHR comprising an accumulation of event compositions.

FIGURE 5 Basic Event-oriented EHR

An important job of the event Composition is to record not only the data from the session, such as observations on the patient, but also to record the clinical context information, i.e. the who, when, where and why of the session. For this reason, a specific class representing clinical context is associated with event compositions in the formal model.

However, in a more sophisticated EHR, there are also likely to be persistent compositions. Many items of long-term interest in the record are separated by clinicians into well-known categories, such as:

- Family history
- Social history
- Problem list
- Current medications
- Therapeutic precautions
- Vaccination history
- Lifestyle
- Care plan

Note that over time, the number of event Compositions is likely to far outstrip the number of persistent Compositions. FIGURE 6 illustrates an EHR containing persistent information as well as event information.

In any clinical session, an event composition will be created, and in many cases, persistent compositions will be modified. How this works is described below under Change Control of the EHR on page 27.

FIGURE 6 An EHR containing Event and Persistent Compositions

4.2.2 Folders

As Compositions accumulate over time in the EHR, they form a long-term patient history. Previous work in CEN [23] - [26] and SynEx [21] suggests that it is useful to be able to organise Compositions using a hierarchy of folders to classify them, much the same way files in a file system are arranged in the directory structure, as visualised by the Windows Explorer and other similar tools. In the *open*EHR model, folders do not contain Compositions by value but by reference, and are completely optional. More than one Folder can refer to the same Composition. Folders might be used to manage a simple classification of Compositions, e.g into event and persistent, or they might be used to create numerous categories, based on episodes or other groupings of Compositions. Folder structures cann be archetyped.

A simple structure showing Folders referencing Compositions is shown in FIGURE 7, in which the following folders are used:

Subject: a composition containing clinically relevant demographic data of the patient;

Persistent: compositions containing information which is valid in the long term;

Event: compositions containing information whose currency is limited to the short term after the time of committal;

Episode_xxx: rather than using a single ‘event’ folder, it may be convenient to group event compositions into episodes (periods of treatment at a health care facility relating to one or more identified problems) and/or other categories such as on the basis of type of healthcare (orthodox, homeopathic, etc).

A justification for these particular categories is based on patterns of access. The persistent category consists of a dozen or so compositions described above, and which are continually required by querying (particularly lifestyle, current problems and medications). The event category consists of clinical data whose relevancy fades fairly quickly, including most measurements made on the patients or in pathology. Compositions in this category are thus potentially very numerous over the patient’s lifetime, but of decreasing relevance to the clinical care of the patient in time; it therefore makes sense to separate them from the persistent compositions.

Regardless of the folder structure used, the folder concept in itself poses no restrictions, nor does it add any clinical meaning to the record - it simply provides a logical navigational structure to the “lumps” of information committed to the record (remembering that inside compositions, there are other means of providing fine-grained structure in entries).

Note that neither the folder names nor the composition names described and illustrated above are part of the *open*EHR EHR architecture: all such details are provided by archetypes; hence, EHR structures based on completely different conceptions of division of information, or even different types of medicine are equally possible.

The folder structure of an EHR constitutes a thrid category of information which must be controlled over time, in order to allow changes to the folder structure to be remembered along with changes to content. The contents of a typical EHR now resemble FIGURE 8.

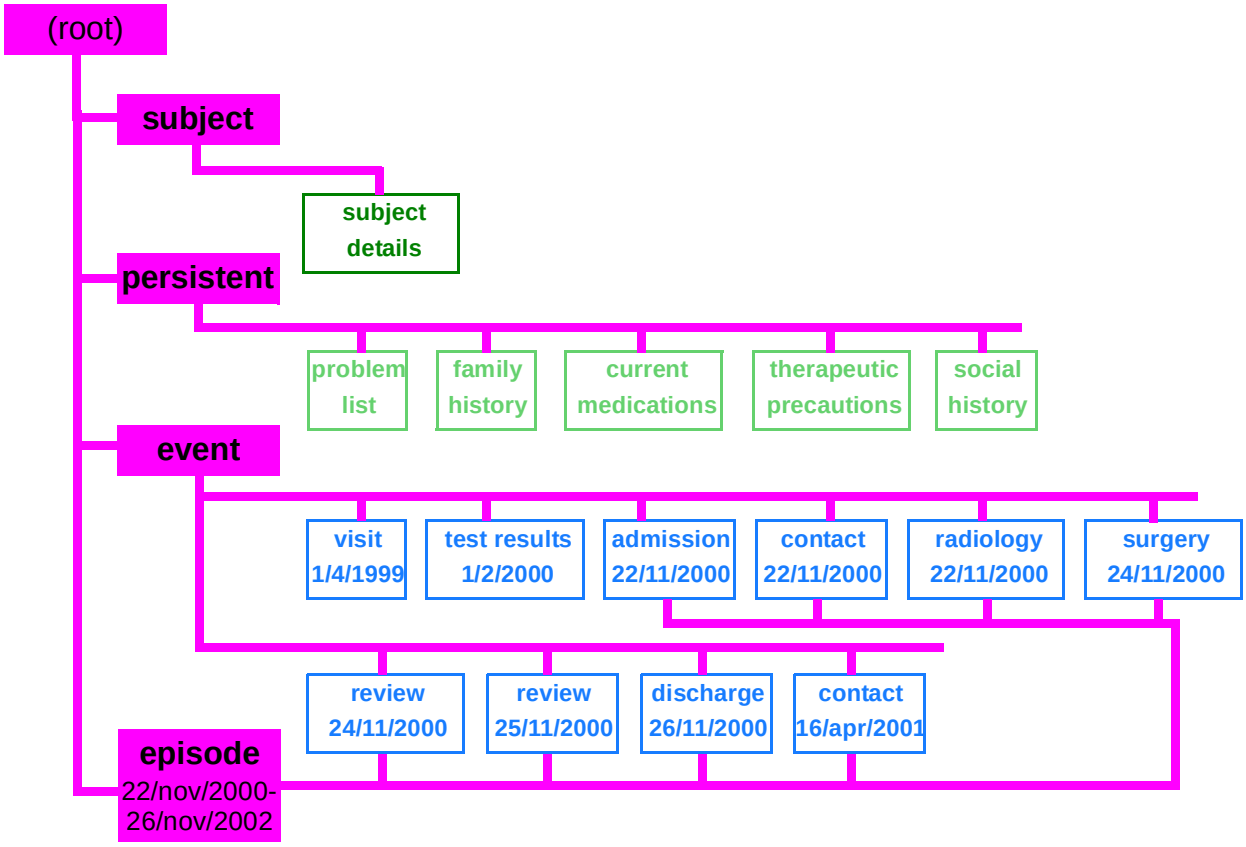

FIGURE 7 Using Folders to Group Compositions

FIGURE 8 An EHR containing Compositions and Folder Structure CIs

A final category of information which may be needed in the change-controlled EHR is that of “technical contextual information”, which includes environment settings, software application names and version ids, identification and versions of data resources such as terminologies and possibly even actual software tools, configuration files, keys and so on. Such information is commonly versioned in software configuration management systems, in order to enable the reconstruction of earlier versions of software with the correct tools. One reason to store such information at all is that it adds to medico-legal support when clinicians have to justify a seemingly bad decision: if it can be shown that the version of software in use at the time was faulty, they are protected, but to do this requires that such information be recorded in the first place. We therefore add a fourth category of content to the notional controlled EHR - that of “environment”, as shown in FIGURE 9.

FIGURE 9 A Comprehensive Medico-legal EHR

4.2.3 Change Control of the EHR

The EHR described thus far is essentially a logical arrangement of Compositions within a directory structure of Folders. However we have not described the semantics of update to the record, or audit-trailing of changes.

A number of requirements and design considerations lead to the final design of Compositions in the *open*EHR EHR. The first is the *system interaction context*, in which a healthcare agent (usually a human, but may be a software process) interacts with the EHR system to enter data. This is the context during which Compositions and Folder structures are created or modified on the system, and once again, the contextual details of who, when and where must be recorded. As described in [3], these details might be quite different from the context details of the *clinical session*, as is the case when the EHR is updated some time after a contact, and by other personnel.

Given an EHR in which there is a folder structure, and event and persistent compositions, the general model of update of the EHR is that any of these might be created and/or modified during the update. The simplest, most common case is probably the creation of a single contact Composition, which is placed in an "events" folder. A very common case will be the creation of an event Composition, and modification of one or more persistent Compositions, e.g. due to facts learned in the consultation about family history, or due to prescription of new medications. Other types of updates include corrections to existing compositions, and acquisition of compositions from another site such as a hospital. Any of these updates might also include a change to the folder structure, or the moving of existing Compositions to other Folders. Naturally these scenarios depend on a structure of the record including event and persistent compositions, and a folder structures; in the extreme, an EHR consisting only of event Compositions and no folders, will experience only the creation of a single Composition for most updates, with acquisitions being the exception.

Recording of contextual information is not the only requirement of the EHR. Numerous projects (GEHR/Europe, GEHR/Australia, SynEx, Synapses etc) as well as standards (CEN 13606, the emerging ISO 18308 EHRRA Requirements) and academic work all agree on the need to satisfy a number of medico-legal requirements of the EHR. These are essentially: that all additions and changes to the record be audit-trailed and that all previous states of the record be available for the purposes of medico-legal investigation. The former is satisfied by the recording of context details in the relevant places (including at the Entry level, dealt with later in this specification). The latter requirement leads us to the use of version control of information items, and eventually, to a formal change management approach.

Change management of information is a non-trivial business, and requires a well-defined approach, such as the "configuration managment" (CM) paradigm described in the *open*EHR Common Information Model. Under this paradigm we can visualise how changes occur to the EHR. FIGURE 10 shows a number of *contributions* (known as "change sets" in CM) to the EHR as follows:

- The first is due to a patient contact, and causes the creation of a new contact composition; it also causes changes to the problem list, current medications and care plan compositions (once again, in a differently designed record, all this information might have been contained in a single event Composition; likewise, it might be been distributed into even more Compositions).
- The next contribution is the acquisition of test results from a pathology laboratory.
- The third is another contact in which both family history and the folder structure are modified, and the fourth is a correction.
- This fourth is an error correction (e.g. a mispelled name, wrongly entered value), and shows that there can be a contribution even when there is no clinical session.

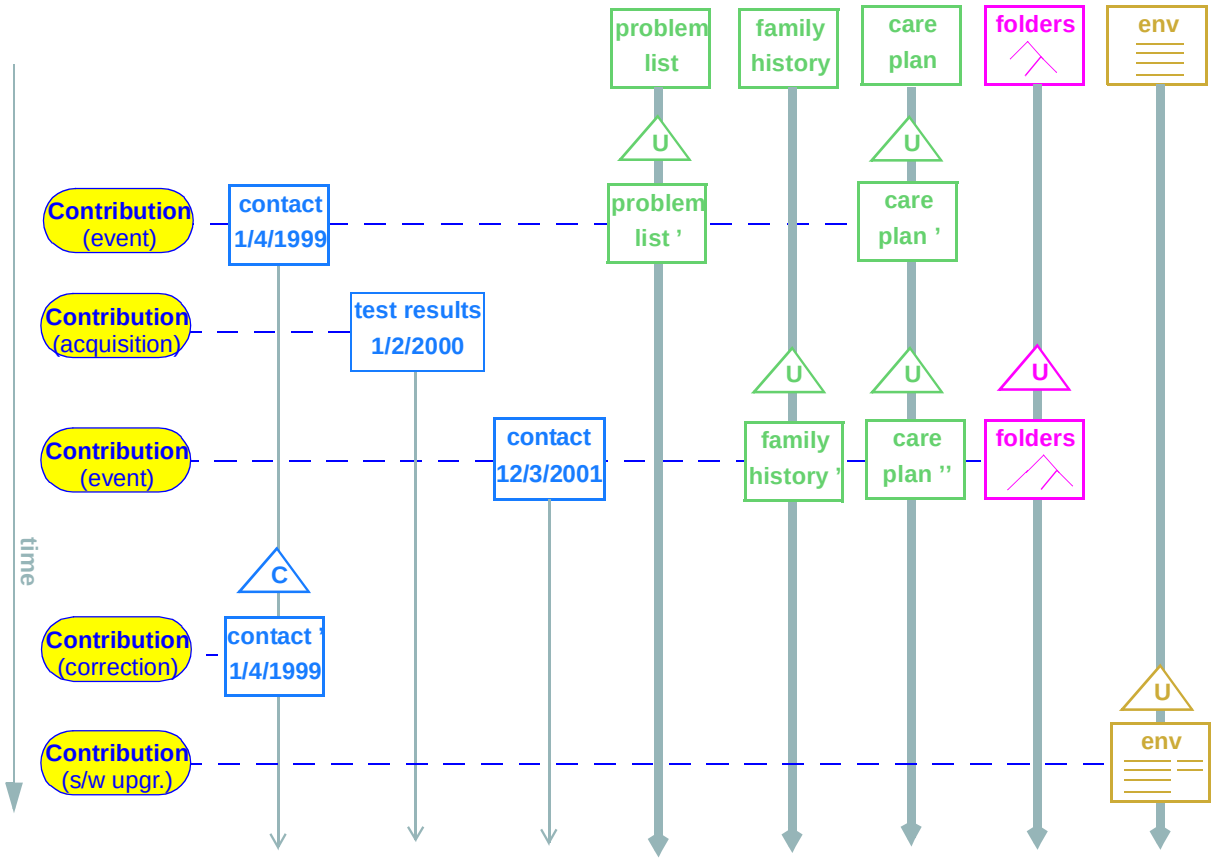

FIGURE 10 Contributions to the EHR

- The last is an update to the environmental information in the EHR, due to a software upgrade.

We can now see that the CM paradigm is a suitable approach. Consider the EHR described thus far:

- it is a repository of information about the patient, which is separated into distinct entities, (Compositions);
- it may have a directory of folders acting as a navigational structure of the compositions;
- there can be multiple, simultaneous users of the repository;
- changes to the information occur due to interactions of users with the repository, and must be attested with revision history information;
- previous states in time of the repository must be available upon request.

Thus, the EHR corresponds very closely to the general model of a change-controlled repository. The implication is that we should consider updates to the EHR to be the "contributions" described in the CM paradigm, where each contribution causes the creation or modification of one or more Compositions (configuration items, or "CIs") and/or changes to the folder structure (directory structure).

4.2.4 Versioning of Compositions

Versioning of Compositions is achieved with the `VERSIONED<T>` type from the Change Control package, which in the Composition package is explicitly bound to the `COMPOSITION` class, via the class `VERSIONED_COMPOSITION` which inherits from the type `VERSIONED<COMPOSITION>`. This is done because versioned Compositions are self-standing entities - they are not contained by value in

any other class. Consequently, there is no other class where the type binding VERSIONED<COMPOSITION> can take place.

The effect of version control on Compositions is visualised in FIGURE 11. The versions (each "version" being a COMPOSITION) shown here in a VERSIONED_COMPOSITION are the same versions shown along each vertical line in FIGURE 10, this time shown with their associated audit items. The set of versions should be understood as a set of successive modifications of the same data in time.

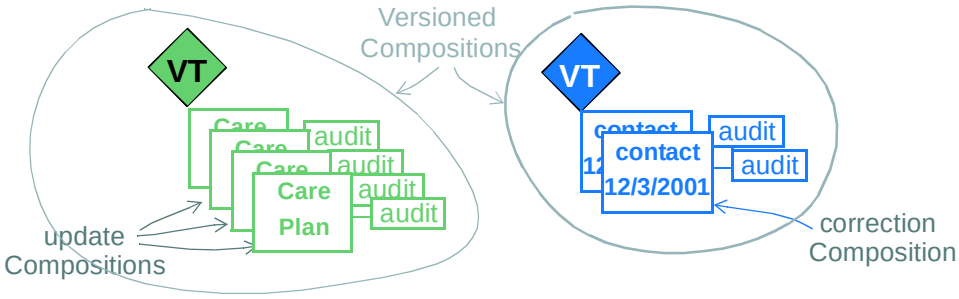

FIGURE 11 The versioned composition

The VERSIONED_COMPOSITION can be thought of as a kind of intelligent repository: how it stores successive versions in time is an implementation concern (there are a number of intelligent algorithms available for this sort of thing), but what is important is that its functional interface enables any version to be retrieved, whether it be the latest, the first, or any in between.

Returning to Compositions, the logical types "event composition" and "persistent composition" are modelled using the class COMPOSITION, a coded attribute *category*, and a Boolean-returning function *is_persistent*. The function returns true for those categories deemed to be persistent categories, and can be implemented as required on each system. For event Compositions this function should return false. There will usually be a *context* attribute, which carries the clinical context information corresponding to the event, although in continuous care situations this might not always be used. Persistent Compositions do not include this, although if it is necessary to determine what clinical session (if any) caused an update to a persistent Composition, the contribution can be found from the audit, and then checked for the presence of an event Composition.

4.2.5 Versioning Scenarios

The following scenarios for creating new COMPOSITION versions have been identified as follows.

Case 0: information is authored locally, causing the creation of a new VERSION<COMPOSITION>. If this is the first version, a new VERSIONED_COMPOSITION will be created first.

Case 1: information is modified locally, such as for the correction of a wrongly entered datum in a composition. This causes the creation of a new VERSION<COMPOSITION> in an existing VERSIONED_COMPOSITION, in which the AUDIT_DETAILS.*change_type* is set to "correction".

Case 2: information received from a feeder system, e.g. a test result, which will be converted and used to create a new VERSION<COMPOSITION>. This kind of acquisition could be done automatically. If the receiver system needs to store a copy of the original feeder system audit details, it writes it into the COMPOSITION.*feeder_audit*.

Case 3: a VERSION<COMPOSITION> (such as a family history) received as part of an EHR_EXTRACT from another *open*EHR system, which will be used by a local author to

create a new COMPOSITION which includes some content chosen from the received VERSION<COMPOSITION>. In this case, the new VERSION<COMPOSITION> is considered as a locally authored one, in which some content has been obtained from elsewhere. If it is the first version, a VERSIONED_COMPOSITION is first created. The AUDIT_DETAILS documents the committal of this content, and the clinician may choose to record some details about it in the audit *description*.

In summary, the AUDIT_DETAILS is always used to document the addition of information locally, regardless of where it has come from. If there is a need to record original audit details, they become part of the content of the versioned object.

5 EHR Package

5.1 Overview

The ehr package is illustrated in FIGURE 12. The EHR class is the root of the EHR information structure, and is a change-controlled repository of the kind described in the *open*EHR Common Information Model. Accordingly, it contains a directory, in the form of a versioned Folder structure, logical reference to the VERSIONED_COMPOSITIONs which are the versioned data containers of the EHR, and references to the CONTRIBUTIONs which document all changes so far to the EHR. The directory structure is optional, and consists of FOLDERs, enabling the construction of a hierarchical directory of any complexity. Each folder in the structure can contain any number of references to versioned Compositions. The structure as a whole acts as a directory for organising Compositions in the record.

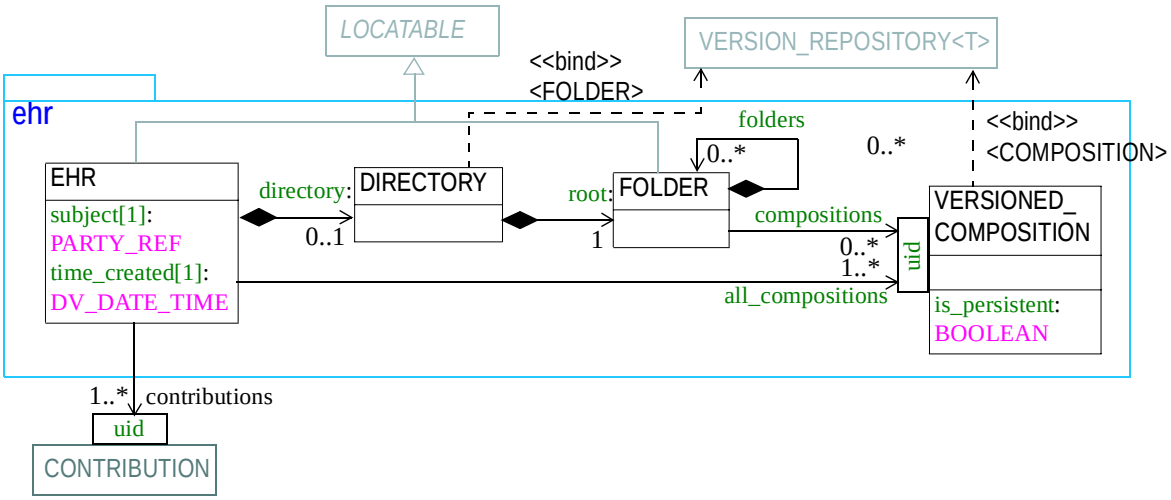

FIGURE 12 rm.ehr Package

The use of references between Folders and Compositions allows more than one folder to refer to the same Composition, in turn allowing multiple ways of finding or classifying Compositions. This arrangement is akin to a computer directory system which also allows links (Unix) or “shortcuts” (Windows); in the EHR it allows for example a contact Composition to be grouped with other Compositions in an episode, and also in a group corresponding to a type of problem. The sophistication of the folder structure is completely under the control of the designers of EHR systems, and can be as simple or complex as required, according to the use of archetypes. The whole folder structure may correspond to one archetype, or there may be multiple archetypes used to create it.

References rather than containment by value are also used for the *all_compositions* relationship between the EHR and VERSIONED_COMPOSITION classes, reflecting the vast majority of retrieval scenarios in which only select (usually recent) Compositions are needed. Containment by value would lead to systems which retrieved all VERSIONED_COMPOSITION objects every time the EHR object was accessed. (However, while undesirable from the resource usage point of view, there is nothing semantically incorrect with containment by value between the EHR and its subparts).

Exactly the same logic holds for the relationship between the EHR and its CONTRIBUTIONs. The role of Contributions, as documented in the Common Information Model, is to record the set of versions added to a repository during a single logical update, along with the audit details of the change. In the context of the EHR as one such repository, each instance of the CONTRIBUTION class potentially includes in its list of versions not only compositions, but also the folder structure itself, if this was

changed during the update. Since the *versions* attribute of the CONTRIBUTIONs class is in fact a list of OBJECT_REFs, the same list can accommodate both types of object.

5.2 Class Descriptions

5.2.1 EHR Class

CLASS	EHR	
Purpose	The EHR class is the centre node of the EHR "repository" for a subject of care.	
CEN	EHCR class	
Synapses	RecordFolder class	
GEHR	G1_EHR	
Inherit	LOCATABLE	
Attributes	**Signature**	**Meaning**
	subject: PARTY_REF	The subject of this EHR.
	time_created: DV_DATE_TIME	Time of creation of the repository
	contributions: List<OBJECT_REF>	List of contributions causing changes to this EHR. Each contribution contains a list of versions, which may include references to any number of VERSION instances, i.e. items of type VERSIONED_COMPOSITION and DIRECTORY.
	directory: DIRECTORY	Optional directory structure for this EHR.
	all_compositions: List <OBJECT_REF>	Master list of all composition references in this EHR
Invariants	***Is_archetype_root***: is_archetype_root ***Directory_valid***: directory /= Void ***and then*** directory.latest_version.data.is_archetype_root ***Subject_exists***: subject /= Void ***Time_created_exists***: time_created /= Void ***Contributions_valid***: contributions /= Void ***and then not*** contributions.is_empty ***and then*** contributions.for_all(type.is_equal("CONTRIBUTION")) ***All_compositions_valid***: all_compositions /= Void ***and then*** all_compositions.for_all(type.is_equal("VERSIONED_COMPOSITION"))	

5.2.1.1 EHR Path

The path to the EHR as a whole is formed from the value of the EHR name attribute. The name attribute is usually the patient identifier, i.e. the *subject.id.value* attribute (usually a meaningless identifier), optionally with the EHR node id appended. The syntax is as follows:

```
'[' EHR.name.value "]
```

Possible EHR paths are as follows:

```
[10290494@st_vincents.health.au]
[04959900021]
```

Paths *from* the EHR must include the appropriate attribute names, as follows:

- path to any folder:
  `'[' EHR.name.value "]/directory" folder_path`
  e.g. `[04959900021]/directory/root/folders[family history]`
- path to any composition (bypassing the folder directory):
  `'[' EHR.name.value "]/all_compositions" composition_path`
  e.g. `[04959900021]/all_compositions[family history]`
- path to a contribution:
  `'[' EHR.name.value "]/contributions[" contribution.uid ']'`
  e.g. `[04959900021]/contributions[uid=deepak_patel@indrisi_clinic.health.in @2000-10-03 12:34:00]`

5.2.2 DIRECTORY Class

CLASS	DIRECTORY	
Purpose	A version-controlled hierarchy of FOLDERs.	
Inherit	VERSION_REPOSITORY <FOLDER>	
Attributes	**Signature**	**Meaning**
	root: FOLDER	Root FOLDER of the directory.
Invariants	***Root_exists***: root /= Void ***Owner_id_valid***: owner_id.type.is_equal("EHR")	

5.2.3 FOLDER Class

CLASS	FOLDER
Purpose	The concept of a named folder.
CEN	FOLDER class
Synapses	RecordFolder class
Inherit	LOCATABLE

CLASS	FOLDER	
Attributes	**Signature**	**Meaning**
	folders: `List<FOLDER>`	Sub-folders of this `FOLDER`.
	compositions: `List<OBJECT_REF>`	The list of references to versioned compositions in this folder. Since more than one folder can include the same composition, this relationship is an association.
Invariants	***Folders_valid***: folders /= Void ***implies not*** folders.empty ***Compositions_valid***: compositions /= Void ***implies*** (***not*** compositions.empty ***and then*** compositions.for_all(type.is_equal("VERSIONED_COMPOSITION")))	

5.2.3.1 Folder Paths

Folder paths are built using Folder *name* attribute values, which will usually be derived from the value of the *archetype_node_id* attribute, plus a uniqueness modifier if required. The syntax is as follows:

```
'[' FOLDER.name.value "]/folders[" FOLDER.name.value "]/" ...
'/folders[" FOLDER.name.value "]/" ...
```

Example folder paths:

```
[hospital episodes]
[patient entered data]/folders[diabetes monitoring]
[homeopathy contacts]
```

Uniqueness modifiers are appended in parentheses, and only needed to differentiate folders at the same node that would otherwise have the same names, e.g.

```
[hospital episodes]
[hospital episodes(car accident Aug 1998)]
```

5.2.3.2 EHR and FOLDER Instance Structure

To Be Continued:

5.2.4 VERSIONED_COMPOSITION Class

CLASS	VERSIONED_COMPOSITION	
Purpose	Version-controlled composition abstraction, defined by inheriting `VERSION_REPOSITORY<COMPOSITION>`.	
Use		
GEHR	`G1_VERSIONED_COMPOSITION`	
Inherit	`VERSION_REPOSITORY<COMPOSITION>`	
Function	**Signature**	**Meaning**

CLASS	VERSIONED_COMPOSITION	
	is_persistent: Boolean	Indicates whether this composition set is persistent; derived from first version.
Invariants	***Archetype_node_id_valid***: all_versions.for_all (data.archetype_node_id.is_equal(all_versions.first.data.archetype_node_id)) ***Persistent_valid***: all_versions.for_all (data.is_persistent = all_versions.first.data.is_persistent) ***Owner_id_valid***: owner_id.type.is_equal("EHR")	

5.2.4.1 VERSIONED_COMPOSITION Path

The path of a VERSIONED_COMPOSITION is taken from the *name* attribute value of the contained compositions. Thus, typical paths to VERSIONED_compositions include:

```
[family history]
[current medications]
[current medications (chinese)]
[patient contact]
```

The path to any particular version within a VERSIONED_COMPOSITION is given by adding the value of the version_id of the version as a uniqueness modifier. This may be a combination like {*committer, ehr_node, time_committed*}, taken from the attributes in the AUDIT_DETAILS object attached to the relevant VERSION object, or it might be the special symbolic version identifiers "first", "latest". The patterns for the path to a version are therefore:

```
"[" name "(" committer "@" ehr_node @ time_committed ")]"
"[" name "(" first ")]"
"[" name "(" latest ")]"
```

Example paths to individual versions include:

- [current medications (latest)]
- [patient contact (sam_heard@park_rd_clinic.health.au@2002-04-26 12:34:00)]
- [test results (p_athologist@dbh.health.au)]

5.3 Historical Views of the Record

It is important to understand that the COMPOSITION versions at a previous point in time represent a previously available *informational state* of the EHR, at a particular EHR node. Such previous states include only those compositions from other sources as have been acquired by that point in time, *regardless of whether the acquired information pertains to clinical information recorded earlier.* A previous historical state of the EHR thus corresponds to what users of a system could see at a particular moment of time. It is important to differentiate this from previous *clinical* states of the patient: previous *informational* states of the EHR might include acquired information which is significantly older than the point in time when merging occurred. A previous clinical state of the patient would be a view of the EHRs in all locations for the patient - what is sometimes called the virtual EHR - at a given point in time, minus acquired Compositions, since these constitute (usually out-of-date) copies of Compositions primarily available elsewhere.

It is previous informational states with which we are concerned for medico-legal purposes, since they represent the information actually available to clinicians at a health-care facility, at a point in time. But previous clinical views may be useful for reconstructing an actual sequence of events as experienced by the patient.

6 Composition Package

6.1 Overview

FIGURE 13 illustrates the `composition` package.

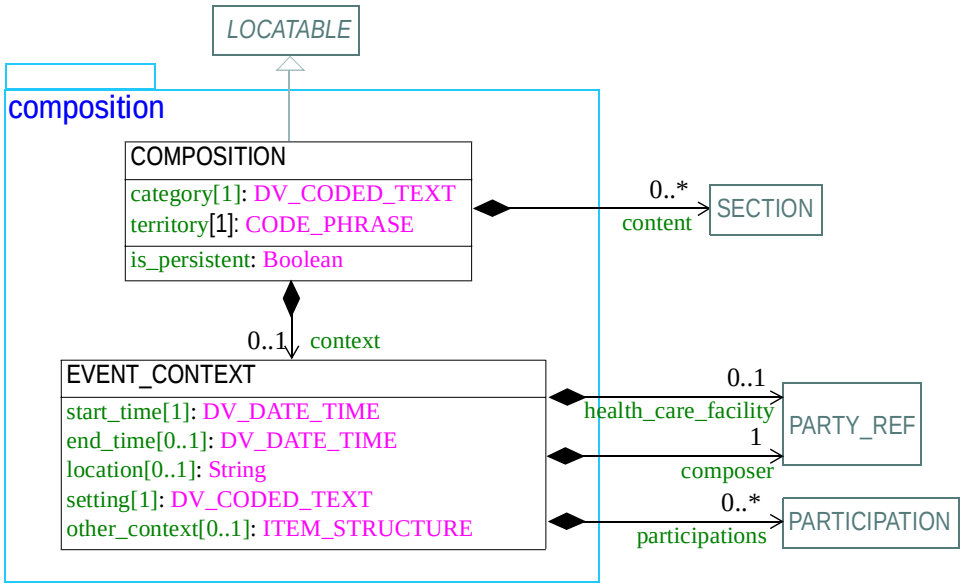

FIGURE 13 rm.composition Package

Instances of the `COMPOSITION` class can be considered as self-standing data aggregations, or documents in a document-oriented system (similar to HL7 CDA "documents"). The majority of the use of paths in *open*EHR is likely to within Compositions.

6.1.1 Event Context

The `EVENT_CONTEXT` of a `COMPOSITION` is used to record information describing the real world (usually clinical) event which gave rise to the creation or changes in Compositions in the record. An Event context is created only for Compositions to which the context information relates. Persistent Compositions do not have an event context. In general, a Contribution to the EHR consists of one or more Compositions, which were created or modified due to some activity. Within such a set, there will usually be one Composition relating directly to the event, such as the patient contact - this is the Composition containing the doctor's observations, nurses activities etc, during the visit. This Composition will include an `EVENT_CONTEXT` object documenting the time, location and participants in the event. Other Compositions changed during the same event (e.g. updates to medication list, family history and so on) do not require an event context, since they are part of the same Contribution, and the event context of the primary Composition can always be retrieved if desired.

The times recorded in the Event context object represent the time of an encounter or other activity undertaken by a health provider to/for/on behalf of the patient. The time is represented as a mandatory start time and optional end time. It is assumed that where there is a clinical session (i.e. an `EVENT_CONTEXT` object does exist), e.g. a patient encounter, the start time is known or can be reasonably approximated. It is quite common that the end time of a consultation or encounter is not recorded, but rather inferred from e.g. average consult times, or the start_time of the next consult for the same physician.

Event context is used even if the additions are made to the EHR long after the event took place, such as happens when a doctor writes his notes into the record system at night, after all patients have been seen. In such cases, the versioned Composition audit trail indicates the context of when the data were entered, as distinct from the context of when the clinical interaction took place.

In some cases, updates are made to the record where no event context is recorded, for example, if a secretary corrects an error in a Composition previously recorded for a patient visit; in such cases, the Event context of any Compositions from the original commit will remain intact and unchanged (unless the correction is to the event context itself of course), and will correctly reflect the fact that no new clinical interaction occurred. Event context will often not be used in hospital situations, since many patient 'contacts' are simply nurse's observations.

6.1.2 Participations

As part of the Event context, participations can be recorded to describe who participated, and how. Usually this will be used to record the details of patient and clinician participation. Each participation object describes the "mode" of participation as well, such as direct presence, video-conference and so on. There are no general rules about who participates. For example, while there will be a patient participation during a GP visit, there will be no such participation recorded when the clinical event is a tissue test in a laboratory. Conversely, a patient might record some observations and self drug administration in the record, in which case there will be no clinician participation. Consequently, the use of participations will mostly be archetype-driven.

A few 'participations' are predefined: *health_care_facility* and *author*. These are formally defined as follows:

health_care_facility (HCF): the health care facility under whose care the event took place. This is the most specific identifiable (by the health system) workgroup or care delivery unit within a care delivery enterprise which was involved in the care event. The identification of the HCF can be used to ensure medico-legal accountability. Often, the HCF is also where the encounter physically took place, but not in the case of patient home visits, internet contacts or emergency care; the HCF should not be thought of as a physical place, but as a care delivery management unit. The physical place of care can be separately recorded in `EVENT_CONTEXT`.*location*.

The *health_care_facility* attribute is optional to allow for cases where the clinical event did not involve any care delivery enterprise, e.g. self-care at home by the patient, emergency revival by a non-professional (e.g. CPR by lifeguard on a beach), care by a professional acting in an unofficial capacity (doctor on a plane asked to aid a passenger in difficulty). In all other cases, it is mandatory. Archetypes are used to express this.

composer: the person who was primarily responsible for the content of the Composition (not necessarily its entry into the EHR system). This is the identifier which should appear on the screen. It could be a junior doctor who did all the work, even if not legally responsible, or it could be a nurse, even if later attested by a more senior clinician; it will be the patient in the case of patient-entered data. It may or may not be the person who entered the data. it may also be a software agent. This attribute is mandatory, since all content must be been created by some person or agent.

location: the physical location where the care delivery took place, and should document a reasonably specifically identifiable location possible. Examples include "bed 5, ward E", "home". This attribute is optional, since the location is not always known, particularly in legacy data.

setting: this attribute documents the "setting" of the care event. In clinical record keeping, this has been found to be a useful coarse-grained classifier of information. The *open*EHR Terminology "setting" group is used to code this attribute. It is mandatory, on the basis that making it optional will reduce its utility for querying and classification.

6.1.3 Composition Content

The data in a Composition is stored in the *content* attribute. There are four ways the *content* attribute can be populated:

- it may be empty. Although for most situations, there should be content in a Compostion, there are at least two cases where an empty Composition makes sense:
  - the first is a Composition in 'draft' editing state (`VERSION`.*lifecycle_state* = 'draft')
  - the second is for systems that are only interested in the fact of an event having taken place, but want no details, such as so-called clinical 'event summary' systems, which might record the fact of visits to the doctor, but contain no further information. This can be achieved using Compositions with event context, and no further content.
- it may contain one or more `SECTIONs` which are defined in the archetype of the Composition;
- it may contain one or more Section trees, each of which is a separately archetyped structure;
- it may be a mixture of the last two possibilities.

6.2 Class Descriptions

6.2.1 COMPOSITION Class

CLASS	COMPOSITION	
Purpose	One version in a `VERSIONED_COMPOSITION`. A composition is considered the unit of modification of the record, the unit of transmission in record extracts, and the unit of attestation by authorising clinicians. In this latter sense, it may be considered equivalent to a signed document.	
CEN	Composition	
GEHR	`G1_COMPOSITION_VERSION`	
Synapses	Composition class	
HL7	CDA `DOCUMENT`	
Inherit	`LOCATABLE`	
Attributes	**Signature**	**Meaning**
	content: `List<SECTION>`	The clinical session content of this Composition, i.e. the information generated in the clinical session.

CLASS	COMPOSITION	
	context: EVENT_CONTEXT	The clinical session context of this Composition, i.e. the contextual attributes of the clinical session.
	category: DV_CODED_TEXT	Indicates what broad category this Composition is belogs to, e.g. “persistent” - of longitudinal validity, “event”, “process” etc.
	territory: CODE_PHRASE	Name of territory in which this Composition was written. Coded from openEHR “countries” code set, which is an expression of the ISO 3166 standard.
Functions	**Signature**	**Meaning**
	is_persistent: Boolean	True if category is a “persistent” type, False otherwise. Useful for finding Compositions in an EHR which are guaranteed to be of interest to most users.
Invariants	***is_archetype_root***: is_archetype_root ***content_valid***: content /= Void **implies not** content.is_empty ***Category_validity***: category /= Void **and then** terminology(“openehr”).codes_for_group_name(“composition category", “en”).has(category.defining_code) ***Is_persistent_validity***: is_persistent **implies** context = Void ***name_value***: not is_persistent **implies** name.value.is_equal(context.health_care_facility.as_string + context.start_time.as_string) ***territory_valid***: territory /= Void **and then** code_set(“countries”).has(territory)	

6.2.1.1 Composition Path

Like all other EHR components, the path of a COMPOSITION is derived from its *name* attribute. Because VERSIONED_COMPOSITIONs are archetyped, the runtime value of *name* is normally related to, if not the same as the value of the meaning of the *archetype_node_id* attribute (i.e. the “text” value for the relevant *archetype_node_id* code, in the local archetype ontology). The COMPOSITION runtime *name* value for both persistent and event compositions is based on the *archetype_node_id* value, although it can be any other value if required. The general model for the *name* value is:

- meaning “(” modifier “)”

Here “modifier” is an addition to the *name* attribute to ensure uniqueness. For persistent compositions such as “family history” and “current medications”, there would normally only be one instance, although there is no guarantee of this, for example when different groups of practitioners (e.g. western medicine clinicians and chinese herbalists) maintain separate instances of the major persistent compositions. The modifer will then be be the name of the clinician group. There are many other reasons why persistent compositions might be split. Typical name values include:

- “family history”

- “current medications”
- “current medications (chinese)”
- “therapeutic precautions (food and allergy)”
- “therapeutic precautions (drug)”

To Be Continued: to be reviewed: in the following, in fact there is no need for uniqueness on event composition names, since the paths will be made unique at the next level up.

For event Compositions for which there will normally be numerous instances, the name requires a different type of uniqueness modifier. One approach would be to use ordinal numbers, leading to name values of the form:

- “patient contact (1)”
- “patient contact (2)”
- ...
- “patient contact (73)”

This creates problems for merging Compositions from other records for the same patient. since independent changes might be made to more than one copy of a previously-shared Composition held on two or more EHR systems.

A better approach uses a modifier made from attributes from the `EVENT_CONTEXT` object, which all event Compositions must include. Specifically, the attributes *health_care_facility*, *time* should guaranteed uniquness within a given patient record, and constitute an appropriate uniqueness modifier. The name value for an event Composition will thus be of the form:

- meaning “(” health_care_facility “@” time “)”

Example Compositions paths are therefore:

```
[patient contact (Park Rd Clinic@1997-09-12 12:24:00)]
[test results (QML Taringa Lab@2001-01-08 15:35:00)]
```

6.2.2 EVENT_CONTEXT Class

CLASS	EVENT_CONTEXT	
Purpose	Documents the clinical context of the clinical session (or encounter). The context information recorded here are independent of the attributes recorded in the version audit, which document the “system interaction” context, i.e. the context of a user interacting with the health record system. Clinical sessions include patient contacts, and any other business activity, such as pathology investigations which take place on behalf of the patient.	
CEN	Composition class	
Synapses	Composition class	
HL7	TBD	
Attributes	**Signature**	**Meaning**

CLASS	EVENT_CONTEXT	
	health_care_facility: PARTY_REF	The health care facility under whose care the event took place. This is the most specific workgroup or delivery unit within a care delivery enterprise which has an official identifier in the health system, and can be used to ensure medico-legal accountability.
	start_time: DV_DATE_TIME	Start time of the clinical session or other kind of event during which a provider performs a service of any kind for the patient.
	end_time: DV_DATE_TIME	Optional end time of the clinical session.
	composer: PARTY_REF	The person primarily responsible for the content of the Composition (not necessarily its entry into the EHR system). This is the identifier which should appear on the screen. It may or may not be the person who entered the data.
	participations: List <PARTICIPATION>	Parties involved in the *clinical* session. These would normally include the physician(s) and often the patient (but not the latter if the clinical session is a pathology test for example).
	location: String	The actual location where the session occurred, e.g. “microbiol lab 2”, “home”, “ward A3” and so on.
	setting: DV_CODED_TEXT	The setting in which the clinical session took place. Coded using the *open*EHR Terminology, “setting” group.
	other_context: ITEM_STRUCTURE	Other optional context which will be archetyped.
Invariants	***composer_exists***: composer /= Void ***start_time_exists***: start_time /= Void ***participations_validity***: participations /= Void ***implies not*** participations.empty ***location_valid***: location /= Void ***implies not*** location.is_empty ***setting_valid***: setting /= Void ***and then*** Terminology(“openehr”).codes_for_group_name(“setting”, “en”).has(setting.defining_code)	

6.2.3 COMPOSITION Instance Structures

To Be Continued:

7 Navigation Package

7.1 Overview

The `navigation` Package defines a hierachical heading structure, in which all individual headings are considered to belong to a “tree of headings”. Each heading is an instance of the class `SECTION`, illustrated in FIGURE 14.

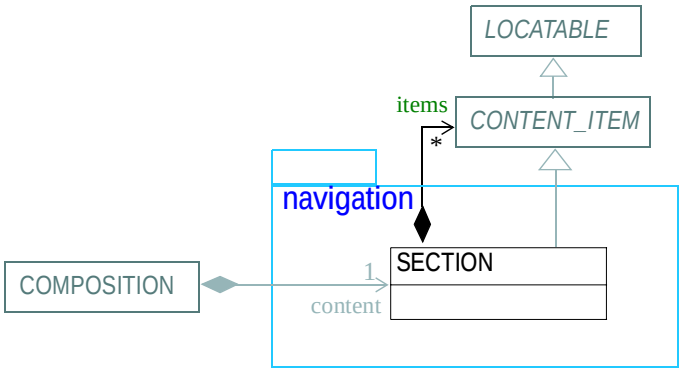

FIGURE 14 rm.composition.content.navigation Package

Sections provide both a logical structure for the author to arrange `ENTRYs`, and a navigational structure for readers of the record, whether they be human or machine. Sections are archetyped in trees with each tree containing a root Section, one or more sub-sections, and any number of Entries at each node. Section trees which are separately archetyped, such as the SOAP headings, or the heading structure for a physical examination, can be combined at runtime to form one large heading structure.

In terms of understanding of clinical data, section structures are not essential in a Composition - in theory they could be removed. They do not contain *primary instances* of meaning-modifying terms or headings. This means that if there are meaning-modifying terms, such as “history of”, “risk of” etc, such terms are not *primarily* recorded as sections. However, this does not mean that they cannot appear in sections *as well*, as would typically occur with data about family history. Thus, the section “family history” appearing in an section structure is not to be taken as the definitive indicator that any entries are about family history of the subject (rather than directly about the subject); rather, it is a guide to the kind of information to be found under it. If there are indeed entries recording family history information, the fact that the subject of that information is not the patient but a member of his/her family will be formally recorded in the `ENTRYs` themselves.

Despite the above, section structures do not have to be regarded as *ad hoc* or unreliable structures. On the contrary, as they are archetyped, their structures can be relied upon in the same way as any other structure in the record can be relied on to conform to its archetype. Accordingly, solid assumptions can be made about sections, based on their archetypes, for the purposes of querying. In fact, the main benefit of Sections is that they may provide significant performance benefits to querying, whether by interactive application or by automated systems.

One potentially confusing aspect of an section structure is that while the root section is logically a section, it would not appear in a display or printed form as a visible section, due to the fact that humans don’t usually write down top-level headings for anything, since there is always a containing structure acting as a top-level organising context (such as the piece of paper one is writing on). For example, consider the way a clinician writes down the problem/SOAP headings on paper. She writes the name of the first problem, then under that, the S/O/A/P headings, then repeats the process for fur-

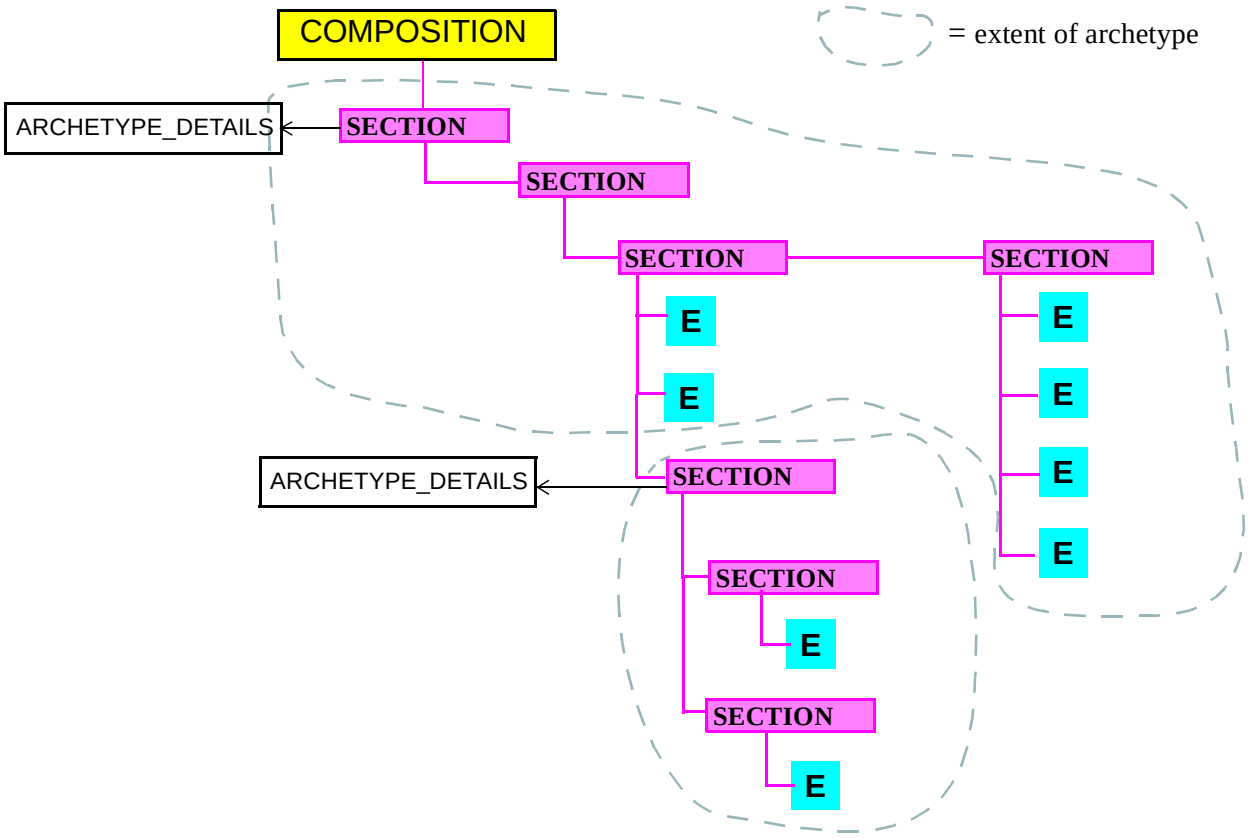

FIGURE 15 Section View of a General Practice Contact Composition

ther problems. But she doesn't write down a heading above the level of the problems, even though there must be one from a data structure point of view.

7.2 Class Descriptions

7.2.1 SECTION Class

CLASS	SECTION
Purpose	Represents a heading in a heading structure, or "section tree".
Use	Created according to archetyped structures for typical headings such as SOAP, physical examination, but also pathology result heading structures.
MisUse	Should not be used instead of `ENTRY` hierarchical structures.
CEN	Headed_section
OMG HDTF	COAS::CompositeObservation (COAS does not distinguish between the semantics of Sections and hierarchical structure inside Entries, modelled in *open*EHR by Cluster).
GEHR	`G1_ORGANISER`

CLASS	SECTION	
HL7	CDA Heading.	
Inherit	CONTENT_ITEM	
Attributes	Signature	Meaning
	items: List<CONTENT_ITEM>	Ordered list of content items under this section, which may include: • more SECTIONs • ENTRYs
Invariants	***Items_exists***: items /= Void ***implies not*** items.is_empty	

7.2.2 Section Paths

Section paths are built using the values of the *name* attribute of each SECTION in an section structure, as follows:

```
'[' SECTION.name.value "]/items[" SECTION.name.value "]/" ...
'/items[" SECTION.name.value "]/" ...
```

Examples include:

```
[SOAP headings]/items[diabetes mellitus]/items[Plan]
/items[diabetes mellitus]/items[Plan]
```

7.3 Section Instance Structures

7.3.1 Problem/SOAP Headings

An example of an section tree representing the problem/SOAP heading structure is shown in FIGURE 16.

7.3.2 Care Plans

To Be Continued: this section to be developed

General form:

- therapies & indications
- review
- monitoring
- managers (who) - participants, roles
- goals
- preventative health care (screening, education)
- triggers

To Be Continued:

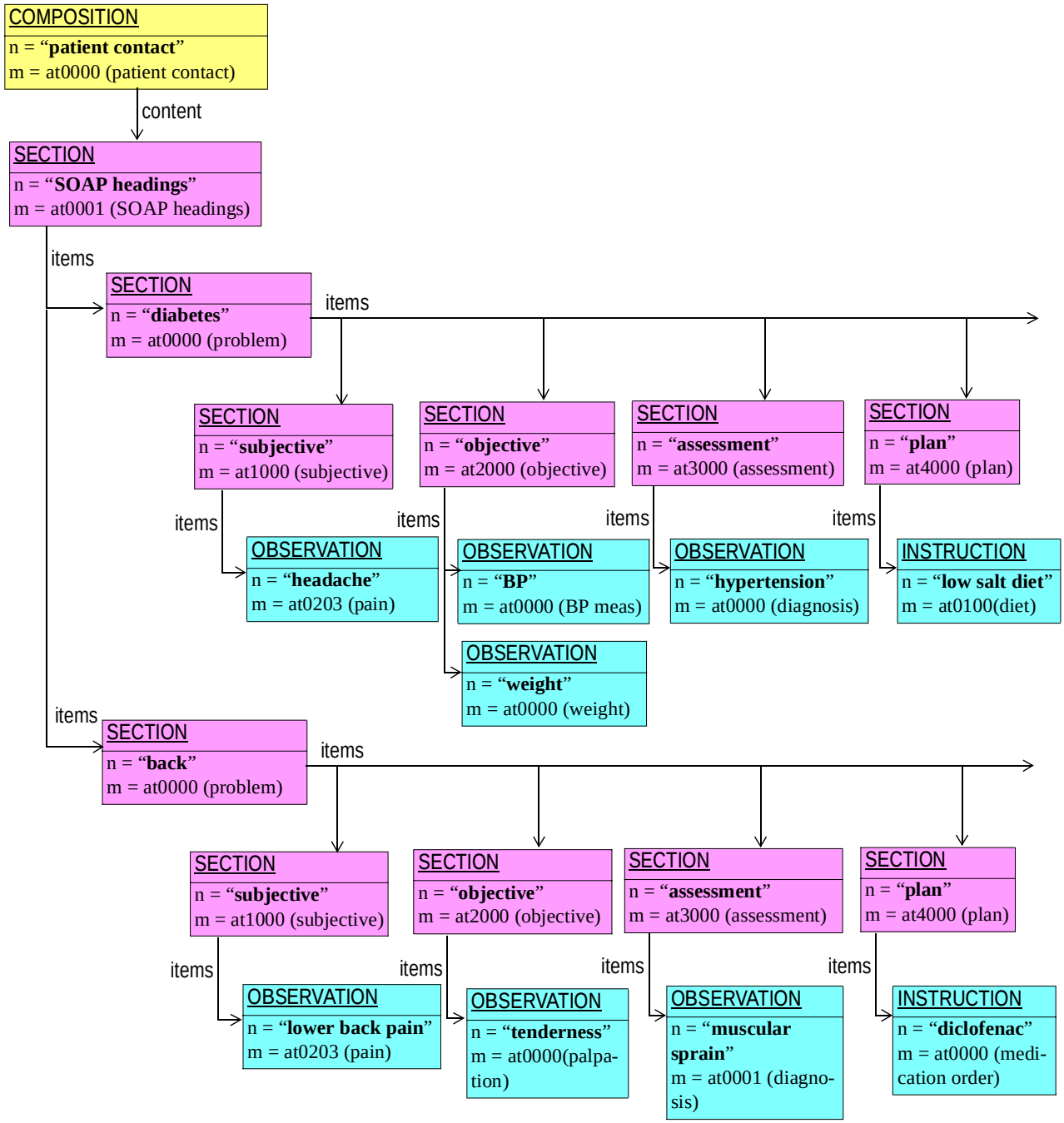

FIGURE 16 "problem/SOAP" Section Structure

8 Entry Package

8.1 Overview

All information which is created in the “clinical statement” context (see [3]) is expressed in terms of Entry instances. Entry subtypes contain instances of structure, history and action structures, and ultimately data items. Thus, when we speak of a logical “entry”, we mean the entirety of the ENTRY including all the structure below it. The ENTRY class defines the attributes common to all Entry subtypes, while specific subtypes define only attributes to that type of Entry. FIGURE 17 illustrates the entry package.

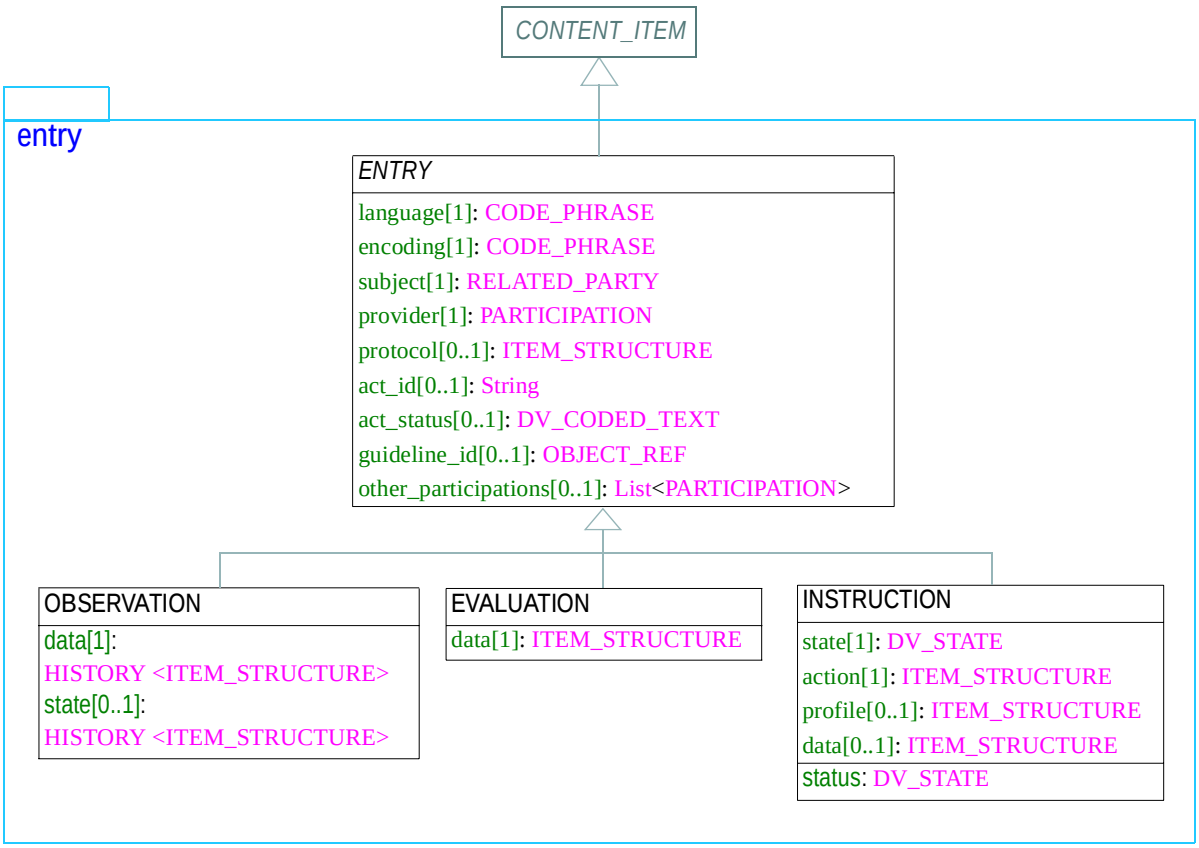

FIGURE 17 rm.composition.content.entry Package

8.1.1 Common Attributes

All Entries have a number of attributes in common. Firstly, the language and encoding attributes indicate how all text data within the Entry are to be interpreted. Normally, the language will be the same throughout the entire Entry, but in cases where it is not, the optional language attribute of DV_TEXT can be used to override the value in the enclosing ENTRY (or other enclosing structure, if a DV_TEXT is being used in some other context). Character encoding is also indicated in all Entries via the *encoding* attribute. As with language, the value is assumed for all text items within an Entry unless overridden by the *encoding* attribute within DV_TEXT.

8.1.2 Entry Subtypes

The choice of subtypes of ENTRY is based on the analysis of knowledge types described in Design Principles for the EHR [3] (which is itself mainly a synthesis of previous research such as described in Rector, Nowlan *et al* [5], and knowledge representation theory). These are as follows.

- *Observation*: statements due to observations of any phenomenon of interest in the care of the subject; typically but not restricted to clinical phenomena. For example, a pathology result, a blood pressure reading, the family history and social circumstances as told by the patient to the doctor, self-entered answers to a psychological assessment questionaire by a patient.
- *Evaluation*: statements created by the author as a result of interpretation or analysis of observations. These include hypotheses, decisions, diagnoses, plans, goals etc. Usually clinical but not necessarily so.
- *Instruction*: statements describing actions to be enacted. Instructions are detailed enough to be enactable without further details. E.g. while an evaluation may mention that "oral cortico-steroids are indicated at a peak flow of 40 l/m", an instruction is required to detail which actual drug, route, dose, frequency, and so on. Instructions may also describe non-clinical intentions such as consent.

Variations on these types are widely recognised in existing systems and standards and proposals, such as the OBSERVATION, SUBJECTIVE_OBSERVATION, INSTRUCTION types in the GEHR (Australia) model and the HL7v3 moods. The three types appear in the *open*EHR model in the form of the classes OBSERVATION, EVALUATION, and INSTRUCTION.

All types contain their data in generic structures as follows:

- OBSERVATION: since observation data is situated in time, it is expressed as a HISTORY<T>, which itself contains instances of ITEM_STRUCTURE, such as ITEM_LIST, ITEM_TABLE etc.
- EVALUATION contains its data directly in ITEM_STRUCTURE instances, since there is no need for timing information (apart from the time of authoring, which is recorded in the COMPOSITION).
- INSTRUCTION represents its data in terms of ACTION_SPECIFICATION objects, which in turn contain their data in ITEM_STRUCTURE instances.

The VIEW subtype is also defined, for expressing derived views on other data.

To Be Continued: VIEW semantics to be defined

8.1.3 Contextual Information

Contextual attributes defined on ENTRY are as follows:

subject: person or other demographic subject of the information contained in the ENTRY. This is expressed in the form of a RELATED_PARTY, which describes the kind of relationship, and optionally, identifies the demographic entity when the relationship is not "self".

provider: the person who provided the information. This is usually the patient or the clinician, but may be someone else.

other_participations: other participations which existed for this ENTRY, e.g. a nurse who administered a drug in an INSTRUCTION ENTRY; only required in cases where participants other than the subject of the information and the provider of the information need to be recorded.

Time is expressed for observations in instances of the HISTORY class inside OBSERVATION, and as time specifications within INSTRUCTIONs.

8.1.4 Workflow and Guidelines

The attributes *act_id* and *guideline_id* have been included, respectively, to enable workflow systems to retain identifiers of particular acts, and the EHR to record identifiers of guidelines which gave rise to particular Entries.

8.1.5 Relationship to HL7 Moods

ENTRYs in the *open*EHR record correspond to the moods defined in the HL7v3 RIM in the following way:

OBSERVATION: moods OBS (Observation) and EVN (Event)

EVALUATION: moods INT (Intent), PRP (Proposal), RMD (Recommendation)

INSTRUCTION: moods ORD (Order), APT (Appointment), ARQ (Appointment Request)

Moods which are not mapped 1:1 to *open*EHR types are DEF (Definition), EVN.CRT (Event criterion) and OPT (Option). These appear to be mappable to one of the ENTRY types depending on their context of use. The significance of such a mapping is that HL7v3 messages can be converted to ENTRYs in an *open*EHR EHR system.

8.1.6 Action Specifications

To Be Continued: this section to be completely replaced in release 1.0

Statements of future actions constitute a major category of information in the health record. In this section, we are concerned with “actionable” statements which are described in sufficient detail that they can be enacted, and where the executor is stated or at least obvious by implication. This is in contrast with “plans” which are statements of intent, but are not in themselves formally processable action statements.

Action specifications are created in the EHR due to decision-making processes. They may be directly entered by the clinician, or created by a decision support application, typically as the result of the evaluation of a guideline.

It is worth considering the relationship between guidelines, care plans and the EHR in order to understand what information needs to be stored in the EHR. *Guidelines* are formal, general (i.e. independent of particular patients) models for managing care, consisting of decision and action nodes, and can be represented, maintained and processed in decision support systems. Sometimes guidelines are customised for a patient into a personal *care plan* (e.g. for management of asthma). Such care plans may take the same form as a guideline (networks of decision and action nodes), or may be able to be expressed as a *guideline profile*. In either case, they would be stored in the record in their original form (e.g. a GLIF, EON, or Proforma statement) inside a DV_ENCAPSULATED object.

The evaluation of guidelines in a decision support tool results in action specifications being written into the record. These are sufficiently concrete statements (e.g. a particular drug and administration has to be chosen by the clinician) to be actionable and/or automatically processable.

Here we are concerned with modelling the action concept, i.e. any action corresponding to a single clinical concept (and consequently, a single archetype). Examples include:

- simple medication orders, e.g. an antibiotic prescription. Medication orders may include a termination condition;

- "chained" medication orders, including single medication prescriptions which have administration directions which change in time, e.g. changed dose, form, route, frequency of the drug;
- recalls and reviews, where the action is to cause a notification in the health system which will result in a patient coming in for observation;
- monitoring directions, i.e. directions to a patient or health professional to engage in an observational or data-gathering activity, such as measuring peak flow or blood sugar over a period of time;
- referrals, which are essentially a direction for transfer of care.

Each of these concepts is describable by a single archetype, and can be represented under a single `ENTRY` as illustrated in FIGURE 18.

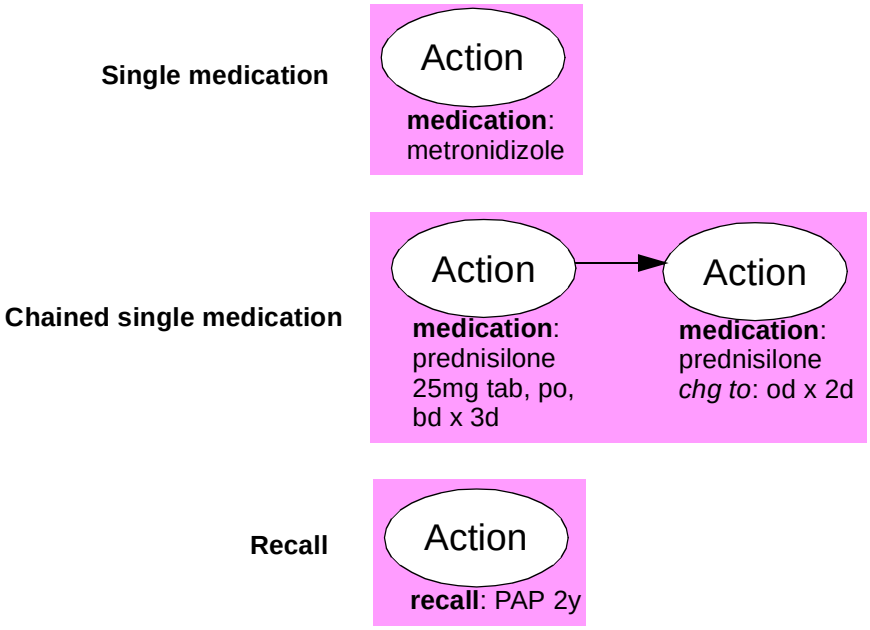

FIGURE 18 Simple Action Specifications

8.1.7 Semantics of Actions

There are two kinds of semantics which can be imputed to any statement of action: "real-world" and "system". Consider an action specification describing a medication administration for the patient. The "action" here is understood to be the real-world instruction to administer a particular drug to a patient. This kind of action specification stored in the record is on its own, passive, and must be executed by a real world actor, such as the patient or a nurse.

In contrast, a specification for a recall in the record is only useful if it specifies actions which can be executed by the "system", i.e. some application or service in the information infrastructure containing the EHR. Actual actions of this kind might be of the form `notify(recall, pat12093, "PAP due on 12/3/2001, ....)`, which can be directly processed by the system, in order to generate a real-world action, such as mailing or phoning the patient.

Ultimately, all action specifications express specific things that are intended to occur in the real world. Typically, they include conditions or criteria for starting, stopping, repeating, delaying, etc, which can be influenced by real world *events*. For instance, an action specification for commencing oral corticosteroids for asthma would be conditional on the measured peak flow falling to (say) 40 - 60% of the normal. Similarly, chemotherapy might be required to stop if a measured haemoglobin fell

below a certain threshold. A general way to understand this is that actions progress through meaningful states of execution due to events occurring. For example, medication actions often obey a state machine such as the one shown in FIGURE 19. The logical events “start”, “order”, “suspend”, “finish” etc all have real-world counterparts, which might be monitored in a hospital and made available to the application whose job it is to manage the medication action. In contrast, the state machine for most simple GP-prescribed medications would probably consist solely of the states `PROPOSED`, `COMPLETED` and `ABORTED`, since the action is completely executed by the patient, without reference to the computerised environment.

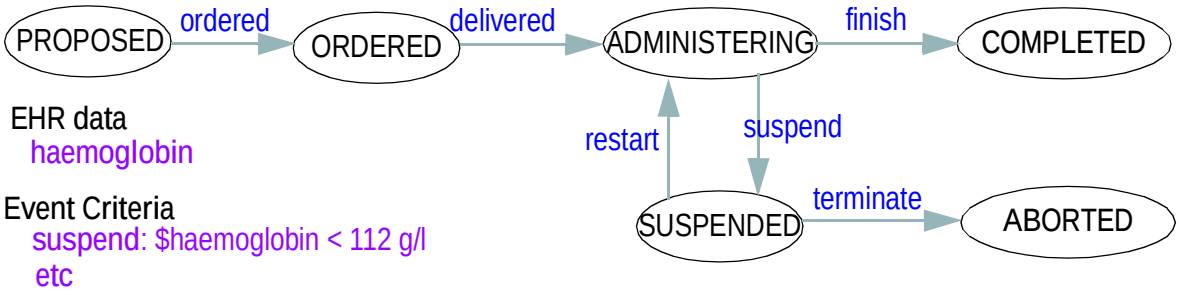

FIGURE 19 Simple State Machine for a Hospital Medication Order

In a similar way, recalls obey a state machine similar to the one shown in FIGURE 20. In this case, transition actions, or what we will call “notifications” here, are shown as well as events for some transitions. These notifications should be distinguished from the primary “action” of the specification, i.e. to cause a recall event in the real world - they are instructions to some part of the system, not to the outside world. Taken together, events and notifications constitute a two-way communication between

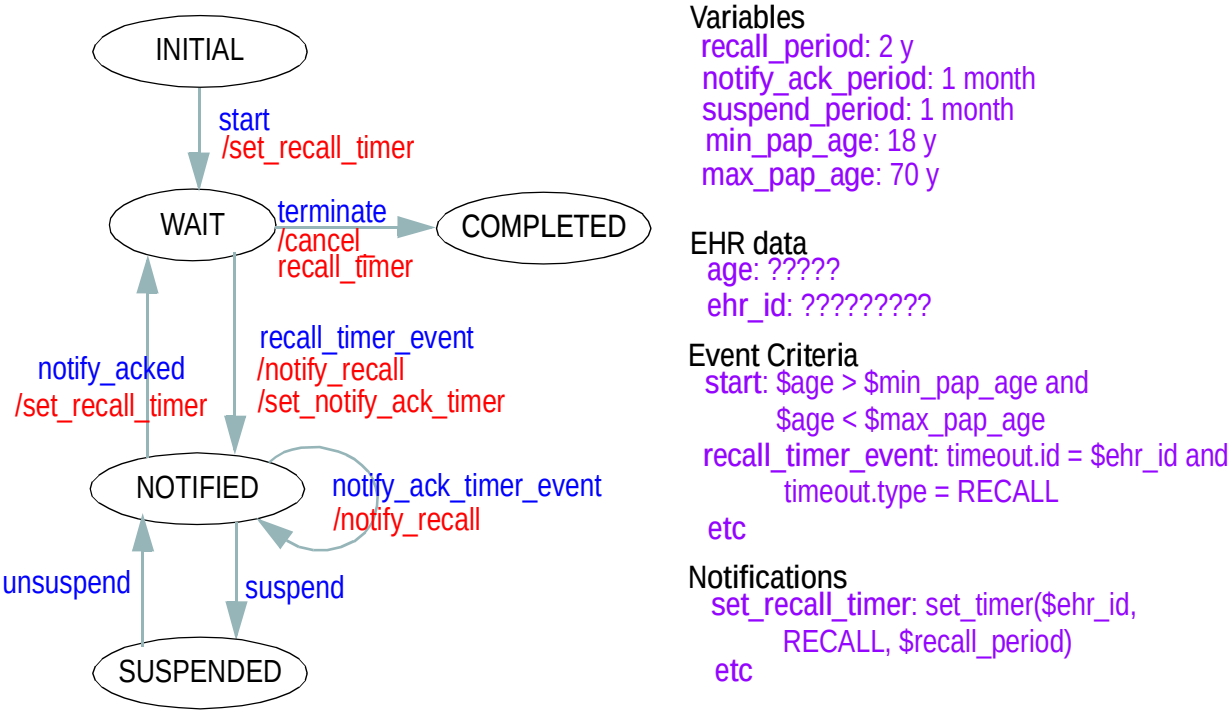

FIGURE 20 Simplified state Machine for System-based Recall Management

an action-processing application and the surrounding computing environment. Both the events and the notifications shown in the model of FIGURE 20 are logical, as might be found in a standard model of medication orders or recalls. If we consider just how one such state machine might be exe-

cuted for a particular patient, it soon becomes obvious that logical models have to be translated into computational expressions, such as those indicated under the headings “variables”, “EHR data”, “event criteria” and “notifications” in the diagrams. While the particular approach illustrated in these figures is not in any way normative, it is likely to be representative in its general form of how models of action specifications are likely to be interpreted or augmented to enable their execution in a particular system environment, for a particular patient.

We can now begin to consider the question of what needs to be modelled in the EHR information model. Clearly, the details of processing actions are not the business of the health record. System-oriented actions will be executed by other applications, or by human beings who have read the action specification in the record. Similarly, models of state machines or other ways of representing the rules of events and notifications for a particular type of action are not the business of the EHR as such: they are knowledge models, which should be expressed in a convenient form such as an archetype.

A general scheme of action-specification processing which takes account of these facts is illustrated in FIGURE 21. In the system as a whole, models of action specifications are described in a knowledge base, such as an archetype repository. Particular models are chosen by human users or applications, and used to write action specifications into the record, including “profiles” (see below); they are also used by applications to execute action specifications. During execution, events from the system environment will be received (such as timers, record writes, notifications of external events) and notifications may also be created.

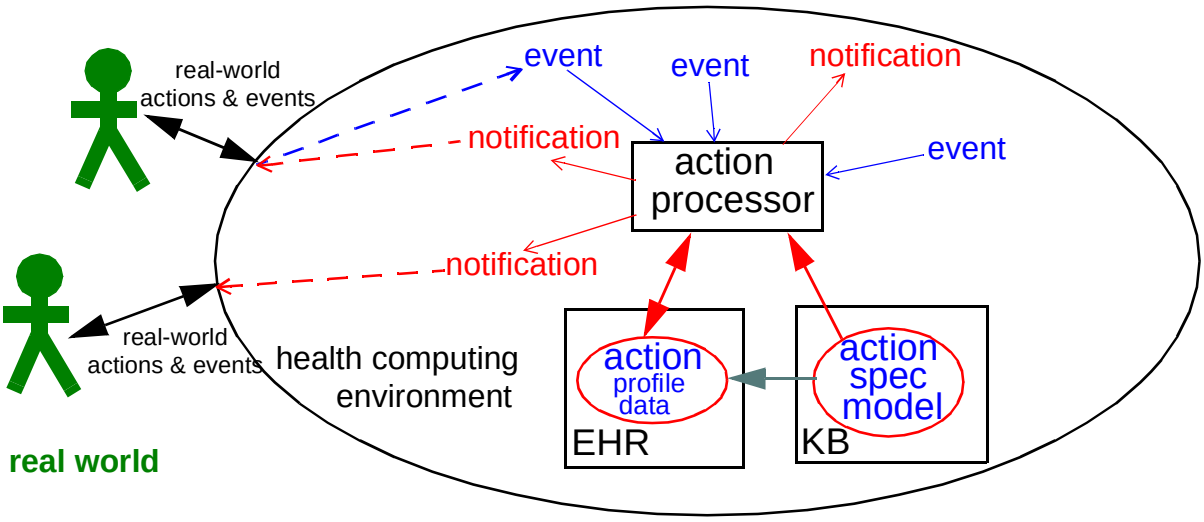

FIGURE 21 General Scheme for Processing Action Specifications

8.1.8 General Model of Action Specifications

From this scheme, and the above discussion, we can state the following conceptual model for an action specification in the EHR:

- current *state* of execution of an action specification;
- a *profile* including:
  - list of variables
  - list of EHR data items
  - list of event criteria
  - list of notifications
  - etc

- any other state data used during the execution of the action;
- references to the generating guideline where relevant;
- following actions.

Expression Syntax

In the profile category above, items such as event criteria take the form of a {name, expression} pair. While the form and syntax of such items is likely to be completely dependent on applications and/or other syntaxes, there is an argument for describing a standard default syntax. The elements of such a syntax are as follows:

- Operators:
  - boolean operators (AND, OR, NOT)
  - relational operators (<, >, <=, >=, =, <>)
  - arithmetic operators (*, /, +, -, ^, ...)
  - set operators (count, sum, average, ...)
  - functional operators (log, sin, 1/x, etc)
- operands:
  - EHR data items, such as $weight, $blood_pressure;
  - variables obtained from the outside world, i.e. from a user or machine, e.g. $pulse, $breathing, $temperature; $blood_pressure;
  - environmental variables, such as $time;
  - constants of appropriate types.
- nesting.

To Be Continued: Sensible behaviour must occur in distributed systems where copying of EHR compositions occurs, i.e. actions are enacted only in one place, or else multiple executions are explicitly understood and handled in a clinically proper way.

8.2 Class Descriptions

8.2.1 ENTRY Class

CLASS	*ENTRY (abstract)*	
Purpose	The abstract parent of all ENTRY subtypes. An ENTRY is the root of a logical item of "hard" clinical information created in the "clinical statement" context, within a clinical session. There can be numerous such contexts in a clinical session. Observations and other Entry types only ever document information captured/created in the clinical session documented by the enclosing Composition. (The effect of this is that pathology results etc, if represented in the EHR, must be expressed as Entries inside Compositions representing the session in the pathology lab, not within Compositions created during a consultation during which a clinician might read the pathology result and make some decision based on it.) An ENTRY is also the minimal unit of information any query should return, since a whole ENTRY (including subparts) records spatial structure, timing information, and contextual information, including meaning-changing modifiers as well as the subject and generator of the information.	
CEN	Cluster OCC	
OMG HDTF	COAS::HealthRecordEntry and COAS::ObservationQualifier, a generic class which is used to represent context attributes which are concretely modelled here.	
Synapses	The Item class is the closest match for Entry as described here.	
GEHR	*_CONTENT	
HL7	Act	
Inherit	CONTENT_ITEM	
Attributes	**Signature**	**Meaning**
	language: CODE_PHRASE	Mandatory indicator of the localised language in which this Entry is written. Coded from *open*EHR Code Set "languages".
	encoding: CODE_PHRASE	Name of character set in which text values in this Entry are encoded. Coded from *open*EHR Code Set "character sets".

CLASS	*ENTRY (abstract)*	
	subject: RELATED_PARTY	Id of human subject of this ENTRY; usually the patient, but may be: • organ donor • foetus • a family member • another clinically relevant person Relationship of subject of this ENTRY to the subject of the record. May be coded. If it is the patient, coded as "self".
	provider: PARTICIPATION	Id of provider of statement in this ENTRY, which might be: • the patient • a patient agent, e.g. parent, guardian • the clinician • a device or software
	protocol: ITEM_STRUCTURE	Description of *how* and/or *why* the information in this entry was arrived at. For OBSERVATIONs, this is a description of the method or instrument used. For EVALUATIONs, how the evaluation was arrived at. For INSTRUCTIONs, how to execute the instruction. This may take the form of references to guidelines, including manually folowed and executable; knowledge references such as a paper in Medline; clinical reasons within a largercare process.
	act_id: String	Optional act identifier for this Entry, used by e.g. a workflow system for an act to which this ENTRY corresponds. This identifier might have internal syntax and meaning to an external processor.
	act_status: DV_CODED_TEXT	Optional act status indicator for this entry.
	guideline_id: OBJECT_REF	Optional external identifier of guideline creating this action if relevant
	other_participations: List <PARTICIPATION>	Other participations at ENTRY level - archetypable.

CLASS	*ENTRY (abstract)*
Invariants	***Language_valid***: language /= Void ***and then*** code_set("languages").has(language) ***Encoding_valid***: encoding /= Void ***and then*** code_set("character sets").has(encoding) ***Subject_exists***: subject /= Void ***Provider_valid***: provider /= Void ***Act_status_valid***: act_status /= Void ***implies*** terminology("openehr").codes_for_group_name("act status", "en").has(act_status.defining_code) ***Other_participations_valid***: other_participations /= Void ***implies not*** other_participations.is_empty ***Archetype_root_point***: is_archetype_root

8.2.1.1 ENTRY Paths

Entry paths are based on the value of *name*, which will usually be the *archetype_node_id* attribute, and may have a uniqueness modifier, if there is more than one entry based on the same archetype, under the same `SECTION`, thus:

```
'[' meaning '(' uniqueness_modifier ')' ']'
```

Where "meaning" stands for `(ENTRY.archetype_node_id)`. Examples include:

```
[ECG results]
[blood pressure (before exercise)]
[blood pressure (after exercise)]
```

The paths to access the structures connected by the *data*, *state* and *protocol* attributes are built in the usual way, using the names of these attributes followed by the relative paths to the node of interest, i.e.:

```
'[' meaning '(' uniqueness_modifier ')' ']' '/' attribute_name '/' subpart_path
```

Examples include:

```
[ECG results]/data[history]/items[event_16]/item[ECG result]/items[lead 3]
[blood pressure measurement (after exercise)]/data[history]/items[1 min]/item[blood pressure]/items[systolic pressure]
[blood pressure measurement (after exercise)]/protocol[BP protocol]/items[position]
```

Semantics of Paths

One of the concerns of EHR data is the role and handling of 'meaning-modifying' terms, e.g. "risk of", "family history of" and so on. This problem was discussed in [3], where it was suggested that modifying terms can only sensibly be handled as part of larger semantic structures, of exactly the kind the Structure package describes. Previous efforts in this area have concentrated on finding a special place or kind of marker in the information model to indicate the presence of a modifier, mainly to guarantee that query engines do not falsely match things like "family history of coronary disease" when searching for "coronary disease" as a current problem of the subject of the record.

However it can easily be shown that the problem of meaning modification is far larger than just the prepending of one term to another. Consider the following examples from [3]:

- a blood sugar level after a 75gm oral loading vs a fasting blood sugar

- the phrase “fear of” associated with the phrase “lung cancer” vs the phrase “cause of death” associated with the same phrase.
- “total hip replacement” in the context of a “planned procedure"
- “meningitis” in the context of a “differential diagnosis”

Clearly the modification of the meaning cannot be modelled in a hard way in the information model, since meaning-modifying terms might appear anywhere in a semantic structure. Consequently, there is no special attribute or place for modifying terms in the spatial classes or elsewhere in the information model. The question remains of how querying will function properly: how will false positive matches for terms with meaning modifiers be avoided? The key to ensuring safe querying is to always use *whole paths*, not partial ones, i.e. complete paths always starting from the root of a spatial structure. It is apparent that if full paths e.g. [differential diagnosis]/items[meningitis] is used compared to [current problems]/items[meningitis] there will be no ambiguity in querying. Conversely, errors will occur if paths not commencing at the root are used. Paths in spatial and temporal structures used inside Entries are described in the Data Structures Information Model.

8.2.2 OBSERVATION Class

CLASS	OBSERVATION	
Purpose	Entry subtype for all clinical data in the past or present, i.e. which (by the time it is recorded) has already occurred. `OBSERVATION` data is expressed using the class `HISTORY<T>`, which guarantees that it is situated in time.	
Use	`OBSERVATION` is used for all notionally objective (i.e. measured in some way) observations of phenomena as well as all statements or opinions (i.e. subjective data) about things in the past.	
MisUse	Not used for future statements of any kind, including instructions, intentions, plans etc.	
CEN	Cluster	
GEHR	`G1_OBSERVATION_CONTENT, G1_SUBJECTIVE_CONTENT`	
HL7	Observation	
Inherit	`ENTRY`	
Attributes	**Signature**	**Meaning**

	data:
	`HISTORY`
	`<ITEM_STRUCTURE>`
		The data of this observation, in the form of a history of values which may be of any complexity.

	state:
	`HISTORY`
	`<ITEM_STRUCTURE>`
		The state of subject of this observation during the observation process, in the form of a history of values which may be of any complexity. Optional.

Invariants	***Data_exists***: data /= Void	

8.2.2.1 OBSERVATION Instance Structures

Periodic Series

FIGURE 22 illustrates three events over 35 seconds, with the first event 15 seconds after the origin.

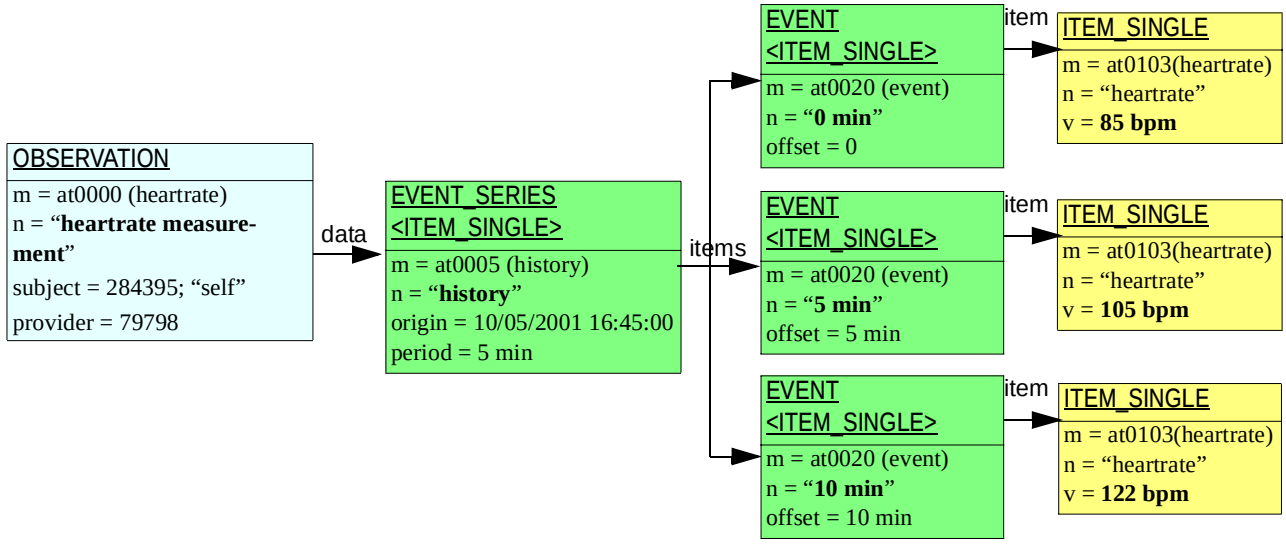

FIGURE 22 Periodic series Instance Structure

Blood Pressure with Protocol

FIGURE 23 illustrates a blood pressure observation with protocol.

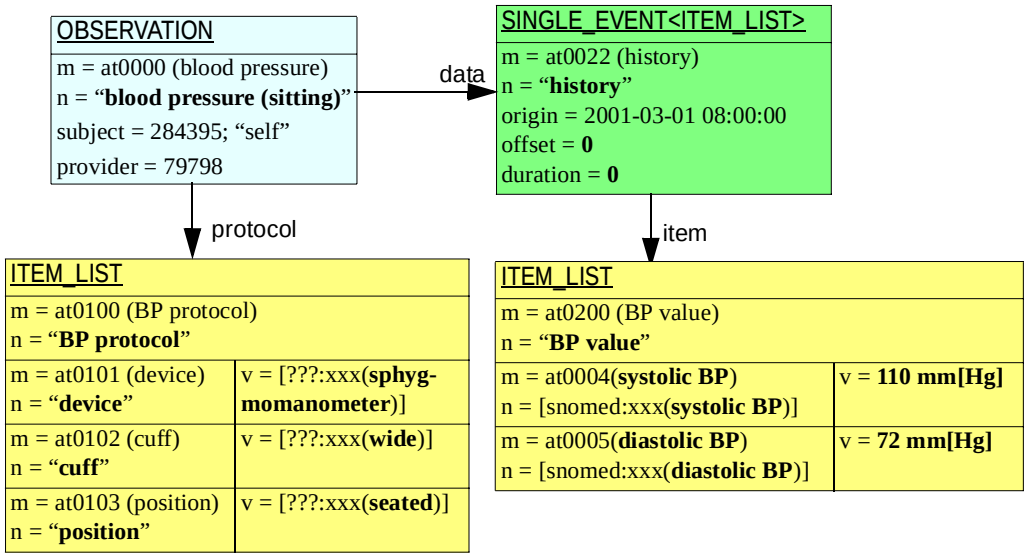

FIGURE 23 Blood Pressure Measurement Observation

Glucose Tolerance Test

An oral glucose tolerance test takes the following form, although the number and timing of the blood sugar levels may be slightly different in practice:

- challenge: no calories fasting from 12pm to 8am
- datum: BSL - 8am

- challenge: 75 g glucose orally - 8:01 am

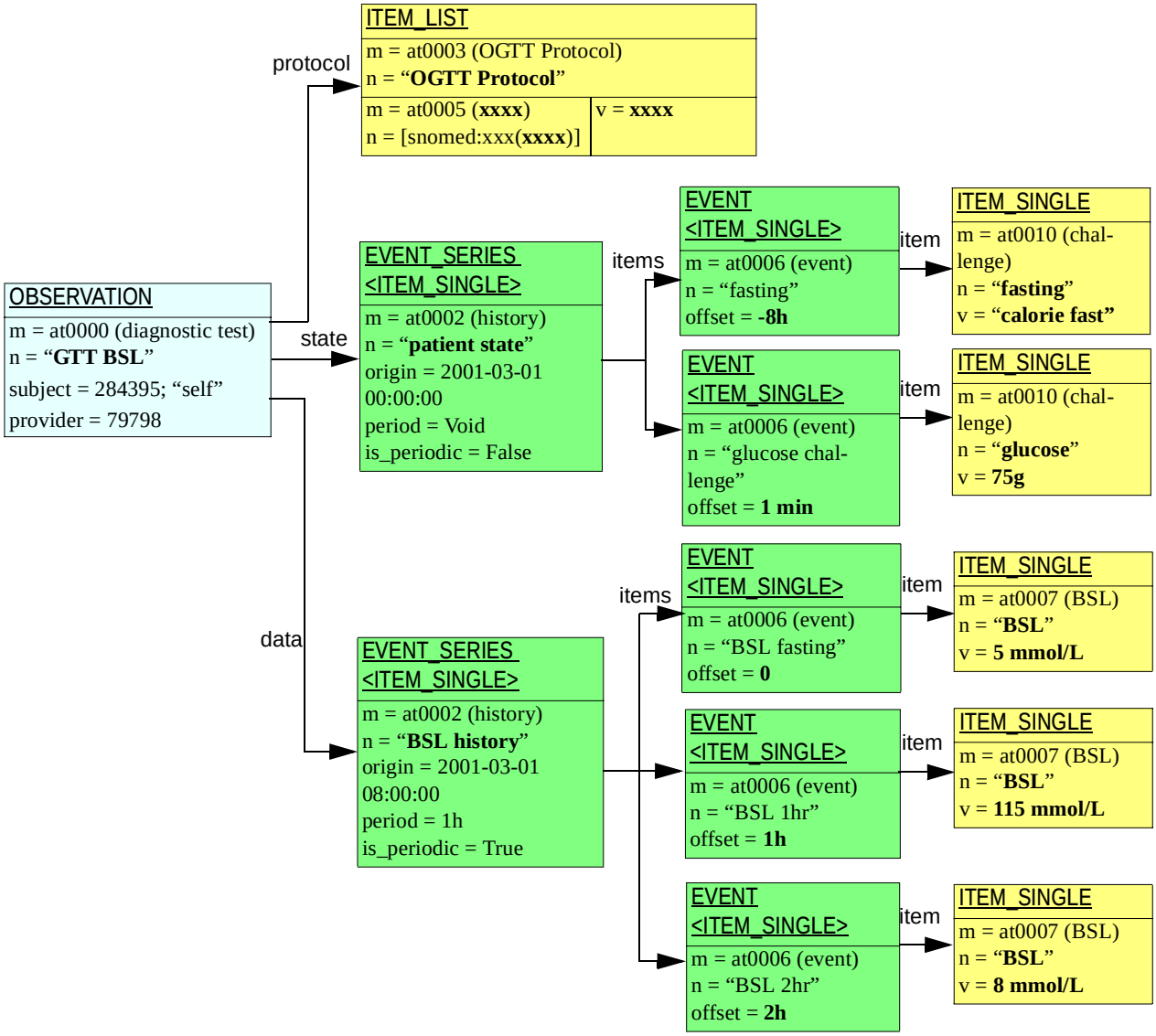

FIGURE 24 OGTT Instance Structure

- datum: BSL - 9 am
- datum: BSL - 10 am

The OGTT is treated as a single clinical concept, and thus requires only one archetype. As with any other clinical concept, there is undoubtedly more than one way to represent an OGTT. The model provides one very obvious one however, in which the two challenges, and the set of three blood sugar levels are each represented by a `HISTORY` under an `ENTRY`, as illustrated in FIGURE 24.

Clinical Observation Set

The observation periodically made by a nurse may occur every 15 mins up to once/twice day in hospital, and might be weekly in a nursing home. The following four data items are recorded:

- pulse
- BP
- temperature
- respiratory rate

To Be Continued: is this an event history? depends when it is entered into the record (might be recorded in some other system, and summarised into the EHR.

8.2.2.2 Apgar

To Be Continued:

8.2.3 EVALUATION Class

CLASS	EVALUATION	
Purpose	Entry type for evaluation statements.	
Use	Used for all kinds of statements which evaluate other information, such as interpretations of obvservations, diagnoses, differential diagnoses, hypotheses, problem assessments and plans.	
MisUse	Should not be used for actionable statements such as medication orders - these are represented with the `INSTRUCTION` type.	
CEN	Cluster	
GEHR	`G1_SUBJECTIVE_CONTENT`	
HL7	Observation	
Inherit	`ENTRY`	
Attributes	**Signature**	**Meaning**
	data: `ITEM_STRUCTURE`	The data of this evaluation, in the form of a spatial data structure.
Invariants	***Data_valid***: data /= Void	

8.2.3.1 EVALUATION Instance Structures

Differential Diagnosis (EVALUATION)

To Be Continued:

Plan

Plans are typically created for a patient based on published plans, and have a structure which usually includes the elements:

- therapies
- monitoring
- review
- prevention
- education

Each of these might be represented as EVALUATION and/or INSTRUCTION instances, depending on whether they are intended to be general statements of intention to be interpreted at a later time by a clinician, or actual executable statements, such as medication orders.

FIGURE 25 illustrates a partial asthma management plan in which monitoring (peak flow) with dependent actions (review and admission to ER) and therapy (bronchodilator) are shown. In a com-

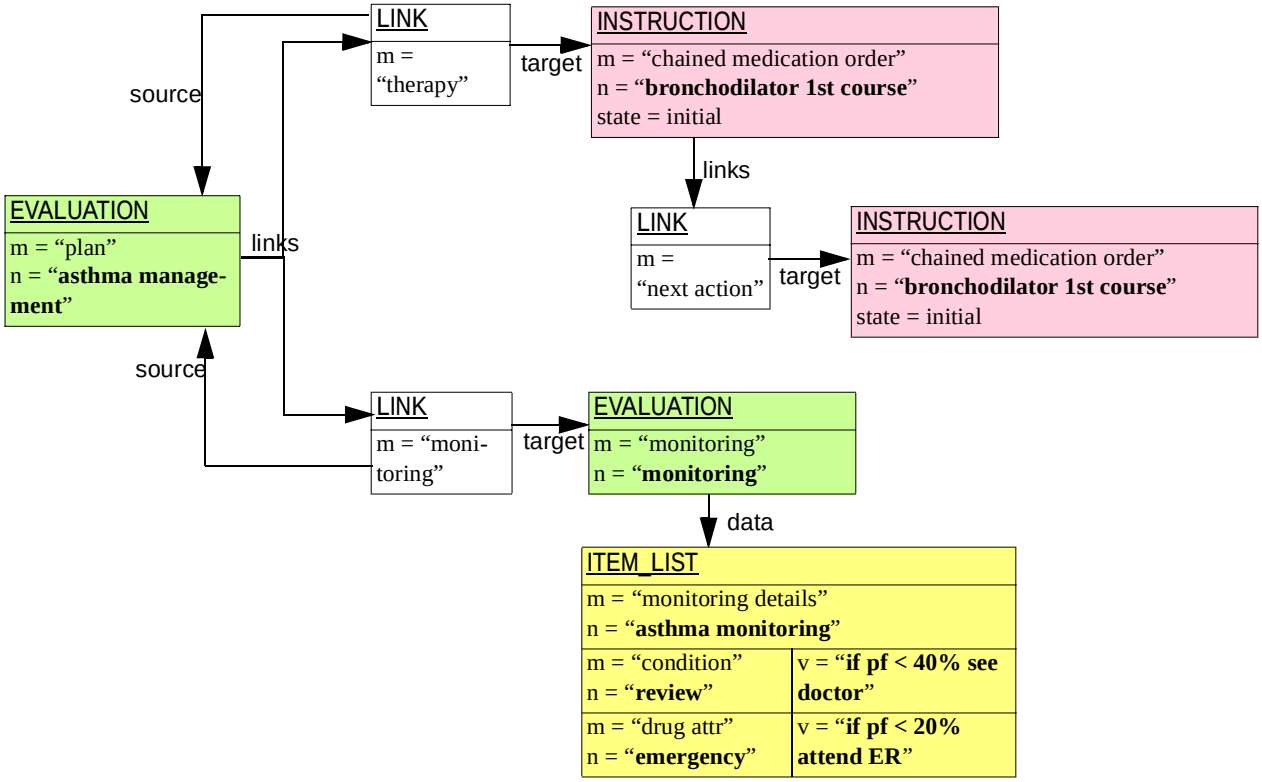

FIGURE 25 Partial Asthma Management Plan

plete plan, symptom monitoring and other medications might be shown. The parts of the plan are linked to the root EVALUATION node via the *links*: Set<LINK> attribute inherited from the LOCATABLE class.

Future experience with archetypes and plan design will show in better detail how plans should be represented.

8.2.4 INSTRUCTION Class

CLASS	INSTRUCTION
Purpose	The root of an action specification.
Use	Used for any actionable statement such as medication and therapeutic orders, monitoring, recall and review. Enough details must be provided for the specification to be executed by an actor.
MisUse	Not to be used for plan items which are only specified in general terms, e.g. "commence bronchodilator".

CLASS	INSTRUCTION	
CEN	n/a	
GEHR	G1_INSTRUCTION	
HL7	Act subtype Substance_administration, any Act type which is really an action specification (cf an Act in the past)	
Inherit	ENTRY	
Attributes	**Signature**	**Meaning**
	state: DV_STATE	current state of the action in a state machine description
	action: ITEM_STRUCTURE	description of the action to be executed.
	profile: ITEM_STRUCTURE	configuration data mappings from archetyped model of action.
	data: ITEM_STRUCTURE	state data of action execution.
Function	**Signature**	**Meaning**
	next_actions: List <INSTRUCTION>	Next actions in chain, derived from *links* attribute - any LINK instance with *name* = “next actions”.
	status: DV_STATE	Overall status, derived from the *state* values of all linked INSTRUCTIONs in the chain.
Invariants	***state_exists***: state /= Void ***action_exists***: action /= Void	

Instruction Status

The value of the *status* function of an INSTRUCTION is derived from its own *state*, and the *states* of other INSTRUCTIONs in the action specification chain, i.e. connected by LINKs whose meaning is “next action”. In the majority of cases, there is only one INSTRUCTION, and the *status* is a copy of the *state* value. If there are several INSTRUCTION objects, the value of INSTRUCTION.*status* is determined from the *state* values of the relevant INSTRUCTION objects.

It should be noted that the *status* attribute might not be of interest in some cases. In primary care, GPs or allied health practitioners prescribing standard medicines for self-administration would not normally bother with anything but the original prescription, which is effectively a proposal and order rolled into one. The patient is expected to take responsibility for the rest. This might also be the case for other routine treatements like physiotherapy or dialysis. In contrast, in secondary care, *status* is much more likely to be used, since most treatments (including antibiotics and pain relief) require nurse or machine-assisted administration.

The main use of *status* is for determining which instructions are current, overdue, executing, etc in the record. The list of medication type instructions in the “currently executing” state is equivalent to part of the current medications list.

8.2.4.1 INSTRUCTION Instance Structures

Simple Medication Order

FIGURE 26 illustrates the objects representing a medication order for Flagyl (Metronidizole) over 10 days. It consists only of an action, since there is no intent to do any system processing, there is no need for a profile or execution data, apart from the state. As with any medication, this represents only the 'normal' situation, and does not attempt to describe exceptions, which are the business of the clinician.

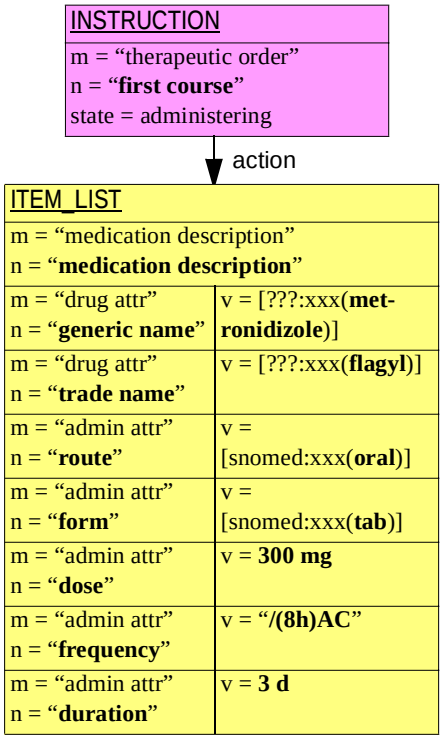

FIGURE 26 Action for a Simple Medication Order

Chained Medication Order

Often, a medication order for one drug consists of segments in which one or more of the administration details of route, form, frequency, dose etc is changed. In hospitals, intravenous antibiotics and pain relief drugs may be followed by a tablet form of the same drug to be taken orally. Other examples are common in general practice, such as the following order:

- trade name = Panafcortelone; generic name = Prednisolone; form = tablets; dose = 25mg; route = oral; freq = bd x 3 days; od x 2 days.

Bearing in mind that the details of the structure representing such an order are archetyped, there are various ways to model it; the most obvious approach is illustrated in FIGURE 27, where for each segment of the order (technically, each *separate* order) after the first, only the changes are described. It would be equally possible (controlled by archetyping) to fully describe all segments. The choice is up to archetype designers, and best practice will emerge as more implementation and clinical experience is gained.

Again, no profile or execution data apart from state are required.

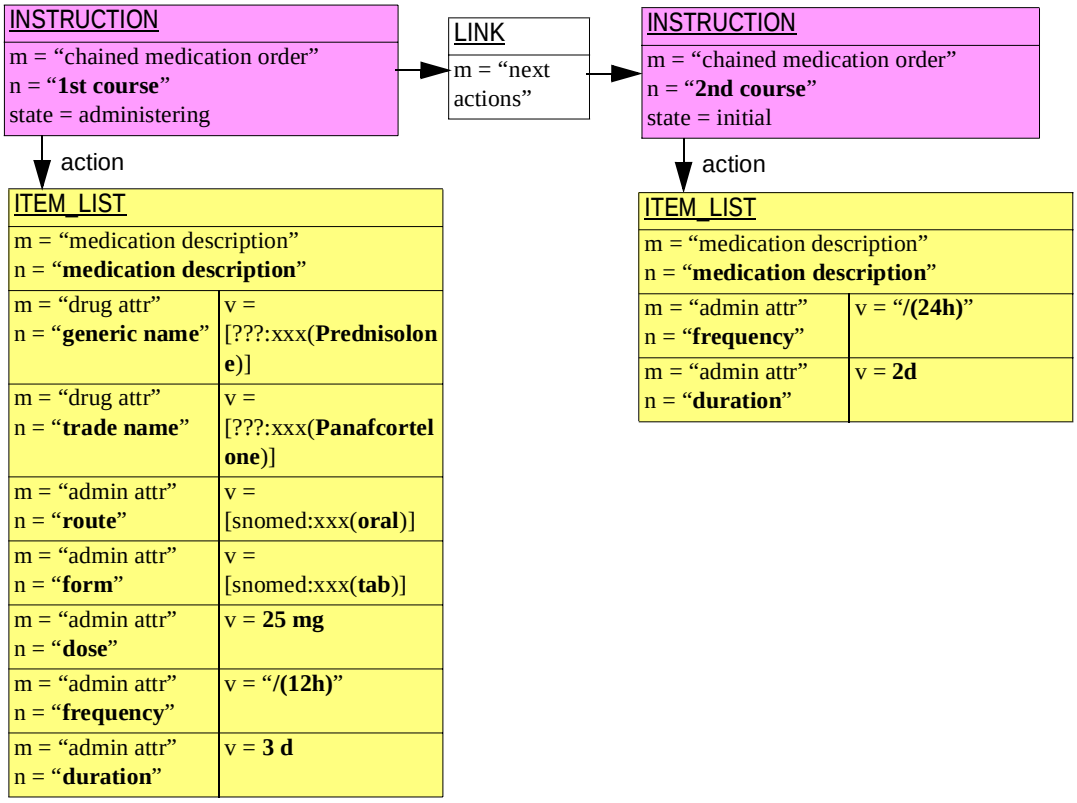

FIGURE 27 Plan for a Chained Medication Order

Recall

The general form of a recall is a communication action or appointment which needs to occur at a certain point in time. Recalls can be generated by management plans, particularly for chronic patients, e.g. opthalmology and podiatry checkups for diabetics, or by legislated or otherwise standardised public health procedures, such as Pap smear checkups, vaccination programs and so on. The main differences between recalls and other plans are that the time periods involved tend to be long.

In this example recall management is assumed to be done by the computer system, so there is a requirement for a recall profile and execution state data. FIGURE 20 shows a state machine model and profile for a recall, although in reality, the state machine and profile specification are likely to be somewhat more complex. In any case, the design of the model and profile are to be found in archetypes rather than the record. Thus, for the purposes of the example here, we simply need to give an idea of the kind of profile and data that might be created in the EHR. A Pap recall is illustrated in FIGURE 28, showing representative profile and data items.

Encapsulated Guideline

FIGURE 29 illustrates how a guideline which is expressed in an external syntax could be represented using the action class. Instead of the structure explicitly modelling the intended events as in previous examples, it provides containment for a guideline (expressed as a DV_PARSABLE), along with its execution state and state data. Storing a guideline in this way is probably only sensible for patient-specific guidelines; normally one would expect a reference to a standard guideline. However, the ability to store state data could be very useful - it assumes that the EHR system is a preferred or more reliable per-patient persistence mechanism than the guideline system (and there is no reason why a guideline

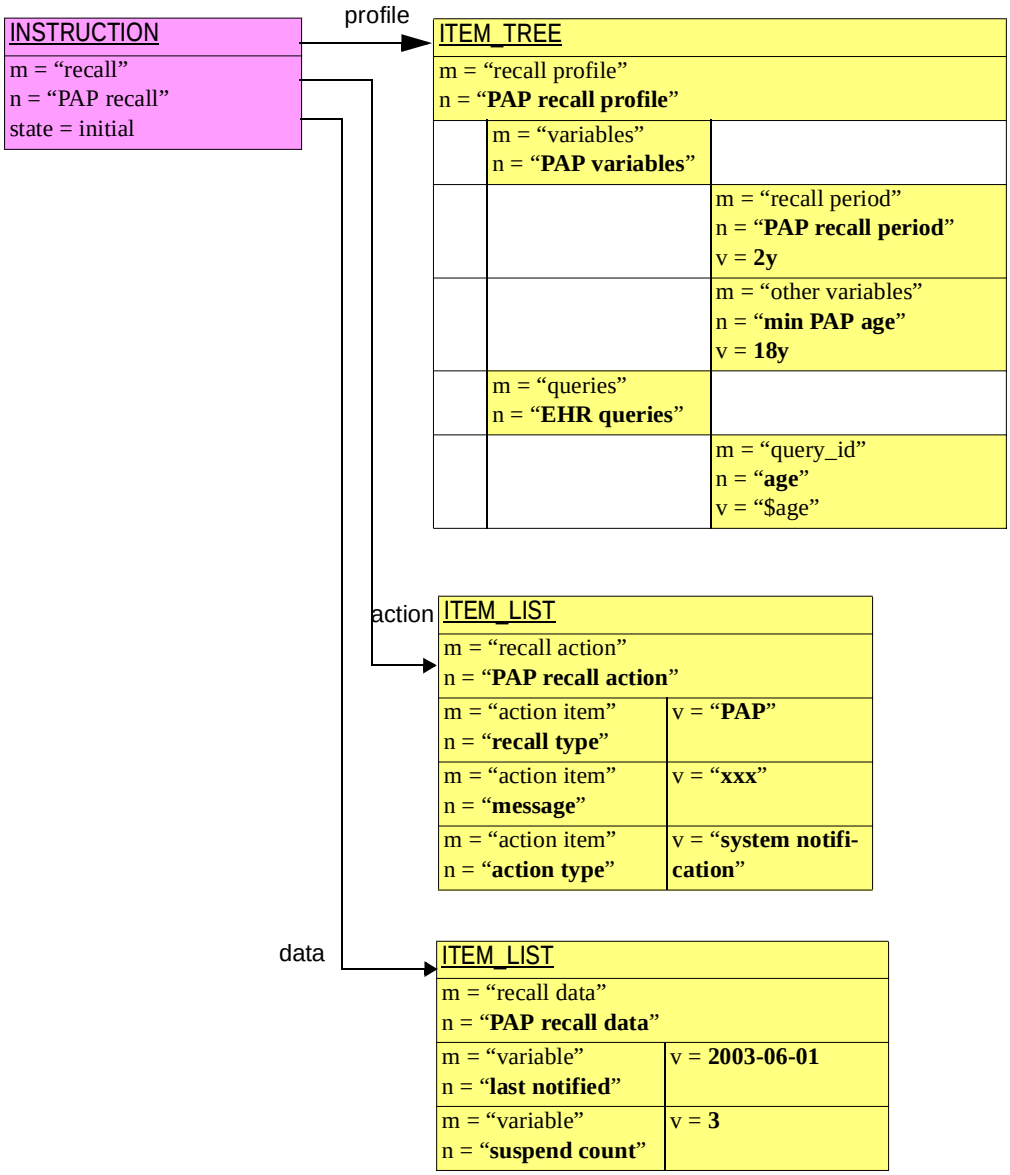

FIGURE 28 Action for a Recall

system should not be constructed on the assumption that execution state data would be stored outside itself).

8.2.4.2 INSTRUCTION Paths

Paths for plans follow the same form as for other structures, i.e. via the concatenation of relevant *name* values and attribute names. The form of the path is as follows:

- "[" `INSTRUCTION`.*name* "]/action"
- "[" `INSTRUCTION`.*name* "]/profile"
- "[" `INSTRUCTION`.*name* "]/data"

For example:

- `[therapeutic_order]/action[2nd course]/data[generic name]`

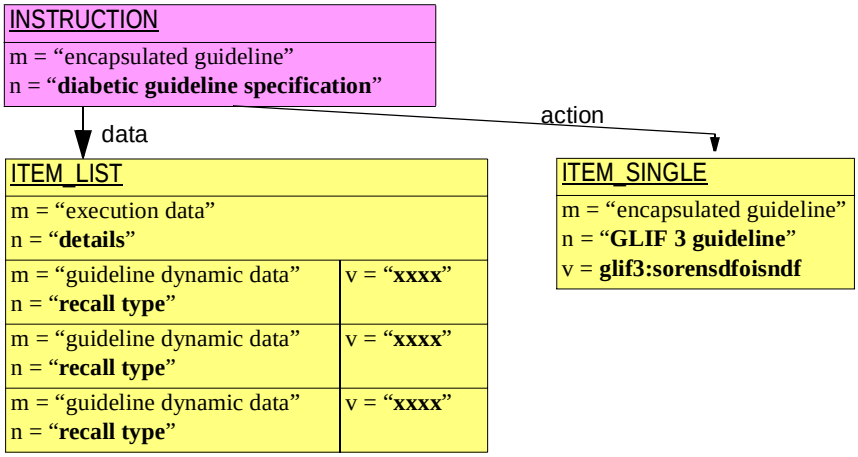

FIGURE 29 Encapsulated Guideline

A Glossary

A.1 *open*EHR Terms

HCA	Health care agent - any doctor, nurse or other recognised staff member, or software or device
HCF	Health care facility - any place where EHRs are kept
HCP	Health care professional - any doctor, nurse or other recognised staff member of an HCF

A.2 Clinical Terms

Care Pathway	A global care management strategy for a patient, showing management of health problems or issues in a time-based framework, similar to a project management view of an engineering work.
Contribution	
Episode	A series of clinical events linked in time, such as a hospital admission or a surgical episode.
Event	
Extract	
Issue	A problem as identified by the patient, e.g. “inability to do exercise due to breathing difficulty”; may be the object of wider health care, e.g. social workers, physiotherapists etc.
Section	
Problem	A health problem of the patient, as identified by its underlying medical cause, e.g. asthma; the object of medical care.
Composition	

A.3 IT Terms

.NET	
API	Application programmer’s interface - the software interface to a library or module.
COM	Microsoft’s Component Object Model; designed to enable integration of binary components obeying stated exported interfaces.
CORBA	Common Object Request Broker Architecture - an object-oriented middleware architecture enabling the construction of 3-tier systems, in which backend data providers (DBMSs etc) are known only by the services they export to the network. CORBA is an open standard managed by the Object Management Group (OMG).
DCOM	Distributed version of Microsoft COM. Similar in its aim to CORBA.
J2EE	
ODMG-93	A standard for object databases, which includes an object definition language (ODL) for writing schemas, an object query language (OQL) for querying, and several language bindings

B References

B.1 General

1 Berners-Lee T. "Universal Resource Identifiers in WWW". Available at http://www.ietf.org/rfc/rfc2396.txt. This is a World-Wide Web RFC for global identification of resources. In current use on the web, e.g. by Mosaic, Netscape and similar tools. See http://www.w3.org/Addressing for a starting point on URIs.

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

3 Beale T *et al*. *Design Principles for the EHR*. See http://www.deepthought.com.au/openEHR.

4 Gray J, Reuter A. *Transaction Processing Concepts and Techniques*. Morgan Kaufmann 1993.

5 Rector A L, Nowlan W A, Kay S. *Foundations for an Electronic Medical Record*. The IMIA Yearbook of Medical Informatics 1992 (Eds. van Bemmel J, McRay A). Stuttgart Schattauer 1994.

6 Rossi-Mori A, Consorti F. *Assembling clinical information*. Note sent to HL7 discussion list, 2001-04-19.

7 Sowa J F. *Knowledge Representation: Logical, philosophical and Computational Foundations*. 2000, Brooks/Cole, California.

8 Schadow G, McDonald C J. *The Unified Code for Units of Measure*, Version 1.4, April 27, 2000. Regenstrief Institute for Health Care, Indianapolis. See http://aurora.rg.iupui.edu/UCUM

9 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.

10 Sottile P.A., Ferrara F.M., Grimson W., Kalra D., and Scherrer J.R. The holistic healthcare information system. Toward an Electronic Health Record Europe 1999. Nov 1999; 259-266.

B.2 European Projects

11 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.

12 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.

13 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.

14 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.

15 Lloyd D, et al. *EHCR Support Action Deliverable 3.1&3.2 "Interim Report to CEN"*. July 1998. Available at http://www.chime.ucl.ac.uk/HealthI/EHCR-SupA/documents.htm.

16 *Deliverable 4: GEHR Requirements for Clinical Comprehensiveness.* GEHR Project 1992

17 *Deliverable 7: Clinical Functional Specifications.* GEHR Project 1993

18 *Deliverable 8: Ethical and legal Requirements of GEHR Architecture and Systems.* GEHR Project 1994

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

20 Grimson W. and Groth T. (Editors). The Synapses User Requirements and Functional Specification (Part B). EU Telematics Application Programme, Brussels; 1996; The Synapses Project: Deliverable USER 1.1.1b.

21 Kalra D. (Editor). The Synapses User Requirements and Functional Specification (Part A). EU Telematics Application Programme, Brussels; 1996; The Synapses Project: Deliverable USER 1.1.1a. 6 chapters, 176 pages.

22 Kalra D. (Editor). Synapses ODP Information Viewpoint. EU Telematics Application Programme, Brussels; 1998; The Synapses Project: Final Deliverable. 10 chapters, 64 pages.

B.3 CEN

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

24 ENV 13606-2 - *Electronic healthcare record communication - Part 2: Domain term list.* CEN/ TC 251 Health Informatics Technical Committee.

25 ENV 13606-3 - *Electronic healthcare record communication - Part 3: Distribution rules*. CEN/ TC 251 Health Informatics Technical Committee.

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

B.4 GEHR Australia

27 Heard S. *GEHR Project Australia, GPCG Trial.* Available at http://www.gehr.org/gpcg/ehra.htm.

28 Beale T, Heard S. *GEHR Technical Requirements*. See http://www.gehr.org/technical/requirements/gehr_requirements.html.

B.5 HL7

29 Schadow G, Russler D, Mead C, Case J, McDonald C. HL7 version 3 deliverable: *The Unified Service Action Model : Documentation for the clinical area of the HL7 Reference Information Model.* (Revision 2.4+).

30 Schadow G, Biron P. HL7 version 3 deliverable: *Version 3 Data Types.* (DRAFT Revision 1.0).

B.6 OMG

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

32 CORBAmed document: *Clinical Observations Access Service*. (Jan 2000)
3M, Care Data Systems, CareFlow/Net, HBO & C, LANL, and others.

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

B.7 Software Engineering

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

35 Walden K, Nerson J. *Seamless Object-oriented Software Architecture*.
Prentice Hall 1994

36 Gamma E, Helm R, Johnson R, Vlissides J. *Design patterns of Reusable Object-oriented Software*
Addison-Wesley 1995

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

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

39 Booch G, Rumbaugh J, Jacobsen I. *The Unified Modelling Language User Guide*. Addison esley 1999.

B.8 Resources

40 Arden Syntax. http://www.cpmc.columbia.edu/arden/

41 Asbru / The Asgaard Project. http://smi-web.stanford.edu/projects/asgaard/

42 Digital Imaging ad Communications in Medicine (DICOM). http://medical.nema.org/dicom.html.

43 EON ref required

44 GLIF (Guideline Interchange Format). http://www.glif.org/.

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

46 ProForma language for decision support. http://www.acl.icnet.uk/lab/proforma.html.

47 SynEx project, UCL. http://www.chime.ucl.ac.uk/HealthI/SynEx/.

END OF DOCUMENT