REFERENCE MODEL

The *open*EHR EHR Information Model

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

Revision: 5.0rc7

Pages: 83

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

The *open*EHR foundation

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

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

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

Copyright Notice

Amendment Record

Issue	Details	Raiser	Completed
5.0rc7	**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_SUMMARY class to EHR package.	S Heard S Heard T Beale T Beale 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	15 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

Thanks to...

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

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

1 Introduction

1.1 Purpose

This document describes the *open*EHR EHR Information Model, which is a model of an interoperable EHR in the ISO RM/ODP information viewpoint. This model is somewhat different in scope from models such as the CEN ENV 13606 pre-standard and the HL7v3 Clinical Document Architecture (CDA), in that it describes a logical EHR information architecture rather than just an architecture for communication of EHR extracts or documents between EHR systems. The EHR equivalent of these latter specifications is given in the *open*EHR EHR_EXTRACT Information Model.

The intended audience includes:

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

1.2 Related Documents

Prerequisite documents for reading this document include:

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

Other documents describing related models, include:

- The *open*EHR EHR_EXTRACT Information Model
- The *open*EHR Demographic Information Model

1.3 Status

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

The latest version of this document can be found in PDF format at http://svn.openehr.org/specification/BRANCHES/Release-1.0-candidate/publishing/architecture/rm/ehr_im.pdf. New versions are announced on openehr-announce@openehr.org.

Blue text indicates sections under active development.

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

1.4 Peer review

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

To Be Continued: more work required

Reviewers are encouraged to comment on and/or advise on these paragraphs as well as the main content. Please send requests for information to info@openEHR.org. Feedback should preferably be provided on the mailing list openehr-technical@openehr.org, or by private email.

1.5 Conformance

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

2 Background

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; [19]), the following broad requirements areas were identified:

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

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

2.1.2 GEHR Australia Requirements

The GEHR Australia project (1997-2001; [32], [33]) introduced further requirements, including:

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

These requirements can be found at [33]. GEHR Australia produced a proof of concept implementation in which clinical archetypes were developed and used. See [3] for the technical description of archetypes.

2.1.3 European Synapses and SynEx Project Requirements

Following the original Good European Health Record project the EU-funded Synapses (1996-8; [25]) 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) [26];

- 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) [27];
- the core technical requirements of and interfaces for a federation middleware service [25].

2.1.4 European EHCR Support Action Requirements

This EU Support Action project ("SupA"; [17], [18], [20]) consolidated the requirements published by a wide range of European projects and national health informatics organisations as a Consolidated List of Requirements [16].

2.1.5 ISO EHR Requirements

The above requirements publications and the recent experience of *open*EHR feed into the definition of a set of EHR requirements by ISO Technical Committee 215 (Health Informatics) - ISO TS 18308. The present draft [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 for the *open*EHR proposal, based on previous experience in the projects mentioned above include the following:

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

2.2 Design Principles

There are numerous considerations outside the requirements which influence the information model, some of which have been described by Beale & Heard, e.g. [4], [5], [6]. These are summarised as follows:

- System-of-systems understanding of information infrastructures (i.e. a set of collaborating middleware components and services), described in more detail below;
- Design paradigm: two level modelling, archetypes;
- Separation of models according to responsibilities;
- Ontological framework within which reference models and archetypes fit;
- A "context model" of information acquisition which describes contexts for each information fragment from the most detailed information structures to the context of the clinical session, and the enterprise;

- The principle of relegating the vagaries of implementation technologies to separate specifications, rather than compromising the central model in any way.

2.2.1 The System-of-systems EHR Context

Health care information systems, both at the level of a single enterprise and regional healthcare networks, increasingly comprise a set of software components collaborating through middleware, enabling distributed operations and data exchange. This "system-of-systems" view embraces the distributed object paradigm (exemplified by CORBA and .net), the message-based paradigm (exemplified by HL7), and in a less formal sense, most of today's organically evolved multi-database hospital IT environments. The distributed systems paradigm is also the generally accepted theoretical and standards view, as exemplified by any modern textbook on information processing, by the ISO RM/ODP standard, by health informatics standards such as the OMG Corbamed [37], [38], [39] specifications and CEN family of health information standards ([28] and many others), and by a wide range of international projects, national health information strategies and demonstrator pilots.

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

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

2.3 Relationship to other Health Information Models

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

2.3.1 CEN TC/251 prEN13606

These models have been influenced by and have also influenced the models in CEN prEN13606; accordingly, relationships to 13606 have been documented fairly precisely. There are some parts of this pre-standard which are ambiguous, or confusing, particularly where similar semantics appear in both parts I and IV, in which case the best attempt to understand the standard has been made.

Since January 2002, the 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).

2.3.2 HL7 Version 3

Correspondences to some parts of HL7 version 3 (ballot 5, July 2003) are also documented where possible, however, it should be understood that there are a number of difficulties with this. Firstly, while the HL7v3 Reference Information Model (RIM) - the closest HL7 artifact to an information model - provides similar data types and some related semantics, it is not intended to be a model of the EHR. In fact, it differs from the information model presented here (and for that matter most published information models) in two basic aspects: a) it is an amalgam of semantics from many systems which would exist in a distributed health information environment, rather than a model of just one (the EHR); b) it is also not a model of data, but an “analysis pattern” in the sense of Fowler [43] from which further specific models - subschemas - are developed by a process of custom restriction, in order to arrive at message definitions. As a consequence, data in messages are not instances of HL7v3 RIM classes, as would be the case in other systems based on information models of the kind presented here.

Despite the differences, there are some areas that appear to be candidates for fairly direct mapping, specifically the data types and terminology use, and the correspondence between *open*EHR Compositions and the HL7 Clinical Document Architecture (CDA).

2.3.3 OMG HDTF

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

3 The EHR Information Model

3.1 Overview

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

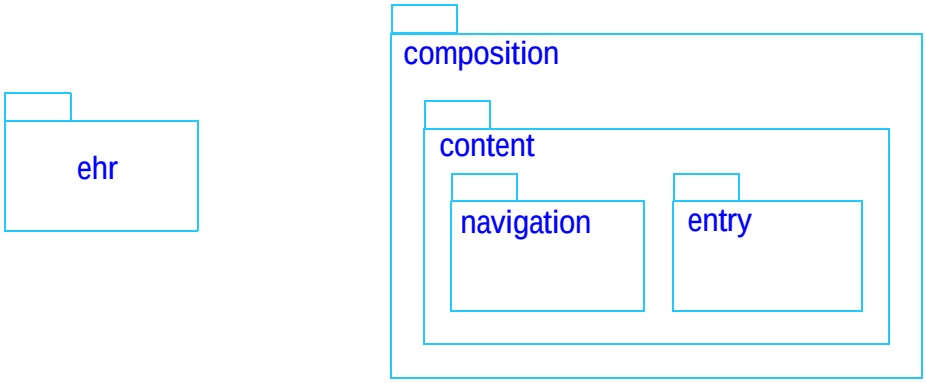

FIGURE 1 ehr Package Structure

The packages are as follows:

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

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

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

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

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

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

3.2 Archetypes

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

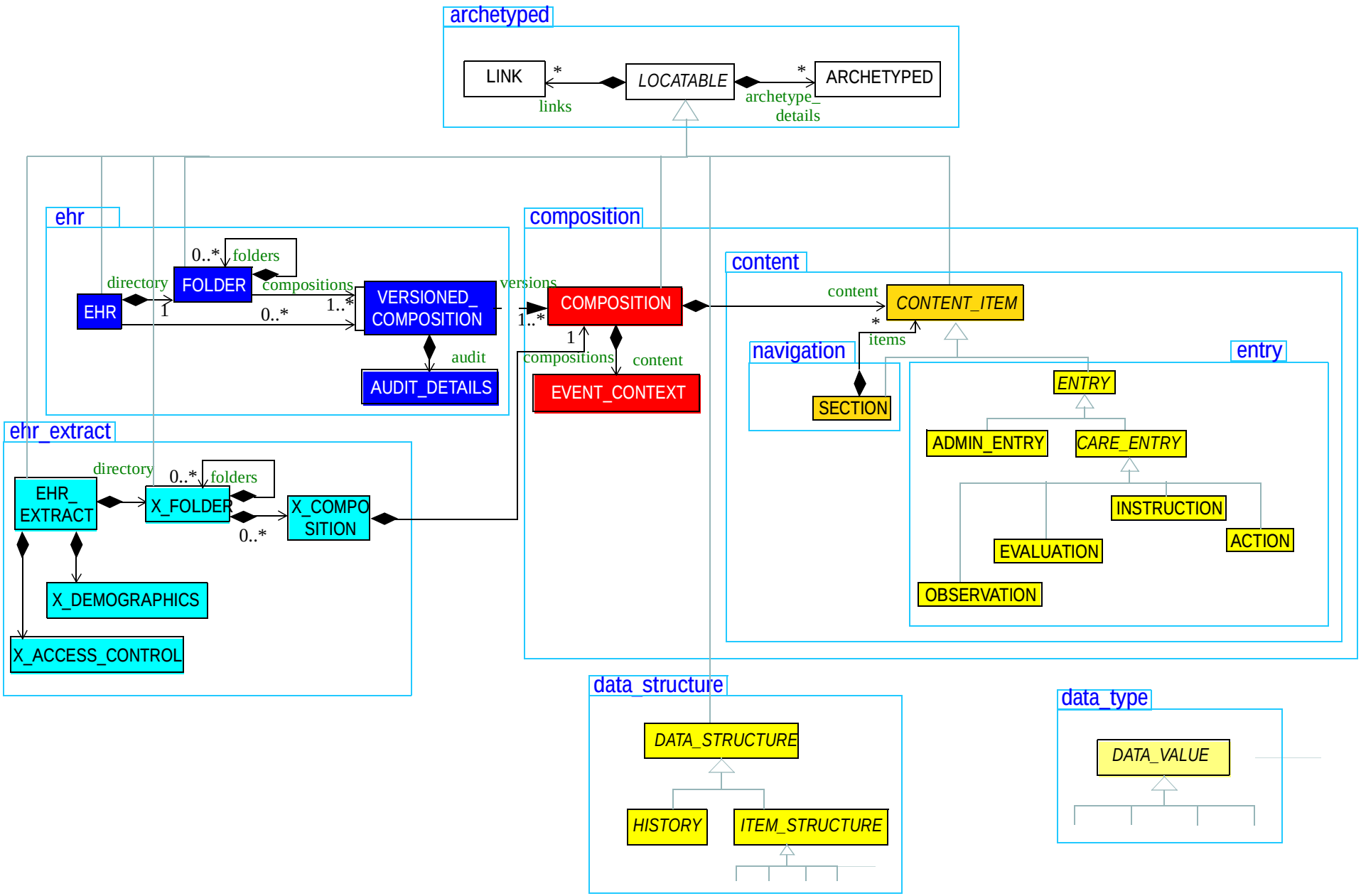

FIGURE 2 *open*EHR and EHR Extract Information Model Overview

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

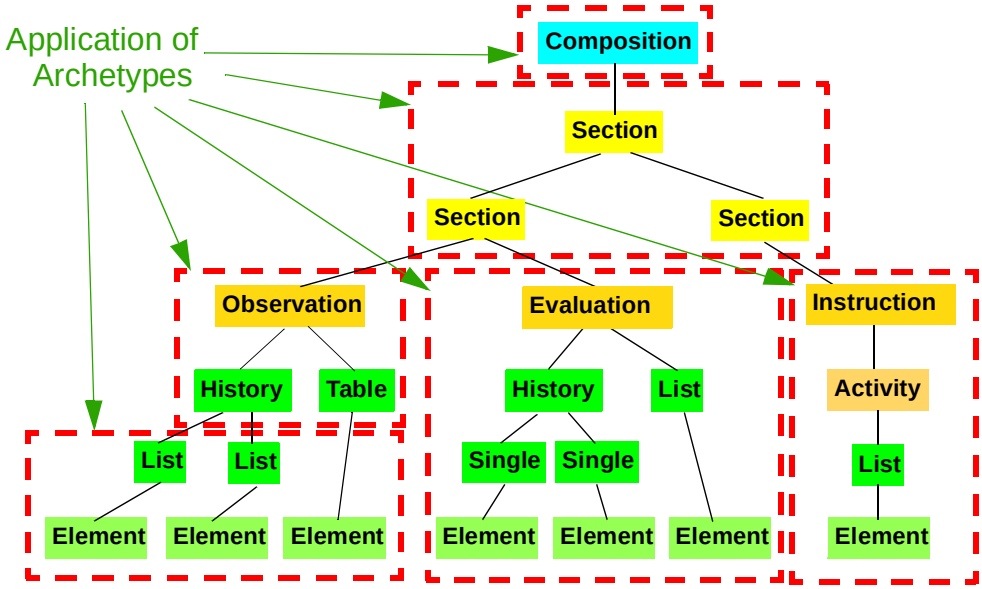

FIGURE 3 How Archetypes apply to Data

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

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

3.3 Paths

3.3.1 Runtime Path Syntax

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

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

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

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

This gives rise to paths of the form:

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

/[root_obj_id]/attr1

3.3.2 Path Values and Language

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

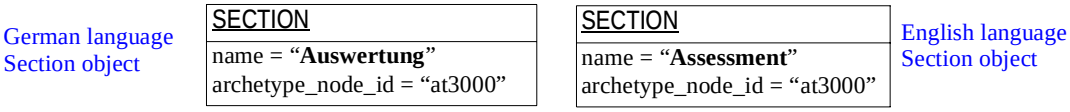

FIGURE 4 Names and archetype_node_ids in EHR Data

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

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

3.3.3 Concrete Path Structure

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

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

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

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

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

folder_path: path to any object in the EHR, via the folder structure. See Folder Paths in the directory package of the Common IM.

composition_path: path to a `COMPOSITION` (a version of a `VERSIONED_COMPOSITION`) as defined in Composition Path on page 41.

section_path: path to a particular `SECTION`, as defined in Section Paths on page 46.

entry_path: path to an item in an `ENTRY`, as defined in ENTRY on page 65.

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

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

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

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

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

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

3.3.4 Archetype Paths

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

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

```
ENTRY[at0000] matches {   -- blood pressure measurement
        name matches {...}
        data matches {
           HISTORY[at9001] matches {        -- history
              events cardinality matches {1..*} matches {
                 EVENT[at9002] {0..1}  matches {-- baseline
                    name matches {...}
                    data matches {
                       ITEM_LIST[at1000] matches {-- systemic arterial BP
                          items cardinality matches {2..*; ordered} matches {
```

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

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

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

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

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

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

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

```
                    >
                >
            >
```

The correspondending runtime paths from this data are as follows:

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

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

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

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

4 The Record

4.1 Overview

The design of the *open*EHR EHR is based on concepts and experience from the GeHR Australia project [32], the Synapses project [25], [26], [27], the CEN prEN13606 standard (2005), and the HL7v3 Clinical Document Architecture (CDA) Release 1 [35]. The structure consists of Compositions (equivalent to the CEN 13606 Composition and the HL7 CDA Document), organised by a directory of Folders (a CEN 13606 concept). The *open*EHR EHR is also versioned: every Composition is versioned, and so is the directory structure. This means that every previous state of the EHR - i.e. all states of Compositions and the Folder structure - is available. The versioning facilities are provided by the concepts defined in the `rm.common.change_control` package. All changes to the EHR are created by *contributions*, where a contribution can include creation, update or correction of Compositions, creation or modification of the Folder structure, move of Compositions in the Folder structure, and deletion of Compositions or part of the Folder structure. This approach guarantees that the EHR progresses from one valid state to another, regardless of what changes occur in any particular contribution.

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

4.2 General Organisation of the EHR

4.2.1 Compositions

The Composition concept in the *open*EHR EHR originated from the the Transaction concept of the GEHR project [21], [22], [23], [24], which was based on the concept of a unit of information corresponding to the interaction of a healthcare agent with the EHR. It was originally designed to satisfy the following needs (which include the well-known ACID characterisation of transactions [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 has since 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 basically two types of information at the coarse level which exist in the EHR: *event* items, and longitudinal, or *persistent* items, of which there might be various kinds.

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

FIGURE 5 Basic Event-oriented EHR

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

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

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

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

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 of the EHR on page 27.

4.2.2 Folders

As Compositions accumulate over time in the EHR, they form a long-term patient history. Previous work in CEN [28] - [31] and SynEx [26] suggests that it is useful to be able to organise Compositions using a hierarchy of folders to classify them, much the same way files in a file system are arranged in the directory structure, as visualised by the Windows Explorer and other similar tools. In the *open*EHR model, folders do not contain Compositions by value but by reference, and are completely optional. More than one Folder can refer to the same Composition. Folders might be used to manage a simple classification of Compositions, e.g into event and persistent, or they might be used to create numerous categories, based on episodes or other groupings of Compositions. Folder structures cann be archetyped.

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

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

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

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

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

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

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

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

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

A final category of information which may be needed in the change-controlled EHR is that of “technical contextual information”, which includes environment settings, software application names and version ids, identification and versions of data resources such as terminologies and possibly even

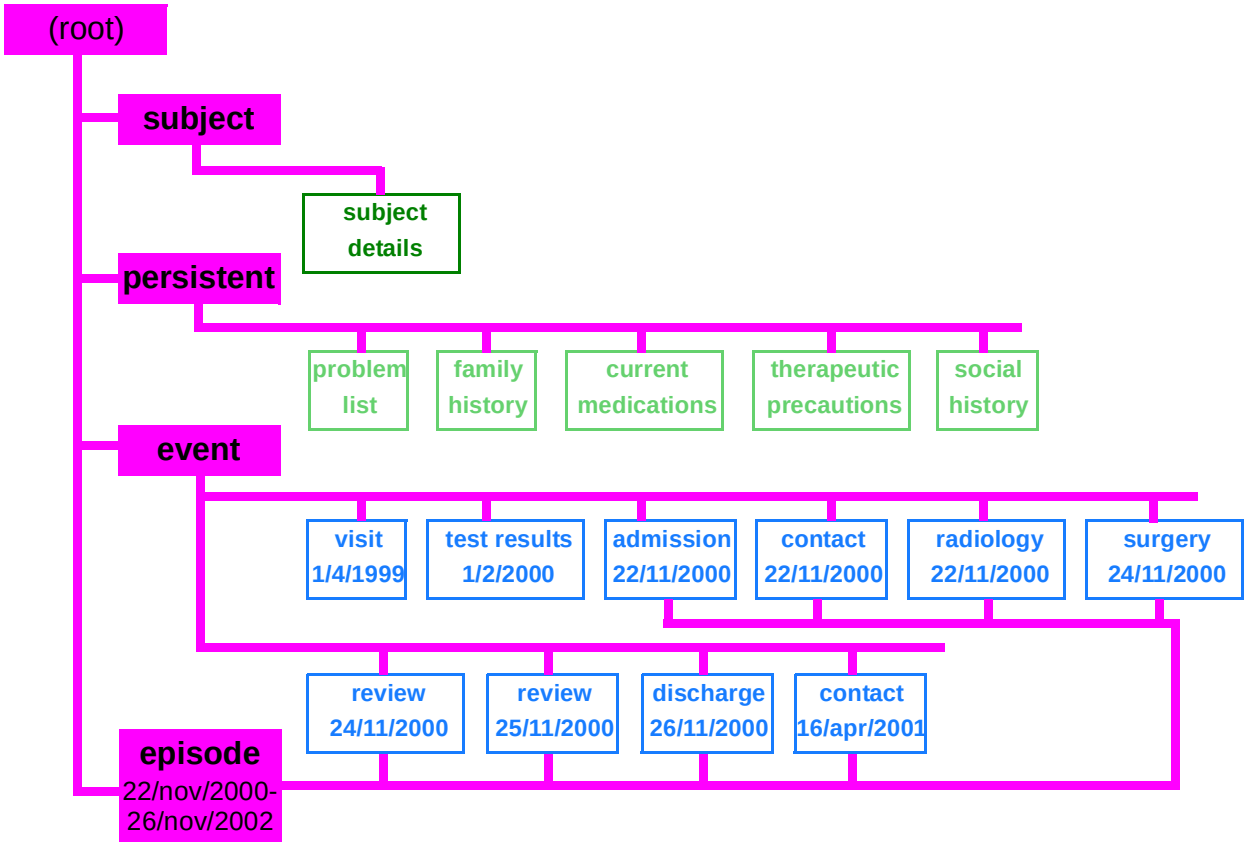

FIGURE 7 Using Folders to Group Compositions

FIGURE 8 An EHR containing Compositions and Folder Structure CIs

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

FIGURE 9 A Comprehensive Medico-legal EHR

4.2.3 Change Control of the EHR

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

A number of requirements and design considerations lead to the final design of Compositions in the *open*EHR EHR. The first is the *system interaction context*, in which a healthcare agent (usually a human, but may be a software process) interacts with the EHR system to enter data. This is the context during which Compositions and Folder structures are created or modified on the system, and once again, the contextual details of who, when and where must be recorded. As described in [5], these details might be quite different from the context details of the *clinical session*, as is the case when the EHR is updated some time after a contact, and by other personnel.

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

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

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

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

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

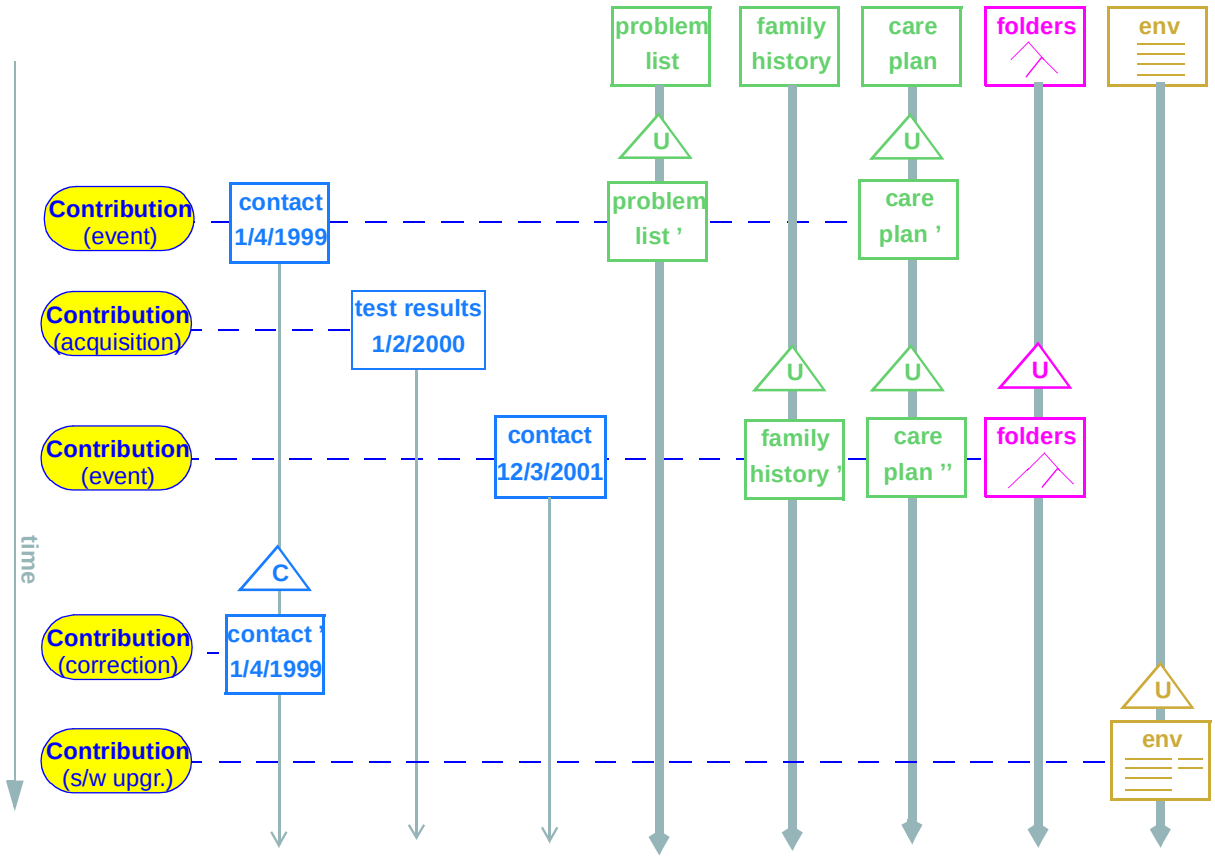

FIGURE 10 Contributions to the EHR

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

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

4.2.4 Versioning of Compositions

Versioning of Compositions is achieved with the `VERSIONED<T>` type from the Change Control package, which in the Composition package is explicitly bound to the `COMPOSITION` class, via the class `VERSIONED_COMPOSITION` which inherits from the type `VERSIONED<COMPOSITION>`. This is done because versioned Compositions are self-standing entities - they are not contained by value in any other class. Consequently, there is no other class where the type binding `VERSIONED<COMPOSITION>` can take place.

The effect of version control on Compositions is visualised in FIGURE 11. The versions (each “version” being a `COMPOSITION`) shown here in a `VERSIONED_COMPOSITION` are the same versions

shown along each vertical line in FIGURE 10, this time shown with their associated audit items. The set of versions should be understood as a set of successive modifications of the same data in time.

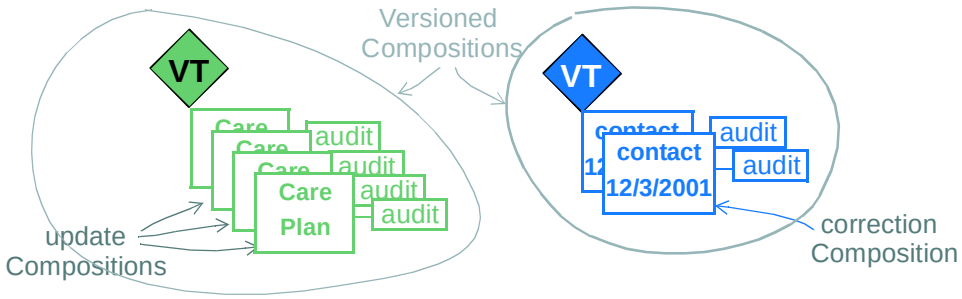

FIGURE 11 The versioned composition

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

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

4.2.5 Versioning Scenarios

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

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

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

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

Case 3: a VERSION<COMPOSITION> (such as a family history) received as part of an EHR_EXTRACT from another *open*EHR system, which will be used by a local author to create a new COMPOSITION which includes some content chosen from the received VERSION<COMPOSITION>. In this case, the new VERSION<COMPOSITION> is considered as a locally authored one, in which some content has been obtained from elsewhere. If it is the first version, a VERSIONED_COMPOSITION is first created. The AUDIT_DETAILS

documents the committal of this content, and the clinician may choose to record some details about it in the audit *description*.

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

5 EHR Package

5.1 Overview

The ehr package is illustrated in FIGURE 12. The EHR class is the root of the health record for a patient, and is a change-controlled repository of the kind described in the *open*EHR Common Information Model. Accordingly, it contains the identifiers of various versioned objects, as well as the list of Contributions made to it. The versioned objects consist of: a directory, in the form of a VERSIONED_FOLDER instance, and Compositions in terms of VERSIONED_COMPOSITIONs. The directory structure is optional, and consists of FOLDERs, enabling the construction of a hierarchical directory of any complexity. Each folder in the structure can contain any number of references to versioned Compositions. The structure as a whole acts as a directory for organising versioned Compositions in the record.

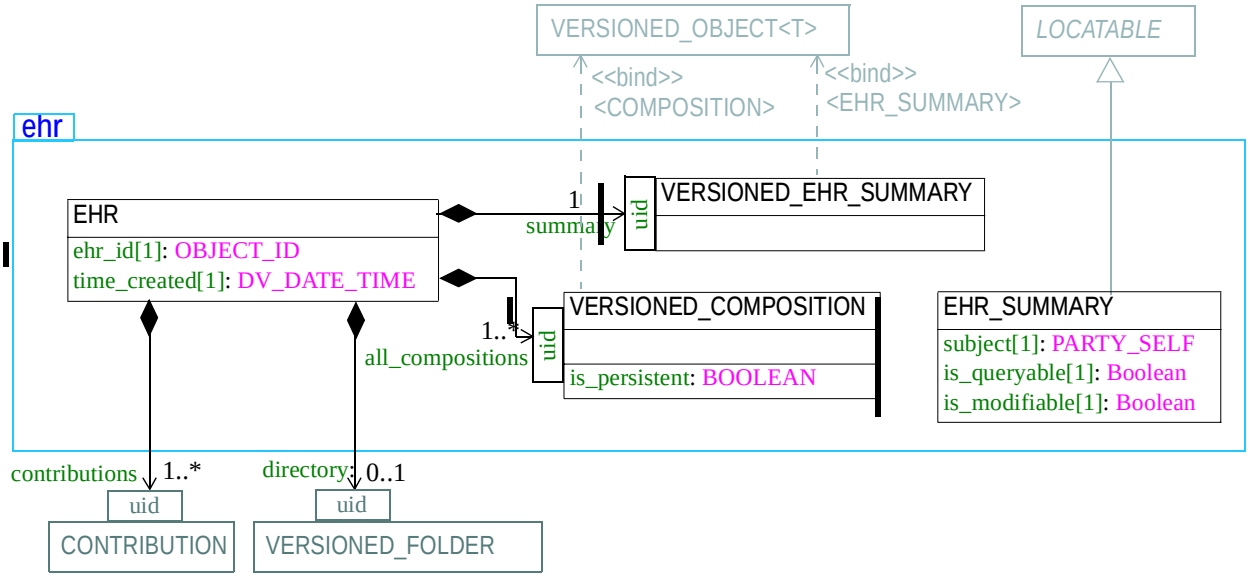

FIGURE 12 rm.ehr Package

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

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

Exactly the same logic holds for the relationship between the EHR and its CONTRIBUTIONs. The role of Contributions, as documented in the Common Information Model, is to record the set of versions added to a repository during a single logical update, along with the audit details of the change. In the

context of the EHR as one such repository, each instance of the CONTRIBUTION class potentially includes in its list of versions not only compositions, but also the folder structure itself, if this was changed during the update. Since the *versions* attribute of the CONTRIBUTIONs class is in fact a list of OBJECT_REFs, the same list can accommodate both types of object.

5.2 Historical Views of the Record

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

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

5.3 Path Semantics

5.3.1 EHR Path

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

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

Possible EHR paths are as follows:

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

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

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

5.3.2 VERSIONED_COMPOSITION Path

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

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

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

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

Example paths to individual versions include:

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

5.4 Class Descriptions

5.4.1 EHR Class

CLASS	EHR	
Purpose	The EHR object is the root object and access point of an EHR for a subject of care.	
CEN	EHCR class	
Synapses	RecordFolder class	
GEHR	G1_EHR	
Attributes	**Signature**	**Meaning**
1..1	**ehr_id**: 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	**summary**: OBJECT_REF	Summary object for this EHR.
0..1	**directory**: OBJECT_REF	Optional directory structure for this EHR.
1..1	**all_compositions**: List <OBJECT_REF>	Master list of all composition references in this EHR
Invariants	***Ehr_id_exists***: ehr_id /= Void ***Time_created_exists***: time_created /= Void ***Contributions_valid***: contributions /= Void ***and then*** contributions.for_all(type.is_equal("CONTRIBUTION")) ***All_compositions_valid***: all_compositions /= Void ***and then*** all_compositions.for_all(type.is_equal("VERSIONED_COMPOSITION")) ***Directory_valid***: directory /= Void ***implies*** directory.type.is_equal("VERSIONED_FOLDER")	

5.4.2 VERSIONED_EHR_SUMMARY Class

CLASS	VERSIONED_EHR_SUMMARY	
Purpose	Versioning container for EHR_SUMMARY instance.	
Inherit	VERSIONED_OBJECT<EHR_SUMMARY>	
Attributes	Signature	Meaning
Invariants		

5.4.3 EHR_SUMMARY Class

CLASS	EHR_SUMMARY	
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.

CLASS	EHR_SUMMARY	
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.
1..1	**is_modifiable**: `Boolean`	True if this EHR is allowed to be written to.
Invariants	***Subject_exists***: subject /= Void ***No_parent***: parent = Void	

5.4.4 VERSIONED_COMPOSITION Class

CLASS	VERSIONED_COMPOSITION	
Purpose	Version-controlled composition abstraction, defined by inheriting `VERSIONED_OBJECT<COMPOSITION>`.	
Use		
GEHR	`G1_VERSIONED_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 (data.archetype_node_id.is_equal(all_versions.first.data.archetype_node_id)) ***Persistent_valid***: all_versions.for_all (data.is_persistent = all_versions.first.data.is_persistent) ***Owner_id_valid***: owner_id.generating_type.is_equal("EHR")	

6 Composition Package

6.1 Overview

The Composition is the key 'data container' in the *open*EHR reference model and is the root point of clinical content and context information. Instances of the `COMPOSITION` class can be considered as self-standing data aggregations, or documents in a document-oriented system (similar to HL7v3 CDA "documents"). The key information in a `COMPOSITION` is its *content*, *context*, and *composer*. The majority of the use of paths in *open*EHR is likely to within Compositions. FIGURE 13 illustrates the `composition` package.

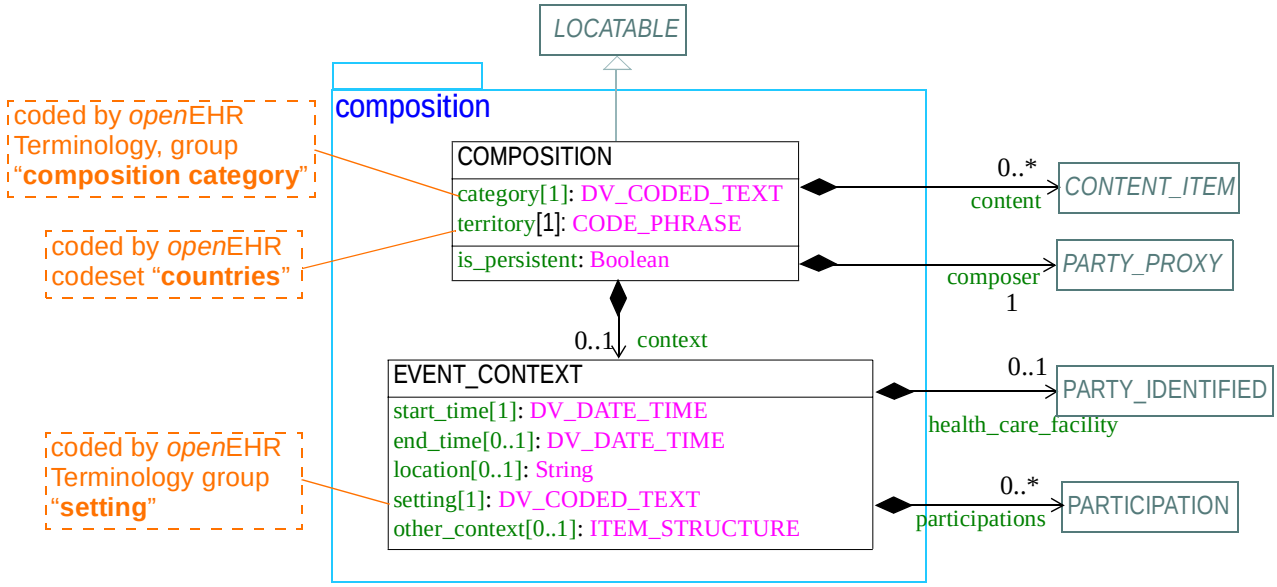

FIGURE 13 rm.composition Package

6.2 Design Principles

6.2.1 Composition Composer

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

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*.

6.2.2 Event Context

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'. 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 optionally further details defined by the relevant archetype.

Healthcare events that require an Event context 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 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 depend upon Composition-level archetypes.

Occurrence in Data

For the situations in which it is used in data, an EVENT_CONTEXT object is created only for Compositions to which the context information directly relates, as shown in FIGURE 14.

Consider a Contribution 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 doc-

tor'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.

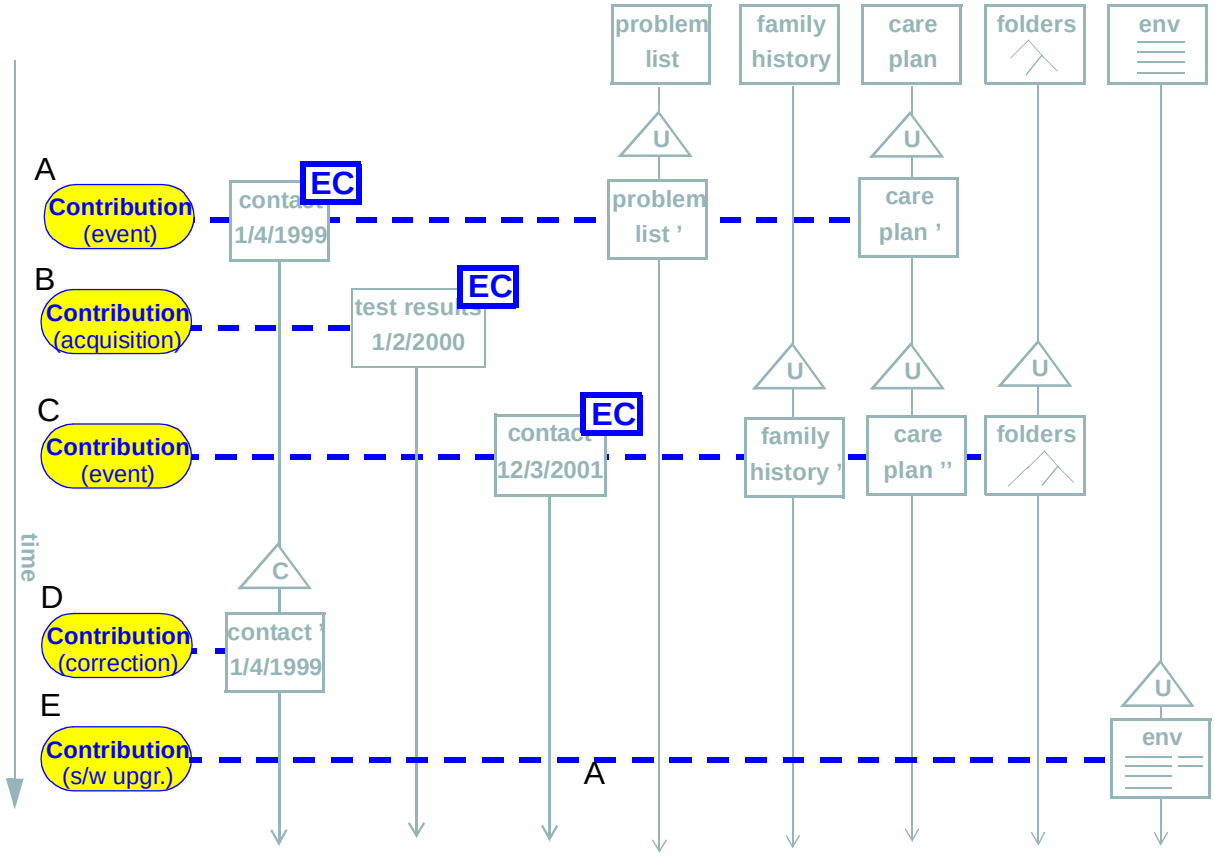

FIGURE 14 Use of Event context

Contributions A, B and C in the figure illustrate this case.

In some cases, Contributions are made to the record which include no event context, for example, if a secretary corrects an error in a Composition previously recorded for a patient visit; in such cases, the Event context of any Compositions from the original commit will remain intact and unchanged (unless the correction is to the event context itself of course), and will correctly reflect the fact that no new clinical interaction occurred. This is the case with Contribution D in the figure.

Persistent Compositions do not have an Event context.

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 notes into the record system at night, after

all patients have been seen. In such cases, the versioned Composition audit trail indicates the context of when the data were entered, as distinct from the context of when the clinical interaction took place.

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.

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 express this.

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

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.

6.2.3 Composition Content

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

- it may be empty. Although for most situations, there should be content in a Compostion, there are at least two cases where an empty Composition makes sense:
  - the first is a Composition in 'draft' editing state (VERSION.*lifecycle_state* = 'draft')

- the second is for systems that are only interested in the fact of an event having taken place, but want no details, such as so-called clinical 'event summary' systems, which might record the fact of visits to the doctor, but contain no further information. This can be achieved using Compositions with event context, and no further content.

- it may contain one or more SECTIONs which are defined in the archetype of the Composition;
- it may contain one or more SECTIONs trees, each of which is a separately archetyped structure;
- it may contain one of more ENTRYs directly, with no intermediate SECTIONs;
- it may be any combination of the last three possibilities.

6.3 Path Semantics

6.3.1 Composition Path

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

- meaning "(" modifier ")"

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

- "family history"
- "current medications"
- "current medications (chinese)"
- "therapeutic precautions (food and allergy)"
- "therapeutic precautions (drug)"

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

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

- "patient contact (1)"
- "patient contact (2)"
- ...
- "patient contact (73)"

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

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

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

Example Composition paths following this model include:

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

6.4 Class Descriptions

6.4.1 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.	
CEN	Composition	
GEHR	G1_COMPOSITION_VERSION	
Synapses	Composition class	
HL7v3	CDA 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 (not necessarily its entry into the EHR system). This is the identifier which should appear on the screen. It may or may not be the person who entered the data. 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	**territory**: CODE_PHRASE	Name of territory in which this Composition was written. Coded from openEHR “countries” code set, which is an expression of the ISO 3166 standard.
Functions	**Signature**	**Meaning**
	is_persistent: Boolean	True if category is a “persistent” type, False otherwise. Useful for finding Compositions in an EHR which are guaranteed to be of interest to most users.
Invariants	***Is_archetype_root***: is_archetype_root ***Composer_exists***: composer /= Void ***Content_valid***: content /= Void **implies not** content.is_empty ***Category_validity***: category /= Void **and then** terminology(“openehr”).codes_for_group_name(“composition category", “en”).has(category.defining_code) ***Is_persistent_validity***: is_persistent **implies** context = Void ***Name_value***: not is_persistent **implies** name.value.is_equal(context.health_care_facility.as_string + context.start_time.as_string) ***Territory_valid***: territory /= Void **and then** code_set(“countries”).has(territory) ***No_parent***: parent = Void	

6.4.2 EVENT_CONTEXT Class

CLASS	EVENT_CONTEXT	
Purpose	Documents the clinical context of the clinical session (or encounter). The context information recorded here are independent of the attributes recorded in the version audit, which document the “system interaction” context, i.e. the context of a user interacting with the health record system. Clinical sessions include patient contacts, and any other business activity, such as pathology investigations which take place on behalf of the patient.	
CEN	Composition class	
Synapses	Composition class	
HL7v3	TBD	
Attributes	**Signature**	**Meaning**

CLASS	EVENT_CONTEXT	
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 *clinical* session. These would normally include the physician(s) and often the patient (but not the latter if the clinical session is a pathology test for example).
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_exists***: start_time /= Void ***participations_validity***: participations /= Void ***implies not*** participations.empty ***location_valid***: location /= Void ***implies not*** location.is_empty ***setting_valid***: setting /= Void ***and then*** Terminology(“openehr”).codes_for_group_name(“setting”, “en”).has(setting.defining_code)	

7 Navigation Package

7.1 Overview

The `navigation` Package defines a hierachical heading structure, in which all individual headings are considered to belong to a “tree of headings”. Each heading is an instance of the class `SECTION`, illustrated in FIGURE 15.

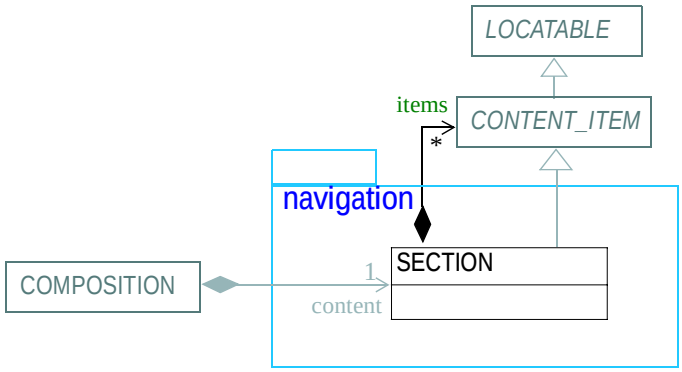

FIGURE 15 rm.composition.content.navigation Package

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 to form one large heading structure.

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 52.

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

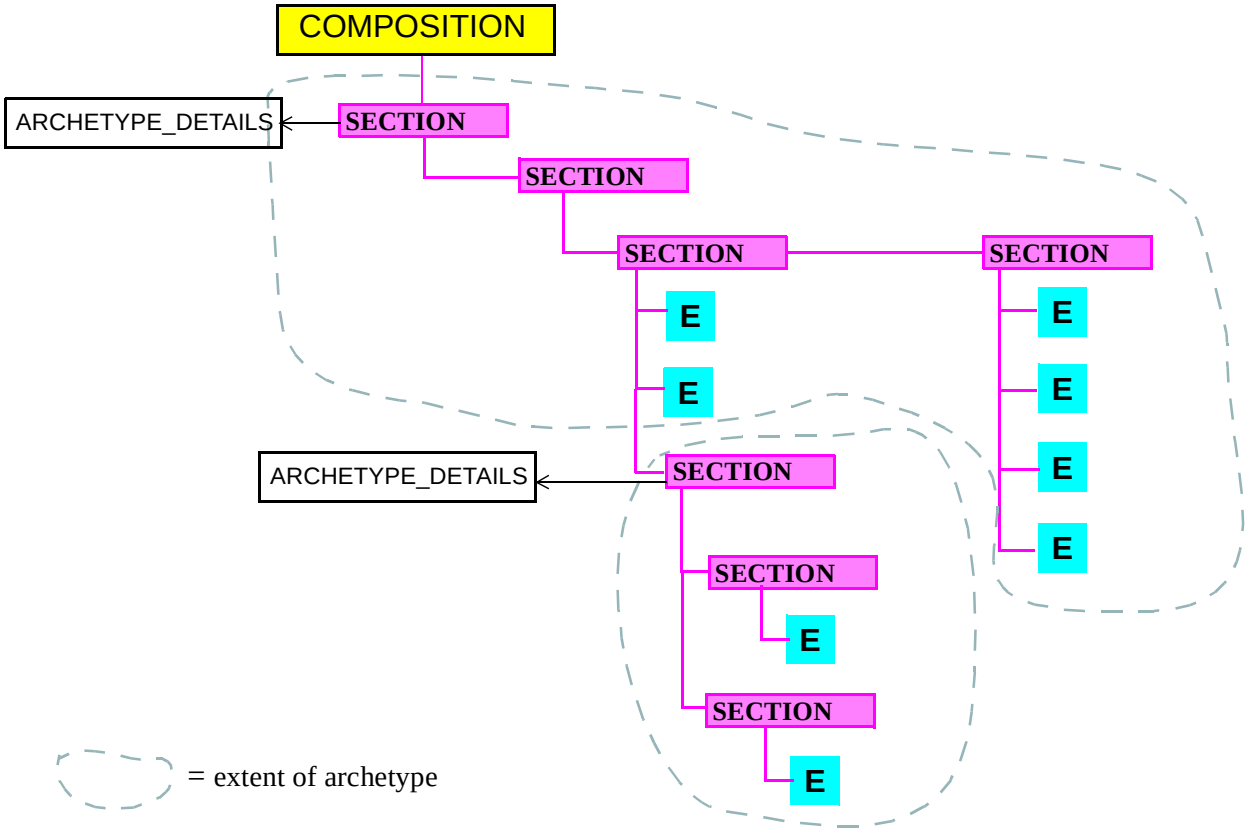

FIGURE 16 Section View of a General Practice Contact Composition

7.2 Section Paths

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

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

Examples include:

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

7.3 Class Descriptions

7.3.1 SECTION Class

CLASS	SECTION
Purpose	Represents a heading in a heading structure, or "section tree".
Use	Created according to archetyped structures for typical headings such as SOAP, physical examination, but also pathology result heading structures.
MisUse	Should not be used instead of `ENTRY` hierarchical structures.
CEN	Headed_section

CLASS	SECTION	
OMG HDTF	COAS::CompositeObservation (COAS does not distinguish between the semantics of Sections and hierarchical structure inside Entries, modelled in *open*EHR by Cluster).	
GEHR	`G1_ORGANISER`	
HL7v3	CDA Heading.	
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_exists***: items /= Void ***implies not*** items.is_empty	

7.4 Section Instance Structures

7.4.1 Problem/SOAP Headings

An example of an section tree representing the problem/SOAP heading structure is shown in FIGURE 17.

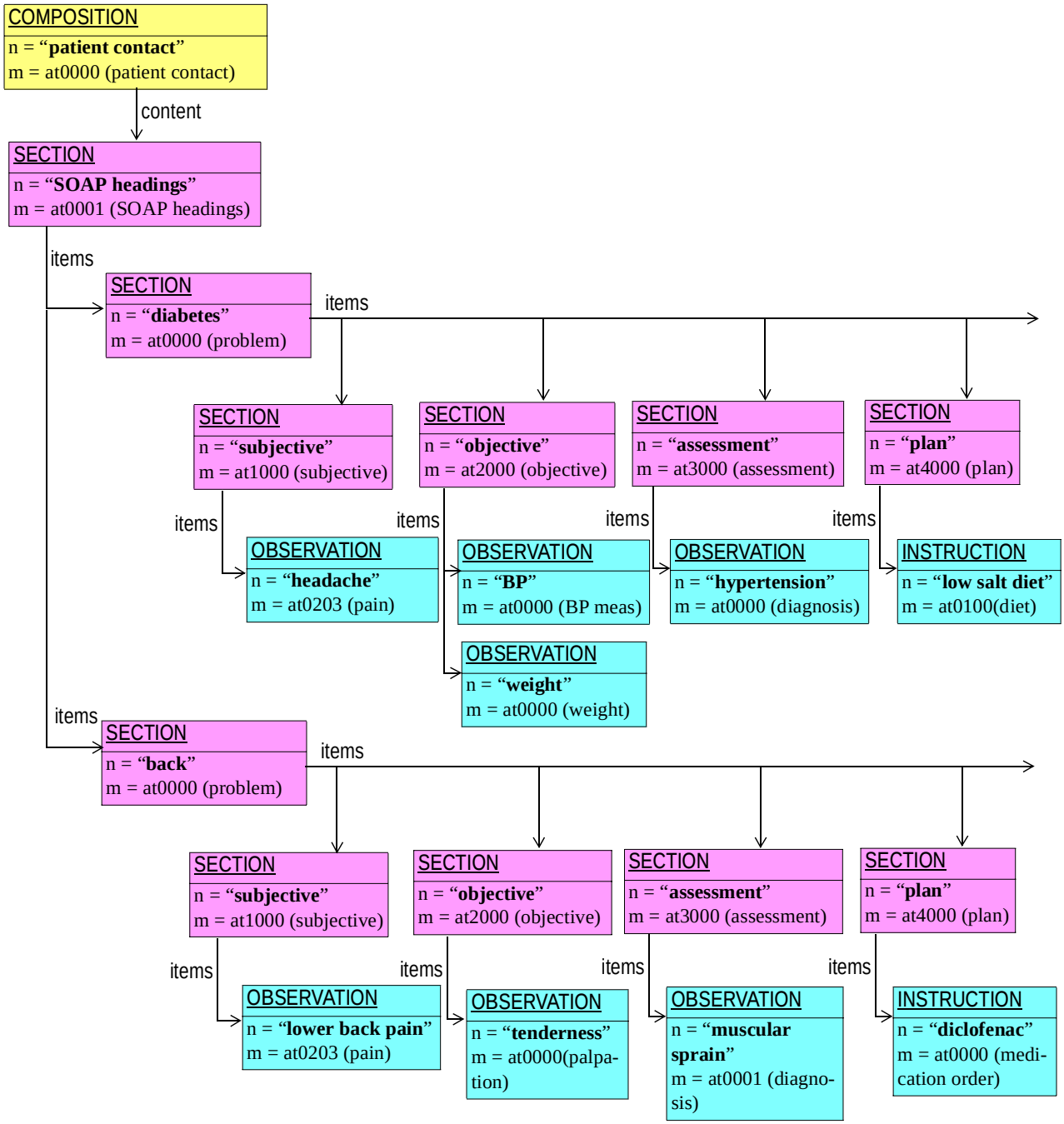

FIGURE 17 "problem/SOAP" Section Structure

8 Entry Package

8.1 Design Principles

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', but may contain a significant amount of data, e.g. an entire microbiology result, a psychiatric examination, a complex prescription. 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 achetyped, and in fact, archetypes for Entries make up the vast majority of important clinical archetypes defined.

The design of Entry package is based on past experience and upon a solid theoretical model of clinical recording, described in Beale & Heard [4], which includes a typology of information categories shown in the FIGURE 18 (the *open*EHR Entry class names are annotated next to categories which are directly modelled).

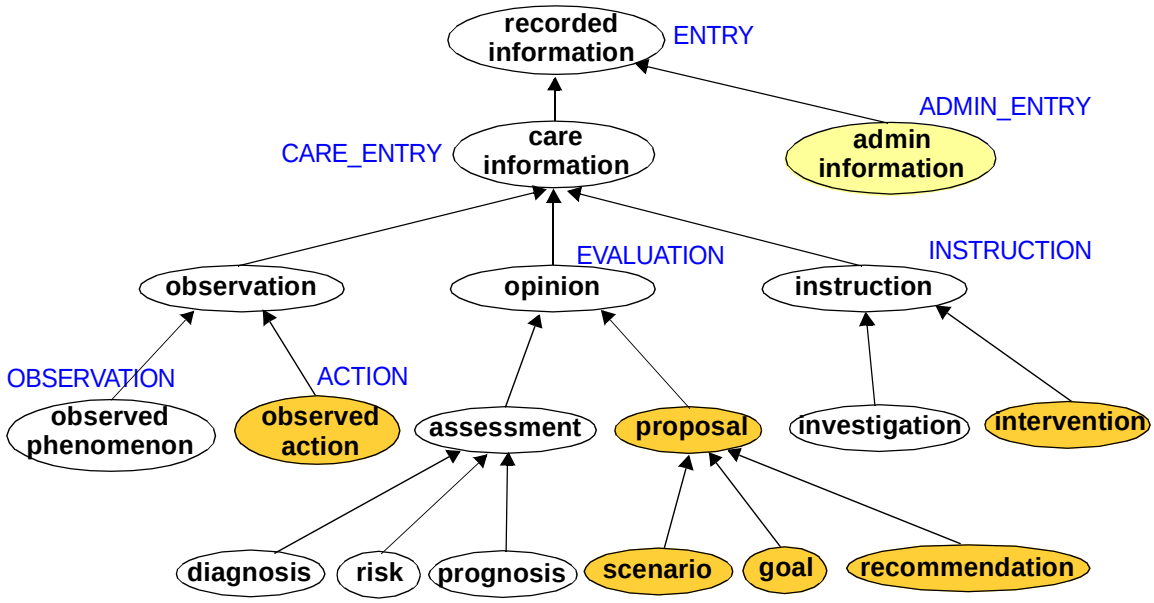

FIGURE 18 Typology of Recorded Information (adapted from Beale & Heard, 2006)

The key top-level categories in the typology are 'care information' and 'administrative information'. The former encompasses all statements that might be recorded at any point in the clinical care process, and consists of the major subcategories 'observation', '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, but relates to organising it, such as appointments and admissions. Categories that relate to the patient system as observed are shown as white bubbles, while categories that relate to intervention into the patient system are shown as shaded.

Where the main information categories occur in the clinical process is shown in FIGURE 19. This figure shows the cycle of information in an iterative, investigation process typical not just of clinical medicine, but of science itself. The major category of Observation leads 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. 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 administer themselves.

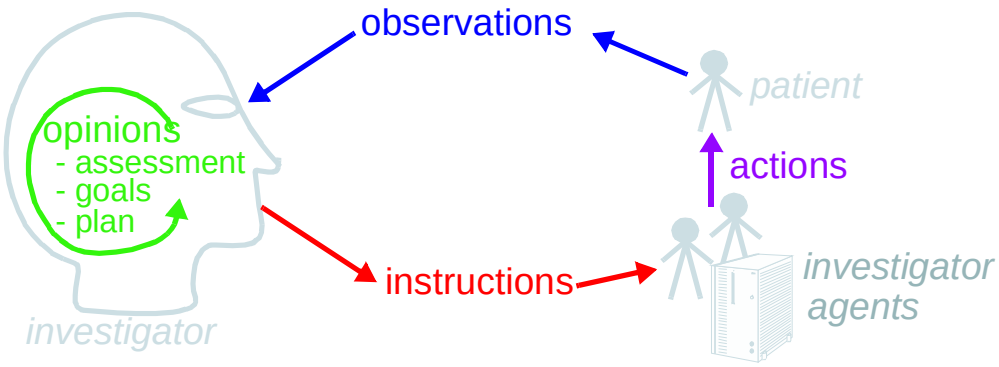

FIGURE 19 Categories of information in the care process

The Entry model is defined by the `composition.content.entry` package, shown in FIGURE 20.

There are two key justifications for using such a typology as a basis for class design. Firstly, although for all categories in the typology 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, 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 typology. 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);

Another set of statement types which 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) ⇒ 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);

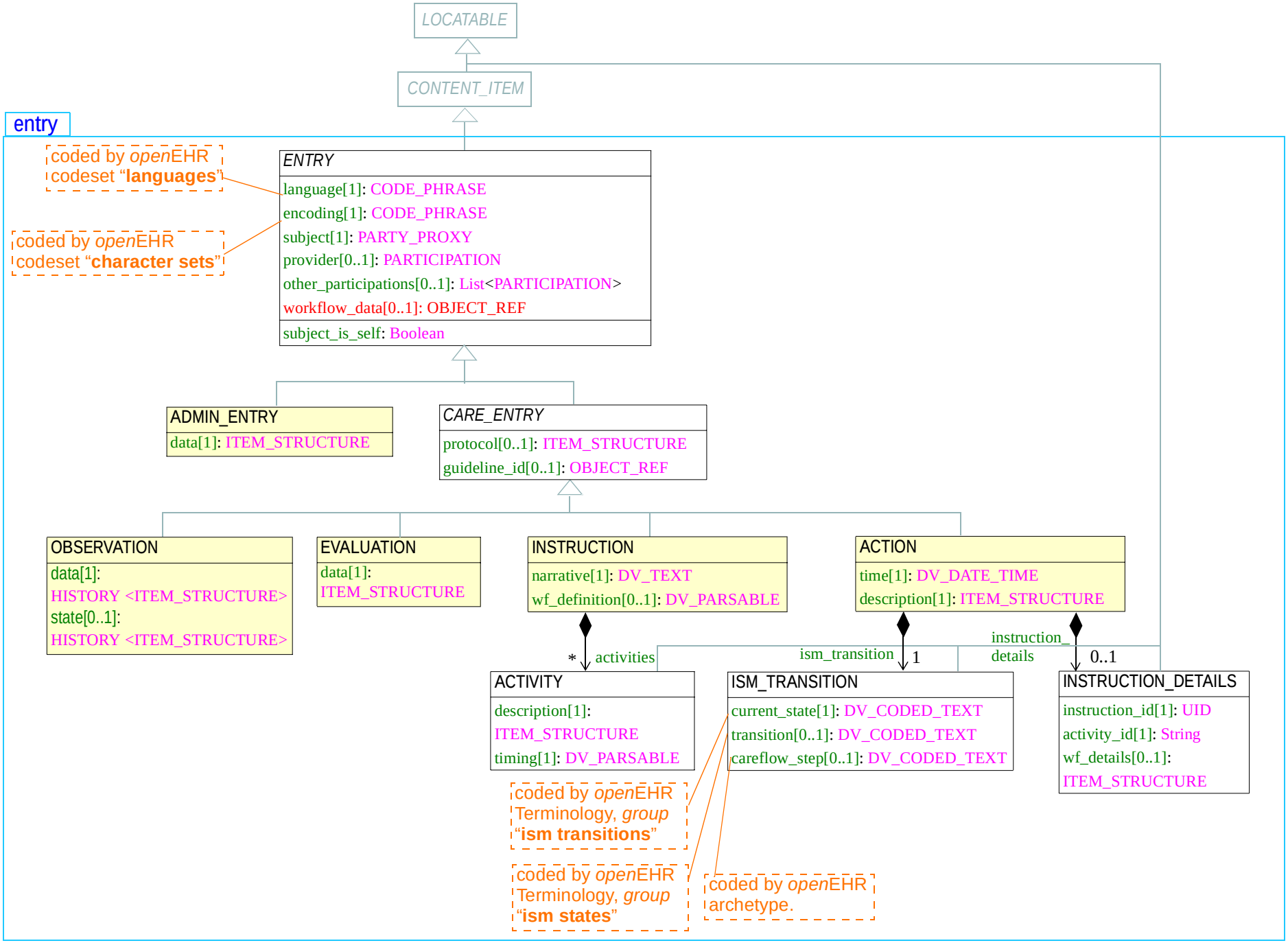

FIGURE 20 rm.composition.content.entry Package

- I (ordered) ⇒ Instruction (of I for patient for some date in the future).

Correct use of the categories of the typology 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.

8.2 Entry and its Subtypes

The choice of subtypes of ENTRY is based on the typology shown in FIGURE 18 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 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). Additionally, for categories such as Opinion and Instruction, the subcategories shown in the typology (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 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, 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). Similarly character encoding can be overridden by the *encoding* attribute within DV_TEXT.

The other common Entry features 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. 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. Expressed as a PARTICIPATION to allow the mode (e.g. by phone, video-conference) to be 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.

To Be Continued: probably need a workflow id or ITEM_STRUCTURE in ENTRY for future large scale workflow automation.

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);
- that person 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 which 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 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 admnistrative 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.

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*.

TBD_1: guideline_id is too specific and should probably be replaced by reasoning: ITEM_STRUCTURE

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.

It is worth noting that another high-level division exists, namely between EHR and demographic information. The general approach of *open*EHR is to eable 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.

8.2.3 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, and self-entered answers to a psychological assessment questionnaire. Observations are distinguished from Actions by excluding interventions; they record only information relating to the situation of the patient, not what is done to her.

The main data of Observation is expressed in the *data* and *state* attributes, and the inherited *protocol* attribute. This trio of attributes is of key importance in recording observations, and may be understood 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: any particular information about the state of entity (usually the patient) 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.

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 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` pack-

age[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 using the EVENT attribute *is_instantaneous*; interval events have the *width* attribute is set to the duration of the interval.

8.2.3.2 Valid Contents of a History

The intention of this model is to represent single sample and 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. Compositions, in the EHR). 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 should 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 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 entry into 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).

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

The second situation is when the sample does decay in some way, and the delay is relevant. Most such cases are in pathology, 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.

(It should be noted that if *open*EHR were deployed in the pathology laboratory itself, it would be quite reasonable to use Observation structures to directly record lengthy testing processes and intermediate result states. In this case, the subject of the Observation would be the sample, not the patient. Such data would however have to be processed into a clinical Observation result as described here to be compatible with the target EHR).

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 21 illustrates the two methods of recording state.

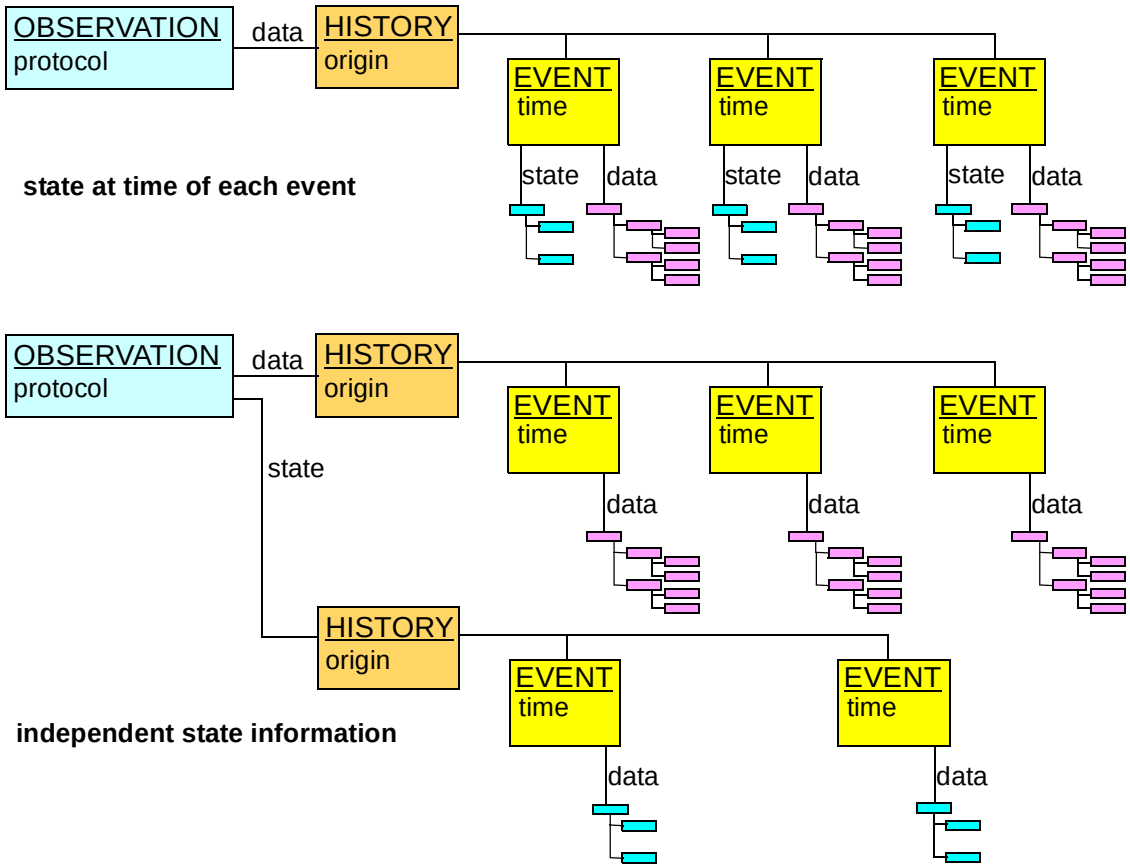

FIGURE 21 Alternative ways of recording state

8.2.4 Evaluation

According to the typology 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.

plan: a description in general terms of a care plan 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”, “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.*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 as in Observation, since a diagnosis is usually the result of thinking on the part of the physician; an exception to this would be 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 typology (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 described 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 guideline explained by his/her GP.

In the typology of FIGURE 18, 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 key class, ACTION, is used to model the information recorded due to the execution of an Instruction by some agent.

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 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 possible states and transitions for any Instruction, no matter whether it be a PAP recall or a complex course of chemotherapy. The use of standardised states means that the execution state of any given Instruction 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 third principle is to provide a way of mapping steps in any care pathway (i.e. healthcare business process) to states in the Instruction state machine. A care pathway is a process which covers the entirety of steps required to effect an Instruction, including prescribing, booking, dispensing, referring, suspension etc. Any such step when performed leaves the Instruction in one of the states of the ISM. This is supported even for Instructions with no formal workflow definition.

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, which is appropriate because .

The following sections describe the model in further detail.

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, generally limited to the active admnistration activity or activities. It also includes all the attributes inherited from the CARE_ENTRY class, including *subject*, *participations* and so on.

Most 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. Overall, there are three levels of representation of an Instruction, as shown in FIGURE 22. It is expected that the vast majority of *open*EHR systems for the forseeable future will support only the minimal and basic levels.

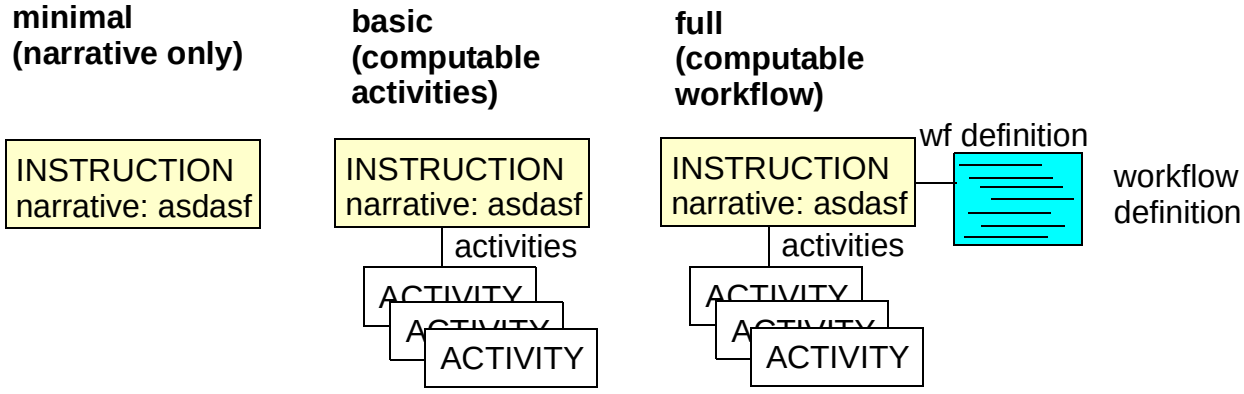

FIGURE 22 Levels of Instruction representation

When some part of an Instruction is executed (or in some cases where there is none), an ACTION instance is created describing it, and committed to the EHR. Actions include the inherited CARE_ENTRY attributes, along with the time of being performed, a description of what was performed, and a ISM_DETAILS object indicating which state of the ISM the Instruction is now in. If the Action did correspond to an Instruction, there will be an INSTRUCTION_DETAILS instance, which indicates which Activity of which Instruction was executed, including workflow execution details if relevant. Actions for which there is no Instruction (or at least not in the current EHR) do occur fairly often, and will not have the INSTRUCTION_DETAILS part of the information. However, they do have the ISM_DETAILS instance, and it should always be possible for the committer of an *ad hoc* ACTION to determine what state in the ISM it corresponds to (e.g. to at least be able to say that it is active, suspended, completed etc). In this way, the ISM state of all things happening to the patient can be ascertained by querying, regardless of whether explicit Instructions exist.

8.2.5.4 The Standard Instruction State Machine (ISM)

When a healthcare process is unfolding in a clinical context, EHR users want to know things such as the following:

- what is the current step in the process for the patient?

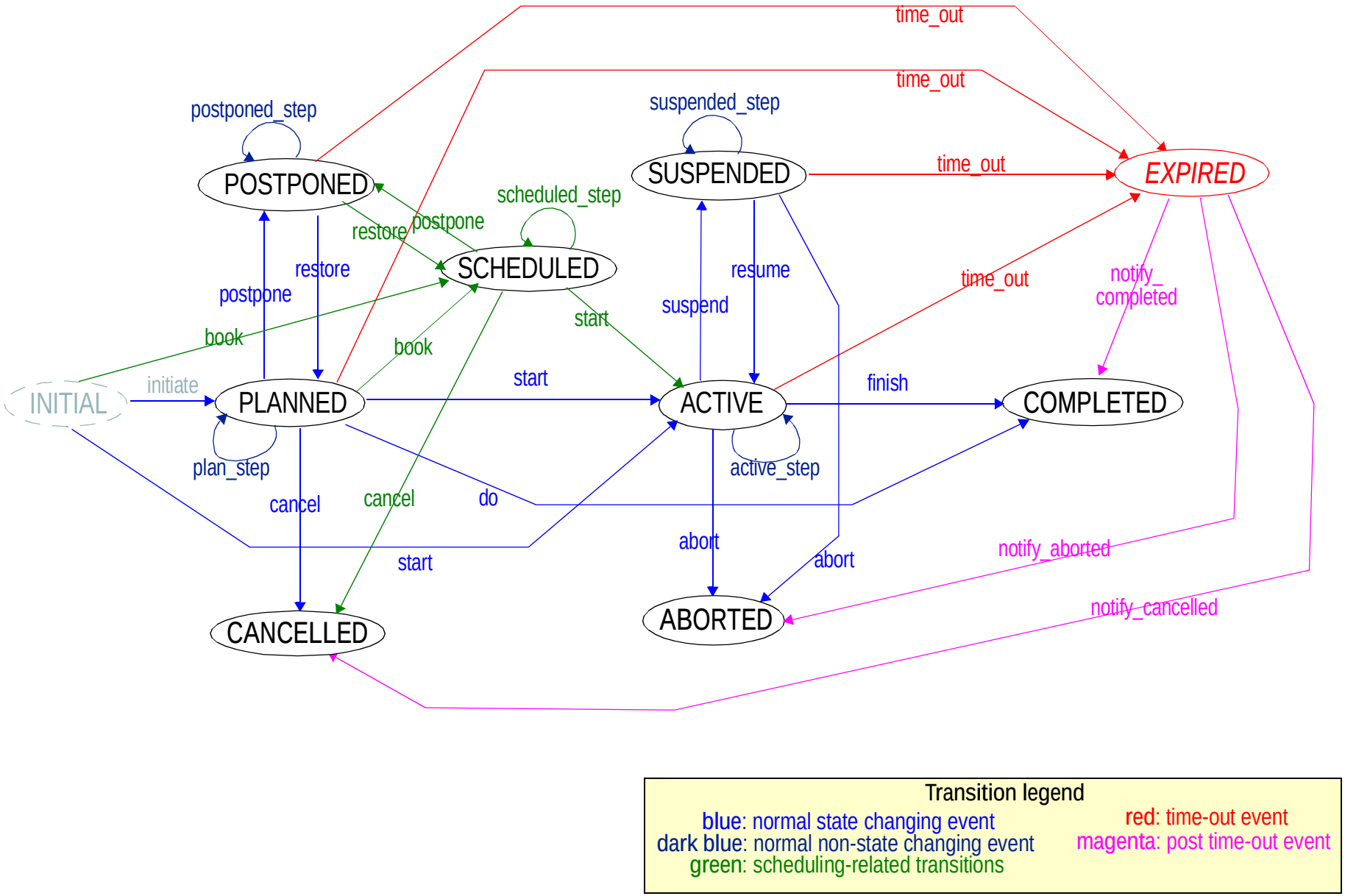

FIGURE 23 *open*EHR standard Instruction State Machine

- for a given patient, what processes are planned, active, suspended, completed?
- for the populations of patients, what is the state of a particular workflow, such as a recall?

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 Instruction. The state machine is illustrated in FIGURE 23. 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 not as yet taken place.

`POSTPONED`: the action has not taken place 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 take place, and has been booked in a scheduling system.

`CANCELLED`: the action was defined, but was cancelled before anything happened; it has not and will not take place.

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

`SUSPENDED`: the action has 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.

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 whole Instruction 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 which 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.

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

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

8.2.5.5 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 what 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 the archetype for the Instruction. When each Action corresponding to a particular Instruction is performed, it will be known both which careflow step it corresponds to and which ISM state the Instruction 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_DETAILS` object records the step performed and the ISM state and transition. The careflow step must be one of the steps from the corresponding Instruction.

8.2.5.6 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 marking 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], [7] 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 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.

- 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 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 espressed.

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.

8.3 Path Semantics

8.3.1 ENTRY

Entry runtime paths are constructed in the usual way by the concatenation of model attribute names and object *name* values. Examples include:

```
/data/items[event_16]/item[ECG result]/items[lead 3]
/data/items[1 min]/item[blood pressure]/items[systolic pressure]
/protocol[BP protocol]/items[position]
```

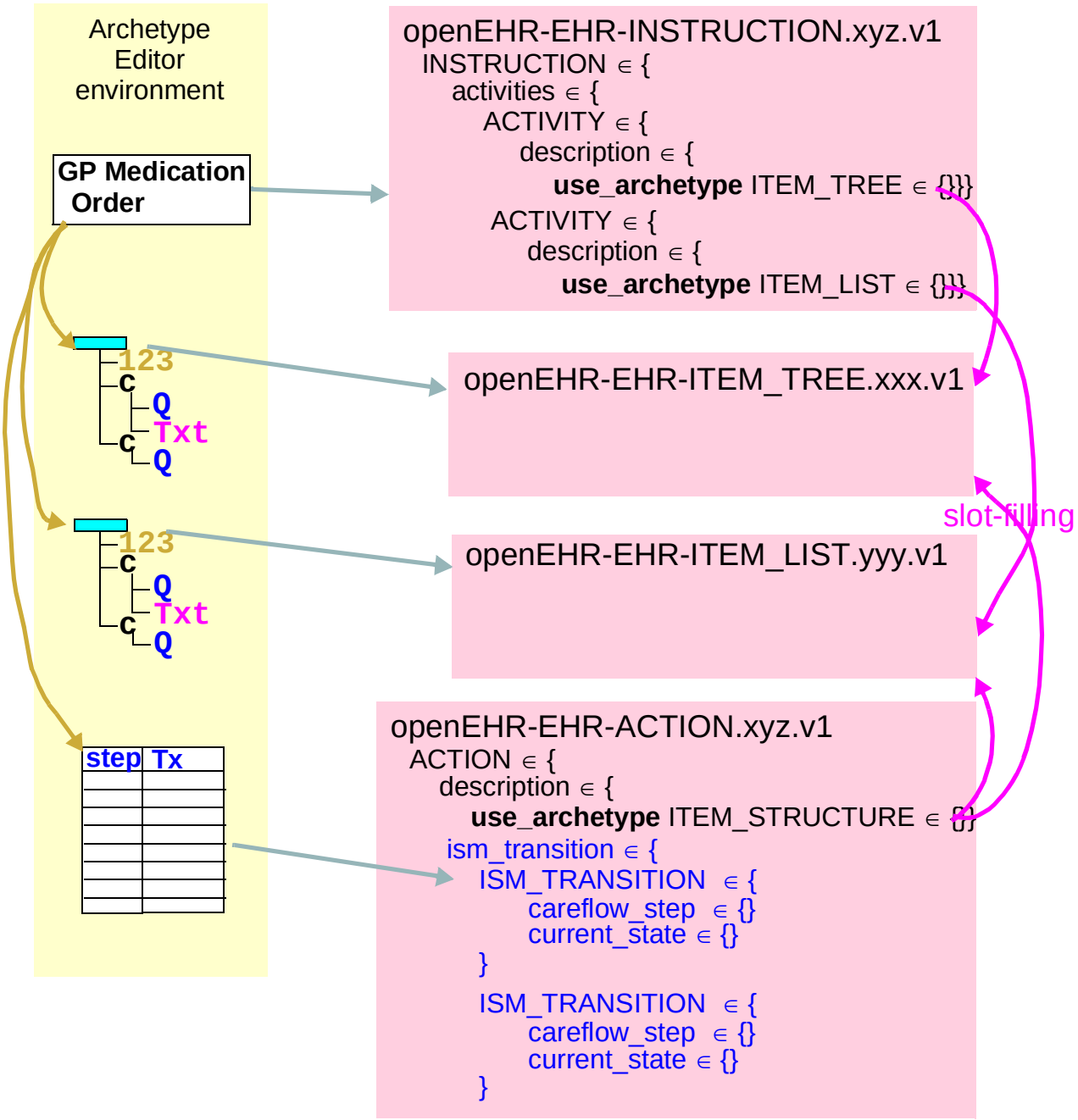

FIGURE 24 Archetypes, Instruction, and Action

8.4 Class Descriptions

8.4.1 ENTRY Class

CLASS	*ENTRY (abstract)*	
Purpose	The abstract parent of all ENTRY subtypes. An ENTRY is the root of a logical item of "hard" clinical information created in the "clinical statement" context, within a clinical session. There can be numerous such contexts in a clinical session. Observations and other Entry types only ever document information captured/created in the clinical session documented by the enclosing Composition. (The effect of this is that pathology results etc, if represented in the EHR, must be expressed as Entries inside Compositions representing the session in the pathology lab, not within Compositions created during a consultation during which a clinician might read the pathology result and make some decision based on it.) An ENTRY is also the minimal unit of information any query should return, since a whole ENTRY (including subparts) records spatial structure, timing information, and contextual information, including meaning-changing modifiers as well as the subject and generator of the information.	
CEN	Cluster OCC	
OMG HDTF	COAS::HealthRecordEntry and COAS::ObservationQualifier, a generic class which is used to represent context attributes which are concretely modelled here.	
Synapses	The Item class is the closest match for Entry as described here.	
GEHR	*_CONTENT	
HL7v3	Act	
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**: PARTICIPATION	Optional identification of provider of the informatoin 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. Modelled as a PARTICIPATION to allow function and mode to be recorded.
0..1	**other_participations**: List <PARTICIPATION>	Other participations at ENTRY level - archetypable.
0..1	**workflow_data**: OBJECT_REF	Workflow engine meta-data.
Functions	**Signature**	**Meaning**
	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("languages").has(language) ***Encoding_valid***: encoding /= Void ***and then*** code_set("character sets").has(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	

8.4.2 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.	
Use	Typically 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	

8.4.3 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 *how* and/or *why* the information in this entry was arrived at. For OBSERVATIONs, this is a description of the method or instrument used. For EVALUATIONs, how the evaluation was arrived at. For INSTRUCTIONs, how to execute the instruction. This may take the form of references to guidelines, including manually folowed and executable; knowledge references such as a paper in Medline; clinical reasons within a largercare process.
0..1	**guideline_id**: OBJECT_REF	Optional external identifier of guideline creating this action if relevant
Invariants		

8.4.4 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 as well as all statements or opinions (i.e. subjective data) about things in the past.
MisUse	Not used for future statements of any kind, including instructions, intentions, plans etc.

CLASS	OBSERVATION	
CEN	Cluster	
GEHR	G1_OBSERVATION_CONTENT, G1_SUBJECTIVE_CONTENT	
HL7v3	Observation	
Inherit	CARE_ENTRY	
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>	The state of subject of this observation during the observation process, in the form of a history of values which may be of any complexity. Optional.
Invariants	***Data_exists***: data /= Void	

8.4.5 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 with the INSTRUCTION type.	
GEHR	G1_SUBJECTIVE_CONTENT	
HL7v3	In HL7v3, diagnoses etc seem to be Observations.	
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	

8.4.6 INSTRUCTION Class

CLASS	INSTRUCTION	
Purpose	Used to specify actions in the future. Enables simple and complex specifications to be expressed in a 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.	
GEHR	`G1_INSTRUCTION`	
HL7v3	Act subtype Substance_administration, any Act type which is really an action specification (cf an Act in the past)	
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 definition.
0..1	**wf_definition**: `DV_PARSABLE`	Optional workflow engine executable expression of the Instruction.
Invariants	***Narrative_exists***: narrative /= Void	

8.4.7 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 a GTS or iCal string.
Invariants	***Description_exists***: description /= Void ***Timing_exists***: timing /= Void	

8.4.8 ACTION Class

CLASS	ACTION	
Purpose	Used to record a clinical action, e.g. 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	Time at which this action completed.
1..1	**description**: ITEM_STRUCTURE	Description of the activity to be performed, 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	
Invariants	***Time_exists***: time /= Void ***Description_exists***: description /= Void ***Ism_transition_exists***: ism_transition /= Void	

8.4.9 INSTRUCTION_DETAILS Class

CLASS	INSTRUCTION_DETAILS	
Purpose	Used to record details of the Instruction causing an Action.	
Inherit	LOCATABLE	
Attributes	**Signature**	**Meaning**
1..1	**instruction_id**: UID	Id of causing Instruction.
1..1	**activity_id**: String	Identifier of Activity within Instruction, in the form of its archetype path.
0..1	**wf_details**: ITEM_STRUCTURE	List of notifications which actually occurred, with all variables substituted.
Invariants	***Instruction_id_valid***: instruction_id /= Void ***Activity_path_valid***: activity_id /= Void ***and then not*** activity_id.is_empty	

8.4.10 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	LOCATABLE	
Attributes	**Signature**	**Meaning**
1..1	**current_state**: DV_CODED_TEXT	The ISM current state.
0..1	**transition**: DV_CODED_TEXT	The ISM transition which occurred to arrive in the *current_state*.
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”. Defined in archetype.
Invariants	***Current_state_valid***: current_state /= Void	

8.5 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).

8.5.1 OBSERVATION

Heartrate Measurement Series

FIGURE 25 illustrates three heartrate measurements over 10 minutes.

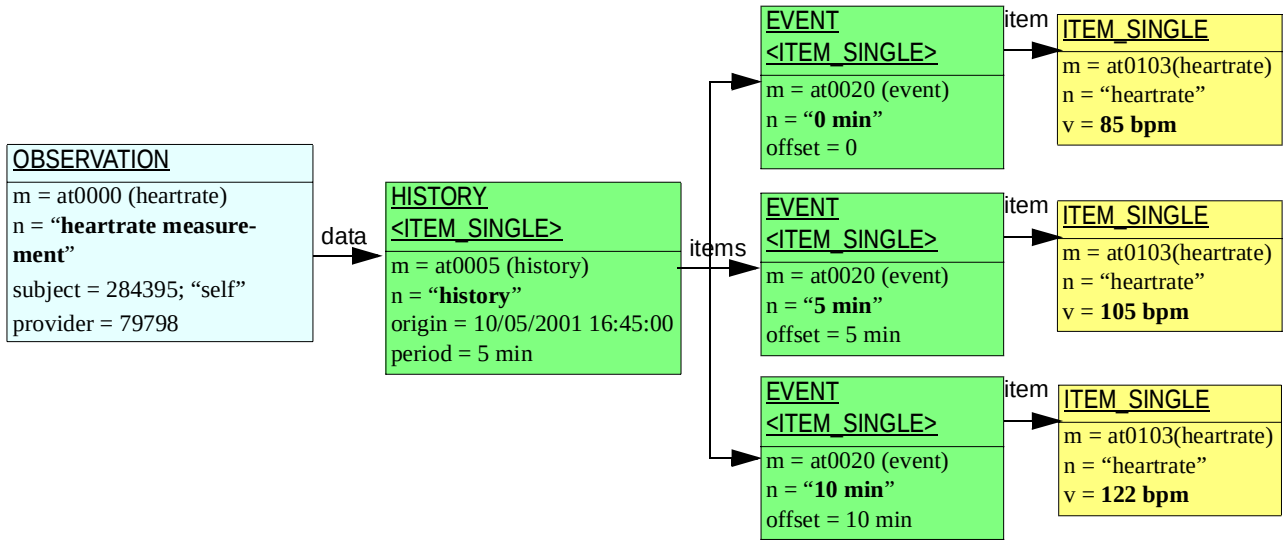

FIGURE 25 Periodic series Instance Structure

Blood Pressure with Protocol

FIGURE 26 illustrates a blood pressure observation with protocol.

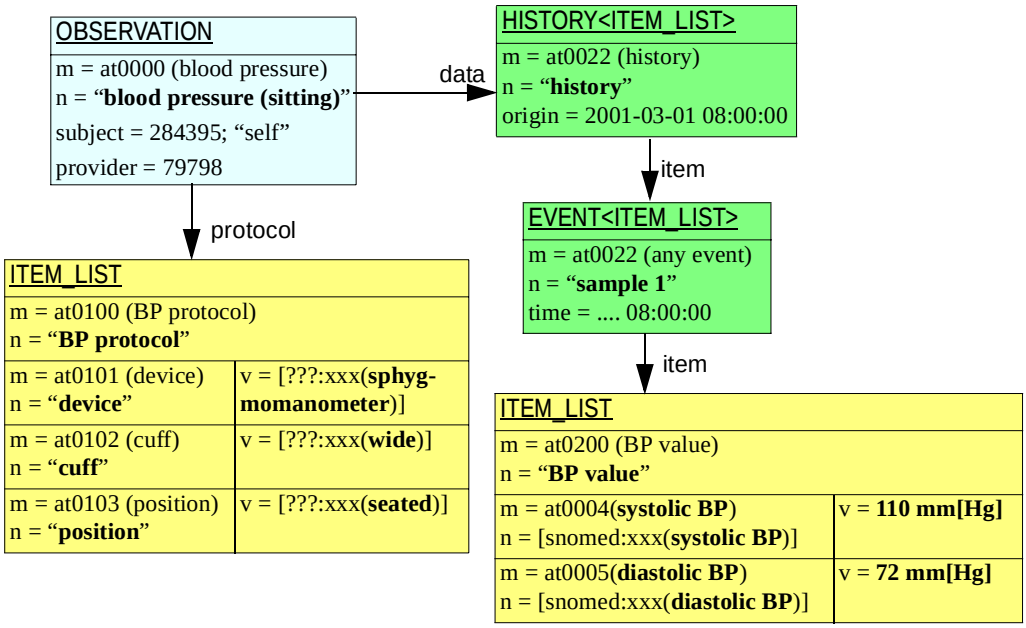

FIGURE 26 Blood Pressure Measurement Observation

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 `EVENTS`, with the fasting and glucose challenges being expressed as states on the relevant events.

8.5.2 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

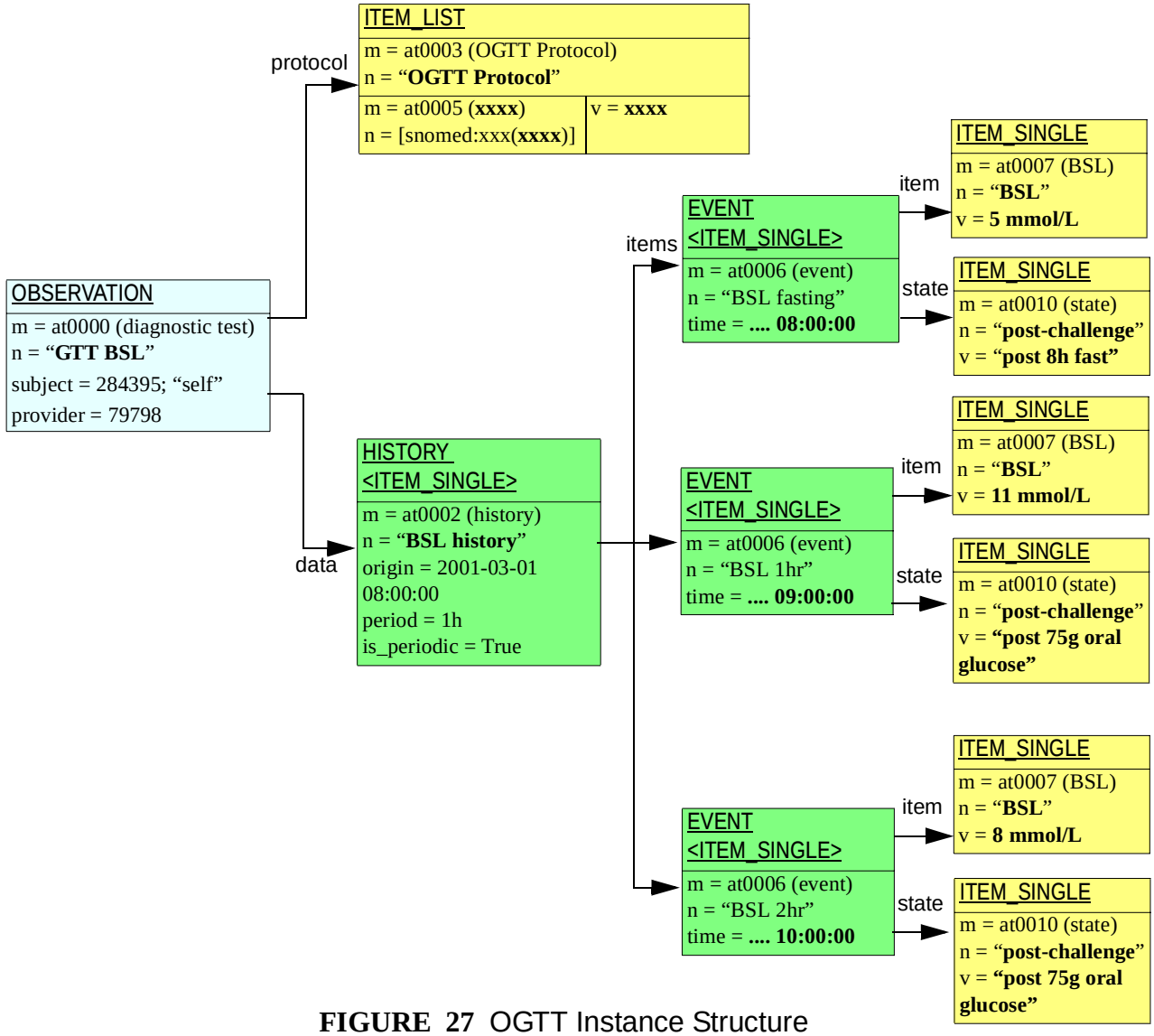

FIGURE 27 OGTT Instance Structure

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

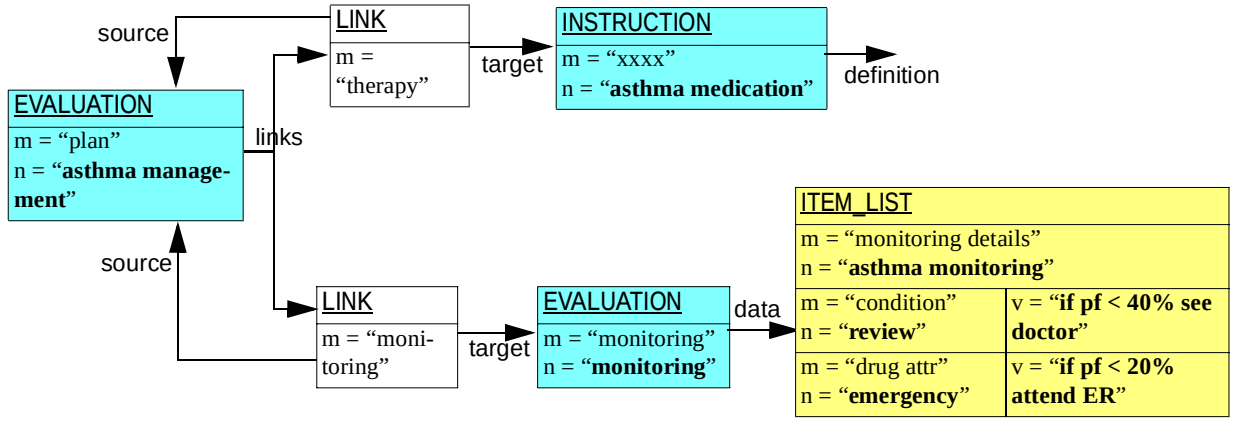

FIGURE 28 Partial Asthma Management Plan

8.5.3 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).

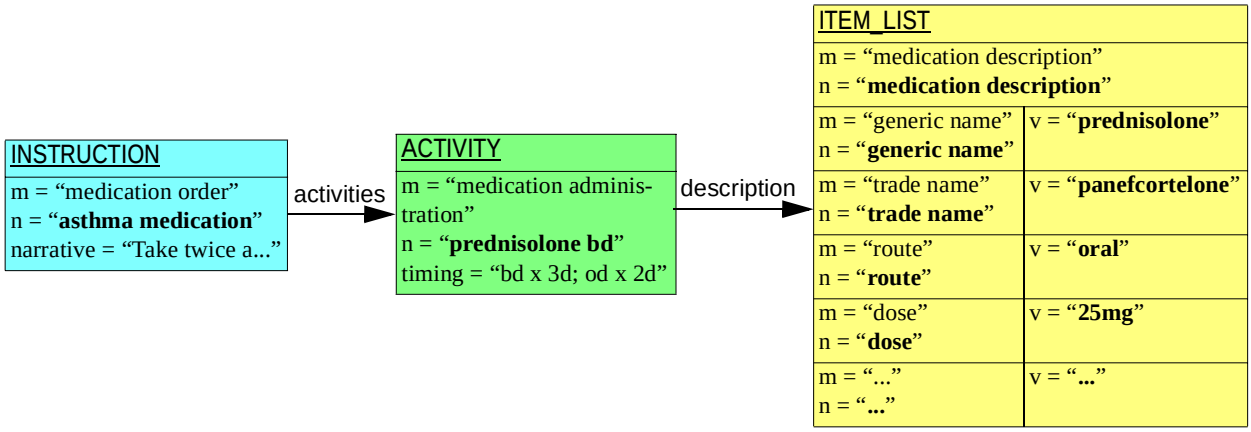

FIGURE 29 Chained medication order

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.

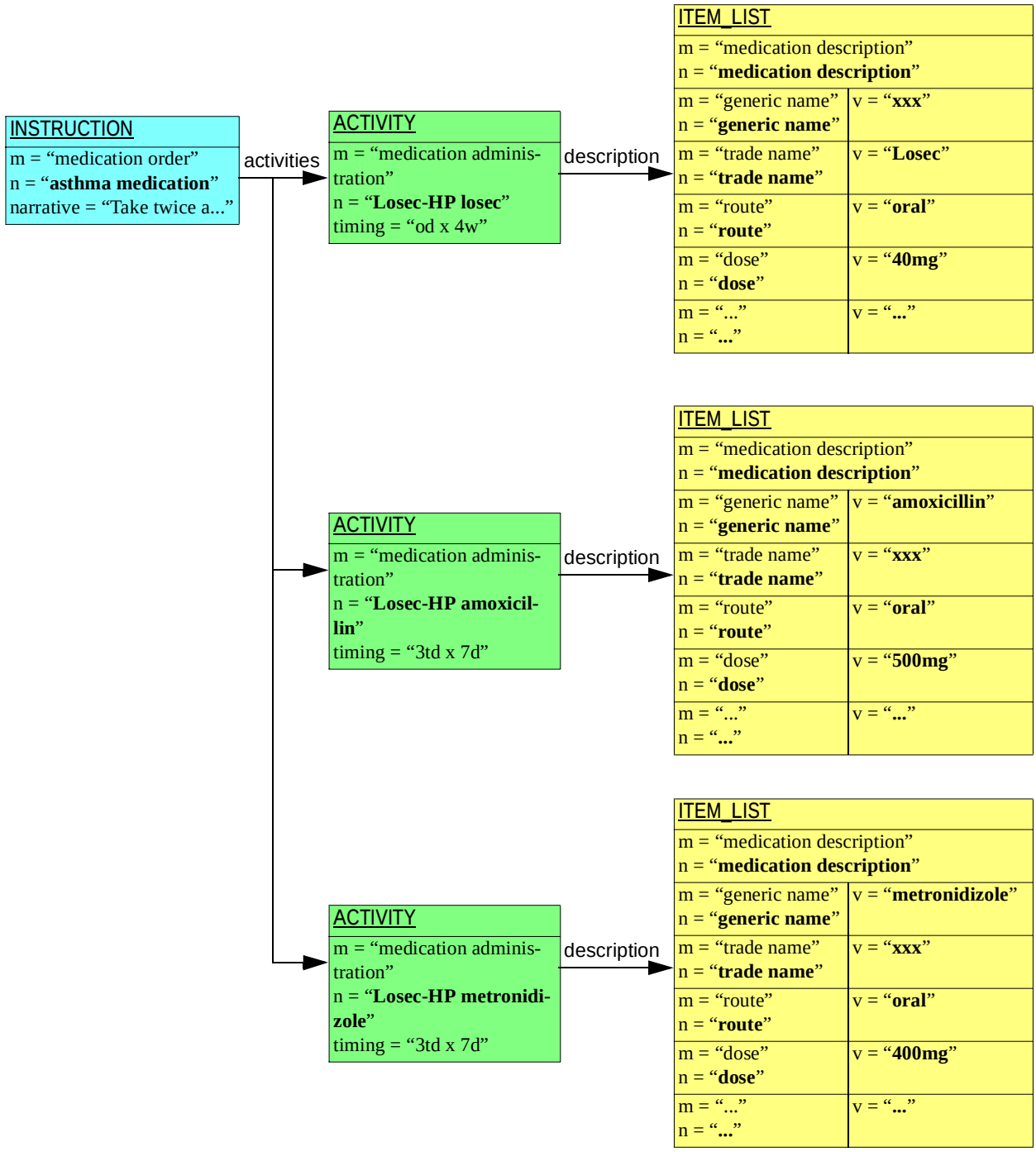

FIGURE 30 Multi-drug therapy Instruction

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. *A Systematic Theory of Clinical Recording*. 2006. Publication to be announced.

5 Beale T, Heard S. *A Model of Clinical Context for EHR Recording. 2006.* Publishing to be announced.

6 Beale T. *An Ontological Framework for Understanding Clinical Information*. 2006. Publishing to be announced.

7 Browne E D. Workflow Modelling of Coordinated Inter-Health-Provider Care Plans. 2005. PhD dissertation, University of South Australia. Available at XXXXXXXXXXXX.

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 XXXXXXXXXX.

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.

B.2 European Projects

16 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.

17 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.

18 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.

19 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.

20 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.

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

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

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

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

25 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.

26 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.

27 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

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

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

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

31 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

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

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

B.5 HL7v3

34 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

35 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+).

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

B.6 OMG

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

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

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

B.7 Software Engineering

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

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

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

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

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

45 Booch G, Rumbaugh J, Jacobsen I. *The Unified Modelling Language User Guide*. Addison es-

ley 1999.

B.8 Resources

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

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

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

49 EON ref required

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

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

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

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

END OF DOCUMENT