Distributed Knowledge Development Model

Editors: T Beale[a]		
Revision: 0.6	*Pages:* 25	*Date of issue:* 01 Jun 2009
Status: DEVELOPMENT		

[a] Ocean Informatics

Keywords: EHR, health records, repository, governance

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

Copyright Notice

Amendment Record

Issue	Details	Who	Completed
0.6	Rewritten from 2009 Governance document	T Beale	01 Jun 2009
RELEASE 1.0.1			
0.5	Restructured and rewritten	T Beale	12 Mar 2007
RELEASE 1.0			
RELEASE 0.95			
RELEASE 0.9			
0.3.1	Updated diagram; added notes on dissemination network.	S Heard	20 Dec 2003
0.3	Initial Writing	T Beale	29 Nov 2003

Acknowledgements

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe a model of distributed development of *open*EHR knowledge artefacts, i.e. archetype, template and terminology subset formalisms. This includes such artefacts created by the *open*EHR Foundation itself, as well as by other organisations. The model can be applied to any artefact type founded on the principle of two-level modelling.

The aim is that managed (rather than *ad hoc*) development is supported, such that most consumer organisations have the ability to a) find artefacts that suit their purposes, b) adapt found artefacts to their own purposes via various forms of localisation and translation, and c) be able to verify the quality and origin of used artefacts.

The model of development described here is closely based on the moderm model of distributed software development. Some of the key aspects dealt with include:

- an overview of the problem of distributed development and governance;
- requirements of a development and governance model for *open*EHR;
- a description of a development model and its nomenclature;
- an overview of other specifications describing the details of the model, including identification, release management, change management, quality assurance and governance.

Unless otherwise stated, in this document, the term 'artefact' refers specifically to these artefact types.

1.2 Related Documents

This document describes a model of development and governance. The details of various aspects of this model are found in various other specifications:

- Knowledge Artefact Identification System;
- Knowledge Quality System;
- Archetype and Template Semantics.

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://www.openehr.org/svn/specification/TRUNK/publishing/architecture/am/dist_dev_model.pdf. New versions are announced on openehr-announce@openehr.org.

2 Overview

2.1 The General Problem

Within e-health and other domains, the production of computational knowledge artefacts is growing. Two categories of knowledge artefact can be identified: semantic networks, such as ontologies and terminologies, and discrete self-standing models, such as *open*EHR archetypes, computerised clinical guidelines, and workflow definitions. Both are usually distinct from source software, being consumed in some way at runtime.

Two key challenges being faced are the same as within the software world: how to manage development and sharing among large numbers of organisations, and how to manage change over time.

> It has taken the ICT industry some 40 years to develop schemes for identifying software artefacts and to define rules for managing change. Only with the relatively recent phenomenon of large-scale open source projects like Linux, Eclipse and Apache.org have some of the necessary elements been worked out.

In order to avoid the logistical difficulties encountered during 40 years' software development, there is a need to define a scheme for managing the development and use of distributed knowledge artefacts. For both the 'semantic network' and the 'discrete models' category, the problems are the same, but due to the nature of the artefacts, the development models are likely to be somewhat different. This document describes a model of distributed development for *open*EHR artefacts. The scheme is likely to be applicable to all discrete knowledge artefacts.

2.2 The *open*EHR Environment

The *open*EHR methodology consists of the use of an information model to define domain-invariant generic concepts, over which archetypes, templates and terminology are used to define domain-specific models. The information model is designed to underpin the data interoperability of future e-health systems. The methodology can be used with any information model and in any industry or domain. *open*EHR's own information model (usually known as the *open*EHR 'reference model' or RM) defines concepts like `EHR`, `SECTION` and `OBSERVATION`, in all around 100 classes (including data types) used to model the generic characteristics of health information. Archetypes defined over this model are each based on a root class, and express concepts like 'blood pressure measurement', 'antenatal examination' and 'medication order' – in short, topic-based models of possible arrangements of the underlying RM objects. Templates are an artefact that are used to define a compositional structure of particular archetype data points, corresponding to a specific business purpose. A third kind of artefact, terminology subsets (also known as 'ref sets') can be deployed with archetypes and templates. Terminology subsets are referenced from archetypes and templates via 'bindings'. Finally, a fourth kind of artefact is the *open*EHR data query. A number of related syntaxes exist, with all relying on the use of archetype paths, and all being independent of database schemata. Not all queries merit being treated as designed entities, but many will be, particularly those designed for use in research studies and also decision support systems. All four kinds of artefact are usually developed by domain experts, in the case of *open*EHR, health professionals.

Archetypes, templates, terminology subsets and queries are separate from software, and have their own environment for development. They are disseminated separately from software, and are consumed at runtime by the software system, where they are used to create and/or validate data within information systems. As such, they do not yet have their own 'standard model' of development or governance, in the way that software does today.

There is accordingly a requirement for a defined model of development, dissemination, and runtime use of these artefacts. Underlying this is a need for an artefact identification system, and an overall governance framework.

2.3 Existing Models of Development

In order to construct a workable yet theoretically sound model of development of the knowledge artefacts produced in *open*EHR, we consider the precedents in the 'semantic network' and 'discrete model' categories, i.e. ontologies and software.

2.3.1 A Centralised Model – Terminologies and Ontologies

The first model is the development of terminologies such as ICDx and SNOMED-CT, in which a single, central ontological artefact is elaborated in a shared concept space managed by a particular organisation. All additions and changes to the artefact are assumed to be either within the core ontology, if created via the central change process, or within controlled peripheral repositories, if created in immediate response to local needs. The default assumption for most terminology users is that if the terms they require are not in the current release, they have to wait until they are put in by the central change process unless they are clearly terms that are only meaningful locally anyway. The alternative is that local environments create modifications to the central terminology. Such 'modifications' are necessarily a consistent part of the central terminology (otherwise they would constitute a separate terminology).

This approach appears to be sound because the artefact being created is a) highly interconnected and b) semantically 'strong'. Terminology development can be characterised as very centralised, with an accordingly weak concept of 'namespace', since the main idea is to populate the central concept space. The primary reason for this is that the 'atoms' of terminology derive most of their meaning from their relationships with other terms, rather than from their own names/linguistic definitions. In this world, the problem of independent creation of same-named terms with competing meanings is insignificant.

2.3.2 A Distributed Model - Software Development

The second model of development is that of re-usable software libraries. Software libraries are built within many companies, universities and other organisations today. They usually rely heavily on libraries (often called 'framework' libraries) from elsewhere. In most programming technologies, there is a hierarchy of library producers in which a small number of key organisations produce libraries relied on by most other development efforts. In the Java programming world for example, most developments rely on the core libraries from Sun Microsystems ('java' namespace). Other major libraries come from IBM (e.g. SWT, Eclipse-related libraries), Apache.org and other such sources. Most such core libraries slowly absorb improvements and additions over time both from internal development and donations from external sources.

Identification of software source artefacts in all major programming languages is 'ontological', i.e. Each artefact is named after its purpose, such as `LinkedList.java` rather some machine style of identification such as ISO Oids or GUIDs. Classes are normally arranged into 'packages', which are also identified with ontological names. A combination of hierarchical package structure, plus organisational naming leads naturally to the use of name-spacing to globally qualify software source artefacts, e.g. java.lang.reflect refers to a package of reflection classes in the core Java libraries issued by Sun Microsystems.

Binary artefacts, i.e. pre-built libraries and executables are typically identified with human-readable names that include version identifiers. For example, older Unix shared libraries use major and minor library revision numbers (`libfoo.so.1.2`) while contemporary Unixes use major revision numbers (`libfoo.so.1`)[1]. Note that the granularity of such artefacts is usually a library incorporating numerous source artefacts. In this sense, libraries are acting like 'named releases', and the library version identifier is normally the same as the source release identifier.

Software library development can thus be characterised as being highly distributed and having a strong notion of namespace, that allows the disambiguation of same-named concepts from different authors, e.g. competing ideas of a `LinkedList<T>` class. Unlike terms in a terminology, most software classes derive most of their meaning from their own definition rather than from their relationships with other classes; this is because they are 'encapsulated' artefacts with their own internal features (methods, attributes etc). Some large proportion of software libraries could be considered semantically strong, particularly those that implement basic concepts from computer science such as data structures.

2.3.3 Change Management

Managing changes over time is a key element of both the centralised and distributed development models. In the terminology/ontology world, the usual need is to perform a series of modifications to a current edition of the terminology, and then perform actions like 'diff' and 'patch' to extract or apply the changes to another edition, such as a client copy, or a 'trunk' version. The entire terminology is normally the unit of versioning. The semantics of changes can quickly become complex due to ripple effects through the relationship network.

In the software world, change control is typically enabled by file-system versioning tools. Software source artefacts do not mention versions of referenced artefacts, e.g. in class inheritance or association. Instead, the version of any referenced class is resolved by compilation tools against the source files currently available in the local (or sometimes a remote) repository. In other words, the author of a class that inherits or uses another class does not specify the use of a particular version of that class in the class source file – they make that decision at the level of their workspace (e.g. checked out from Subversion). When writing a new class file that references another class, they assume the 'latest available' locally, and the compilation tools take care of inconsistencies. Classes are generally updated in whole libraries or at least 'packages' that have been released as a unit. This is a sensible approach, because there will generally be multiple classes referencing any other given class; individually setting the versions of the referenced artefact in each source file is likely to lead to chaos in terms of version control.

2.4 A Development Model for *open*EHR

If we consider archetypes, templates and terminology subsets, we see that they have similarities with both terminology and software. Archetypes, subsets, and some templates are semantically 'strong' and need to be located within a well-designed concept space. As with terminology, they are essentially designed for interoperability, which is better served by having fewer large and highly coherent repositories, rather than numerous disconnected developments. On the other hand, they are like software classes in that they are encapsulated artefacts each containing their own interior definition which provides most of their meaning. In common with software, this latter fact means that useful archetypes can easily be developed by independent organisations rather than by a purely centralised process.

Also in common with software, there is a strong interest among archetype and template users in one or a few high-level shared libraries of high-quality artefacts rather than scattered development with poor quality control. In an ideal world, there might be only a single repository, or a purely hierarchical

[1] See wikipedia http://en.wikipedia.org/wiki/Library_(computing)

set of repositories. However, we know from experience with software development that this is extremely unlikely. Each organisation initially starts out with its own point of view, timelines and priorities. If a coherent ecosystem of cooperating repositories, or a single international repository were ever to exist, it would be as an emergent result of collaboration among initially separate authoring groups, rather than being established at the outset. In mid-2008, a repository was created at http://www.openEHR.org/knowledge. Other national and institutional repositories are likely to come online from 2009 onward. This trend appears to be similar to the emergence of large-scale open source projects.

> Based on the above considerations, the 'modern' model of software development is assumed to provide a suitable paradigm for *open*EHR knowledge artefact development, with emphasis on collaborative development of one or a small number of well-known artefact repositories.

3 Requirements

The key requirements of a distributed development model can be stated based on the assumption of the existence of the following distinct activities, within a distributed environment similar to that of software production:

- *development*, i.e. authoring, review, quality assurance, publication;
- *dissemination*, i.e. a means of publishing and distributing artefacts from development locations to user environments;
- *runtime use*, i.e. consumption of the artefacts by computing systems or applications.

In an *open*EHR-enabled world, each of these activities will occur whilst a given information system continues to be used operationally. The knowledge artefacts will evolve, and new versions will be consumed by the system, in many cases while maintaining references to previous versions of the same artefact. This further strengthens the requirement for a sound identification model.

3.1 Artefact Types

3.1.1 Archetypes

Archetypes are a semantically 'strong' artefact in that they generally define concepts that can be agreed and used across relatively large numbers of users, across multiple technologies, and are likely to remain valid over long periods of time. In technical terms, archetypes are expressed as a set of constraints on an underlying information model, augmented with concept identifiers.

Archetypes therefore exist within a 2-dimensional ontological space. The first dimension is defined by the information model – any object-oriented information model can be used as a basis for creating archetypes. A well-defined information model acts as a 'base-level' ontology. Within the space defined by a given information model, the second dimension is a domain concept space, in which each archetype defines a constraint model encompassing one or more (often numerous) variations of a basic domain information structure, such as 'blood pressure measurement' i.e. the content and structure of information captured to record a measurement of patient blood pressure. All such data instances 'conform' to the archetype in the sense that the instance structures match the archetype constraints in the way defined by the openEHR ADL and AOM specifications.

Within the concept dimension, specialised archetypes can be defined, whose constrained data instances are guaranteed to conform to their parent archetypes, recursively.

Modified forms of a given archetype over time are also accounted for, and this corresponds to a 3rd dimension. Changes that maintain data compatibility give rise to new 'revisions'. Any 'revision' of an archetype is guaranteed to be able to validate and process data created by previous revisions. Where a required change is incompatible with the current revision, a new 'version' is created, which is effectively a new archetype.

In summary, there are three dimensions which can be thought of as a 2-dimensional ontological space (giving the 'meaning' of the artefact, e.g. 'blood film observation') moving through a time/version dimension, in which changes due to ongoing development occur.

3.1.2 Templates

Technically, *open*EHR templates are expressed similarly to archetypes, with the additional function of being able to aggregate archetypes. They have more variable semantic significance than archetypes. Some templates are likely to be defined internationally and/or nationally, such as 'discharge summary', ASTM CCR[1] etc, which can be considered semantically strong, while the vast majority are likely to be defined in local environments outside of which they have little semantic significance. 'Strong' templates exist within a recognised concept space in the same way as for archetypes, while weak ones can be considered to exist within very local concept spaces corresponding to the authoring organisations.

Templates do not permit specialisation, but do allow inclusion of other templates.

3.1.3 Terminology Subsets

Some archetypes and many templates will specify the set of terms that are allowed to be used to provide the value of a particular element or name of a node. Such value sets of terms are known as terminology 'subsets' or 'ref sets', e.g. within LOINC[2], SNOMED[3], ICD[4] or ICPC[5]. There is often local agreement, within a hospital for example, to use an agreed subset.

Subsets of a terminology need to be integrated into the archetype and template development environment so that it is clear which subset can be used to populate which element in the archetype.

3.1.4 Operational Forms

*open*EHR knowledge artefacts are authored in a 'source' form, in a manner similar to software. They also have a compiled, or 'operational' form, which can be described as follows.

Archetypes: two operational forms:

- *flat archetype*: the flattened form of an archetype obtained by taking into account all parents (for specialised archetypes), and with internal references expanded out. This form is used for publishing in authoring tools, as a means of showing the ‘effective’ definition of a specialised archetype.
- *operational archetype*: same as the flattened form but also taking into account the relevant classes in the reference model. May optionally include only a subset of the languages and terminology bindings defined originally in the archetype. Used as the input to operational template generation.

Templates: one operational form:

- *operational template*: the combination of operational archetypes and template-introduced constraints, fully flattened into a single large archetype structure.

Terminology Subsets: one operational form:

- *subset result*: the result of evaluating the subset query statement against a target terminology.

3.1.5 Queries and the Query Language

Archetyped / templated data are queried in *open*EHR using a language such as AQL, that relies on archetype paths. Queries (other than *ad hoc* queries created at runtime) are designed based on clinical guidelines or other models, such as found in research studies. They are dependent on available archetypes, but not templates, and can be created independently of the latter.

[1] See http://www.astm.org/Standards/E2369.htm

[2] See http://loinc.org/

[3] See http://www.ihtsdo.org

[4] See http://www.who.int/classifications/icd/en/

[5] See http://www.woncaeurope.org/

The set of paths for an archetype is known as its 'path map'. The path map of an archetype is derived from the flat-form of the archetype. For specialised archetypes, this will in general include paths from parent archetypes, both unchanged and overridden. Archetype paths are designed so that a query based on a path from a given archetype should be constructable to find not only data created directly from that archetype, but also data created from specialisation children archetypes that override that path.

3.2 Development-related requirements

Within the development context, the following requirements can be identified.

3.2.1 Identification

An identification system for source and operational artefacts is needed that can prevent clashes in all of the following dimensions:

Ontological: archetypes and templates must be distinguishable within the 2-dimensional ontological space for which these artefacts were designed. In concrete terms, an ontological identifier needs to include the relevant information model identifier as well as the domain concept identifier. This would allow, for example, the disambiguation of:

- Archetypes for the same abstract domain notion, e.g. 'heart rate' based on a class from two distinct information models, e.g. the *open*EHR `OBSERVATION` class and the HL7 CDA `Observation` class;
- Archetypes based on different classes from the same information model to have the same name, e.g. An archetype for 'vital signs' headings based on the `SECTION` class, and a 'vital signs' archetype based on `OBSERVATION`.
- Archetypes for different domain notions to be created based on the same class from the same model, e.g. 'blood pressure measurement' and 'heart rate' based on the *open*EHR `OBSERVATION` class.

Versioning: a way is needed of distinguishing the versions of an artefact as it changes over time.

Publisher: given the existence of multiple authors and publishers of artefacts, it must be possible to distinguish artefacts adequately. For example, two archetypes based on the same RM class and using the same domain identifier, e.g. *open*EHR `OBSERVATION` class + 'blood pressure measurement'.

3.2.2 Referencing

Rules are required for how to create references between source artefacts within the development environment, including:

- how a specialised archetype refers to its parent, including a parent defined by another publisher;
- how an archetype slot pattern is defined;
- how an archetype slot filler is referenced from within a template or archetype;
- How a template references another template for purposes of inclusion;
- how a terminology subset is referenced from an archetype or template.

These requirements lead naturally to a need for how an authoring environment can 'import' or reference a whole release of artefacts from elsewhere.

3.2.3 Version and Life-cycle Evolution

Rules for the evolution of artefacts over time are required, including rules for version / revision advancement. Kinds of change include:

- creation of a new artefact;
- factoring of existing archetypes to create a new archetype containing data points already being collected by the previous form of the archetypes;
- addition of a new specialisation of an archetype;
- change to descriptive meta-data;
- change to constraint definition that is consistent with previous revision of archetype;
- change to constraint definition inconsistent with previous revision of archetype;
- addition of, change to or removal of a translation language;
- change to meanings of ontology entries;
- superseding of an artefact.

Changes to terminology subsets include:

- change to subset contents;
- change to 'source binding' reference of subset.

All changes will initially occur in an authoring environment, and will, following quality assurance, trickle out to production environments, finally arriving within operational information systems. The effects of the passage of any given type of change should be describable and testable.

3.2.4 Movement of Artefacts

Rules for identification of artefacts due to moving of artefacts from one publisher environment to another are required.

Traceability of the history of moves of an artefact should be possible.

3.2.5 Quality Assurance

It must be possible to distinguish an artefact that has passed a QA process from one that hasn't, particularly an earlier or later revision of the same artefact.

Artefacts that have never been quality assured should be able to exist, but must be easy to distinguish from quality assured artefacts.

3.3 Dissemination Requirements

3.3.1 Release Management

A way of identifying releases of artefacts created by publishers is required, including the following capabilities:

- to be able to define a release that includes locally created artefacts and / or one or more releases of artefacts from other publishers.

3.3.2 Authentication

A way is needed of authenticating an artefact (say a local copy) which is marked with a given identifier as being a faithful copy of the artefact thus identified by the relevant publisher. Authentication needs to account for the fact that any particular copy of the artefact in question may not be exactly the same as the original in terms of white space or ordering of items within lists.

3.4 User-space Requirements

3.4.1 Disambiguation

It must be possible to assemble in an end-use environment all the desired archetypes, templates and terminology subsets, including from different publishers, without clashes occurring.

It must be possible to determine when a changed version of an existing artefact has been received, and in what way the new version is different.

3.4.2 Referencing from Data

When archetypes and templates are used to create data, adequate references must be included in the data to ensure that at a later point in time, the same data can be retrieved from a persistence system and re-associated with the correct archetypes and templates.

3.4.3 Querying

It must be possible for a query engine to query data based on the archetypes from which they were created. In particular for specialised archetypes, it must be possible to query the data using both the specialised child, as well as any parent archetype in its lineage.

3.4.4 Introspection

TBD_1: A need to be able to identify the provenance of artefact from inside the artefact? (EB)

4 Development Model

4.1 Overview

Responding to the requirements described above entails two things. Firstly an abstract development model must be described, providing structure and nomeclature. Secondly, formal rules are defined with respect to the model, such as for identification, sharing, and change management. This section describes a model, illustrated in FIGURE 1.

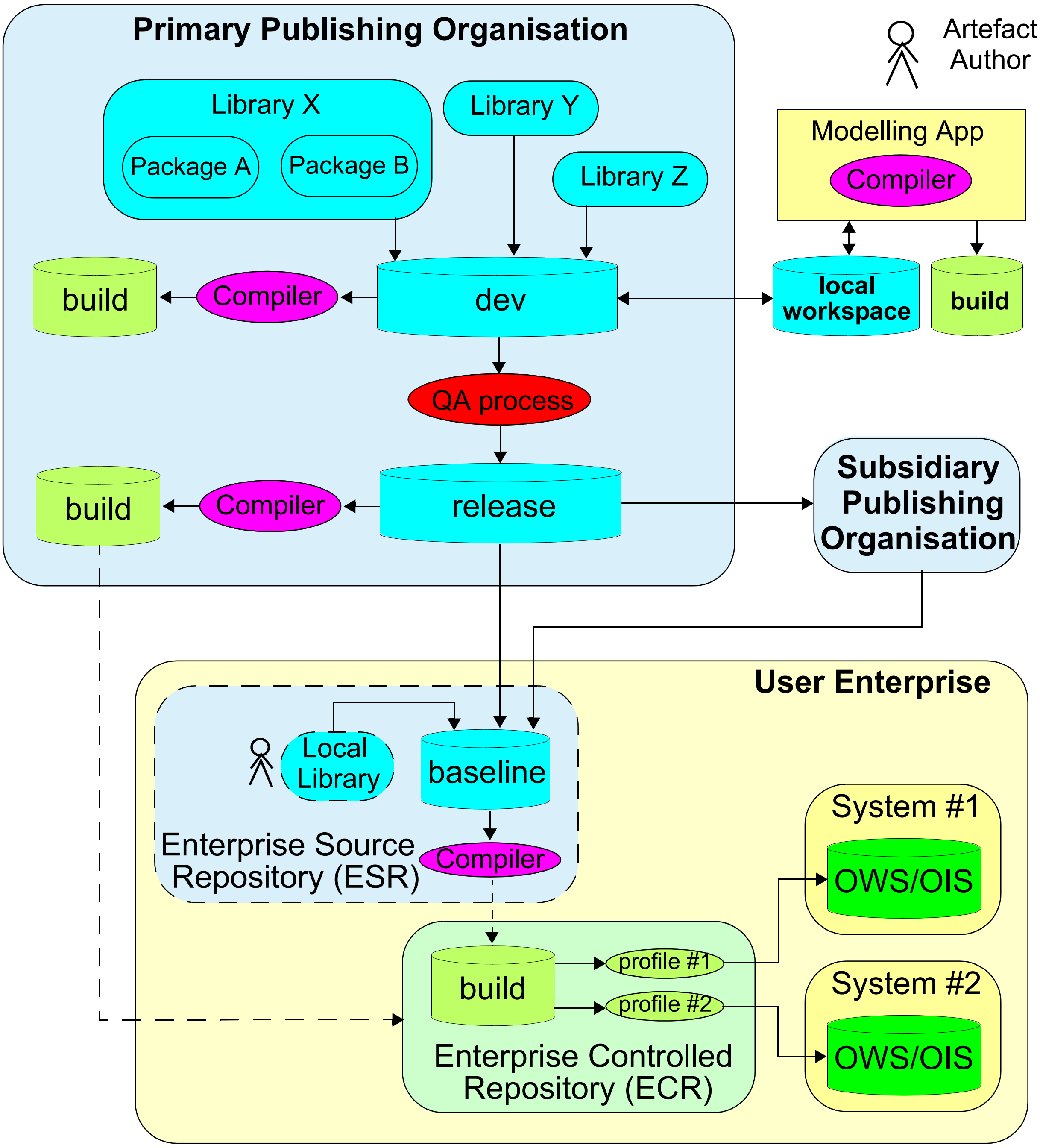

FIGURE 1 Distributed Development Model

The environment is broadly divided into two parts, the *production environment* and the *consumer environment*.

4.2 The Production Environment

The production environment consists of human experts using tools, who author source artefacts. This either occurs via check-out and check-in to the managed environment of a *Publishing Organisation* (PO), or in an uncontrolled manner.

4.2.1 Artefact Organisation

Publishing Organisations (POs) create and maintain *Libraries* of artefacts. A Library is a top-level collection of artefacts based on the same information model, for a particular purpose. A Library is normally versioned and released as a unit.

Libraries are a kind of *Package*, and contain further Packages and/or source artefacts (i.e. Packages are not mandatory within Libraries). A package is a recursive grouping construct containing other Packages and/or source artefacts. If a Library were to be absorbed into another for reasons of consolidation, it would become a new Package within the receiving Library.

Source artefacts within a PO library are identified with qualified identifiers indicating the PO and their Library and Package location. They may also be digitally signed.

New revisions of source artefacts can only be revised by their current Publisher.

Artefacts may also be authored within a User Enterprise, e.g. templates, specialised archetypes, and modified terminology subsets. If these are to be used, they should be developed and quality assured in the same way as for artefacts from POs. Such artefacts will be uniquely identified in a similar manner as for those from PO Libraries.

Uncontrolled authoring can also occur, using the same tools, but without the artefacts being committed to a Library. Uncontrolled authoring typically occurs 'in the wild' such as by students trying tools and physicians creating private models. Uncontrolled artefacts carry unqualified identifiers and cannot be signed.

4.2.2 Authoring Paradigm

Artefact authors have a logically *private workspace* copy of the Library in which they are working, and the ability to refer to other libraries. Whether this is achieved by checkout onto a local file system, or by some web-based method is semantically immaterial.

A *Compiler* generates various *operational* forms of the artefacts. The compilation process ensures that artefacts are technically valid and mutually coherent.

Within a Library of source artefacts, references are always *resolved* against the current and any other Libraries available in the local workspace by the Compiler, i.e. the question of which versions of referenced artefacts are used is addressed by version control mechanisms at the workspace level, in the usual manner of software development.

4.2.3 Publisher Organisation

A PO may maintain multiple Libraries, e.g. created by different authoring groups, or for different purposes. At any moment in time, some version of each library will be at some stage in a quality assurance process, and will be available in an identified release from the PO. A *Primary Publishing Organisation* (PPO) generates all its own artefacts. A *Subsidiary Publishing Organisation* (SPO) imports release(s) from a PPO(s) or other SPO(s), and adds its own artefacts. Its own releases consist of the content of its own artefacts plus references to imported releases against which its own artefacts were built. The PPO containing artefacts based on the *open*EHR reference model for example, is identified as `org.openehr`.

As with software, a means of permanently transferring a Library or package to another PO is required. This occurs when the library is found to be widely applicable and/or when an organisation wants to divest itself of the maintenance responsibility. When this occurs, the Library identifier will usually change, since it will incorporate the domain name of the new organisation.

POs can publish artefact releases both in source and built (i.e. operational) form, in the same way as open source software is usually published in both source and binary form.

4.2.4 Governance

Publishing Organisations can include government agencies, solution vendors and healthcare delivery enterprises, and need to be registered and certified. Certification is carried out by the *Central Governance Authority* (CGA). A PO achieves certification if it can show that it has established a quality assurance system (QAS) meeting the minimum requirements for *open*EHR knowledge artefacts. Certified POs are given a digital key pair that can be used for creating signatures of published artefacts and libraries. The public key can be verified by any consumer organisation by checking with the CGA (e.g. via its website). The private key is used by the PO for signing assets, and will not be available to any other PO.

A vendor may be set up as a registered PO, or it may author its artefacts in an uncontrolled way. The latter situation is not likely to be advantageous to the vendor, since the main aim of most vendors is wide usage of their product. This is particularly true where vendor-produced *open*EHR knowledge artefacts are mixed in with those from the user enterprise or those from other POs. Vendors therefore need to have ways of being recognised as Publishing Organisations or at least using tools and methods that allow them to obey the rules for safe artefact sharing.

4.3 The Consumer Environment

The consumer environment is where knowledge artefacts are used by computing systems. These are usually EHR or related systems, but may be any computational application that uses archetypes and templates to process data or perform some analytic task, or to create queries. The artefacts used in any given system might come from a recognised PO, and/or from local authoring.

4.3.1 User Enterprise

A User Enterprise is a location where one or more information systems using *open*EHR knowledge artefacts are found. In order to manage knowledge artefacts in a safe way across the enterprise, an *Enterprise Controlled Repository* (ECR) is established. This is a repository of operational knowledge artefacts, and is the source point for all knowledge artefact use in the enterprise. The ECR is may be directly populated with release builds from external POs (e.g. a national server), or it may be used to build source releases from POs. It is also used to build any locally produced artefacts. The entirely of artefacts from all sources must form a single coherent release within the ECR. In particular, there cannot be competing revisions of the same artefact present at the same time.

If local artefact production is undertaken, or if source releases are obtained from external POs, the enterprise will also maintain an *Enterprise Source Repository* (ESR), in which a coherent source release is established as an input to building tools.

In order to ensure the consistency of the ECR, various change management rules should be applied to any updates made to it, including:

- any new revision of an existing artefact is valid with respect to the previous revision, according to the rules for the artefact type;
- any specialisation of an existing artefact is consistent with respect to its parent artefacts, according to the rules of specialisation of the artefact type;
- any new or updated artefact is clinically safe and does not produce unexpectedly changed semantics;
- no update invalidates existing data.

Implementing these checks requires not only a Compiler tool, but also a 'diffing' tool that can compare current and previous revisions of a given artefact.

4.3.2 Information Systems

Any knowledge-enabled system or application obtains its operational artefacts from the Enterprise Controlled Repository. Any given system may use only a small subset of the available artefacts, defined by a 'profile' for that system. Within the system or application, the final set of artefacts actually in use is known as the *Operational Working Set* (OWS). In any system of signficance, the OWS will be persisted in some form, possibly in a runtime-efficient format derived from the standard operational form. Regardless of the kind of system or application, the OWS profile should be persisted within the ECR, e.g. as a 'baseline'. For the safety and maintainability of the system, it is crucial that the OWS be established and maintained in a consistent state throughout its life.

As for the ECR, various rules should be used for updating an OWS. The ECR change management rules will ensure overall validity, however, there may be a need to limit updates to any given OWS within an enterprise, due to legacy needs of the corresponding system.

Not all data in a system is guaranteed to be based on artefacts from the OWS, since data may be imported from other systems that use different artefacts. We denote the artefacts referenced in the imported data as the *Operational Import Set* (OIS).

TBD_2: consequences of allowing data based on uncontrolled archetypes into a system

TBD_3: relationship between OWS and OIS

5 Overview of Detailed Specifications

The model described in the previous section provides a definitional framework on which a number of detailed specifications can be based. These are available within other documents, and are summarised here.

5.1 Identification

A primary feature of any organised system of artefact development and use is a system of identification. The requirements section indicated the need for identification that disambiguates artefacts on the basis of:

- *ontological*: 2-dimensional space defined by cross-product of RM type and domain concept type - needs to support the notion of specialised artefacts (i.e. an idea of inheritance);
- *version*: multiple versions of the same artefact;
- *publisher*: originating organisation.

The need for a definition of referencing between various types of source artefact, operational artefact and data was also indicated. The development model in the previous section adds a hierarchical library/package concept as used in modern software development. We can also assume from the world of software development that 'versions' are likely to be needed at coarse- and fine-grained levels, with an accompanying scheme for version/revision advancement in time. Similarly, we can assume the need for a way of defining and identifying releases consisting of multiple artefacts, each at some defined revision in their lifecycle.

Other requirements include: backward compatibility for existing archetype identifiers (defined by the `ARCHETYPE_ID` class in the *open*EHR RM), typified by the following:

```
openEHR-EHR-EVALUATION.problem.v2
```

A system of identification resulting from all the the above considerations turns out to a relatively simple evolution of the existing form of archetype identifier, with the following characteristics:

- a *generic scheme* that applies to *open*EHR archetypes, templates and terminology subsets, and could be applied to other kinds of knowledge artefact;
- the use of *namespaces* to identify Publishing Organisation and Library/Package, e.g.
  `org.openehr.clinical::openEHR-EHR-EVALUATION.problem.v2`
- relaxing the existing rule for identifiers for specialised artefacts, to allow any concept name rather than a concatenation of concept names down the specialisation hierarchy, i.e. 'diagnosis' rather than 'problem-diagnosis'. However the old-style names are still legal and supported;
- a *fine-grained version numbering* scheme, that enables any 'commit' of any 'revision' to be referred to;
- *rules for naming files* and directories for serialised (i.e. file-system persisted) forms of the artefacts;
- an ability to define a *configuration* of source artefacts that were used to make up a particular operational artefact, such as an operational template;
- a flexible method of *referencing archetypes* from data that supports querying with any archetype in the specialisation hierarchy;
- a means of establishing *integrity and authenticity* of published artefacts using digital hashes and signatures.

A system of identification of the ‘release’ concept in *open*EHR is also included. The scheme is described in detail in the *open*EHR Knowledge Artefact Identification specification.

5.2 Service Models

Part of the technical infrastructure required to support the trajectory of knowledge assets from the authoring environment to runtime use is a definition of service interfaces of the various parts of the system. FIGURE 2 illustrate the interfaces from the overall model of FIGURE 1.

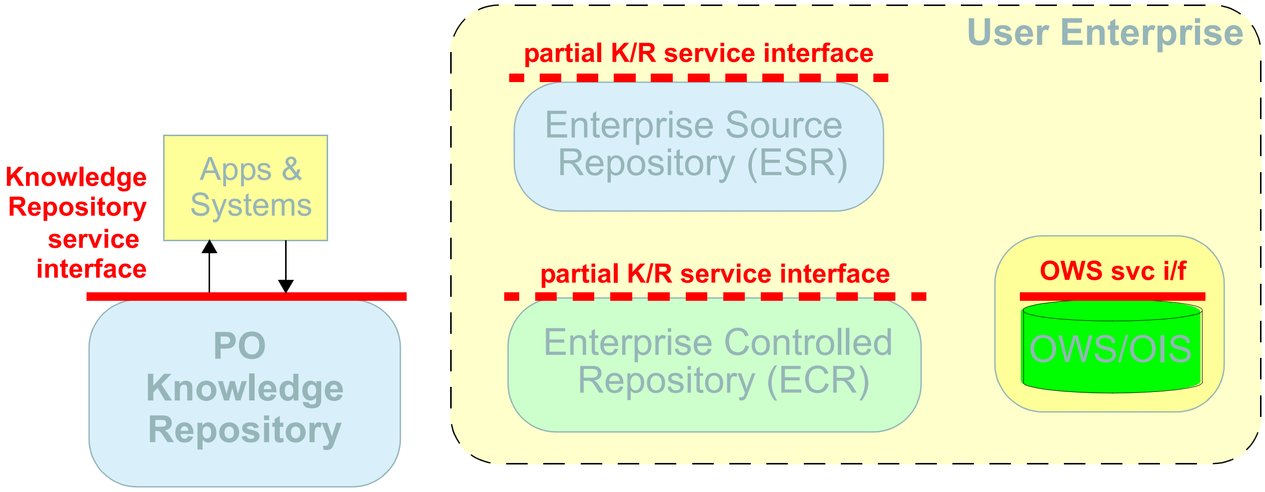
User Enterprise
partial K/R service interface
Enterprise Source Repository (ESR)
Knowledge Repository service interface
Apps & Systems
partial K/R service interface
OWS svc i/f
PO Knowledge Repository
Enterprise Controlled Repository (ECR)
OWS/OIS

FIGURE 2 Service Interface View

These interfaces are defined in detail in other *open*EHR specifications, and are summarised as follows.

5.2.1 Knowledge Repository (K/R) service

The knowledge repository exposed by a Publishing Organisation needs to be able to respond to requests of various kinds of tools and systems. Its service interface is therefore made up of groups of functions, including:

- check-out / check-in application tool interface;
- artefact id-based artefact search & retrieve interface;
- meta-data -based artefact search and retrieve interface, allowing artefacts to be retrieved on the basis of meta-data attributes and ontological classification;
- relationship and dependency querying interface, allowing e.g. the identities of all artefacts dependent on a given artefact to be retrieved;
- statistics / reporting interface, allowing statistical and reporting indicators to be retrieved;
- export interface, allowing artefacts to be obtained in exported formats, e.g. compressed archives.

Both the Enterprise Source Repository and Enterprise Controlled Repository are likely to support a reduced form of the K/R service. One important semantic not found in software repositories is the need to compare previous and current revisions of artefacts before effecting check-in, since only some changes are allowed in a new revision of an existing artefact in *open*EHR.

5.2.2 Operational Working Set (OWS) service

An OWS service should be present in some form in any archetype-based system. Its job is twofold. Firstly it has to serve archetypes, templates and terminology subsets based on identifier, within the local system. Secondly, it must provide an update interface that enables the contents of the OWS to be updated, ensuring (as with the K/R service) that no illegal changes are made. Use of digital signatures is likely to be made in order to implement the latter semantics.

5.3 Governance

The principal function of governance in the distributed knowledge environment is to ensure quality. This is performed by a Central Governance Authority (CGA) by the following means:

- designate (and possibly develop) a minimal Quality Assurance framework against which particular QA programmes used by publishers and user enterprises can be evaluated, and which defines the minimal quality criteria for safe knowledge artefacts;
- register Publishing Organisations, including a private artefact signing key;
- certify quality assurance programmes (QAPs) used by Publishing Organisations;
- certify key open source tools such as compilers and validators with respect to openEHR published specifications.

It is not the CGA's job to directly perform quality assurance; rather this is done by POs and user enterprises. A governance plan for *open*EHR knowledge resources is described in detail in a separate document.

5.4 Quality Assurance

Given that e-health knowledge artefacts are likely to become widely used in health information systems, their validity is clearly of key concern, since their use can materially affect patient outcomes. Quality assurance (QA) of such artefacts is therefore of major importance. Quality assurance is a complex area, and guidelines are in development by various organisations including the *open*EHR Foundation, projects like the EU-funded Q-REC project. Such guidelines need to be formulated into a framework of tools and procedures that can be efficiently (i.e. economically and reliably) applied to knowledge artefacts proposed for use in real systems.

As a basic premise, we can assume several levels of QA. The minimum level of quality is concerned with the technical correctness of an artefact, i.e. whether it is a legal instance of its formalism; then whether its descriptive elements including translations are formed correctly. The next level of quality has to do with domain validity, and encompasses issues such as the following:

- whether the correct balance has been found between structuring used in archetypes and the semantics of terminology (e.g. the well-known problem of expressing body site + laterality structurally using two attributes, or using a terminology post-coordination in one attribute);
- whether term-bindings between archetypes and templates, and terminology, are correct;
- whether correct subsets have been used.

All of the above aspects of quality should be testable relatively objectively. A further group of quality issues falls under the rubric of ‘good design’, and impacts re-usability and clinical correctness. Some of these include:

- minimisation of overlap;
- identification and factoring out of common sub-parts;

- design pattern conformance, e.g. in areas like questionnaires, exclusions etc;
- appropriate clinical scope of an artefact.

These items should be testable within a peer-review context, possibly with some tool support. Further quality aspects that may be difficult to objectively evaluate, but need to be considered include:

- performance in runtime systems, particularly for data queries and terminology subsets;
- the effect of non-breaking changes across revisions to archetypes, particularly in changes to terminology subsets;
- whether good user interface qualities are enabled, e.g. intentional terminology subsets should support an efficient and semantically correct experience at the user interface for a clinical user.

The effect of changes over time and revisions in particular is an important aspect of quality, since the aim is not just to establish quality at one point in time, but to maintain it into the future, recognising that at all points in time, data are being created and processed with the artefacts available at any one moment.

Guidelines for quality assurance of *open*EHR knowledge artefacts is described in a separate document.

END OF DOCUMENT