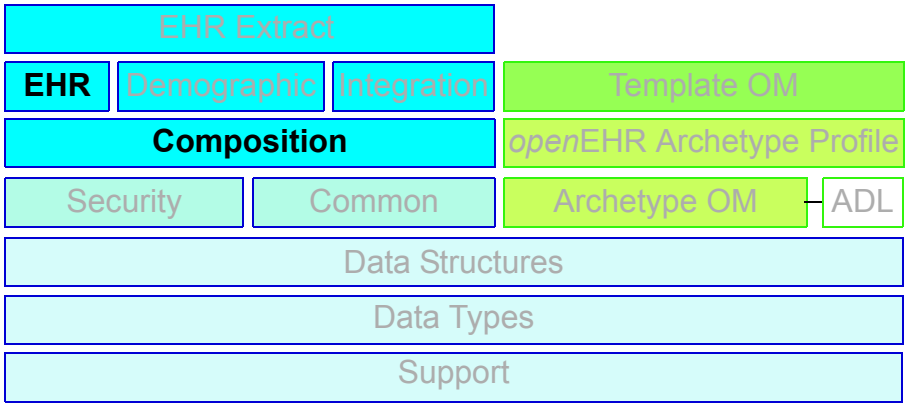
openEHR

The *open*EHR Reference Model

EHR Information Model

Issuer: openEHR Specification Program		
Revision: 5.1.1	*Pages:* 82	*Date of issue:* 16 Aug 2008
Status: STABLE		

Keywords: EHR, reference model, openehr



Support **Issue tracker**: http://www.openehr.org/issues/
Web: http://www.openEHR.org

Amendment Record

Issue	Details	Raiser	Completed
	RELEASE 1.0.2		
5.1.1	**SPEC-274**. Observation should be Evaluation in problem/SOAP structure figure. **SPEC-275**. Update Entry package design principles in EHR IM. **SPEC-253**. Clarify explanation of Instruction/Action model in EHR IM to indicate state machine per Activity	R Chen T Beale T Beale	16 Aug 2008
	RELEASE 1.0.1		
5.1.0	**CR-000200**. Correct Release 1.0 typographical errors. Correct INSTRUCTION_DETAILS.*instruction_id* type to LOCATABLE_REF. Correct inheritance of CONTENT_ITEM to LOCATABLE. **CR-000201**: Add archetype ids to Instruction ACTIVITY class. **CR-000203**: Release 1.0 explanatory text improvements. Minor changes to Entry section. Improved section on "time in the EHR". **CR-000210**: Remove LOCATABLE inheritance from ISM_TRANSITION and INSTRUCTION_DETAILS classes **CR-000130**: Correct security details in LOCATABLE and ARCHETYPED classes. Add EHR_ACCESS class. **CR-000218**: Add *language* attribute to COMPOSITION. **CR-000219**: Use constants instead of literals to refer to terminology in RM. **CR-000244**: Separate LOCATABLE path functions into PATHABLE class. **CR-000246**: Correct *open*EHR terminology rubrics.	S Heard M Forss C Ma S Heard T Beale S Heard T Beale G Grieve R Chen T Beale H Frankel B Verhees M Forss	08 Apr 2007
	RELEASE 1.0		

Issue	Details	Raiser	Completed
5.0	**CR-000014**. Adjust History. **CR-000140**. Redevelop Instruction, based on workflow principles. **CR-000147**: Make DIRECTORY Re-usable. **CR-000162**. Allow party identifiers when no demographic data. Changes to EHR, EVENT_CONTEXT, and ENTRY. **CR-000164**. Improve description of use of times in OBSERVATION. **CR-000174**. Add Admin Entry subtype. **CR-000175**. Make ENTRY.*provider* optional **CR-000177**. Make COMPOSITION.*content* a CONTENT_ITEM. **CR-000180**. Move EVENT_CONTEXT.*composer* to COMPOSITION **CR-000181**: Change ENTRY.*provider* to PARTY_PROXY. **CR-000182**: Rationalise VERSION.*lifecycle_state* and ATTESTATION.*status*. **CR-000187**: Correct modelling errors in DIRECTORY class and rename. **CR-000188**: Add *generating_type* function to ANY for use in invariants **CR-000189**. Add LOCATABLE.*parent*. New invariants in EHR and COMPOSITION. **CR-000190**. Rename VERSION_REPOSITORY to VERSIONED_OBJECT. **CR-000191**: Add EHR_STATUS class to EHR package. **CR-000194**: Correct anomalies with LOCATABLE.*uid* **CR-000195**: Rename EHR.*all_compositions* to *compositions* **CR-000161**. Support distributed versioning. Correct identifier types in EHR, ACTION classes.	S Heard S Heard T Beale R Chen S Heard T Beale S Heard H Frankel S Heard T Beale S Heard S Heard, D Kalra T Beale S Heard T Beale C Ma D Kalra T Beale T Beale S Heard T Beale H Frankel H Frankel T Beale S Heard T Beale H Frankel	25 Jan 2006
RELEASE 0.96			
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

Issue	Details	Raiser	Completed
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
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

Issue	Details	Raiser	Completed
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
3.8	Added *practice_setting* attribute to CLINICAL_CONTEXT, inspired from HL7v3/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

Issue	Details	Raiser	Completed
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 HL7v3 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
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

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 defines a logical EHR information architecture rather than just an architecture for communication of EHR extracts or documents between EHR systems. The openEHR definition of the EHR Extract 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 Architecture Overview
- 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

The status of this specification is 'stable'. It is available at http://www.openehr.org/releases/trunk/architecture/rm/ehr_im.pdf.

The latest version of this document can be found at https://github.com/openEHR/specifications-RM/blob/master/publishing/ehr_im.pdf.

New versions are announced on the openehr-announce mailing list.

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. Feedback should be provided either on the technical mailing list, or on the specifications issue tracker.

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; [20]), 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;

The original deliverables can be reviewed in detail at the GEHR page at CHIME, UCL.

2.1.2 GEHR Australia Requirements

The GEHR Australia project (1997-2001; [33], [34]) introduced further requirements, including:

- support for clinical data structures: lists, tables, time-series etc;
- safer information model than the original (European) GEHR: context attributes only in valid places (but still similar style);
- separate groups for “persistent”, “demographic” and “event” information in EHR, which corresponds closely to real clinical querying patterns;
- introduction of formally specified archetypes, and archetype-enabled information model;
- 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.

GEHR Australia produced a proof of concept implementation in which clinical archetypes were developed and used. See [3] for a 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; [26]) and SynEx (1998-2000; [14]) 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) [27];

- 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) [28];
- the core technical requirements of and interfaces for a federation middleware service [26].

2.1.4 European EHCR Support Action Requirements

This EU Support Action project ("SupA"; [18], [19], [21]) consolidated the requirements published by a wide range of European projects and national health informatics organisations as a Consolidated List of Requirements [17].

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 authored by ISO Technical Committee 215 (Health Informatics) - ISO TS 18308. The present draft [13] 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 generated during the development of *open*EHR included the following:

- major improvements in information models from the point of view of reducing progammer errors and ambiguity;
- better modelling of time and context (temporal/spatial approach);
- better understanding of legacy system / federation problem;
- workflow modelling;
- harmonise with CEN EN 13606;
- integration with HL7v2 and other messaging systems;
- numerous specific clinical requirements.

2.2 Relationship to other Health Information Models

Where relevant, the correspondences to other information models have been documented in this specification. 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. The following sections summarise the other models and standards with which correspondences are shown.

2.2.1 CEN TC/251 prEN13606

These models have been influenced by and have also influenced the models in CEN prEN13606 (2005 revision); accordingly, relationships to 13606 have been documented fairly precisely.

Since January 2002, the prEN13606 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).

Implementation experience with Release 0.9 and 0.95 of *open*EHR has further improved these areas significantly. Nevertheless, *open*EHR is not a copy of CEN, for two reasons. Firstly, its scope includes systems, while EN13606 defines an EHR Extract; secondly, EN13606 suffers somewhat from "design by committee", and has no formal validation mechanism for its models.

2.2.2 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 respects: 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 a combination of "analysis patterns" in the sense of Fowler [44] from which further specific models - subschemas - are developed by a custom process of "refinement by 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 some areas that appear to be candidates for mapping, specifically the data types and terminology use, and the correspondence between *open*EHR Compositions and parts of the HL7 Clinical Document Architecture (CDA).

2.2.3 OMG HDTF

In general, the *open*EHR information models represent a more recent analysis of the required semantics of EHR and related information than can be found in the information viewpoint of the OMG HDTF specifications (particularly PIDS and COAS). However, the computational viewpoint (i.e. functional service interface definitions) is one of the inputs to the *open*EHR srevice model develpment activity.

3 The EHR Information Model

3.1 Overview

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

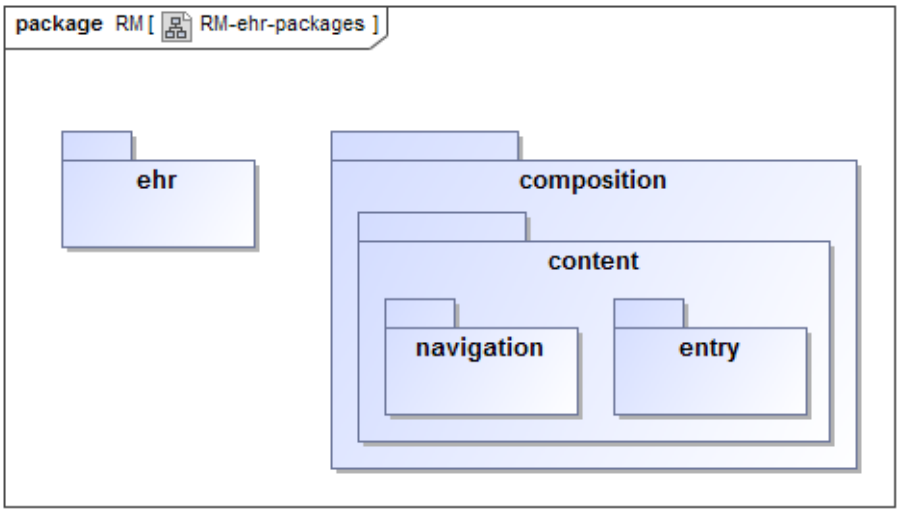


FIGURE 1 ehr Package Structure

The packages are as follows:

`ehr`: This package contains the top level structure, the `EHR`, which consists of an `EHR_ACCESS` object, an `EHR_STATUS` object, versioned data containers in the form of `VERSIONED_COMPOSITIONs`, optionally indexed by a hierarchical directory of `FOLDERs`. A collection of `CONTRIBUTIONs` which document the changes to the EHR over time is also included.

`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. `ENTRYs` and other `SECTIONs` can appear under `SECTIONs`.
- `entry`: This package contains the generic structures for recording clinical statements. Entry types include `ADMIN_ENTRY`, `OBSERVATION` (for all observed phenomena, including mechanically or manually measured, and responses in interview), `EVALUATION` (for assessments, diagnoses, plans), `INSTRUCTION` (actionable statements such as medication orders, recalls, monitoring, reviews), and `ACTION` (information recorded as a result of performing Instructions).

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.

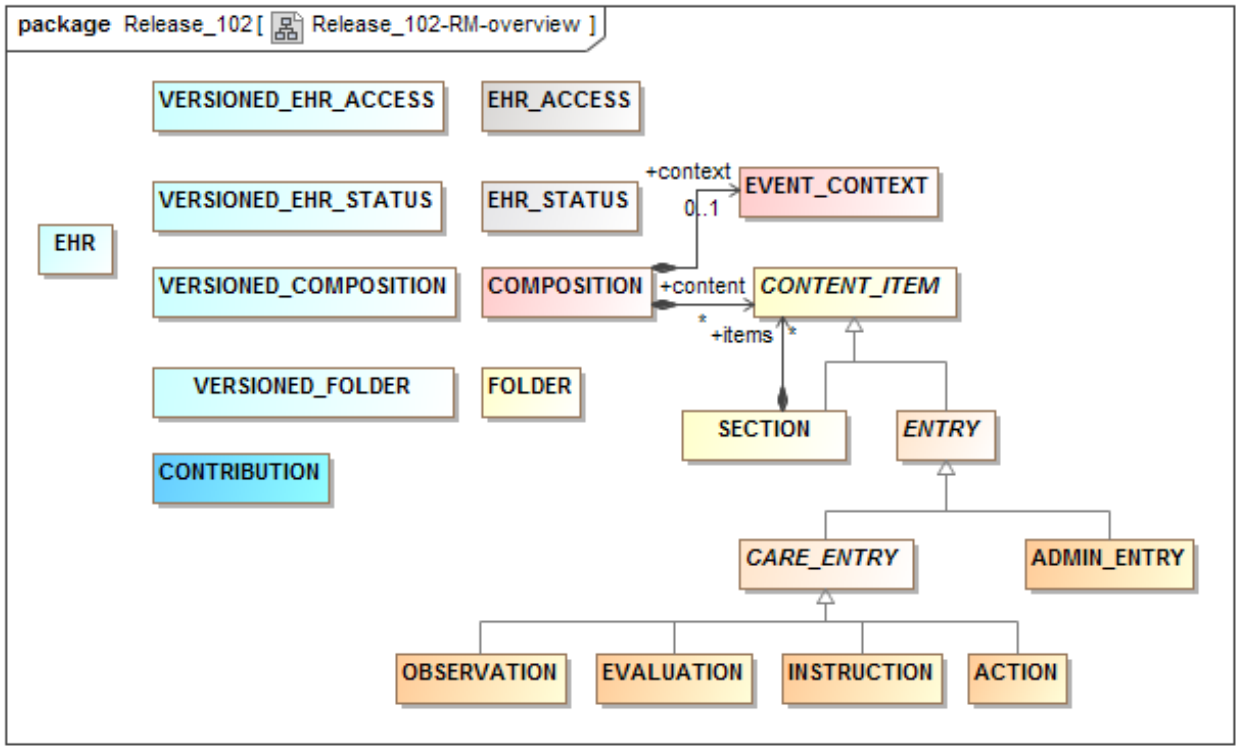


FIGURE 2 *open*EHR EHR Information Model Overview

4 EHR Package

4.1 Overview

The *open*EHR EHR is structured according to a relatively simple model. A central EHR object identified by an EHR id specifies references to a number of types of structured, versioned information, plus a list of Contribution objects that act as audits of change-sets made to the EHR. The high-level structure of the *open*EHR EHR is shown in FIGURE 3.

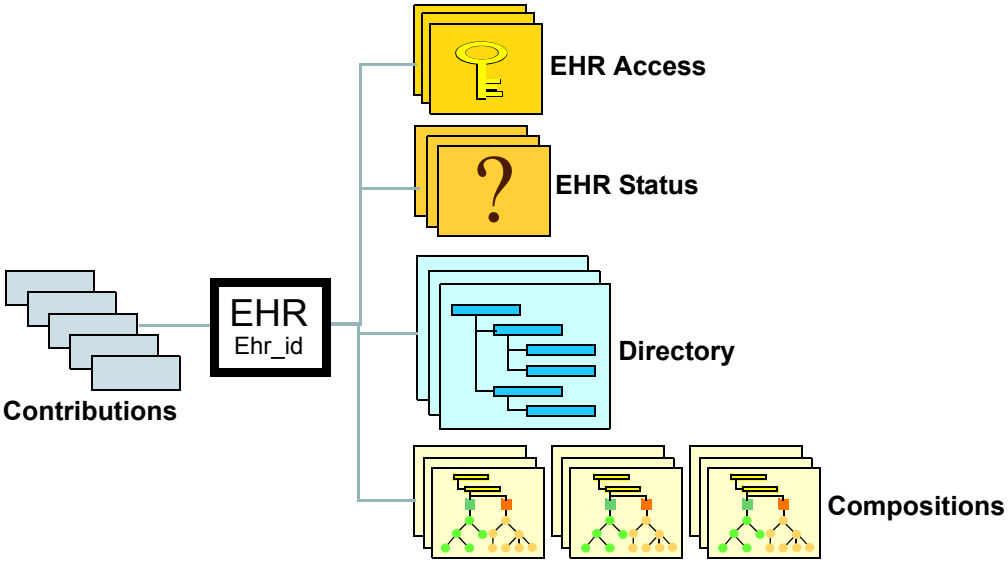


FIGURE 3 High-level Structure of the *open*EHR EHR

In this figure, the parts of the EHR are as follows:

- *EHR*: the root object, identified by a globally unique EHR identifier;
- *EHR_access (versioned)*: an object containing access control settings for the record;
- *EHR_status (versioned)*: an object containing various status and control information, optionally including the identifier of the subject (i.e. patient) currently associated with the record;
- *Directory (versioned)*: an optional hierarchical structure of Folders that can be used to logically organise Compositions;
- *Compositions (versioned)*: the containers of all clinical and administrative content of the record;
- *Contributions*: the change-set records for every change made to the health record; each Contribution references a set of one or more Versions of any of the versioned items in the record that were committed or attested together by a user to an EHR system.

The `ehr` package is illustrated in FIGURE 4. The classes have a more or less one-to-one correspondence with the objects shown in FIGURE 3. Each versioned object of type `XXX` is defined by a class `VERSIONED_XXX`, which is a binding of the type `XXX` to the generic type paremeter T in the generic type `VERSIONED_COMPOSITION` (while such bindings do not strictly require classes of their own, they facilitate implementation in languages lacking genericity).

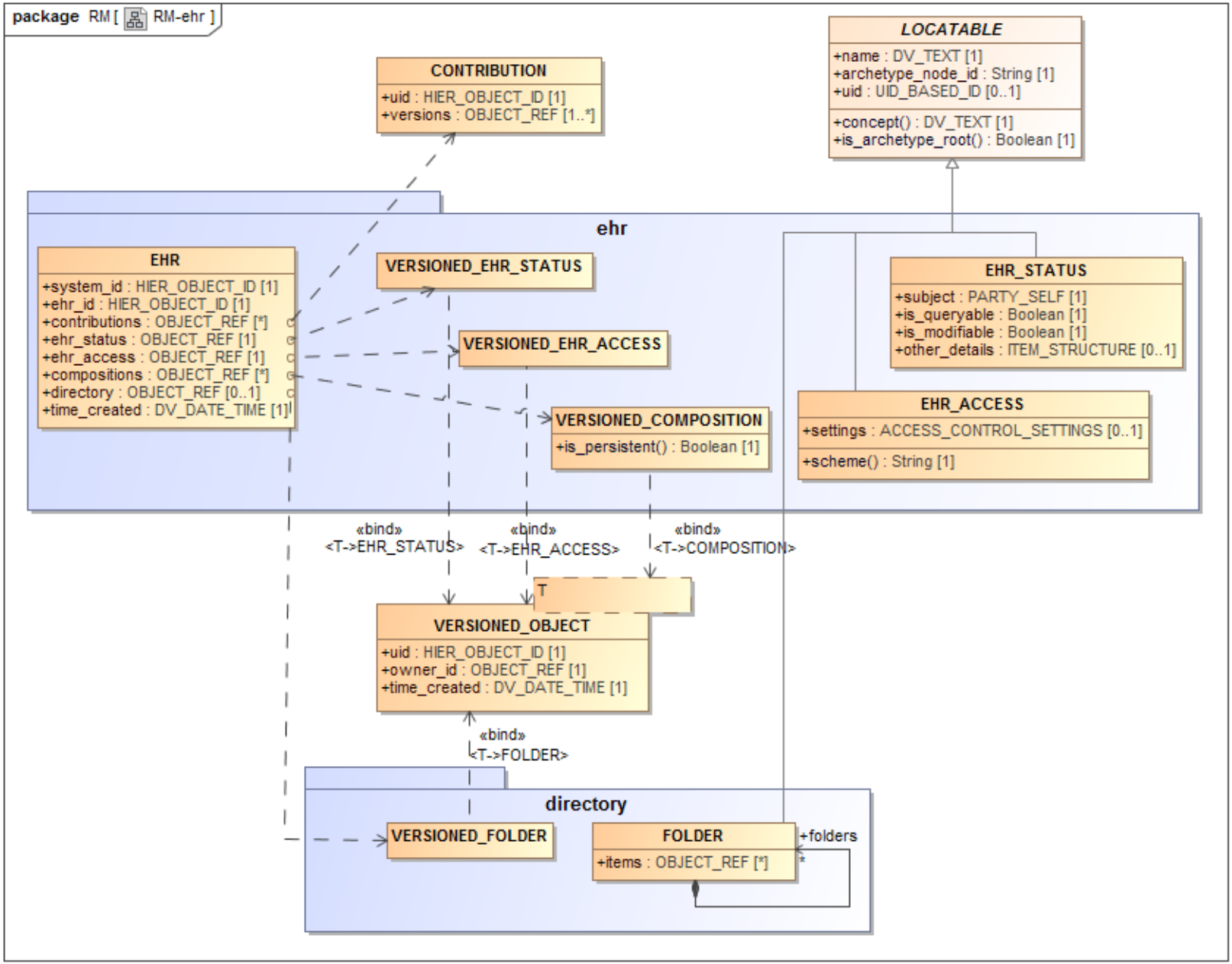


FIGURE 4 rm.ehr Package

4.2 The Parts of the EHR

4.2.1 Root EHR Object

The root EHR object records three pieces of information that are immutable after creation: the identifier of the system in which the EHR was created, the identifier of the EHR (distinct from any identifier for the subject of care), and the time of creation of the EHR. Otherwise, it simply acts as an access point for the component parts of the EHR.

References rather than containment by value are used for the relationship between the EHR and VERSIONED_XXX objects, reflecting the vast majority of retrieval scenarios in which only selected (usually recent) items are needed. Containment by value would lead to systems which retrieved all VERSIONED_XXX objects every time the EHR object was accessed.

4.2.2 EHR Access

Access control settings for the whole EHR are specified in the EHR_ACCESS object. This includes default privacy policy, lists of identified accessors (individuals and groups) and exceptions to the default policies, each one identifying a particular Composition in the EHR. All changes to the EHR Access object are versioned via the normal mechanism, ensuring that the visible view of the EHR at any previous point in time is reconstructable.

Because security models for health information are still in their infancy, the *open*EHR model adopts a completely flexible approach. Only two hard-wired attributes are defined in the EHR_ACCESS class. The first is the name of the security scheme currently in use, while the second (*settings* attribute) is an object containing the access settings for the EHR according to that particular scheme. Each scheme is defined by an instance of a subclass of the abstract class ACCESS_CONTROL_SETTINGS, defined in the Security Information Model.

4.2.3 EHR Status

The EHR_STATUS object contains a small number of hard-wired attributes, and an archetyped ‘other_details’ part. The former are used to indicate who is (currently understood to be) the subject of the record, and whether the EHR is actively in use, inactive, and whether it should be considered queryable. As for elsewhere in the *open*EHR EHR, the subject is represented by a PARTY_SELF object, enabling it to be made completely anonymous, or alternatively to include a patient identifier. The subject is included in the EHR Status object because it can change, due to errors being discovered in the allocation of records to patients. If the anonymous form is used, the change will be made elsewhere in a cross-reference table; otherwise the EHR Status object will be updated. Because it is versioned, all such changes are audit-trailed, and the change history can be reconstructed.

Other EHR-wide meta-data may be recorded in the archetyped part of this object, including runtime 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.

4.2.4 Compositions

The main data of the EHR is found in its Compositions. The Composition concept in the *open*EHR EHR originated from the Transaction concept of the GEHR project [22], [23], [24], [25], 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 characteristics of transactions [8]):

- *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 was later been renamed to “Composition”, which is the name of the equivalent concept in the current CEN EN13606, 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 two general categories of information at the coarse level which are found in an EHR: *event* items, and longitudinal, or *persistent* items, of which there are various kinds.

Event Compositions

Events record what happens during 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). 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 healthcare event, such as observations on the patient, but also to record the event context information, i.e. the who, when, where and why of the event. For this reason, a specific class representing clinical context is associated with event compositions in the formal model.

Persistent Compositions

In a more sophisticated EHR, there is also a need to record items of long-term interest in the record. These are often separated by clinicians into well-known categories, such as:

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

Persistent Compositions can be thought of as *proxies* for the state or situation of the patient - together they provide a picture of the patient at a point in time. For example, the meaning of the "medication list" Composition is always: this is the list of medications currently being taken by patient X. Similarly for the other persistent Composition types given above. In scientific philosophy, the kind of information recorded in a persistent Composition is known as a *continuant* (see e.g. the KR ontology in Sowa [12]). This is in contrast with event Compositions, which do not generally record continuants, but instead record *occurrents*, i.e. things that were true or did happen but have no longevity.

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.

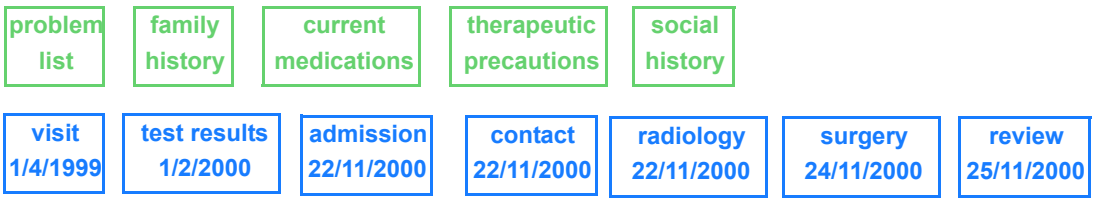


FIGURE 6 An EHR containing Event and Persistent Compositions

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 in the EHR on page 24.

4.2.5 Directory

As Compositions accumulate over time in the EHR, they form a long-term patient history. An optional directory structure can be used in the EHR to organise Compositions using a hierarchy of Folders in much the same way files in a file system are visualised by the directory Explorer tool on Windows and other platforms. In the *open*EHR model, Folders do not contain Compositions by value but by reference. 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 can 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;

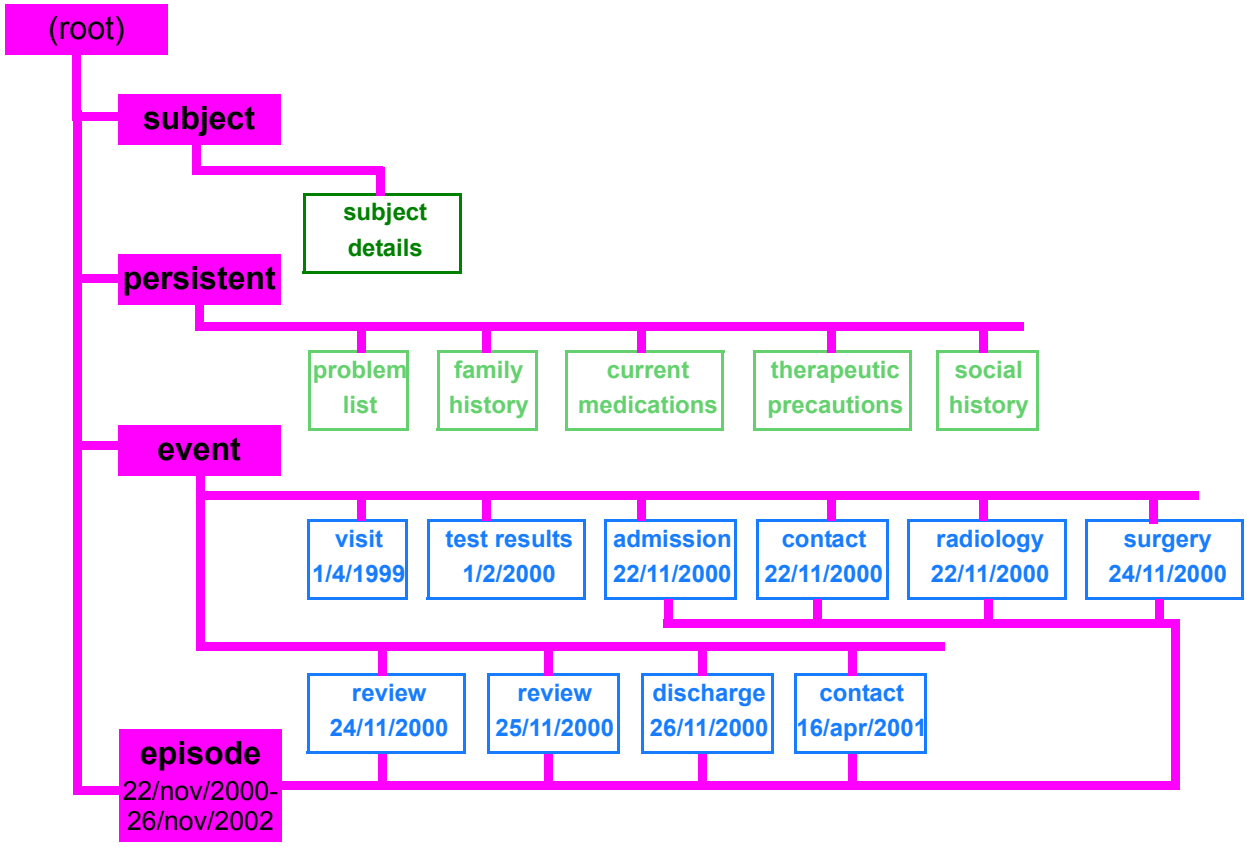


FIGURE 7 Using Folders to Group Compositions

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 reference model: 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 EHR directory is maintained in its own versioned object, ensuring that changes to the classification structure of the Compositions are versioned over time in the same way as changes to the EHR content.

4.3 Change Control in the EHR

Given an EHR in which there is an EHR Access object, EHR Status object, Event and Persistent Compositions, and possibly a directory structure, the general model of update of the EHR is that any of these might be created and/or modified during an update. The simplest, most common case is the creation of a single contact Composition. Another 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 structure. 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. Less often, updates will be made to the EHR Access and EHR Status objects, according to the management and access control needs of the patient and health care providers.

In general, the following requirements must always be met, regardless of the particular changes made at any one time:

- the record should always be in a consistent informational state;
- all changes to the record be audit-trailed;
- all previous states of the record be available for the purposes of medico-legal investigation.

These requirements are satisfied in *open*EHR via the use of the change control and versioning facilities defined in the Common Information Model. A key facet of the approach is the use of change-sets, known as Contributions in *open*EHR. Applied to the EHR, they can be visualised as shown in FIGURE 8:

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

The list of Contributions made to a record is recorded along with changes to the data, so that not only are changes to top-level objects (EHR Acces, Composition etc) captured, but the list of changes forming a change set due to a user commit is always known as well.

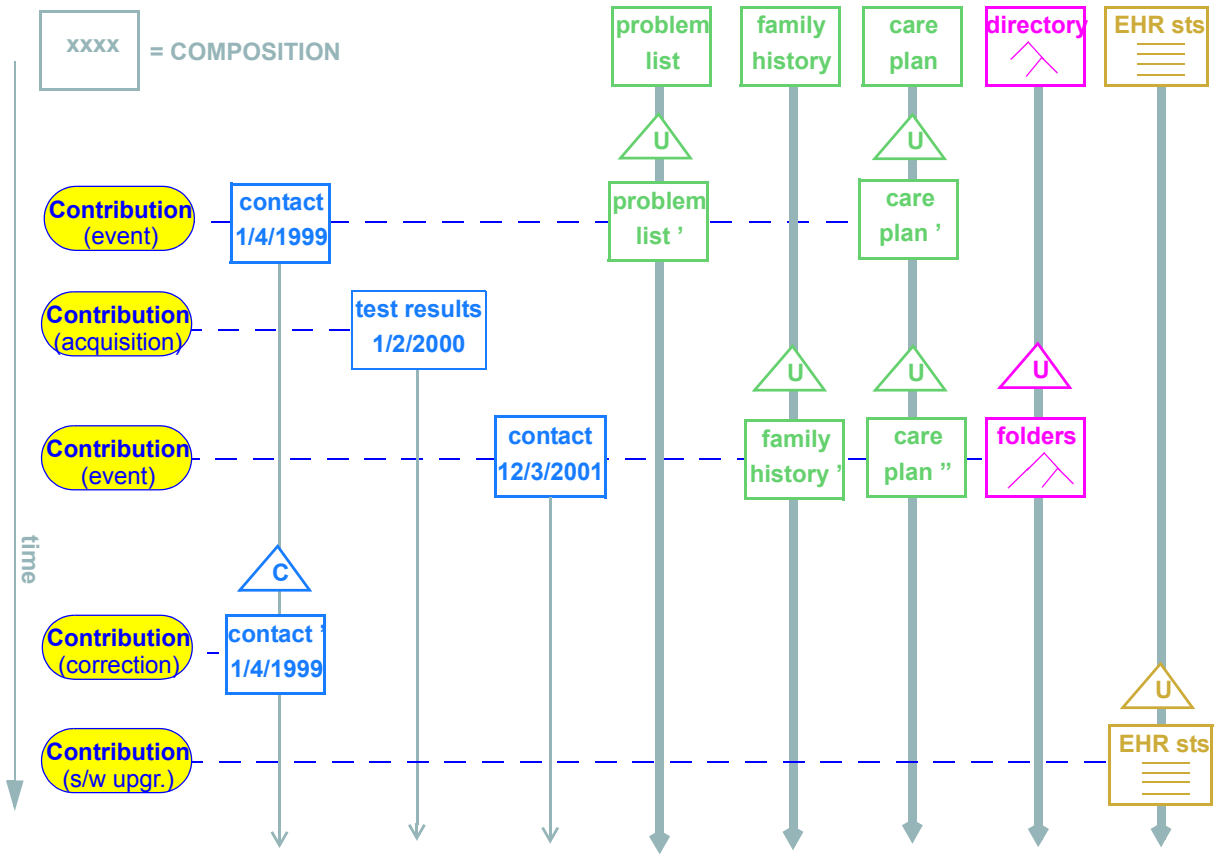


FIGURE 8 Contributions to the EHR

4.3.1 Versioning of Compositions

Versioning of Compositions is achieved with the `VERSIONED_OBJECT<T>` type from the `change_control` package (Common IM), which in the `composition` package is explicitly bound to the `COMPOSITION` class, via the class `VERSIONED_COMPOSITION` which inherits from the type `VERSIONED<COMPOSITION>`.

The effect of version control on Compositions is visualised in FIGURE 9. The versions (each "version" being a `COMPOSITION`) shown here in a `VERSIONED_COMPOSITION` are the same versions shown along each vertical line in FIGURE 8, 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.

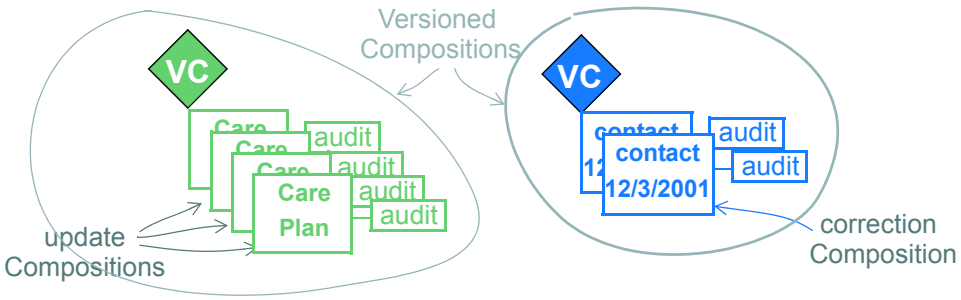


FIGURE 9 The versioned composition

The `VERSIONED_COMPOSITION` can be thought of as a kind of smart 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.

4.3.2 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 that includes some content chosen from the received item. 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.

4.4 EHR Creation Semantics

4.4.1 EHR Identifier Allocation

*open*EHR EHRs are created due to two types of events. The first is a new patient presenting at the provider institution. An EHR may be created due to this event, without reference to any other *open*EHR EHRs that may exist in the broader community or jurisdiction. In this case, the EHR will be allocated a new, globally unique EHR id. This establishes the new EHR as an intentional clone of the source EHR (or more correctly, part of the family of EHRs making up the virtual EHR for that patient).

On the other hand, an *open*EHR EHR may be created in an organisation as a logical clone (probably partial) of an EHR for the patient in some other system. This might happen as a normal part of the front-desk registration / admission process, i.e. the local EHR system is able to interrogate an EHR location service and discover if any other EHR exists for this patient, or it may occur due to purely electronic communications between two providers, i.e. the EHR is created because an Extract of an EHR hs been sent from elsewhere as part of a referral or similar communication. In this second case, the EHR id should be a copy of the EHR id from the other institution.

In theory such a scheme could guarantee one EHR id per patient, but of course in reality, various factors conspire against this, and it can only approximate it. For one thing, it is known that providers routinely create new EHRs for a patient regardless of how many other EHRs already exist for that patient, simply because they have no easy way to find out about those EHRs. Ideally, this situation would be improved in the *open*EHR world, but due to reliance on such things as distributed services and reliable person identification, there are no guarantees. The best that can be said is that the EHR id allocation scheme can help support an ideal EHR id-per-patient situation if and when it becomes possible.

4.4.2 Creation

When an EHR is created, the result should be a root EHR object, an EHR Status object, and an EHR Access object, plus any other house-keeping information the versioning implementation requires. In a normal implementation, the EHR Status and EHR Access objects would normally be created and committed in a Contribution, just as any Composition would be. The EHR Status object has a special status in the EHR, indicating whether the EHR should be included in querying, whether it is modifiable, and by implication, whether it is active. Flags might be set to indicate that it is test record, or for educational or training purposes. The initial creation operation has to supply sufficient parameters for creation of these two objects, including:

- system id
- EHR id
- Subject id (optional; the use of `PARTY_SELF` allows completely anonymous EHRs)
- queryable flag
- modifiable flag
- any other flags required by the EHR Status object in the local implementation.

The EHR id will either be a new globally unique identifier, in the case of first time EHR creation for this patient in the health system, or else the same identifier as an existing EHR for the same subject in another system, in the case of an EHR move or copy. The effect of EHR copying / synchronising between systems is that EHRs with the same identifier can be found within multiple systems. However if the same patient presented at multiple provider locations with no EHR sharing capability, a new EHR with a unique identifier will be created at each place. If a later request for copying occurs (e.g. due to a request for an EHR Extract) between two providers, the requesting institution will perform the merge of the received copy into the existing EHR for the same patient.

The main consequences in a distributed environment are as follows:

- multiple EHR ids for a given patient indicate a mobile patient, but lack of systematic EHR sharing;
- one EHR id everywhere for the patient indicates a seamlessly integrated distributed environment, most likely with a global identification service.

Note that the first situation is not a problem, and is not the same as the situation in which two EHRs are identified as being for different patients (i.e. subject id rather than EHR id is different) when in fact they are for the same person.

4.5 Time in the EHR

There are numerous times recorded in the EHR, at varying levels of granularity. Certain times are a guaranteed by-product of the scientific investigation process, including time of sampling or collec-

tion; time of measurement, time of a healthcare business event, time of data committal. The following figure indicates these times with respect to Observation recordings (generally the most numerous) in the EHR.

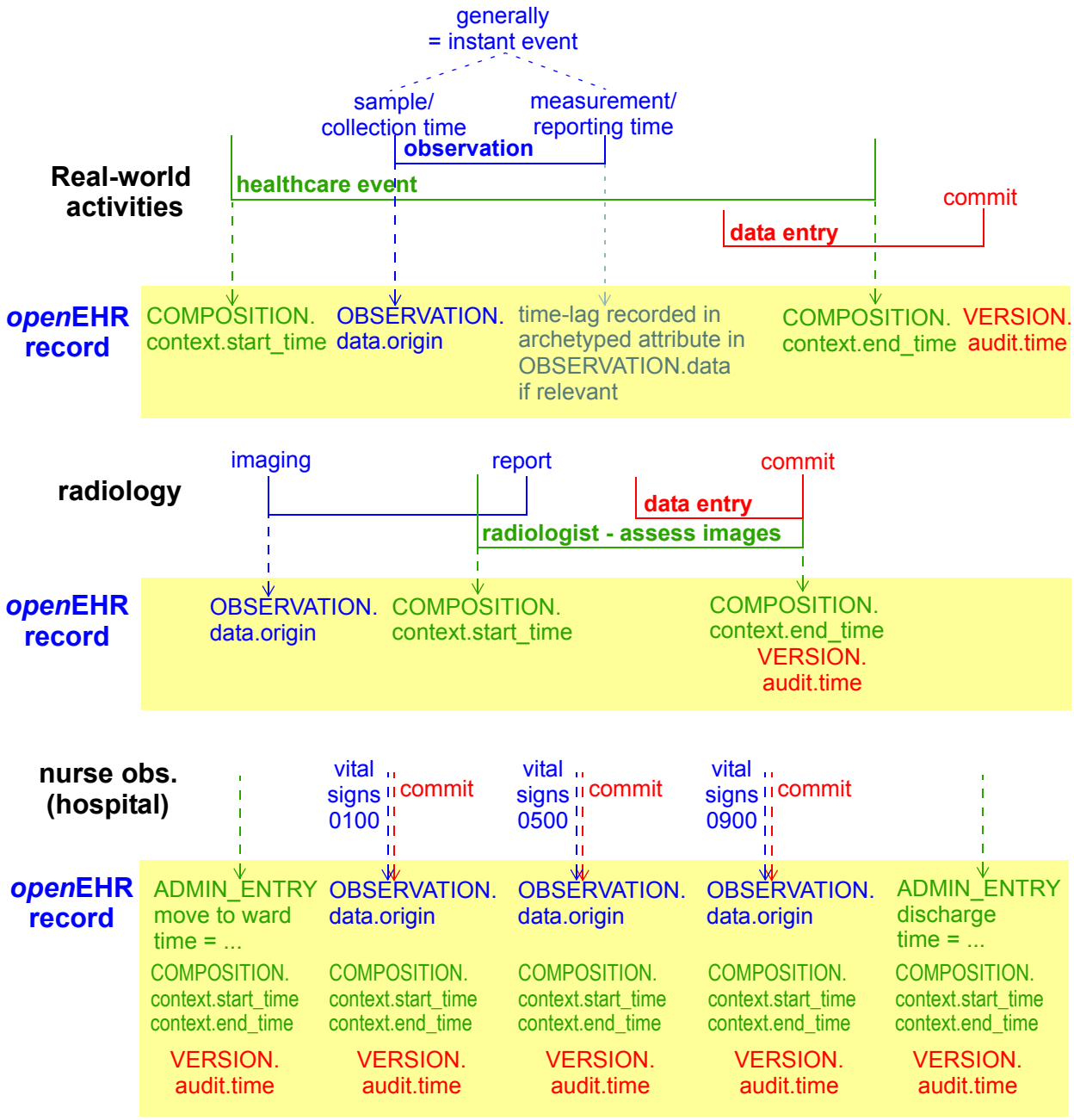


FIGURE 10 Time in the EHR

The top part of FIGURE 10 shows the relationship of times typical for a physical examination at a GP visit. The lower part of the figure shows different relationships common in radiology and microbiology, where the sample (imaging or specimen collection) time can be quite different from the time of assessment and reporting of the sample. The times shown in FIGURE 10 have a close correspondence with the contexts described in Context Model of Recording on page 34; they also have (as shown) concrete attributes in the reference model.

Other times to do with diagnoses (time of onset, time of resolution, time of last episode) and medication management (time started taking medication, time stopped, time suspended, etc) are specific to the particular type of information (e.g. diagnosis versus prognosis versus recommendation) and are generally represented as archetyped date/time values within Evaluation objects. Basic timing information is modelled concretely in the Instruction and Action Entry types, while archetyping is used to express timing information that is specific to particular content.

4.6 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 derivable 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.

4.7 Class Descriptions

4.7.1 EHR Class

CLASS	EHR	
Purpose	The EHR object is the root object and access point of an EHR for a subject of care.	
Attributes	**Signature**	**Meaning**
1..1	**system_id**: HIER_OBJECT_ID	The id of the EHR system on which this EHR was created.
1..1	**ehr_id**: HIER_OBJECT_ID	The id of this EHR.
1..1	**time_created**: DV_DATE_TIME	Time of creation of the EHR
1..1	**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 VERSIONED_FOLDER.

CLASS	EHR	
1..1	**ehr_access**: OBJECT_REF	Reference to EHR_ACCESS object for this EHR.
1..1	**ehr_status**: OBJECT_REF	Reference to EHR_STATUS object for this EHR.
0..1	**directory**: OBJECT_REF	Optional directory structure for this EHR.
1..1	**compositions**: List <OBJECT_REF>	Master list of all composition references in this EHR.
Invariants	***System_id_valid***: system_id /= Void ***Ehr_id_valid***: ehr_id /= Void ***Time_created_valid***: time_created /= Void ***Contributions_valid***: contributions /= Void ***and then*** contributions.for_all(type.is_equal("CONTRIBUTION")) ***Ehr_access_valid***: ehr_access /= Void ***and then*** ehr_access.type.is_equal("VERSIONED_EHR_ACCESS") ***Ehr_status_valid***: ehr_status /= Void ***and then*** ehr_status.type.is_equal("VERSIONED_EHR_STATUS") ***Compositions_valid***: compositions /= Void ***and then*** compositions.for_all(type.is_equal("VERSIONED_COMPOSITION")) ***Directory_valid***: directory /= Void ***implies*** directory.type.is_equal("VERSIONED_FOLDER")	

4.7.2 VERSIONED_EHR_ACCESS Class

CLASS	VERSIONED_EHR_ACCESS	
Purpose	Version container for EHR_ACCESS instance.	
Inherit	VERSIONED_OBJECT<EHR_ACCESS>	
Attributes	Signature	Meaning
Invariants		

4.7.3 EHR_ACCESS Class

CLASS	EHR_ACCESS	
Purpose	EHR-wide access contol object. All access decisions to data in the EHR must be made in accordance with the policies and rules in this object.	
Inherit	LOCATABLE	
Attributes	Signature	Meaning

CLASS	EHR_ACCESS	
0..1	**settings**: *ACCESS_CONTROL_SETTINGS*	Access control settings for the EHR. Instance is a subtype of the type *ACCESS_CONTROL_SETTINGS*, allowing for the use of different access control schemes.
Functions	**Signature**	**Meaning**
1..1	**scheme**: String	The name of the access control scheme in use; corresponds to the concrete instance of the *settings* attribute.
Invariants	***Scheme_exists***: scheme /= Void **and then not** scheme.is_empty	

4.7.4 VERSIONED_EHR_STATUS Class

CLASS	VERSIONED_EHR_STATUS	
Purpose	Version container for EHR_STATUS instance.	
Inherit	VERSIONED_OBJECT<EHR_STATUS>	
Attributes	**Signature**	**Meaning**
Invariants		

4.7.5 EHR_STATUS Class

CLASS	EHR_STATUS	
Purpose	Single object per EHR giving various EHR-wide information.	
Inherit	LOCATABLE	
Attributes	**Signature**	**Meaning**
1..1	**subject**: PARTY_SELF	The subject of this EHR. The *external_ref* attribute can be used to contain a direct reference to the subject in a demographic or identity service. Alternatively, the association between patients and their records may be done elsewhere for security reasons.
1..1	**is_queryable**: Boolean	True if this EHR should be included in population queries, i.e. if this EHR is considered active in the population.

CLASS	EHR_STATUS	
1..1	**is_modifiable**: `Boolean`	True if this EHR is allowed to be written to.
0..1	**other_details**: `ITEM_STRUCTURE`	Any other details of the EHR summary object, in the form of an archetyped Item_structure.
Invariants	***Is_archetype_root***: is_archetype_root ***Subject_valid***: subject /= Void ***No_parent***: parent = Void	

4.7.6 VERSIONED_COMPOSITION Class

CLASS	VERSIONED_COMPOSITION	
Purpose	Version-controlled composition abstraction, defined by inheriting `VERSIONED_OBJECT<COMPOSITION>`.	
Inherit	`VERSIONED_OBJECT<COMPOSITION>`	
Function	Signature	Meaning
	is_persistent: `Boolean`	Indicates whether this composition set is persistent; derived from first version.
Invariants	***Archetype_node_id_valid***: all_versions.for_all (archetype_node_id.is_equal(all_versions.first.archetype_node_id)) ***Persistent_valid***: all_versions.for_all (is_persistent = all_versions.first.data.is_persistent) ***Owner_id_valid***: owner_id.generating_type.is_equal("EHR")	

5 Composition Package

5.1 Overview

The Composition is the primary 'data container' in the *open*EHR EHR and is the root point of clinical content. Instances of the Composition class can be considered as self-standing data aggregations, or documents in a document-oriented system. The key information in a `COMPOSITION` is found in its *content*, *context*, and *composer* attributes. FIGURE 11 illustrates the `composition` package.

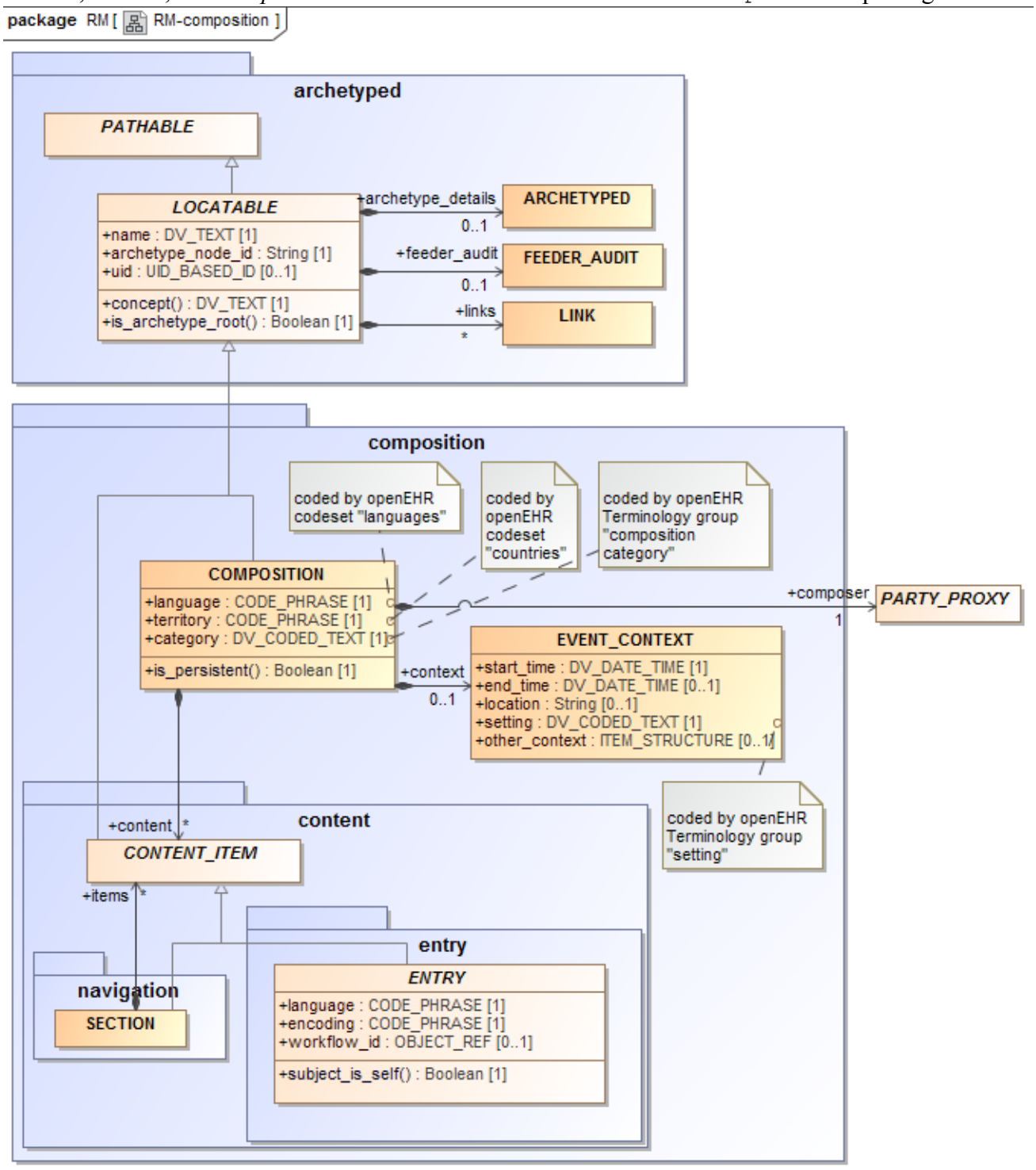


FIGURE 11 rm.composition Package

5.2 Context Model of Recording

5.2.1 Overview

The *open*EHR EHR model takes account of a systematic analysis of "context". Contexts in the real world are mapped to particular levels of the information model in a clear way, according to the scheme shown in FIGURE 12. On the left hand side is depicted the context of a data-entry session in which the information generated by a "healthcare event", containing "clinical statements", is added to the EHR. A healthcare event is defined as any business activity of the health care system for the patient, including encounters, pathology tests, and interventions. A clinical statement is the minimal indivisible unit of information the clinician wants to record. Clinical statements are shown in the diagram has having temporal and spatial structure as well as data values. Each of these three contexts has its own audit information, consisting of who, when, where, why information.

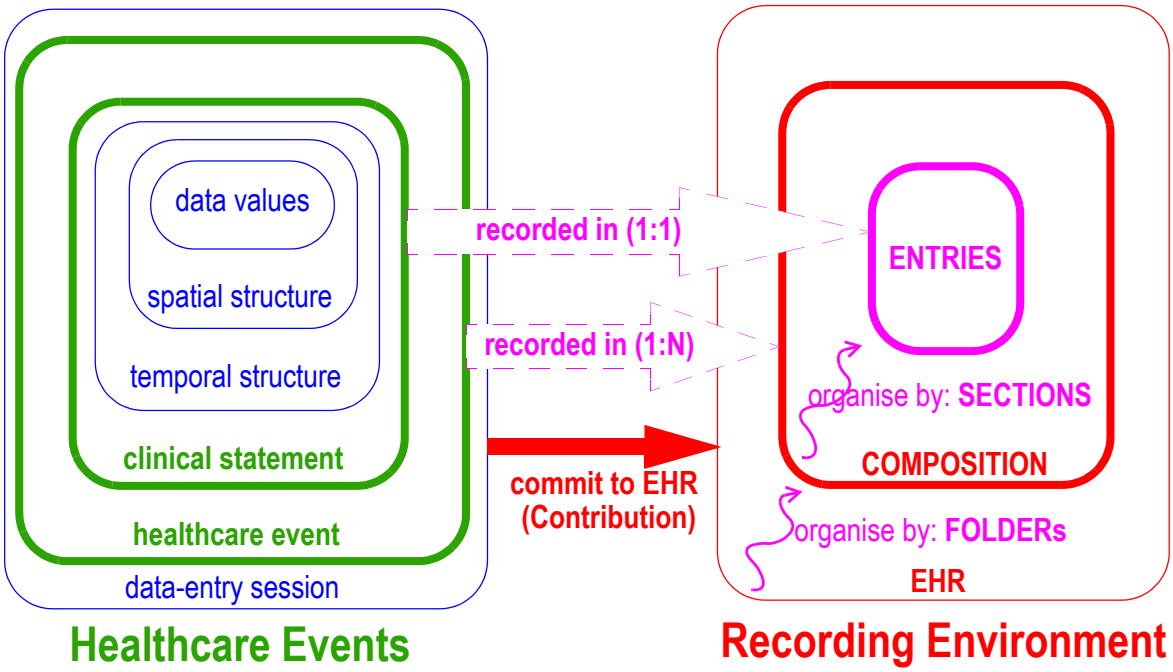


FIGURE 12 General Model of Recording

On the right hand side of FIGURE 12, the EHR recording environment is represented. The EHR consists of distinct, coarse-grained items known as Compositions added over time and organised by Folders. Each Composition consists of Entries, organised by Sections within the Composition. The audit information for each context is recorded at the corresponding level of the EHR.

5.2.2 Composer

The *composer* is the person who was primarily responsible for the content of the Composition. This is the identifier that 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 and committed 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.

Since in many cases Compositions will be composed and committed by the same person, it might seem that two identifiers `COMPOSITION`.*composer* and `VERSION`.*audit.committer* (which are both of type `PARTY_PROXY`) will be identical. In fact, this will probably not be the case, because the kind of

identifier to represent the composer will be a demographic identifier, e.g. “RN Jane Williams”, “RN 12345678”, whereas the identifier in the audit details will usually be a computer system user identifier, e.g. “jane.williams@westmead.health.au”. This difference highlights the different purposes of these attributes: the first exists to identify the clinical agent who created the information, while the second exists to identify the logged-in user who committed it to the system.

In the situation of patient-entered data, the special “self” `PARTY_PROXY` instance (see Common IM `generic` package) is used for both `COMPOSITION`.*composer* and `VERSION`.*audit*.*committer*.

5.2.3 Event Context

5.2.3.1 Overview

The optional *event_context* in the `COMPOSITION` class is used to document the healthcare event causing new or changed content in the record. Here, ‘healthcare event’ means ‘a (generally billable) business activity of the healthcare system with, for or on behalf of the patient’. Generally this will an encounter involving the subject of care and physician, but is variable in a hospital environment. In this sense, a visit to a GP is a single care event, but so is an episode in a hospital, which may encompass multiple encounters. The information recorded in Event context includes start and (optional) end time of the event, health care facility, setting (e.g. primary care, aged care, hospital), participating healthcare professionals, and optional further details defined by an archetype.

Healthcare events that require an Event_context instance in their recorded information include the following.

- Scheduled or booked patient encounters leading to changes to the EHR, including with a GP, hospital consultant, or other clinical professional such as mobile nurse. In this case, the Event context documents the time and place of the encounter, and the identity of the clinical professional.
- Case conferences about a patient, leading to modifications to the health record; here the Event context documents the case conference time, place and participants.
- Pathology, imaging or other test process. In this case, the Event context documents the place and period during which testing and analysis was carried out, and by whom.
- Data resulting from care in the home provided by health professional(s) (often allied health care workers).

Situations in which Event context is optional include the following.

- Nurse interactions with patients in hospital, including checking vital signs, adjusting medication or other aspects of bed situation for the patient. Each instance of a nurse’s observations are generally not considered to be a separate ‘care event’, rather they are seen as the continuation of the general activities of monitoring. In such situations, the overall context is given by `ADMIN_ENTRY` instances in the record indicating date/time and place of admission and discharge.

Situations in which Event context is not used include:

- Any modification to the EHR which corrects or updates existing content, including by administrative staff, and by clinical professionals adding or changing evaluations, summaries etc.
- Patient-entered data where no interaction with health professionals took place; typically readings from devices in the home such as weighing scales, blood glucose measuring devices, wearable monitors etc.

Ultimately, the use of Event context will be controlled by Composition-level archetypes.

5.2.3.2 Occurrence in Data

For situations requiring an EVENT_CONTEXT object to be recorded, it is worth clarifying which Compositions carry such objects. Consider the example shown in FIGURE 13. In this example, a Contribution is made to the EHR, consisting of one or more Compositions that were each created or modified due to some clinical 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, and is therefore the one which contains the EVENT_CONTEXT instance. 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. Contributions A, B and C in the figure illustrate this case.

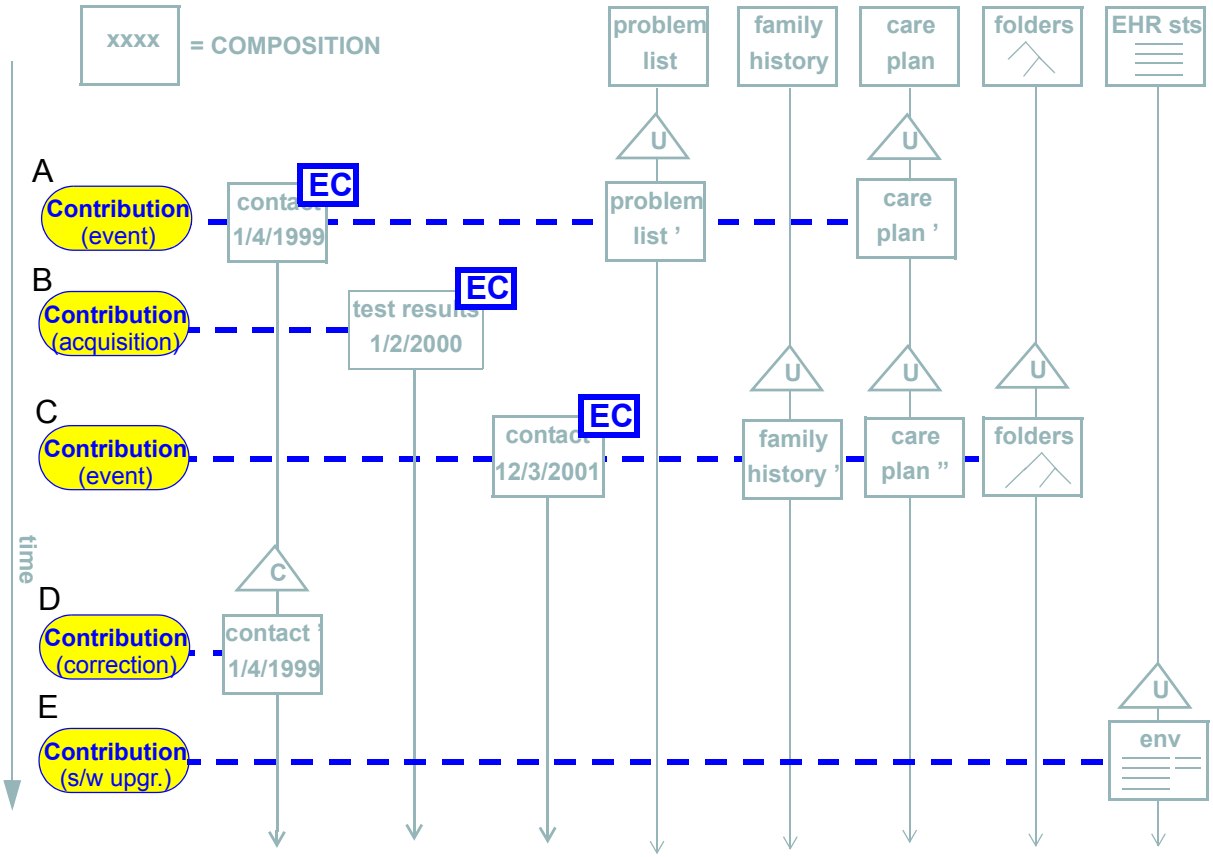


FIGURE 13 Use of Event context

In cases where Contributions are made to the record with no event context, the Event context of any Compositions from the original commit will remain intact and visible (unless the correction is to the event context itself of course), and will correctly reflect the fact that no new clinical interaction occurred. This is the case with Contribution D in the figure.

Persistent Compositions do not have an Event context.

5.2.3.3 Time

The times recorded in the Event context 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), 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 as described above even if the additions are made to the EHR long after the event took place, such as happens when a doctor writes his/her notes into the record system at night, after all patients have been seen. In such cases, the versioned Composition audit trail records the context of when the data were entered, as distinct from the context of when the clinical interaction took place.

5.2.3.4 Participations

As part of the Event context, the *participations* attribute can be used to describe who participated, and how. Each participation object describes the "mode" of participation as well, such as direct presence, video-conference and so on. In many cases such information is of no interest, since the subject of any Entry is known (ENTRY.*subject*) and the clinician will be known (COMPOSITION.*composer*), and the mode of communication is assumed to be a personal encounter. The *participations* attribute is therefore used when it is desired to record further details of how the patient and or physician interacted (e.g. over the internet), and/or other participants, such as family, nurses, specialists etc.

There are no general rules about who is included as a participant. 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 drug self-administration in the record, in which case the composer will be the patient, and there will be no clinician participation. Consequently, the use of participations will mostly be archetype-driven.

5.2.3.5 Healthcare Facility, Location and Setting

The *health_care_facility* (HCF) attribute is used to record the health care facility in whose care the event took place. This is the most specific identifiable (i.e. having a provider id within 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 control this.

Two other context attributes complete the predefined notion of event context in the model: *location* and *setting*. The *location* attribute records: the physical location where the care delivery took place, and should document a reasonably specific identifiable location. Examples include "bed 5, ward E", "home". This attribute is optional, since the location is not always known, particularly in legacy data.

The *setting* attribute is used to document 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.

5.3 Composition Content

The data in a Composition is stored in the *content* attribute. There are four kinds of data structuring possible in the *content* attribute:

- 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* = 'incomplete')
  - 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 contain one or more ENTRYs directly, with no intermediate SECTIONs;
- it may be any combination of the previous three possibilities.

The actual structures used in a Composition at runtime are controlled by a template, which in turn controls the particular combination of archetypes used.

5.4 Class Descriptions

5.4.1 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.	
Inherit	LOCATABLE	
Attributes	**Signature**	**Meaning**
0..1	**content**: List<CONTENT_ITEM>	The content of this Composition.
0..1	**context**: EVENT_CONTEXT	The clinical session context of this Composition, i.e. the contextual attributes of the clinical session.
1..1	**composer**: PARTY_PROXY	The person primarily responsible for the content of the Composition (but not necessarily its committal 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. When it is the patient, the special "self" instance of PARTY_PROXY will be used.

CLASS	COMPOSITION	
1..1	**category**: DV_CODED_TEXT	Indicates what broad category this Composition is belogs to, e.g. “persistent” - of longitudinal validity, “event”, “process” etc.
1..1	**language**: CODE_PHRASE	Mandatory indicator of the localised language in which this Composition is written. Coded from *open*EHR Code Set “languages”. The language of an Entry if different from the Composition is indicated in ENTRY.*language*.
1..1	**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**
1..1	**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 ***Composer_valid***: composer /= Void ***Content_valid***: content /= Void **implies not** content.is_empty ***Category_validity***: category /= Void **and then** terminology(Terminology_id_openehr). has_code_for_group_id(Group_id_composition_category, category.defining_code) ***Is_persistent_validity***: is_persistent **implies** context = Void ***Territory_valid***: territory /= Void **and then** code_set(Code_set_id_countries).has_code(territory) ***Language_valid***: language /= Void ***and then*** code_set(Code_set_id_languages).has_code(language) ***No_parent***: parent = Void	

5.4.2 EVENT_CONTEXT Class

CLASS	EVENT_CONTEXT
Purpose	Documents the context information of a healthcare event involving the subject of care and the health system. 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. Healthcare events include patient contacts, and any other business activity, such as pathology investigations which take place on behalf of the patient.

CLASS	EVENT_CONTEXT	
Inherit	PATHABLE	
Attributes	**Signature**	**Meaning**
0..1	**health_care_facility**: PARTY_IDENTIFIED	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 that has an official identifier in the health system, and can be used to ensure medico-legal accountability.
1..1	**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.
0..1	**end_time**: DV_DATE_TIME	Optional end time of the clinical session.
0..1	**participations**: List <PARTICIPATION>	Parties involved in the healthcare event. 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).
0..1	**location**: String	The actual location where the session occurred, e.g. “microbiol lab 2”, “home”, “ward A3” and so on.
1..1	**setting**: DV_CODED_TEXT	The setting in which the clinical session took place. Coded using the *open*EHR Terminology, “setting” group.
0..1	**other_context**: ITEM_STRUCTURE	Other optional context which will be archetyped.
Invariants	***start_time_valid***: start_time /= Void ***participations_validity***: participations /= Void ***implies not*** participations.is_empty ***location_valid***: location /= Void ***implies not*** location.is_empty ***setting_valid***: setting /= Void ***and then*** Terminology(Terminology_id_openehr).has_code_for_group_id (Group_id_setting, setting.defining_code)	

6 Content Package

6.1 Overview

The `content` package contains the `CONTENT_ITEM` class, ancestor class of all content types, and the `navigation` and `entry` packages, which contain `SECTION`, `ENTRY` and related types.

6.2 Class Descriptions

6.2.1 CONTENT_ITEM Class

CLASS	*CONTENT_ITEM (abstract)*	
Purpose	Abstract ancestor of all concrete content types.	
Inherit	`LOCATABLE`	
Attributes	**Signature**	**Meaning**
Invariants		

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`, visible in the lower left side of FIGURE 11.

Sections provide both a logical structure for the author to arrange Entries, 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 that are separately archetyped, such as the SOAP headings, or the heading structure for a physical examination, can be combined at runtime by type to form one large heading structure, as shown in FIGURE 14.

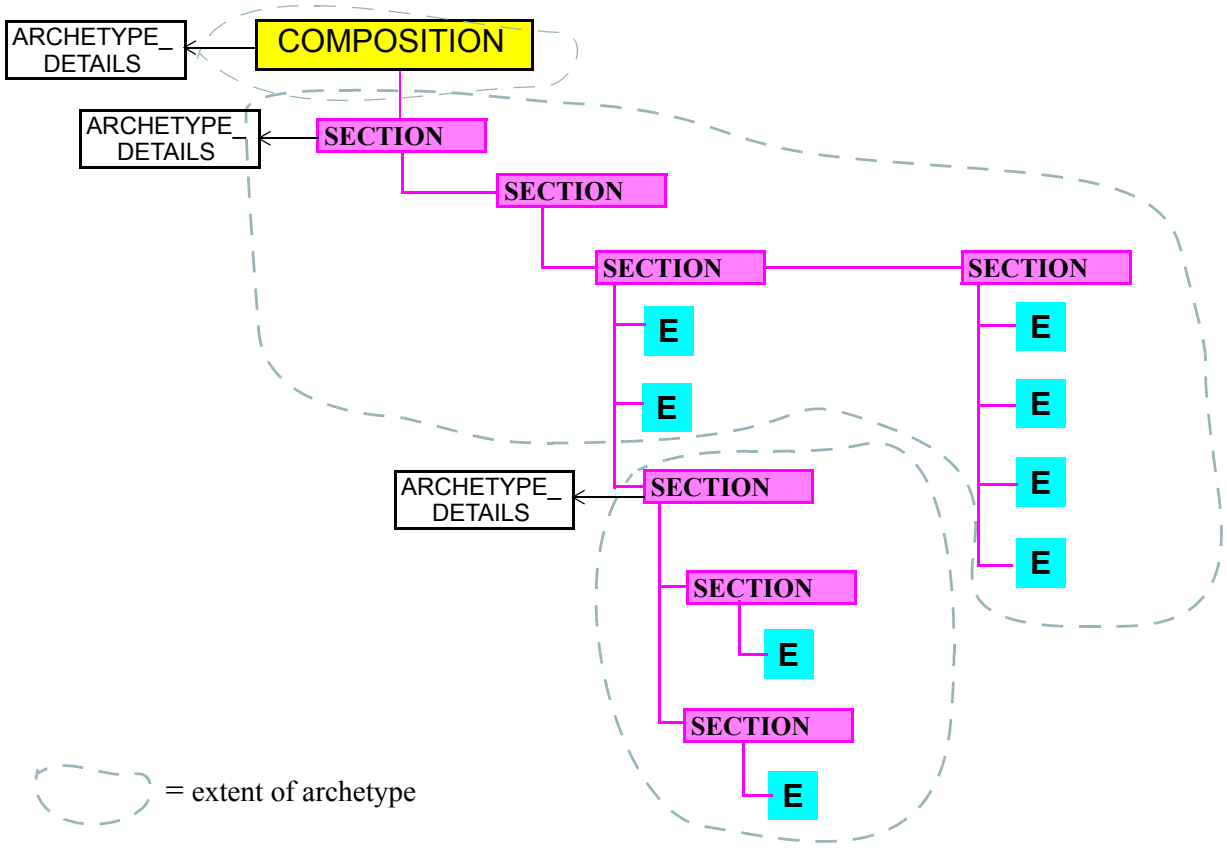


FIGURE 14 Section View of a General Practice Contact Composition

In terms of understanding of clinical data, Section structures are not essential in a Composition - they can always be removed or ignored (typically in machine processing such as querying) without losing the meaning of the Entries in the Composition. While Sections are often used to group Entries according to status, e.g. “family history”, “problems”, “observations”, it is the Entries themselves that indicate the definitive category of information contained therein. This principle is explained in more detail in Entry and its Subtypes on page 49.

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 any Section structure is that while the root Section in a tree is logically a Section, it does not appear in a display or printed form as a visible section. This is 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 further 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.	
Inherit	`CONTENT_ITEM`	
Attributes	**Signature**	**Meaning**
0..1	**items**: `List<CONTENT_ITEM>`	Ordered list of content items under this section, which may include: • more `SECTIONs` • `ENTRYs`
Invariants	***Items_valid***: items /= Void ***implies not*** items.is_empty	

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 15.

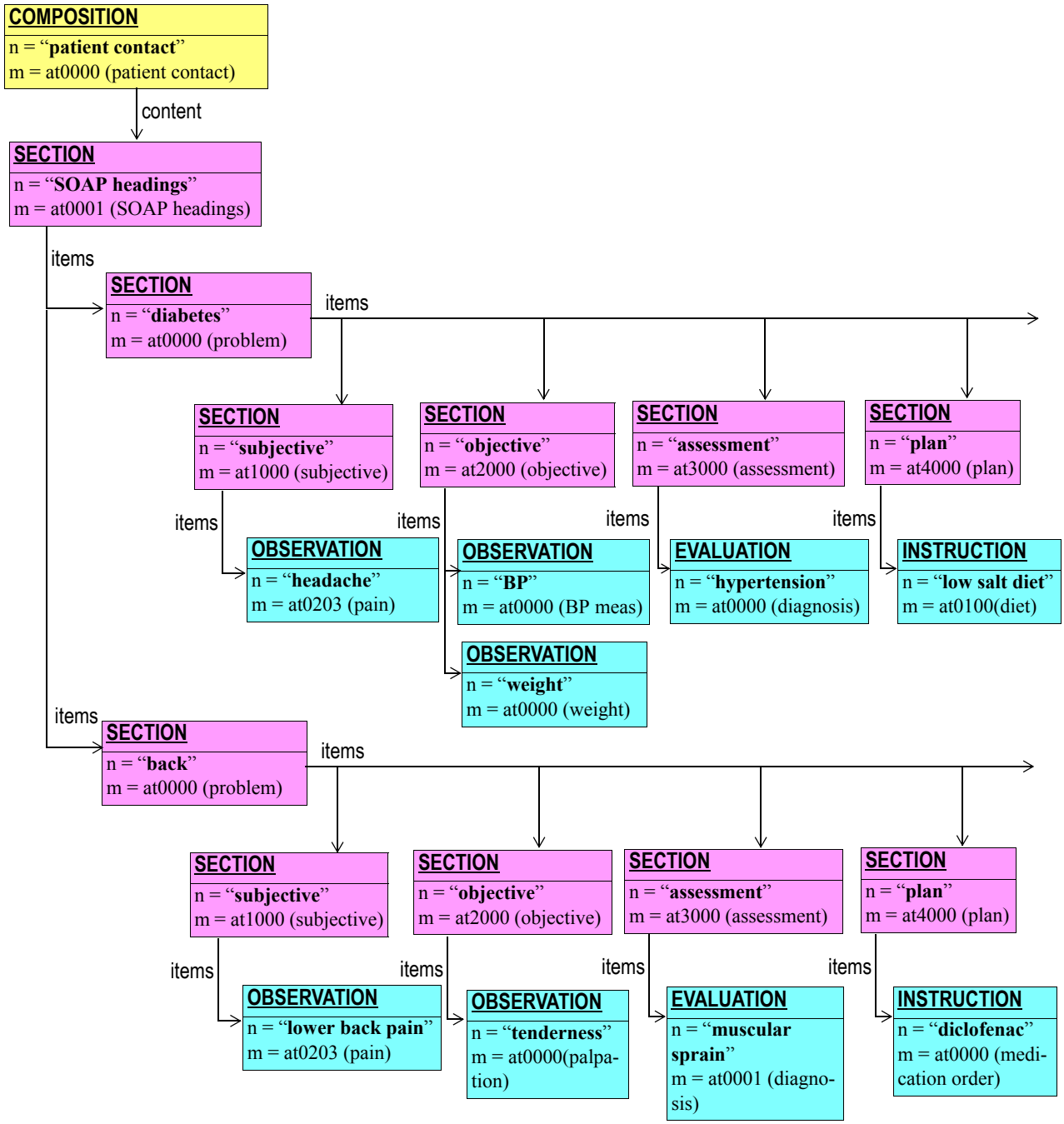


FIGURE 15 "problem/SOAP" Section Structure

8 Entry Package

8.1 Design Principles

8.1.1 Information Ontology

All information which is created in the *open*EHR health record is expressed as an instance of a class in the `entry` package, containing the `ENTRY` class and a number of descendants. An `ENTRY` instance is logically a single 'clinical statement', and may be a single short narrative phrase, but may also contain a significant amount of data, e.g. a microbiology result, a psychiatric examination, a complex prescription. In terms of clinical content, the Entry classes are the most important in the *open*EHR EHR Information Model, since they define the semantics of all the 'hard' information in the record. They are intended to be archetyped, and in fact, archetypes for Entries make up the vast majority of important clinical archetypes defined.

The design of Entry package is based on the Clinical Investigator Recording process and ontology, described in Beale & Heard [4]. The process is shown in the FIGURE 16.

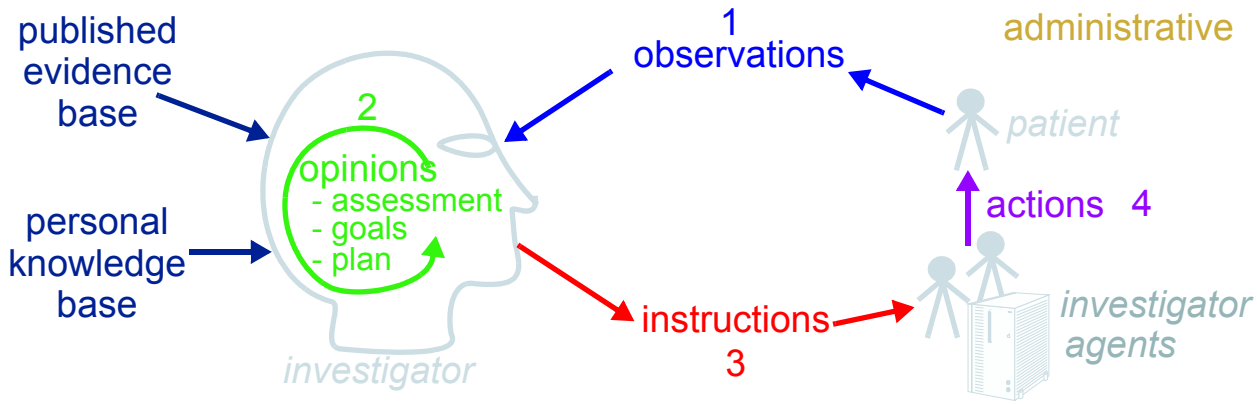


FIGURE 16 Clinical Investigator Recording Process

This figure shows the cycle of information created by an iterative, problem solving process undertaken by a "clinical investigator system", which consists of health carers, and may include the patient (at points in time when the patient performs observational or therapeutic activities).

Starting from the patient (right hand side of figure) observations are made, which lead to opinions on the part of the investigator, including assessment of the current situation, goals for a future situation, and plans for achieving the goals. Personal and published evidence and knowledge almost always play an important part in this process. The latter lead to instructions designed to help the patient achieve the goals. A complex or chronic problem may take numerous iterations - possibly a whole lifetime's worth - with each step being quite small, and future steps depending heavily on past progress. The role of the investigator (and associated agents) is normally filled by health care professionals, but may also be filled by the patient, or a guardian or associate of the patient. Indeed, this is what happens every time a person goes home from the pharmacy with prescribed medication to take at home.

The process illustrated in FIGURE 16 is a synthesis of the "problem-oriented" approach of Weed [16] and the "hypothetico-deductive model" of clinical reasoning described by Elstein *et al* [6]. However, as pointed out in Elstein & Schwarz [7], hypothesis-making and testing is not the only successful process used by clinical professionals - evidence shows that many (particularly those older and more experienced) rely on pattern recognition and direct retrieval of plans used previously with similar

patients or prototype models. The investigator process is compatible with both cognitive approaches, since it does not say how opinions are formed, nor imply any specific number or size of iterations to bring the process to a conclusion. As such, the *open*EHR information model does not impose any process model, only the types of information used.

On the basis of this process, a Clinical Investigator Recording ontology is developed [4], as shown in FIGURE 17. From this ontology, the *open*EHR class model for Entries is derivde. The *open*EHR Entry class names are annotated next to their originating ontological categories.

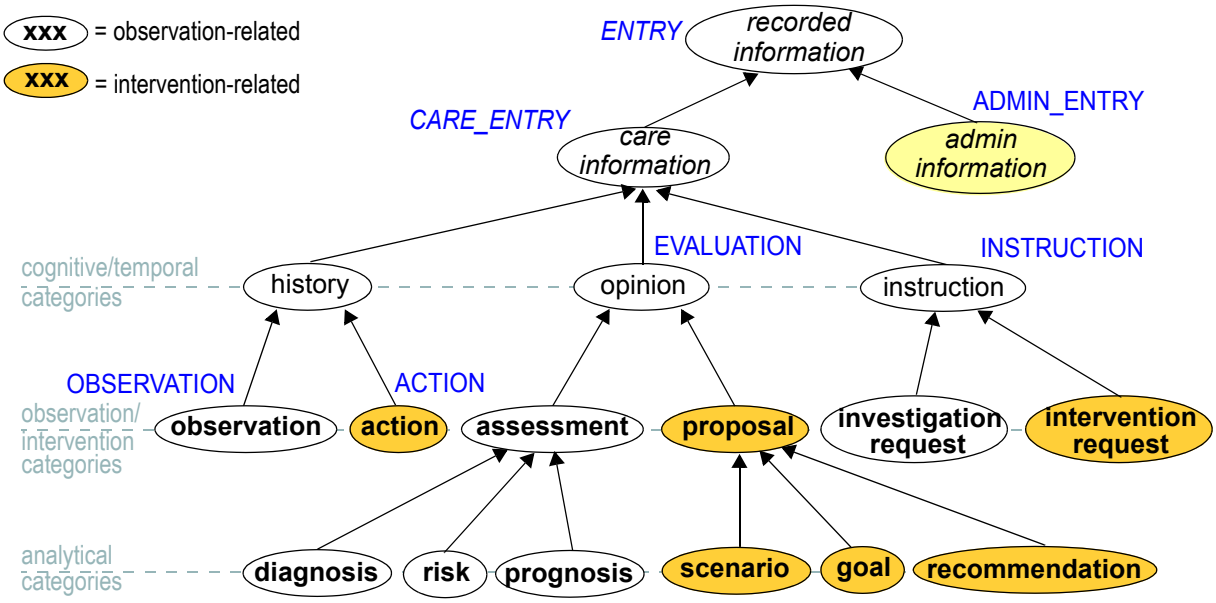


FIGURE 17 The Clinical Investigator Recording (CIR) ontology

The key top-level categories in the ontology are ‘care information’ and ‘administrative information’. The former encompasses all statements that might be recorded at any point during the care process, and consists of the major subcategories ‘history’, ‘opinion’ and ‘instruction’, which themselves correspond to the past, present and future in time (ISO TC215 uses the terms ‘retrospective’, ‘current’ and ‘prospective’). The administrative information category covers information which is not generated by the care process proper, but relates to organising it, such as appointments and admissions. This information is not about care, but about the logistics of care being delivered. Categories that relate to the patient system as observed and understood are shown as white bubbles, while categories that relate to intervention into the patient system are shown as shaded. The opinion category has features of both passive analysis and active intervention.

There are two key justifications for using the ontology of FIGURE 17 as a basis for class design. Firstly, although for all categories in the ontology there is a meaning for ‘contextual’ attributes of time, place, identity, reason and so on, each category has a different structure for these attributes. For example, time in the Observation category has a linear historical structure, whereas in Instruction it has a branching, potentially cyclic structure. The separation of types allows these contextual attributes to be modelled according to the type. Secondly, the separation of types provides a systematic solution to the so-called problem of “status” or “meaning modification” of clinical statement values, as described below.

8.1.2 Clinical Statement Status and Negation

A well-known problem in clinical information recording is the problem of assigning "status", including variants like "actual value of P" (P stands for some phenomenon), "family history of P", "risk of P", "fear of P", as well as negation of any of these, i.e. "not/no P", "no history of P" etc. A proper analysis of these so called statuses [4] shows that they are not "statuses" at all, but different categories of information as per the ontology. The common statement types mentioned here are mapped as follows:

- actual value of P ⇒ Observation (of P);
- no/not P ⇒ Observation (of excluded P or types of P, e.g. allergies).
- family history of P ⇒ Evaluation (that patient is at risk of P);
- no family history of P ⇒ Evaluation (that P is an excluded risk);
- risk of P ⇒ Evaluation (that patient is at risk of P);
- no risk of P ⇒ Evaluation (that patient is not at risk of P);
- fear of P ⇒ Observation (of FEAR, with P mentioned in the description);

In general, negations of the kind mentioned above are handled by using "exclusion" archetypes for the appropriate Entry type. For example, "no allergies" can be modelled using an Evaluation archetype that describes which allergies are excluded for this patient.

Another set of statement types that can be confused in systems that do not properly model information categories concern interventions, e.g. "hip replacement (5 years ago)", "hip replacement (planned)", "hip replacement (ordered for next tuesday 10 am)". Ambiguity is removed here as well, with the use of the correct information categories, e.g. (I stands for an intervention):

- I (distant past/unmanaged/passively documented) ⇒ Observation (of I present in patient);
- I (recent past) ⇒ Action (of I having been done to/for patient);
- I (proposed) ⇒ Evaluation, subtype Proposal (of I suggested/likely for patient);
- I (ordered) ⇒ Instruction (of I for patient for some date in the future).

Related to the problem of clinical status is the need to differentiate between diagnoses that have been made in the present from those that were made in the past, and are reported by the patient. In *open*EHR, diagnosis and indeed all opinion category types are modelled as Evaluations, regardless of when they were made. Time information for diagnosis and other opinions is simply handled within the archetype for Evaluation, enabling a diagnosis (say of diabetes) that was made 15 years ago to have the same status as one made during the period when the EHR was actually operating, and the patient was under the care of the physician using it. Archetypes for the opinion categories tend to have quite a number of times, including "time first noticed", "time of onset", and so on.

Correct use of the categories from the ontology is facilitated by using archetypes designed to map particular kinds of clinical statement to particular `ENTRY` subtypes. In a system where Entries are thus modelled, there will be no danger of incorrectly identifying the various kinds of Entries, as long as the Entry subtype, time, and certainty/negation are taken into account.

8.1.3 Standard Clinical Types of Data

Commonly used types of clinical information are directly representable using the *open*EHR Entry types. As described in the *open*EHR EHR IM, two kinds of Composition are identified: persistent and event. The persistent kind corresponds to information that characterises the patient in the long term, and is maintained by clinicians - it can be thought of as a proxy "model of the patient". Most of the the following are examples of Entry level data belonging inside persistent Compositions:

Basic information (dob, sex, height, weight, pregnant etc): recorded as Observations and/or Admin_entries;

Problem list: maintained as one or more Evaluations which themselves are generated by clinicians as a rsult of observations made elsewhere in the record;

Medications list: derived from Instructions and Actions in the record. The status of Actions allows medications to be displayed as past, current, suspended etc.

Therapeutic precautions (allergies and alerts): recorded as Evaluations of various kinds (adverse reactions are essentially a kind of diagnosis);

Patient preferences: recorded as Evaluations (since they act like contra-indications for certain drugs or treatments);

Patient consents: recorded as instances of Admin_entry;

Family history:

- actual events / conditions in family members are recorded as Observations (e.g. father died of MI at 62)
- patient risks are expressed using Evaluations, often including family history reasons.

Social history/situation: current and previous social situation (e.g. in nursing care, details of feeding, sleeping arrangments) are documented as Observations.

Lifestyle: there are various Observation archetypes for recording aspects of lifestyle, including exercise, smoking/tobacco, alcohol, drug use and so on.

Vaccination record: vaccinations are a kind of Instruction; vaccinations that have actually been given are available as Actions in the record.

Care plan: combinations of goals, targets (Evaluations), monitoring, education (Instructions), etc orgnanised in a care plan Section hierarchy.

The remaining vast majority of remaining clinical information is recorded inside event Compositions, and includes the following:

laboratory results including imaging: recorded as Observations;

physical examinations: recorded as Observations

appointment, admission and discharge: recorded as Admin_entry

prescriptions: one or more medication orders (each one is an Instruction) within a prescription Section structure, within a prescription Composition.

referrals: recorded as Instructions (i.e. a request for care by another provider).

The above concepts are defined in terms of archetypes of Entry and other reference model types in *open*EHR; their definition is therefore completely separate from the reference model.

8.1.4 Demographic Data in the EHR

The general approach of *open*EHR is to enable the complete separation of demographic (particularly patient-identification information) from health records, both in the interests of privacy (in some cases required by national legislation) and separated data management. The Demographic IM therefore defines demographic information. However, there is nothing to prevent certain demographic information occurring in the EHR, and in some cases this is desirable. The two main cases for this are:

- clinically-relevant patient information, such as age, sex, height, weight, eye-colour, ethnicity or ‘race’, occupation;

- identifiers and/or names of health care provider individuals and organisations may be stored directly in the EHR, regardless of whether there is also more detalied information about such entities in the demographic system.

The model of all information intended for a separate *open*EHR demographic service (itself usually a front-end to an existing hospital master patient index or similar) is defined in the *open*EHR Demographic Information Model.

8.2 Entry and its Subtypes

The Entry model is defined by the `composition.content.entry` package, shown in FIGURE 18. The choice of subtypes of `ENTRY` is based on the ontology shown in FIGURE 17 and its associated model. The names do not coincide exactly however, for a number of reasons. Firstly, the category names in the hierarchy were chosen based on a philosophical/scientific model of investigation, and reflecting linguistic norms for the meanings of the terms, whereas the class names used here reflect common health computing and clinical usage of these terms (i.e. names which will make sense to software developers in the health arena). Names in these two cultures do not always coincide. Additionally, for categories such as Opinion and Instruction, the subcategories shown in the ontology (e.g. Assessment, Goal and Plan) are too variable to safely be subtyped in software, and are distinguished only at the archetype level. Only a single class for each of these ontological groups is used in the formal model. The use of different names and a slightly simplified mapping has not however prevented the faithful implementation of the semantics of the model. The model classes are described in the following subsections.

8.2.1 The Entry class

All Entries have a number of attributes in common. The *language* and *encoding* attributes indicate how all text data within the Entry are to be interpreted linguistically and at the character set level. Normally, the language will be the same throughout the entire Entry (if not Composition), 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 can be overridden in the same fashion by the *encoding* attribute within `DV_TEXT`.

The other attributes common to all Entry subtypes are as follows:

subject: this attribute records the subject of the Entry as an instance of a subtype of `PARTY_PROXY`. When this is the record subject, (i.e. the patient), the value is an instance of `PARTY_SELF`. Otherwise it is typically a family member, sexual partner, or other acquaintance of the record subject. It could also be an organ donor. The latter is expressed in the form of a `PARTY_IDENTIFIED` or `RELATED_PARTY` instance, which describes the kind of relationship, and optionally, identifies the demographic entity.

subject_is_self: convenience function returning True when the Entry is about the subject of the record.

provider: the agent who provided the information. This is usually the patient or the clinician, but may be someone else, or a software application or device. If participation details of the provider (e.g. mode of communication) need to be recorded, the details should be recorded once in the `EVENT_CONTEXT`.*participations*. The *provider* attribute is optional, since it is often implicit in the information that was recorded.

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

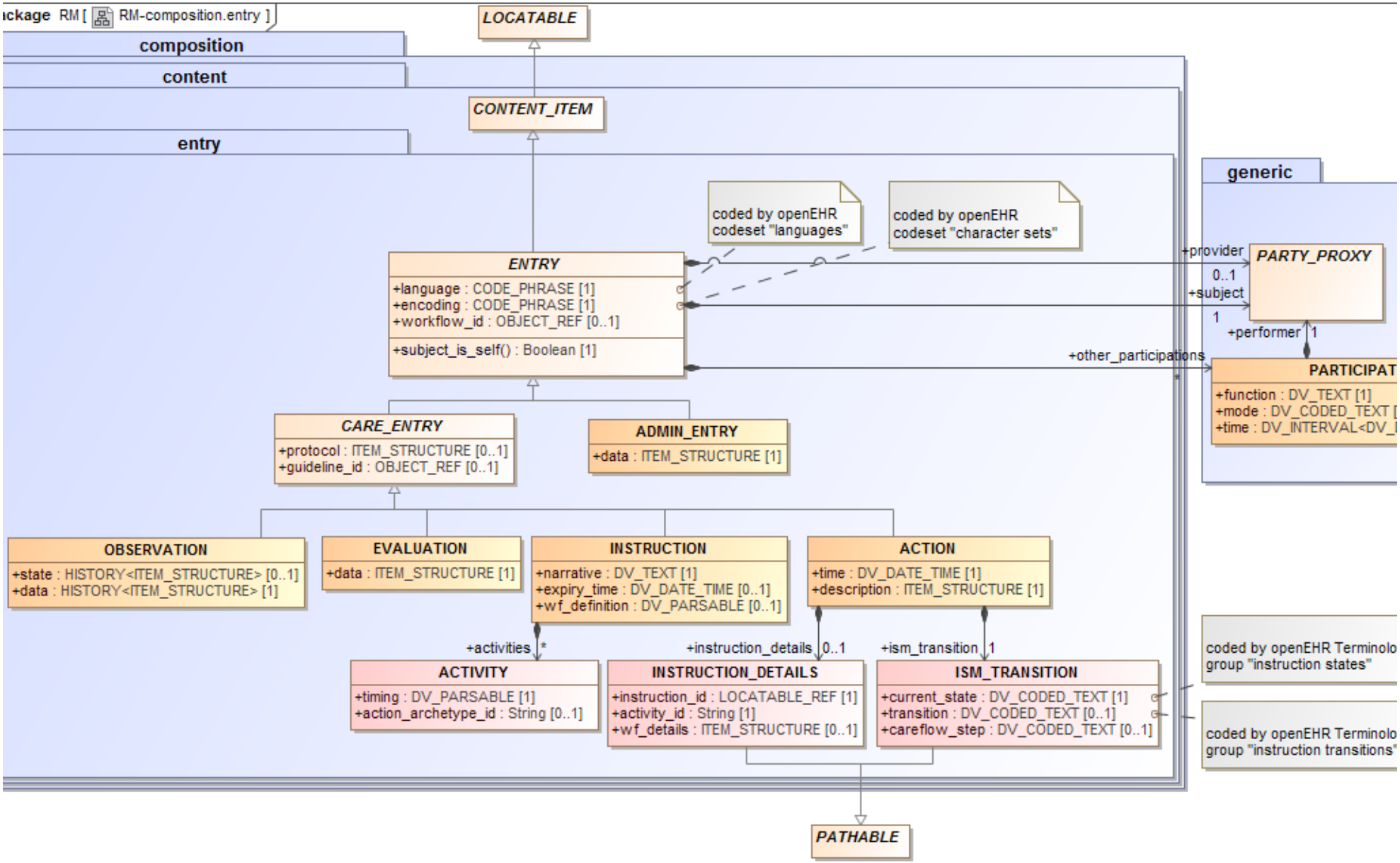


FIGURE 18 rm.composition.content.entry Package

Note that the term 'provider' as used here should not be confused with the more specific healthcare term used in many english-speaking countries meaning 'health care provider', which is usually understood to be a physician or healthcare delivery enterprise such as a hospital. In the model here, it simply means 'provider of information' in the context of an Entry. The information provider is optional, and in many cases will not be recorded, since it will be obvious from the *composer* and *other_participations* of the enclosing Composition. In many cases, it is not sensible to record a provider, e.g. in the most mundane case where a GP asks "where does it hurt" and the patient says "here" - in such cases, it can only be considered mutual. It is expected to be used only when the composer of the Composition really needs to specify the origin of specific statements, such as in the following circumstances:

- the information provider is specifically accountable for the Entry data (it is their opinion, their decision, they carried out the test etc) - they might also need to attest it;
- the information provider is an authoritative source, or has provided information from a unique perspective (e.g. the view of a spouse/ carer on the patient's functional health status or mental state);
- the information provider's view might not reflect the consensus (e.g. a patient opinion not held by the composer, a difference between father and mother on a description of a child's sleeping pattern);
- the information provider is *not* one of the Composition-level participants (e.g. an outside information provider such as someone telephoned during the encounter to provide a lab result, or an automated measurement device, or a decision support software component).

8.2.2 Care_entry and Admin_entry

A basic division occurs between clinical and non-clinical information. The `CARE_ENTRY` class is an abstract precursor of classes that express information of any clinical activity in the care process around the patient, while `ADMIN_ENTRY` is used to capture administrative information. The division may occasionally seem ambiguous at a theoretical level, but at a practical level, it is almost always clear. Administrative information has the following characteristics:

- it is created by non-clinical staff, or clinical staff acting in an administrative capacity (e.g. a nurse or doctor who has to fill out an admission form in A&E);
- it expresses details to do with *coordinating* the clinical process, by recording e.g. admission information (enables clinical staff to know who is in the hospital and where they can be found), appointments (ensuring patient and physician get together at an agreed time and place), discharge/dismissal (allowing clinical staff to know that a patient has been sent home healthy, or transferred to another institution), billing and insurance information (where such information is required in the EHR; it may well be in its own system);
- removing administative information from the EHR would not compromise its clinical integrity, it would simply mean that carers and patients would no longer know when and where they were supposed to meet, or when the patient has entered or left a health care facility.

Conversely, every instance of a `CARE_ENTRY` subtype is clinically significant, even if it also carries information which might be of interest to other health management functions billing (e.g. ICD10 coded diagnoses), practice management (e.g. date, time and place in an order for day surgery). The `CARE_ENTRY` type includes two attributes particular to all clinical entries, namely *protocol* and *guideline_id*.

These attributes allow the "how" and "why" aspects of any clinical recording to be expressed. Protocol is often recorded for Observations (e.g. staining method in microscopy) and Instructions (e.g.

8.2.3 Observation

Instances of the OBSERVATION class record the observation of any phenomenon or state of interest to do with the patient, including pathology results, blood pressure readings, the family history and social circumstances as told by the patient to the doctor, patient answers to physician questions during a physical examination, and responses to a psychological assessment questionnaire. Observations are distinguished from Actions in that Actions are interventions whereas Observations record only information relating to the situation of the patient, not what is done to him/her.

The significant information of an Observation is expressed in terms of "data", "state" and "protocol". The first of these is recorded in the *data* attribute, defined as follows:

data: the actual datum being recorded; expressed in the form of a History of Events, each of which can be a complex data structure such as a List, Table, Single (value), or Tree, in its own right. Examples include blood pressure, heartrate, ECG traces.

State information can be recorded in the *state* attribute of Observation, or within the *state* attribute of each Event in the Observation data attribute (see below and also Data Structures IM for more detailed explanations). The state attribute in Observation is defined as follows:

state: any particular information about the state of the subject of the Entry necessary to correctly interpret the data, which is not already known in the health record (i.e. facts such as the patient being female, pregnant, or currently undergoing chemotherapy). For example, exersion level (resting, post-marathon...), position (lying, standing), post- glucose challenge, and so on. The form of the *state* attribute is the same as that of the *data* attribute: a History of Events of Item_structures.

The inherited *protocol* attribute is defined as follows:

protocol: details of how the observation was carried out, which might include a particular clinical protocol (e.g. Bruce protocol for treadmill exercise ECG) and/or information about instruments other other observational methods. This information can always be safely omitted from the user interface, i.e. has no bearing on the interpretation of the data.

The detailed semantics of Observation data are described in the following subsections.

8.2.3.1 Timing in Observations

Many health information models express observation time as one or more attributes with names like 'observation_time', 'activity_time' and so on. The *open*EHR model departs from this by modelling historical time inside a History/Event structure defined in the data_structures.history package[1]. In short, this package defines the HISTORY class with an *origin* attribute, and a series of EVENT instances each containing a *time* attribute. Instantaneous and interval events are distinguished via the EVENT subtypes POINT_EVENT and INTERVAL_EVENT; interval events have the *width* attribute is set to the duration of the interval.

8.2.3.2 Valid Contents of a History

One of the aims of the model of Observation described here is to represent in the same way single sample and multi-sample time-based data for which *measurement protocol is invariant*. It is not intended for measurements in "coarse" time taken by different people, different instruments, or with any other difference in data-gathering technique. In these cases, separate, usually single-sample histories are used, usually occurring in distinct container objects (e.g. distinct Compositions, in the EHR).

1. Defined in the Data Structures IM; see http://svn.openehr.org/specification/TRUNK/publishing/architecture/rm/data_structures_im.pdf.

Accordingly, in the general practice setting, the use of HISTORY will correspond to measurement series which occur *during* the clinical session (i.e. during a patient contact). In a hospital setting, nurses' observations might occur in 4-hourly intervals, and there is no well-defined clinical session - simply a series of ENTRYs during the time of the episode. Two approaches are possible here.

- If each Observation is to be committed to the EHR as soon as it is made, the result will be distinct COMPOSITIONs in time, each with its *event_context* corresponding to the period of the nurse's presence. Each Composition will contain one or more Observations, each containing in their data a History of one sample of the measured vital sign.
- If Observations are not committed to the EHR immediately, but are stored elsewhere and only committed (say) at the end of each day, then the result will be a single Composition whose *event_context* corresponds to the total data gathering period, and which contains Observations whose data are multi-event Histories representing the multiple measurements made over the day.

Whether time-based data remain outside the record until a series of desired length is gathered, or entered as it occurs is up to the design of applications and systems; the approach taken should be based on the desired availability of the data in the system in question. If for example, it must be visible in the EHR as soon as the appropriate Compositions are written, then it should be represented as Histories in each relevant Composition; if it need only be available at some much later point in time (e.g. because it is known that no-one but the treating clinician is interested in it), then it can be stored in another system until sufficient items have been gathered for committal to the EHR.

8.2.3.3 Clinical Semantics of Event Time

In most cases, the times recorded in a History (HISTORY.*origin* and EVENT.*time*, *width*) can be thought of as "the times when the observed phenomena were true". For example, if a pulse of 88bpm is recorded for 12/feb/2005 12:44:00, this is the time at which the heartrate (for which pulse is a surrogate) existed. In such cases, the *sample* time, and the *measuring* time are one and the same.

However in cases where the time of sampling is different from that of measurement, the semantics are more subtle. There are two cases. The first is where a sample is taken (e.g. a tissue sample in a needle biopsy), and is tested later on, but from the point of view of the test, the time delay makes no difference. This might be because the sample was immediately preserved (e.g. freezing, placed in a sterile anaerobic transport container), or because even if it decays in some way, it makes no difference to the test (e.g. bacteria may die, but this makes no difference to a PCT analysis, as long as the biological matter is not physically destroyed).

The second situation is when the sample does decay in some way, and the delay is relevant. Most such cases are in pathology tests, where presence of live biological organisms (e.g. anaerobic bacteria) is being measured. The sample time (or 'collection' time) must be recorded. Depending on when the test is done, the results may be interpreted differently.

The key question is: what is the meaning of the HISTORY.*origin* and EVENT.*time* attributes in these situations? It is tempting to say that their values are (as in other cases) just the times of the actual act of observation, e.g. microscopy, chromatography etc. However, there are two problems with this. Firstly, and most importantly, all physical samples must be understood as being *indirect surrogates for some aspect of the patient state at the time of sampling,* which cannot be observed by direct, instantaneous means in the way a pulse can be taken. This means that no matter when the laboratory work is done, the time to which the result applies is the *sample* time. It is up to the laboratory to take into account time delays and effects of decay of samples in order to provide a test result which correctly indicates the state of the patient at the time of sampling. The common sense of this is clear when one considers the extreme case where the patient is in a coma or dead (possibly for reasons

completely unrelated to the problem being tested for) by the time laboratory testing actually occurs; however, the test result indicates the situation at the point in time when the sample was taken, i.e. when the patient was alive. The second reason is that some kinds of testing are themselves lengthy. For example fungal specimens require 4-6 weeks to confirm a negative result; checks will be made on a daily or weekly basis to find positive growth. However, the result data reported by the laboratory (and therefore the structure of the Observation) is not related to the timing of the laboratory testing; it is reported as being the result for the time of collection of the specimen from the patient.

The meaning therefore of the HISTORY.*origin* and EVENT.*time* attributes in *open*EHR is always the time of sampling. Where delays between sample and measurement times exist and are significant, they are noted in the protocol section of the Observation; such times are modelled in the appropriate archetypes, and taken into account in results.

8.2.3.4 Two ways of Recording State

State information is optional, and is not needed if the data are meaningful on their own. If it is recorded, it can either be as a History of its own (i.e. using the OBSERVATION.*state* atttribute described above), or else as *state* values within the EVENT instances in the OBSERVATION.*data* History. Both methods are useful in different circumstances. A separate state history is more likely to be used in a correlation study such as a sports medicine study on heartrate with respect to specific types of exercise. In this method, the state information is a History of Events whose times and widths need not match those of the History in the *data* attribute. The state data under this approach generally express the condition of the subject in *absolute* terms, i.e. they are standalone statements about the subject's state at certain points in time, such as "walking on treadmill 10km/h, 10^{o} incline".

The other method will be used in most general medicine, e.g. for recording fasting and post glucose challenge states of a patient undergoing a glucose tolerance test. (See the Data Structures Information Model for more details). State values stored within the *data* History represent the situation in the subject at the time of the Event within the History and usually in relation to it, for example "post 8 hour fast". Recording the latter example in an independent state History would require an Event of 8 hours' duration called 'fast'. The latter would be technically still correct, but would be very unnatural to most clinicians. FIGURE 19 illustrates the two methods of recording state.

8.2.4 Evaluation

According to the ontology described in [4], the Opinion category covers a number of concrete concepts, as follows.

problem/diagnosis: the assignment of a known diagnosis or problem label to a set of observed signs and symptoms in the patient, for the purpose of determining and managing treatment. The physician will usually include a date of initial onset, date clinically recognised, date of last occurrence, date of resolution of last occurrence and possibly other timing information.

risk assessment: an evaluation of the likelihood and timing of a certain event occurring or condition appearing.

scenario: an opinion about the outcome if a certain intervention is proceeded with.

goal: statement of a target, and a time at which it should be reached.

recommendation: a description in general terms of a suggested care approach for the patient, based on diagnosis; includes various times or time-periods for activities, such as monitoring, taking of medications, and review.

The approach taken to modelling these concepts in *open*EHR is heavily based upon the development of archetypes for assessments such as "diagnosis" (various kinds), "goal", "adverse reactions",

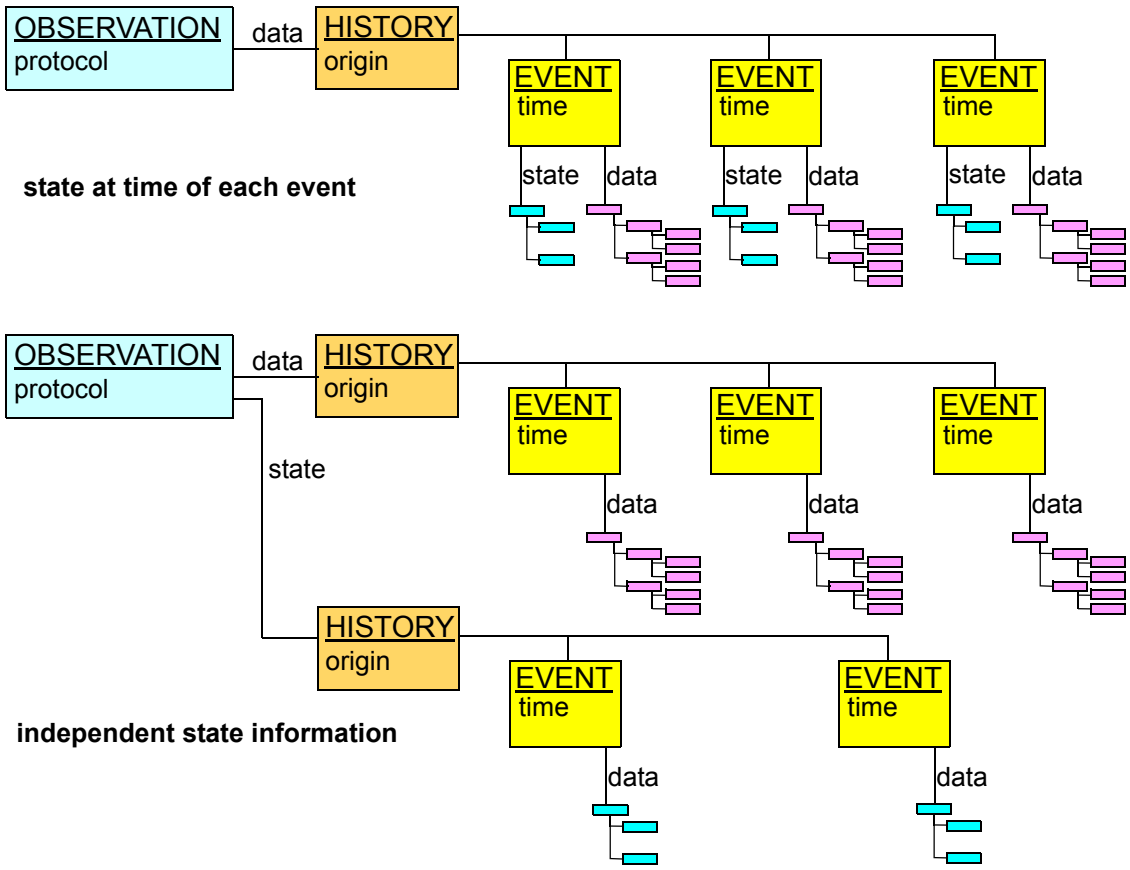


FIGURE 19 Alternative ways of recording state

"alert", "exclusion", "clinical synopsis", "risk based on family history" and so on. Experience has shown that the Opinion category is too variable for safely modelling its sub-categories directly in the reference model. Instead, a single class `EVALUATION` is used for all instances of the Opinion category. (The name Evaluation has been present in *open*EHR for some years, and is retained for reasons of continuity).

The design of the `EVALUATION` class is very simple. In addition to the attributes inherited from `ENTRY` and `CARE_ENTRY`, it has only one attribute, *data*: `ITEM_STRUCTURE`. This structure is intended to be archetyped so as to model all the details of any particular clinical information in the Opinion category. No timing attributes are included, since there is no time associated with creation or capturing of Evaluation information as such, only times included in the information. The only times of generic significance are (potentially) the time of a patient consultation during which the Evaluation was created (recorded in `COMPOSITION`.*event_context.start_time* and *end_time*) and the time of committal to the EHR system (recorded in `VERSION`.*commit_audit* attribute).

The general meaning of the inherited attributes is as for all Entries. In Evaluations, the *provider* is almost always the physician, while the *protocol* may be used to indicate how a particular assessment was made. The *other_participations* attribute is not as likely to be used for Evaluations representing diagnoses, since a diagnosis is usually the result of thinking on the part of the physician; an exception to this would be a case conference or if an expert system were used. However, Plans for complex patients may well be constructed by multiple physicians.

8.2.5 Instruction and Action

Instructions in *open*EHR specify actions to be performed in the future. They differ from information in the Proposal sub-category of Opinion in the ontology (i.e. instances of Evaluation in the class model) in that they are specified in sufficient detail to be directly enacted without further clinical decision-making related to the design of the Instruction, e.g they can be performed by the patient or a nurse. Any decisions that could be made during the performance of an Instruction are either constrained by the Instruction itself (e.g. dose range; suspend if adverse reaction) or else are assumed knowledge of the expected performer. For example, an Evaluation may say that "oral cortico-steroids are indicated at a peak flow of 200 l/m". A corresponding Instruction would indicate the actual drug, route, dose, frequency, and so on. The informed patient might be reasonably expected to be able to vary the dosage on his or her own within a dosage guideline explained by his/her GP.

In the ontology of FIGURE 17, Instructions are further categorised as Investigation and Intervention. However, as for Evaluation, only a single key class, `INSTRUCTION`, is used to model all types of the Instruction category, with archetypes defining the details of the Instruction. A second Entry subtype, `ACTION`, is used to model the information recorded due to the execution of an Instruction by some agent.

The following subsections describe Instruction and Action in some detail.

8.2.5.1 Requirements

The Instruction and Action classes are designed to satisfy the following requirements:

- All kinds of interventions, from simple medication orders to complex multi-drug courses should be representable using the same model;
- Instructions should always have a narrative expression, with an optional machine-processible expression in cases where automation will be used;
- The freedom must exist to model any particular intervention in as much or little detail as required by circumstances;
- Clinicians must be able to specify Instruction steps in their own terms, i.e. using terms like "prescribe", "dispense", "start administering", etc;
- Instructions representing diverse clinical workflows must be queryable in a standard way, so that it can be ascertained what Instructions are 'active', 'completed' and so on for a patient;
- It should be possible to provide a coherent view of the state of execution of an Instruction even if parts of it have been executed in different healthcare provider environments;
- It must be possible to record *ad hoc* actions in the record, i.e. acts for which no Instruction was defined (at least in the EHR in question);
- Instructions must integrate with notification / alert services;
- An interoperable expression of computable workflow definitions of Instructions will be supported.

8.2.5.2 Design Principles

The design approach is based on four principles. The first is that the *specification* of an Instruction as one or more Activities is distinguished from the information representing actions performed as a result. This makes the model and resulting information instances clear to software designers and data users. It also enables workflow engines to determine which parts of the specification have already been executed, and allows for Actions actually performed to differ from those specified. The separation is realised in terms of the `INSTRUCTION` and `ACTION` classes and their helpers. Instances of the

former specify an Instruction, while instances of the latter describe steps which have actually been performed.

The second principle is the use of a standard instruction state machine (ISM) defining standardised states and transitions for each Activity of an Instruction, no matter what its clinical meaning. The use of standardised states means that the execution state of any given Activity can be characterised in exactly the same way (e.g. 'planned', 'active', etc), and that it is therefore possible to query the EHR and find out all interventions of any kind in a particular state. The ISM is formally modelled in *open*EHR. The Instruction itself can also be considered to be in a state derived from the states of its Activities. An informal description of this aggregate state machine is given below.

The third principle is to provide a way of mapping steps in any care pathway process (i.e. healthcare business process) to states in the Instruction state machine. A care pathway process covers the entirety of steps required to effect an Instruction, for example prescribing, booking, dispensing, referring, suspension etc. Any such step when performed leaves the relevant Instruction in one of the states of the ISM.

The fourth principle is to support the expression of the formal workflow definition for an Instruction, where full automation is required. It must be recognised that automation of most therapies and drug admnistration, as well as other interventions like biopsies is minimal today, and is likely to remain so for some time. This is for the simple reason that the cost of automating most tasks is prohibitive compared to human execution, particularly when Instruction activities can often be executed by healthcare professionals already present for other reasons (e.g. ward nurses). It also has to be said that serious research into the use of workflow automation in healthcare is only quite recent, and that so far, there are no standard models for clinical workflow. In the *open*EHR approach to modelling workflow, such uncertainty is dealt with in two ways. Firstly, formal workflow specification of an Instruction is an optional addition to the base model of Instruction and Action classes, and is not required to obtain a basic level of computability, including use of the ISM. Secondly, the formal expression of workflow is in the form of parsable syntax rather than objects. This is a generally appropriate design choice, since the safest and most convenient persistent form of of a complex formalism is the syntax form rather than the parsed fine-grained object form; this both optimises storage and allows for changes in the syntax over time.

8.2.5.3 Model Overview

Instruction definitions are modelled in terms of the `INSTRUCTION` and `ACTIVITY` classes, with optional workflow attributes. These two classes carry the basic information relating to an Instruction, with all formal workflow definition expressed in parsable syntax in the `INSTRUCTION`.*wf_definition* attribute. An `INSTRUCTION` instance includes the narrative description of the Instruction, and a list of `ACTIVITY` instances. It also includes all the attributes inherited from the `CARE_ENTRY` class, including *subject*, *participations* and so on.

Many Instructions will have only one Activity, usually describing a medication to be administered and its timing. Some will have more than one drug or therapy, such as the typical 3 drug Losec-HP regime for treating ulcers, and multi-drug chemotherapy. The base Instruction model does not explicitly try to indicate the exact order, serial or parallel administration, or other dependencies, since the knowledge of how to administer the drugs is known by the relevant clinicians, and/or contained in published guidelines. However, the timing information in each Activity does indicate times, days and the usual specifications of "with meals" etc. The timing information is also sufficient to specify a three drug chemotherapy regime, by indicating which days each drug is administered on. It is only when the Instruction is to be automated by a workflow engine that the full structure of the Activity graph will be given. Activity instances may be completely absent from an Instruction, in which case

only the narrative will be present. This will typically occur with imported legacy data which itself has no structured representation of medications.

There are three possible levels of representation of an Instruction, as shown in FIGURE 20. These are the minimal narrative-only level, a level with formal representation of Instruction activities by ACTIVITY instances, and a level where a formal workflow definition can also be used. It is expected that the vast majority of *open*EHR systems for the forseeable future will support only the minimal and basic levels.

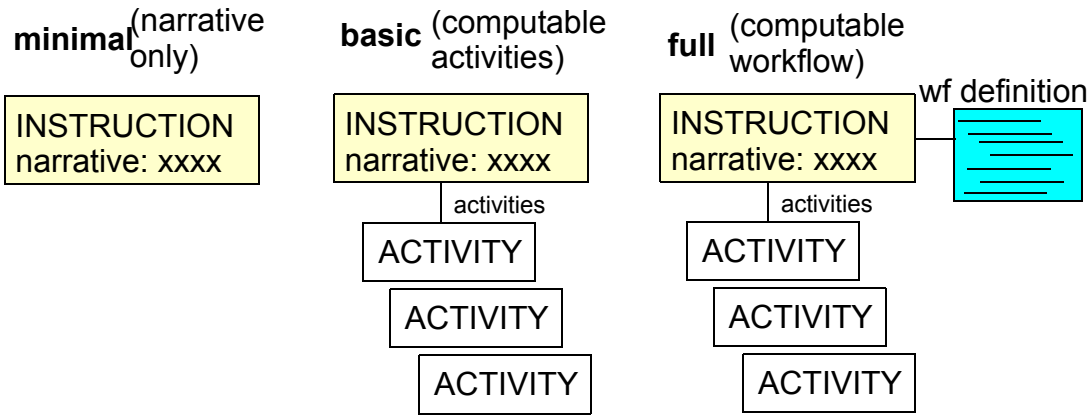


FIGURE 20 Levels of Instruction representation

FIGURE 21 illustrates the correspondence between Instruction and Activity structures, and Action objects that are created due to executing the instructions. Each Activity has a standard Instruction State Machine logically associated with it, indicated in blue. When any step in an Instruction Activity is executed within an ICT-supported environment, an ACTION instance describing what was done should be created and committed to the EHR. In some cases, *ad hoc* Actions are created even when there is no Instruction, such as by self-medicating patients and nurses reacting quickly to patient changes. For such Actions, there is always a notional Activity understood by the health professional.

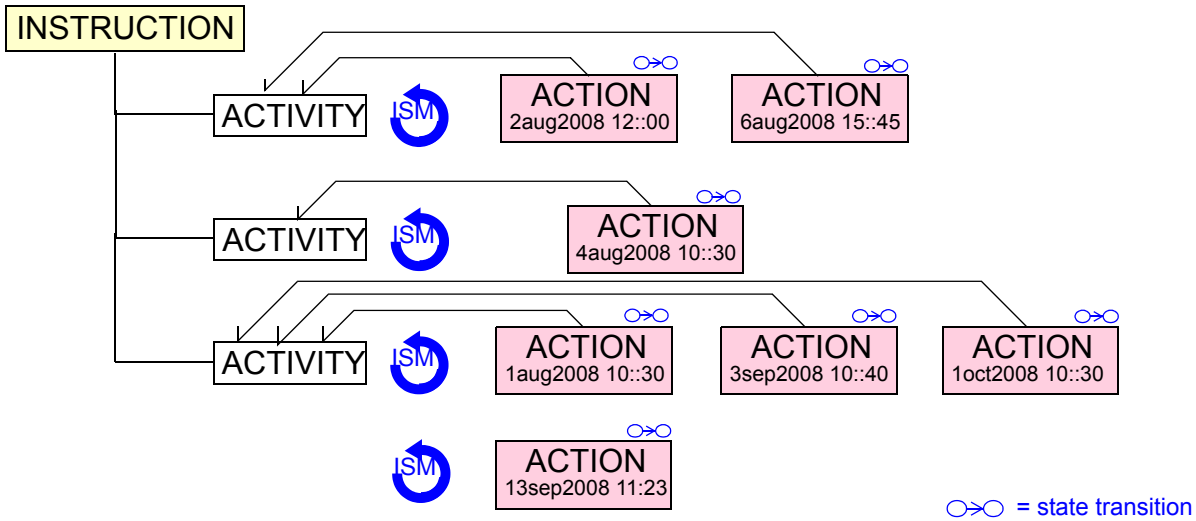


FIGURE 21 Correspondence of Actions to Instructions

All ACTIONs include the inherited CARE_ENTRY attributes, along with the time of being performed, a description of what was performed, and an ISM_TRANSITION object indicating the state of the

Activity (whether explicit in an Instruction or not). The current state of an Activity is thus found not in the ACTIVITY instance but in the most recent ACTION instance for that Activity.

If the Action did correspond to an Instruction, it will also include an INSTRUCTION_DETAILS instance, which indicates which Activity of which Instruction was executed, including workflow execution details if relevant.

Under this scheme, the state of each intervention happening to the patient can be ascertained by querying, regardless of whether explicit Instructions exist..

Note particularly that Actions are often performed in a different provider location, or the home, rather than the provider organisation responsible for the Instruction. Action objects for a given Instruction can thus easily appear in multiple health record systems.

8.2.5.4 The Standard Instruction State Machine (ISM)

When an intervention defined by an Activity is unfolding in a clinical context, EHR users want to know things such as the following:

- what is the current step in the Activity for the patient?
- for a given patient, what Activities are planned, active, suspended, completed?
- for the populations of patients, what is the state of a particular workflow, such as a recall, for each one?

The approach chosen here to support such functionality is to define a standard instruction state machine whose state transitions can be mapped to the steps of a specific care pathway, enabling it to be used as a descriptive device for indicating the state of any Activity. The state machine is illustrated in FIGURE 22. This state machine is the result of long term experience with clinical workflows and act management systems. The states are as follows[1].

INITIAL: initial state, prior to planning activity (default starting state for computable representation of state machine).

PLANNED: the action has been described, but has not as yet been performed.

POSTPONED: the action has not been performed and will not without specific conditions being met. Specifically, events and conditions that would normally ‘activate’ the instruction will be ignored, until a restore event occurs.

SCHEDULED: the action will be performed at some designated future time, and has been booked in a scheduling system.

CANCELLED: the action was defined, but was cancelled before being performed.

ACTIVE: the action is being performed according to its definition. The entire course of medication or therapy corresponds to this state.

SUSPENDED: the action was begun, but has been stopped temporarily, and will not be restarted until explicitly resumed.

ABORTED: the action began but was permanently terminated before normal completion.

COMPLETED: the action began and was completed normally.

EXPIRED: the time during which the action could have been relevant has expired; the action may have completed, been cancelled, or never occurred.

1. The SCHEDULED state was inspired by Van de Velde & Degoulet , Fig 5.5 [15]

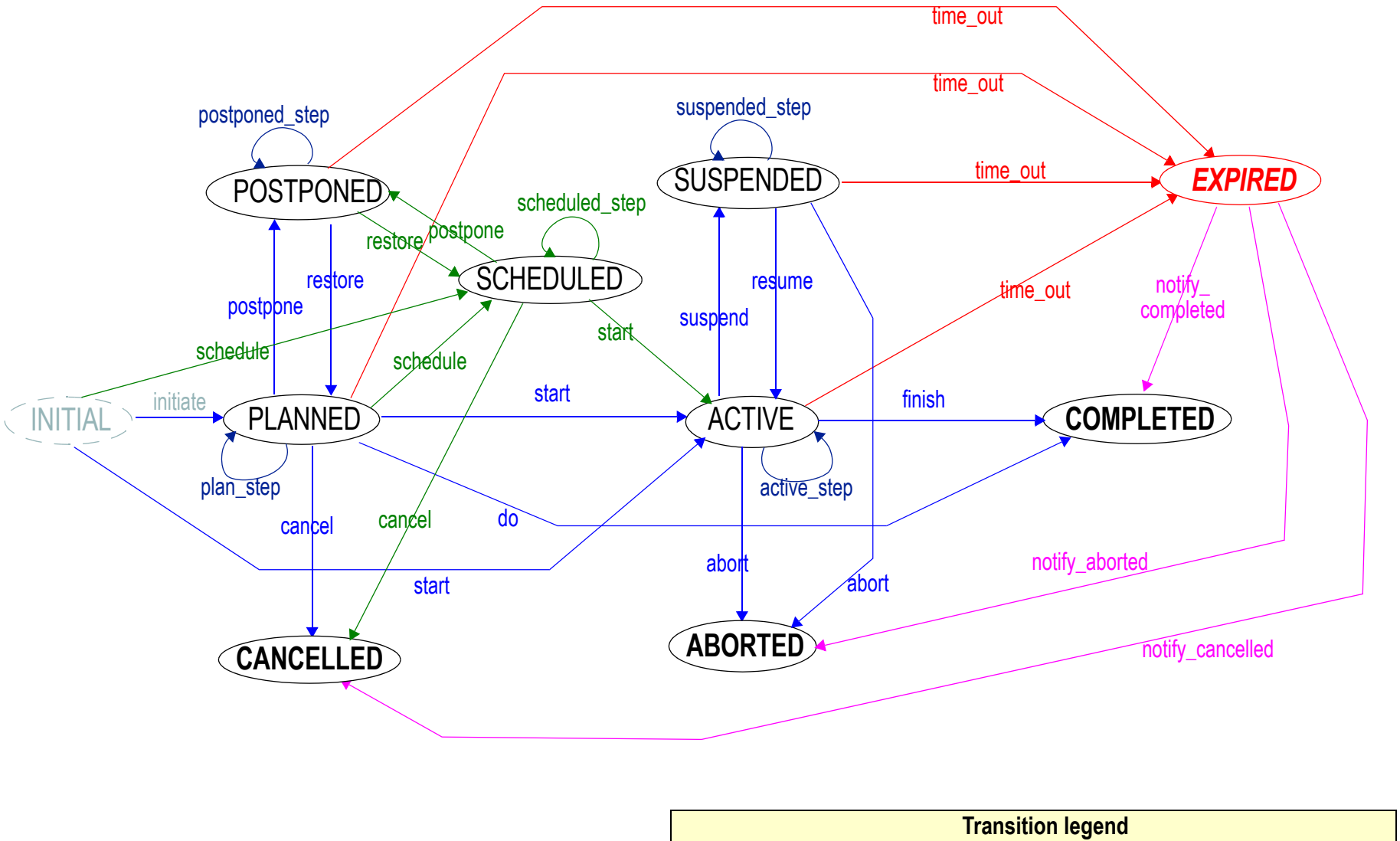


FIGURE 22 *open*EHR standard Instruction State Machine

States CANCELLED, ABORTED and COMPLETED are all terminal states. The EXPIRED state is a pseudo-terminal state, from which transitions are allowed to proceed to any of the true terminal states, due to information being received after the fact (such as a patient reporting that they did indeed finish a course of antibiotics). However it is likely in the EHR that Instructions for many simple medications will finish in the EXPIRED state and remain there.

The transitions are self-explanatory for the most part, however a few deserve comment. The *start* and *finish* events correspond to situations when the administration is not instantaneous, as is the situation with most medications. The *do* event is equivalent to the *finish* event occurring immediately after the *start* event, corresponding to an instantaneous administration, completion of which puts the Activity in the completed state. A single shot vaccination or patient taking a single tablet are typical examples. The states PLANNED, POSTPONED, ACTIVE, SUSPENDED, each have a xxx_step transition which return the state machine to the same state. Workflow steps that cause no transition are mapped to these events and thus leave the Instruction in the same state. An example is a medication review, which will leave the medication in the ACTIVE state, if the physician chooses to continue.

In the future, if delegation of an Instruction to another Instruction is required, nesting of a new Instruction state machine within the Active state of a previous one might need to be supported.

The state machine states and transition names are defined in the *open*EHR terminology groups "Instruction states" and "Instruction transitions" respectively.

8.2.5.5 Instruction Aggregate State

Each Activity within an Instruction constitutes a clinically identifiable medication or therapy of some kind, while the Instruction usually corresponds to a grouping or combination of therapies designed to treat an overall problem. Accordingly, the Instruction can be seen as having an aggregate state derived from the states of the Activities, as shown in FIGURE 23. The rules for entering each aggregate state are indicated in terms of the states of the constituent Activities.

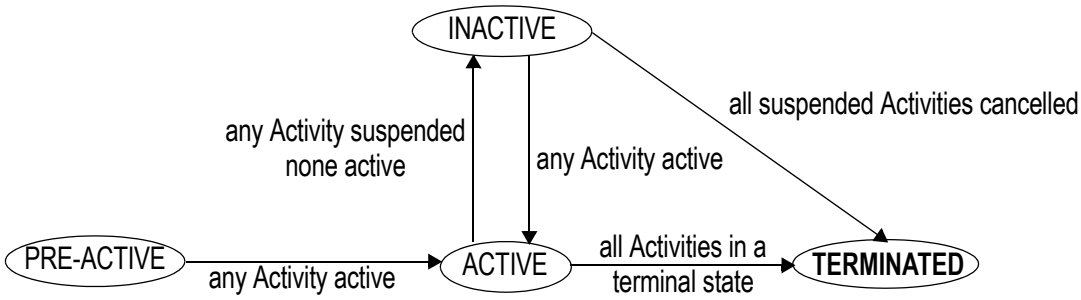


FIGURE 23 Instruction Aggregate State

This state machine is not formally modelled or coded in *open*EHR, although it may be useful to do so within an application.

8.2.5.6 Careflow Process to State Machine Mapping

From a health professional's point of view, a healthcare workflow, or 'careflow', consists of steps and events designed to meet one or more goals. The steps are highly dependent on the particular kind of workflow, and will usually be named in terms familiar to the relevant kinds of clinical professionals, such as "prescribe", "book", "suspend" and so on (note that some of these names may be the same as ISM transitions, but may or may not indicate the same thing). However, the need of users of health information is to know what state the execution of an Instruction is in, regardless of which particular careflow step might have just been executed. This is achieved by defining the mapping between the

steps of a particular careflow to the states of the ISM in an Action archetype. When each Action corresponding to a particular Instruction Activity is performed, it will be known both which careflow step it corresponds to and which ISM state the Activity is now in. The following table provides an example of the mapping for a UK GP medication order workflow.

UK GP Workflow Step	State machine transition
Start	initiate (initial -> planned)
Prescribe	planned_step (planned -> planned)
Dispense	start (planned -> active)
Administer	active_step (active -> active)
Request Renewal	active_step (active -> active)
Re-issue	active_step (active -> active)
Review	active_step (active -> active)
Finish	finish (active -> completed)
Cancel	abort (active -> aborted)

Mappings like this are specified in the archetypes for the Instruction. When an ACTION instance is committed to the EHR, the ISM_TRANSITION object records the step performed and the ISM state and transition. The careflow step must be one of the steps from the corresponding Instruction. See Relationship to Archetypes on page 62 for details of how archetypes are used to represent such mappings.

8.2.5.7 Relationship to Archetypes

Much of the semantics of particular Instructions and Actions derive from archetypes. Currently, archetypes are used to define two groups of Instruction semantics. The first is the descriptions of activities that are defined in Instructions (ACTIVITY.*description*) and executed in Actions (ACTION.*description*). These descriptions are always of the same form for any given Instruction, and it is highly desirable to have the same archetype component for both. An example is where the description is of a medication, commonly consisting of a tree or list of ten or more elements describing the drug, its name, form, dose, route and so on. The same information structure is needed in the Instruction, where it defines what is to be administered, and in the Action, where it describes what has been administered. In any particular instance, there may be small differences in what was administered (e.g. dose or route are modified) even though the archetype model will be the same for both.

In terms of archetypes therefore, definition of say two ACTIVITIES in an INSTRUCTION (see example illustrated in FIGURE 24) will actually create separate archetypes of the Activity structures, each of which will be one of the subtypes of ITEM_STRUCTURE (since this is the type of ACTIVITY.*description* and ACTION.*description*). The archetypes will then be used by both the INSTRUCTION archetype and the ACTION archetype, via the archetype slot mechanism (i.e. the standard way of composing archetypes from other archetypes; see use_archetype statements in FIGURE 24).

The second category of archetypable semantics is the correspondence between steps in a healthcare business process and the standard instruction state machine, as described above. This mapping is an archetype of the *ism_transition* attribute of an ACTION attribute, and therefore defines part of the ACTION archetype. FIGURE 24 shows how logical archetype elements in the archetyped editor environment corresponds to the resulting archetypes.

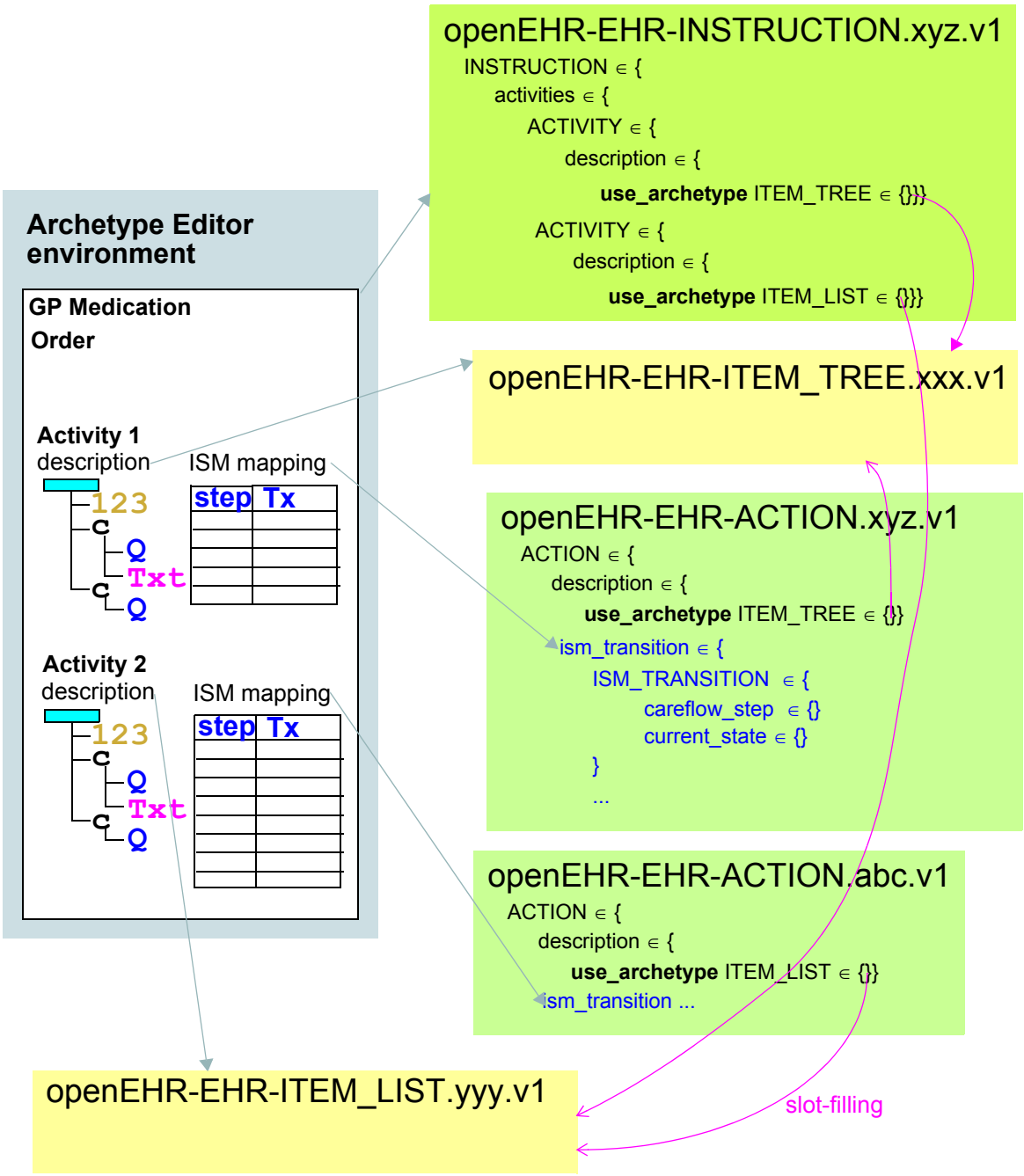


FIGURE 24 Instruction and Action Archetypes

8.2.6 Clinical Workflow Definition

Clinical workflows exist at multiple hierarchical levels, from the health system level (reduce diabetes costs; manage obesity) to the fine-grained (asthmatic medication prescription for a particular patient). At all levels, there are goals, actors and tasks designed to satisfy the goals. At a coarse-grained business process level, workflows may be enacted by more than one actor, and may encompass the whole cycle of Observation, Diagnosis, Instruction and Action. For example a workflow describing the steps "prescribe", "dispense", "administer", "repeat", "review" and so on, around a medication order might include GP, pharmacist, patient as actors. At the finer level of an actual drug or therapy administration, there is usually a single agent or group that performs a specific task, usually within one provider

institution, or at home. The correspondence between workflows at these different levels and *particular* patients, rather than just categories of patients (e.g. all insulin-dependent diabetics), typically increases at finer levels of granularity. Thus from the point of view of automation, it is likely to be fine-grained workflows that have patient-specific definitions which would reasonably appear in the EHR. Most typical medication administrations are in this category. How automated workflow definitions at higher levels of organisational hierarchy are represented and coordinated with lower-level automated workflows is known to be a difficult problem generally, and given that health computing is generally more complex than most domains, implementation of distributed, coordinated (or "orchestrated" or "choreographed", to use the terms of the workflow community) clinical workflows is likely to be some years off.

In this context, the possible scope for formal workflow definition in *open*EHR appears to be as follows.

- To enable the recording of links between *open*EHR Entries and workflow executions, e.g. a particular guideline. This allows *open*EHR data to be integrated with coarse-grained non-patient specific workflows.
- To support a standardised, interoperable representation of fine-grained formal workflow definitions for activities like medication administration.
- To use formal workflow definitions only where automation is actually useful, and is likely to be used. The kind of workflows that are likely to be worth automating are those which run over several days, weeks or longer, i.e. where humans might easily forget to perform a step. In these cases, the output of the workflow system will be reminders for humans to do certain things at certain times, rather than direct machine automation of the task. Execution of such workflow definitions will generate entries in "worklists" for staff or other agents to perform. Examples include asthma drug management for a child and PAP recall management.
- To ensure that any workflow definition takes account of other existing clinical activities, i.e. does not attempt to define all activities that might possibly be relevant. A simple example is a workflow for asthma medication administration probably does not need to explicitly model the taking of peak flow measurements, since this would normally be occurring anyway.

There are various technical challenges with proposing a standard workflow formalism for clinical use. Firstly, executable workflow definitions are essentially structured programs, similar to programs in procedural languages, but with the addition of temporal logic operators, including alternative paths, parallel paths, wait operations, and also references to outside data sources and services. Recent work in clinical workflow modelling e.g. [9], [1], [5] appears to favour a structural (i.e. parse-tree) approach to representation, due to the need to compute potential modifications to an executing workflow, including dropping, replacing and moving nodes. (Whether such "live" modification of executing workflows is realistic from the designer's point of view might be questionable, since it means that the design of each workflow has include every possible exceptional case at a detailed level.)

Secondly, the need to connect workflows to the outside world, i.e. data sources and services like notification and worklist management is crucial in making workflows and guidelines implementable. This problem is probably the main weakness of all guideline and workflow languages to date, including Arden, GLIF, and the various workflow languages such as those mentioned earlier.

The approach taken by the current release of *open*EHR in representing computable workflow is the following.

- Where used at all, formal workflow definitions are expressed in terms of syntax, not structures, since syntax is always a more appropriate representation for persistence (just as object structures, i.e. parse trees are more appropriate for computation).
- Access to patient data items are expressed within the syntax as symbolic queries.
- Actions, such as requests to the notification service are represented as symbolic commands.

The entire definition of a workflow is expressed as an optional parsable string, in the *wf_definition*: DV_PARSABLE attribute of the INSTRUCTION class. Any syntax may currently be used. A workflow syntax is under development by the *open*EHR Foundation, which is designed to incorporate the relevant features of current workflow models and research, while integrating it into the *open*EHR type system and archetype framework. In particular, early versions of this syntax will show how patient data access and service commands can be expressed.

8.3 Class Descriptions

8.3.1 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 event documented by the enclosing Composition. An ENTRY is also the minimal unit of information any query should return, since a whole ENTRY (including sub-parts) records spatial structure, timing information, and contextual information, as well as the subject and generator of the information.	
Inherit	CONTENT_ITEM	
Attributes	**Signature**	**Meaning**
1..1	**language**: CODE_PHRASE	Mandatory indicator of the localised language in which this Entry is written. Coded from *open*EHR Code Set "languages".
1..1	**encoding**: CODE_PHRASE	Name of character set in which text values in this Entry are encoded. Coded from *open*EHR Code Set "character sets".
1..1	**subject**: PARTY_PROXY	Id of human subject of this ENTRY, e.g.: • organ donor • foetus • a family member • another clinically relevant person.

CLASS	*ENTRY (abstract)*	
0..1	**provider :** PARTY_PROXY	Optional identification of provider of the information in this ENTRY, which might be: • the patient • a patient agent, e.g. parent, guardian • the clinician • a device or software Generally only used when the recorder needs to make it explicit. Otherwise, Composition composer and other participants are assumed.
0..1	**other_participations :** List <PARTICIPATION>	Other participations at ENTRY level.
0..1	**workflow_id :** OBJECT_REF	Identifier of externally held workflow engine data for this workflow execution, for this subject of care.
Functions	**Signature**	**Meaning**
1..1	**subject_is_self**: Boolean	Returns True if this Entry is about the subject of the EHR, in which case the *subject* attribute is of type PARTY_SELF.
Invariants	***Language_valid***: language /= Void ***and then*** code_set(Code_set_id_languages).has_code(language) ***Encoding_valid***: encoding /= Void ***and then*** code_set(Code_set_id_character sets).has_code(encoding) ***Subject_validity***: subject_is_self ***implies*** subject.generating_type = "PARTY_SELF" ***Other_participations_valid***: other_participations /= Void ***implies not*** other_participations.is_empty ***Archetype_root_point***: is_archetype_root	

8.3.2 ADMIN_ENTRY Class

CLASS	ADMIN_ENTRY	
Purpose	Entry subtype for administrative information, i.e. information about setting up the clinical process, but not itself clinically relevant. Archetypes will define contained information.	
Use	Used for admistrative details of admission, episode, ward location, discharge, appointment (if not stored in a practice management or appointments system).	
Misuse	Not used for any clinically significant information.	
Inherit	ENTRY	
Attributes	**Signature**	**Meaning**

CLASS	ADMIN_ENTRY	
1..1	**data**: ITEM_STRUCTURE	The data of the Entry; modelled in archetypes.
Invariants	***Data_valid***: data /= Void	

8.3.3 CARE_ENTRY Class

CLASS	*CARE_ENTRY (abstract)*	
Purpose	The abstract parent of all clinical ENTRY subtypes. A CARE_ENTRY defines protocol and guideline attributes for all clinical Entry subtypes.	
Inherit	ENTRY	
Attributes	Signature	Meaning
0..1	**protocol**: ITEM_STRUCTURE	Description of the method (i.e. *how*) 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 followed and executable; knowledge references such as a paper in Medline; clinical reasons within a largercare process.
0..1	**guideline_id**: OBJECT_REF	Optional external identifier of guideline creating this action if relevant
Invariants		

8.3.4 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, and patient-reported phenomena, e.g. pain.
MisUse	Not used for recording opinion or future statements of any kind, including instructions, intentions, plans etc.
Inherit	CARE_ENTRY

CLASS	OBSERVATION	
Attributes	**Signature**	**Meaning**
1..1	**data :** HISTORY <ITEM_STRUCTURE>	The data of this observation, in the form of a history of values which may be of any complexity.
0..1	**state :** HISTORY <ITEM_STRUCTURE>	Optional recording of the state of subject of this observation during the observation process, in the form of a separate history of values which may be of any complexity. State may also be recorded within the History of the *data* attribute.
Invariants	***Data_valid***: data /= Void	

8.3.5 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, risk assessments, goals and plans.	
MisUse	Should not be used for actionable statements such as medication orders - these are represented using the INSTRUCTION type.	
Inherit	CARE_ENTRY	
Attributes	**Signature**	**Meaning**
1..1	**data :** ITEM_STRUCTURE	The data of this evaluation, in the form of a spatial data structure.
Invariants	***Data_valid***: data /= Void	

8.3.6 INSTRUCTION Class

CLASS	INSTRUCTION
Purpose	Used to specify actions in the future. Enables simple and complex specifications to be expressed, including in a fully-computable workflow form.
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 directly executed by an actor, either human or machine.
Misuse	Not to be used for plan items which are only specified in general terms.

CLASS	INSTRUCTION	
Inherit	CARE_ENTRY	
Attributes	**Signature**	**Meaning**
1..1	**narrative** : DV_TEXT	Mandatory human-readable version of what the Instruction is about.
0..1	**activities** : List<ACTIVITY>	List of all activities in Instruction.
0..1	**expiry_time** : DV_DATE_TIME	Optional expiry date/time to assist determination of when an Instruction can be assumed to have expired. This helps prevent false listing of Instructions as Active when they clearly must have been terminated in some way or other.
0..1	**wf_definition** : DV_PARSABLE	Optional workflow engine executable expression of the Instruction.
Invariants	***Narrative_valid***: narrative /= Void ***Activities_valid***: activities /= Void **implies not** activities.is_empty	

8.3.7 ACTIVITY Class

CLASS	ACTIVITY	
Purpose	Defines a single activity within an Instruction, such as a medication administration.	
Inherit	LOCATABLE	
Attributes	**Signature**	**Meaning**
1..1	**description** : ITEM_STRUCTURE	Description of the activity, in the form of an archetyped structure.
1..1	**timing** : DV_PARSABLE	Timing of the activity, in the form of a parsable string, such as an HL7 GTS or ISO8601 string.
1..1	**action_archetype_id** : String	Perl-compliant regular expression pattern, enclosed in '//' delimiters, indicating the valid archetype identifiers of archetypes for Actions corresponding to this Activity specification. Defaults to "/.*/", meaning any archetype.
Invariants	***Description_valid***: description /= Void ***Timing_valid***: timing /= Void ***Action_archteype_id_valid***: action_archetype_id /= Void **and** action_archetype_ids.is_empty	

8.3.8 ACTION Class

CLASS	ACTION	
Purpose	Used to record a clinical action that has been performed, which may have been *ad hoc*, or due to the execution of an Activity in an Instruction workflow. Every Action corresponds to a careflow step of some kind or another.	
Inherit	CARE_ENTRY	
Attributes	**Signature**	**Meaning**
1..1	**time**: DV_DATE_TIME	Point in time at which this action completed.
1..1	**description**: ITEM_STRUCTURE	Description of the activity that was performed (which could differ from the activity prescribed in the originating Instruction), in the form of an archetyped structure.
1..1	**ism_transition**: ISM_TRANSITION	Details of transition in the Instruction state machine caused by this Action.
0..1	**instruction_details**: INSTRUCTION_DETAILS	Details to of the Instruction that caused this Action to be performed, if there was one.
Invariants	***Time_valid***: time /= Void ***Description_valid***: description /= Void ***Ism_transition_valid***: ism_transition /= Void	

8.3.9 INSTRUCTION_DETAILS Class

CLASS	INSTRUCTION_DETAILS	
Purpose	Used to record details of the Instruction causing an Action.	
Inherit	PATHABLE	
Attributes	**Signature**	**Meaning**
1..1	**instruction_id**: LOCATABLE_REF	Reference to causing Instruction.
1..1	**activity_id**: String	Identifier of Activity within Instruction, in the form of its archetype path.

CLASS	INSTRUCTION_DETAILS	
0..1	**wf_details:** ITEM_STRUCTURE	Various workflow engine state details, potentially including such things as: • condition that fired to cause this Action to be done (with actual variables substituted); • list of notifications which actually occurred (with all variables substituted); • other workflow engine state. This specification does not currently define the actual structure or semantics of this field.
Invariants	***Instruction_id_valid***: instruction_id /= Void ***Activity_path_valid***: activity_id /= Void ***and then not*** activity_id.is_empty	

8.3.10 ISM_TRANSITION Class

CLASS	ISM_TRANSITION	
Purpose	Model of a transition in the Instruction State machine, caused by a careflow step. The attributes document the careflow step as well as the ISM transition.	
Inherit	PATHABLE	
Attributes	**Signature**	**Meaning**
1..1	**current_state:** DV_CODED_TEXT	The ISM current state. Coded by *open*EHR terminology group “Instruction states”.
0..1	**transition:** DV_CODED_TEXT	The ISM transition which occurred to arrive in the *current_state*. Coded by *open*EHR terminology group “Instruction transitions”.
0..1	**careflow_step:** DV_CODED_TEXT	The step in the careflow process which occurred as part of generating this action, e.g. “dispense”, “start_administration”. This attribute represents the clinical label for the activity, as opposed to *current_state* which represents the state machine (ISM) computable form. Defined in archetype.
Invariants	***Current_state_valid***: current_state /= Void ***and then*** terminology(Terminology_id_openehr). has_code_for_group_id(Group_id_instruction_states, current_state.defining_code) ***Transition_valid***: transition /= Void ***implies*** terminology(Terminology_id_openehr). has_code_for_group_id(Group_id_instruction_transitions, transition.defining_code)	

8.4 Instance Structures

The following subsections illustrate typical Entry instance structures. For guidance on how to best model particular clinical statements, see the archetype part of the *open*EHR knowledge repository (http://svn.openehr.org/knowledge/project_page.htm).

8.4.1 OBSERVATION

Heartrate Measurement Series

FIGURE 25 illustrates three heartrate measurements over 10 minutes.

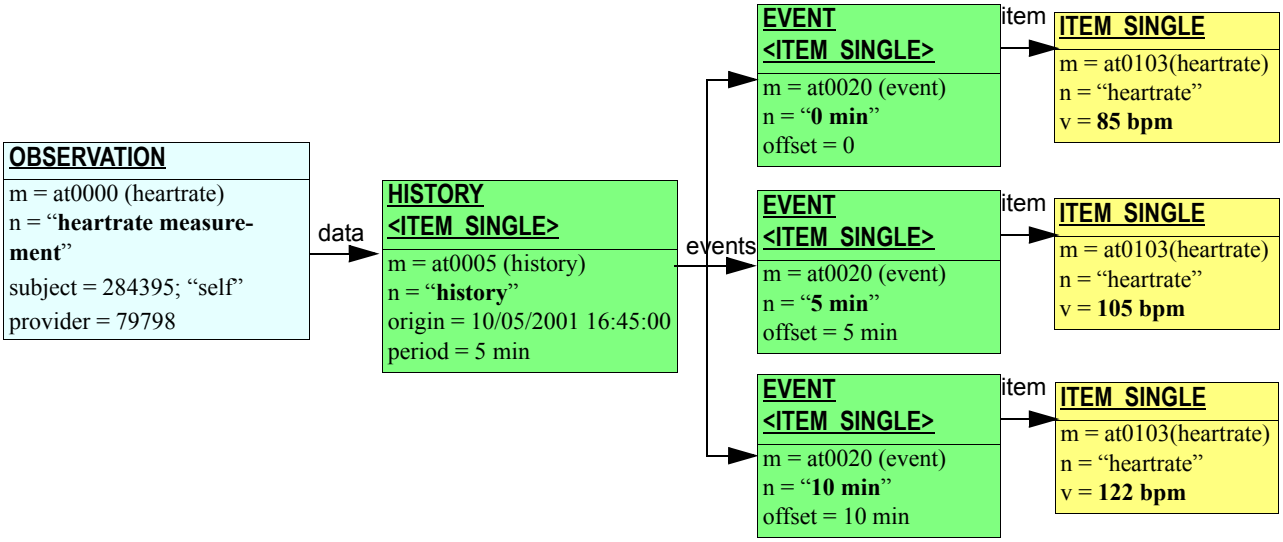


FIGURE 25 Periodic series Instance Structure

Blood Pressure with Protocol

FIGURE 26 illustrates a blood pressure observation with protocol.

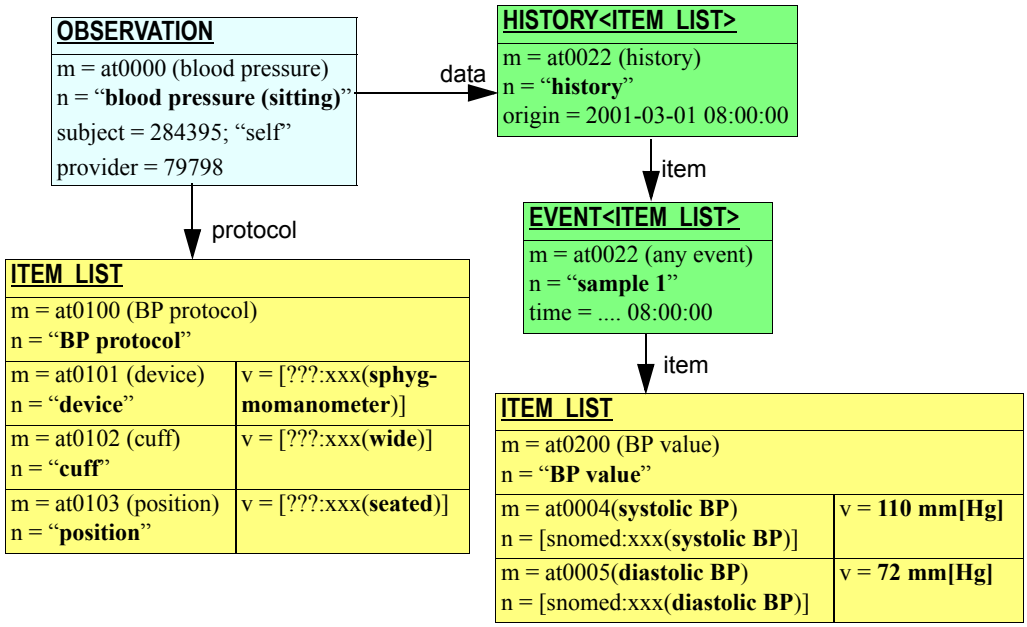


FIGURE 26 Blood Pressure Measurement Observation

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

OGTT is treated as a single clinical concept, and thus requires only one archetype. A typical instance structure is shown in FIGURE 27. In this example, the three blood sugars are represented by `EVENT`s, with the fasting and glucose challenges being expressed as states on the relevant events.

8.4.2 EVALUATION

Partial Asthma Management Plan

FIGURE 28 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 complete plan, symptom monitoring and other medications might be shown. The parts of the plan are

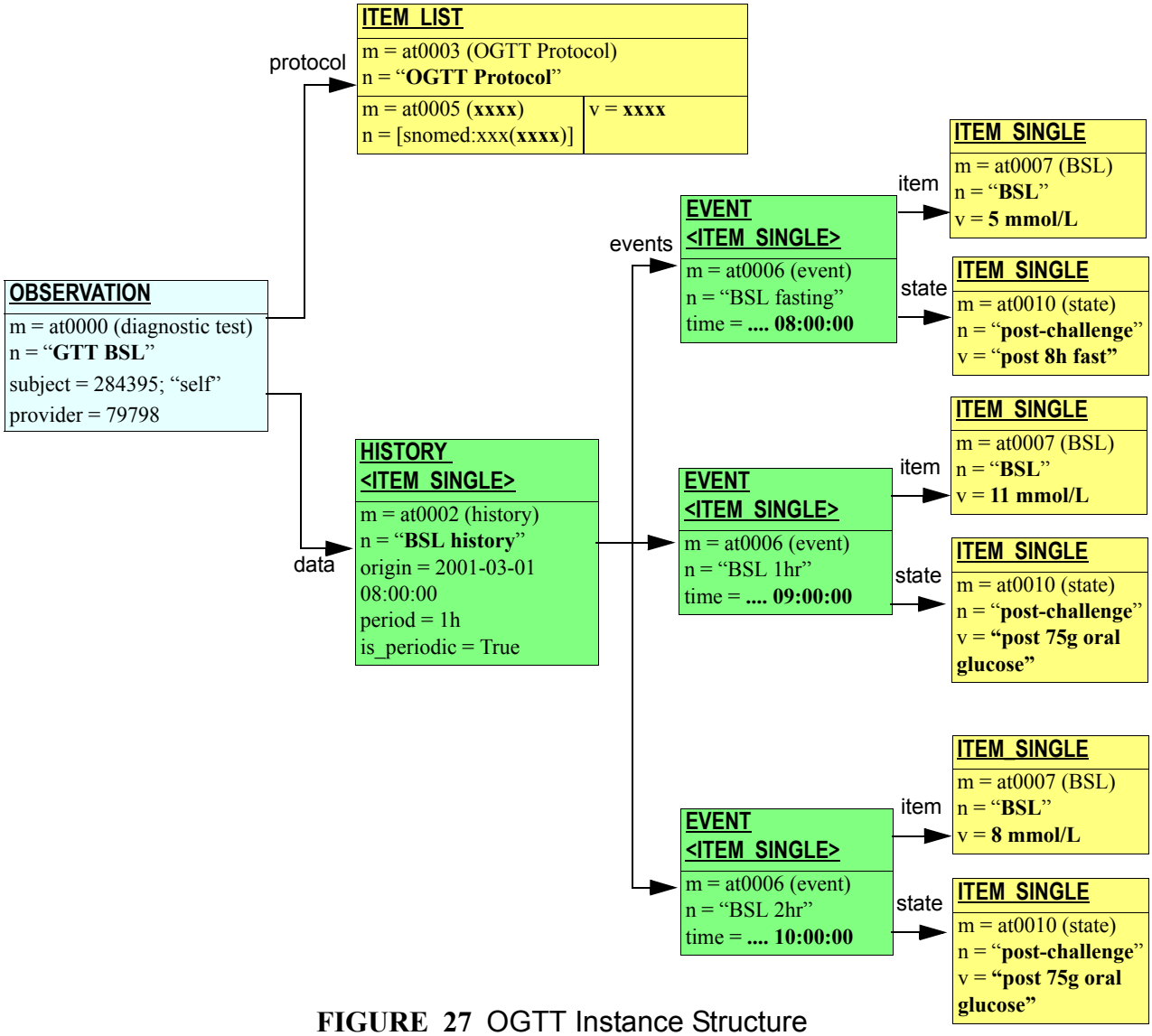


FIGURE 27 OGTT Instance Structure

linked to the root `EVALUATION` node via the *links*: `Set<LINK>` attribute inherited from the `LOCATABLE` class.

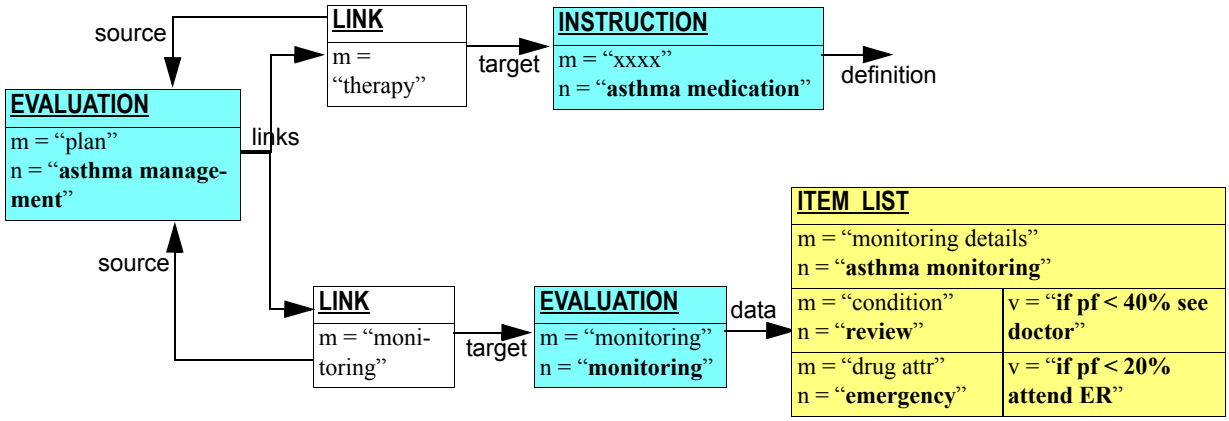


FIGURE 28 Partial Asthma Management Plan

8.4.3 INSTRUCTION

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.

FIGURE 29 illustrates the instance structure for this Instruction. Note that the *timing* attribute of the `ACTIVITY` instance is shown in human-readable form; in reality it will be a GTS string or similar (see Timing Specification section of *open*EHR Data Types IM).

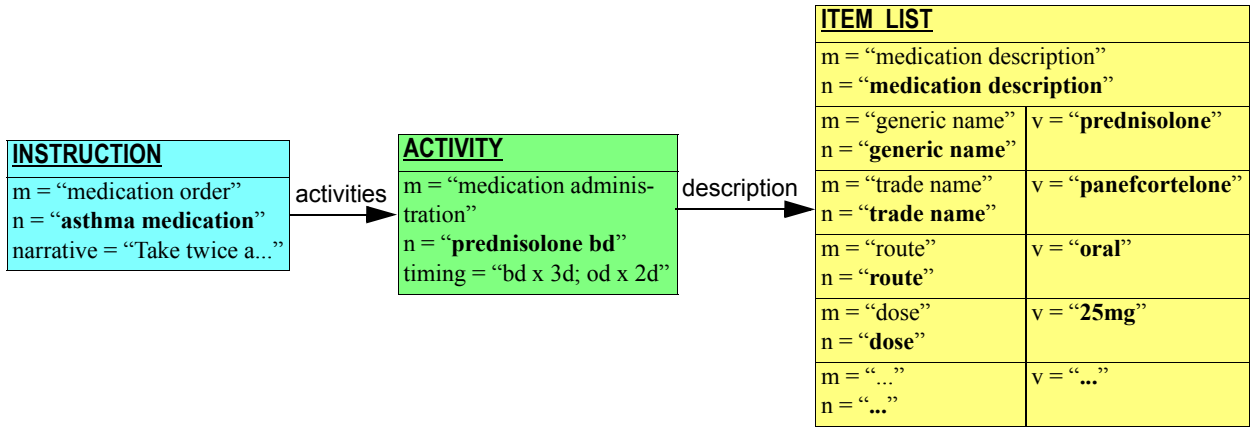


FIGURE 29 Chained medication order

Multi-drug Therapy

A common regime for treating duodenal ulcer and related complaints is using Losec with other drugs, such as in the following combination:

- Losec 40 mg od x 4w or until no symptoms
- amoxicillin 500 mg 3td x 7d
- metronidizole 400 mg 3td x 7d

The Instruction for this therapy is illustrated in FIGURE 30.

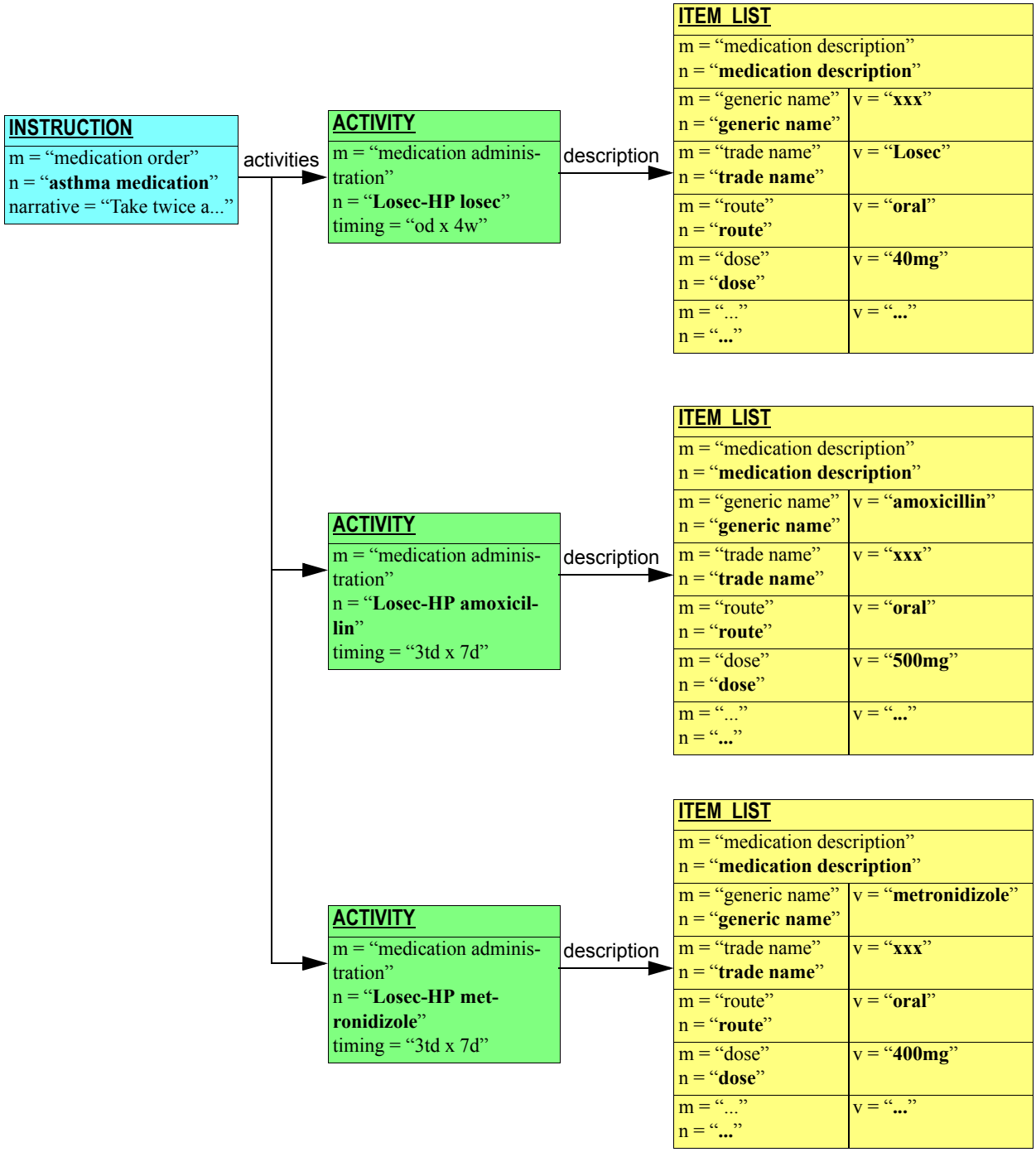


FIGURE 30 Multi-drug therapy Instruction

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 Barretto S A. *Designing Guideline-based Workflow-Integrated Electronic Health Records*. 2005. PhD dissertation, University of South Australia. Available at http://www.cis.unisa.edu.au/~cissab/Barretto_PhD_Thesis_Revised_FINAL.pdf.

2 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.

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

4 Beale T, Heard S. *An Ontology-based Model of Clinical Information*. 2007. pp760-764 Proceedings MedInfo 2007, K. Kuhn et al. (Eds), IOS Publishing 2007. See http://www.openehr.org/publications/health_ict/MedInfo2007-BealeHeard.pdf.

5 Browne E D. Workflow Modelling of Coordinated Inter-Health-Provider Care Plans. 2005. PhD dissertation, University of South Australia. Available at http://www.openehr.org/publications/workflow/t_browne_thesis_abstract.htm.

6 Elstein AS, Shulman LS, Sprafka SA. *Medical problem solving: an analysis of clinical reasoning*. Cambridge, MA: Harvard University Press 1987.

7 Elstein AS, Schwarz A. *Evidence base of clinical diagnosis: Clinical problem solving and diagnostic decision making: selective review of the cognitive literature*. BMJ 2002;324;729-732.

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

9 Müller R. Event-oriented Dnamic Adaptation of Workflows: Model, Architecture, and Implementation. 2003. PhD dissertation, University of Leipzig. Available at http://www.openehr.org/publications/workflow/t_mueller_thesis_abstract.htm.

10 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.

11 Rossi-Mori A, Consorti F. *Assembling clinical information*. Note sent to HL7v3 discussion list, 2001-04-19.

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

13 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.

14 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.

15 Van de Velde R, Degoulet P. *Clinical Information Systems: A Component-Based Approach*. 2003. Springer-Verlag New York.

16 Weed LL. Medical records, medical education and patient care. 6 ed. Chicago: Year Book Medical Publishers Inc. 1969.

B.2 European Projects

17 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.

18 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.

19 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.

20 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.

21 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.

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

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

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

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

26 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.

27 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.

28 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

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

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

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

32 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

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

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

B.5 HL7v3

35 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

36 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+).

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

B.6 OMG

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

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

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

B.7 Software Engineering

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

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

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

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

45 Fowler M, Scott K. *UML Distilled (2nd Ed.)*

Addison Wesley Longman 2000

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

B.8 Resources

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

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

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

50 EON ref required

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

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

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

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

END OF DOCUMENT