LinkEHR-Ed: A multi-reference model archetype editor based on formal semantics

José A. Maldonado[1], David Moner[1], Diego Boscá[1], Jesualdo T. Fernández-Breis[2], Carlos Angulo[1], Montserrat Robles[1].

[1]Grupo de Informática Biomédica (IBIME), Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas (ITACA), Universidad Politécnica de Valencia, Valencia, Spain.
[2]Departamento de Informática y sistemas, Universidad de Murcia, Murcia, Spain.

Address for correspondence:

José Alberto Maldonado-Segura
Grupo de Informática Biomédica (IBIME)
Instituto ITACA
Universidad Politécnica de Valencia
Camino de Vera s/n, Edificio 8G
Acceso B – Nivel 3
46022 Valencia
Spain
Email: jamaldo@upv.es
Fax: +34963877279

Abstract

Purpose. To develop a powerful archetype editing framework capable of handling multiple reference models and oriented towards the semantic description and standardization of legacy data.

Methods. The main prerequisite for implementing tools providing enhanced support for archetypes is the clear specification of archetype semantics. We propose a formalization of the definition section of archetypes based on types over tree-structured data. It covers the specialization of archetypes, the relationship between reference models and archetypes and conformance of data instances to archetypes.

Results. LinkEHR-Ed a visual archetype editor based on the former formalization with advanced processing capabilities that supports multiple reference models, the editing and semantic validation of archetypes, the specification of mappings to data sources, and the automatic generation of data transformation scripts.

Conclusions. LinkEHR-Ed is a useful tool for building, processing and validating archetypes based on any reference model.

Keywords: electronic health records, biomedical informatics, standardization, archetypes, semantic description, semantic interoperability

1. Introduction

Health care is a sector where the need to share information is the norm rather than the exception. However, the information about the health of one person is usually scattered among all of the different health facilities where she/he has ever been attended leading to distributed and heterogeneous data resources and making the exchange of data across systems very difficult. This situation has created a large gap between the potential and actual value of the information contents of health data repositories.

Due to the special sensitivity of health data and the wide range of ethical and legal constraints, the exchange of data must be done in a meaningful way, avoiding the possibility of misunderstanding or misinterpretation. The faithful communication and multiple usability of Electronic Health Records (EHR) crucially depend on the standardization of its syntax, structure and semantics, i.e. on the standardization of the EHR architecture and vocabulary used to communicate data [1].

Currently there are several international organizations working on the definition of an EHR architecture [2][3]. Health Level 7 (HL7) [4] maintains the XML-based Clinical Document Architecture (CDA) [5] that specifies the structure and semantics of clinical documents for exchange. The Technical Committee 251 (health informatics) of the European Committee for Standardization (CEN/TC251) has completed a European Standard for the communication of the EHR called EN13606 [6] whose part 1 (reference model) [7] became an ISO standard in February 2008. The openEHR consortium [8] maintains an architecture designed to support the constructions of distributed, patient-centered, life-long, shared care health records. Finally, ISO provides a set of clinical and technical requirements for an EHR architecture that support using, sharing and exchanging EHRs in the technical specification TS 18308:2004 [9].

OpenEHR and EN13606 utilize the dual model architecture to describe the EHR structure. The dual model or two-level methodology is a novel approach for the development of EHR systems. It aims to overcome the disadvantages of traditional “one-model” methodologies in which domain concepts are hard-coded directly into database models and software, when applied to complex and changing domains such as medicine. As stated by Rector [10] medicine is not only big but also open ended in breadth, depth and complexity. As a consequence health information systems that follow these methodologies need frequent, complex and expensive updates to accommodate new requirements which if not made, cause the systems to suffer “creeping obsolescence” and as result becomes obsolete. The dual model methodology distinguishes a reference model and archetypes [11]. In a broad sense, a reference model is an abstract representation of the entities and relationships of a domain which is designed to provide a basis for the development of more concrete models and implementations. The generality of the reference model is complemented by the particularity of archetypes [12][13]. Archetypes are formal definitions of clinical concepts in the form of structured and constrained combinations of the entities of the reference model.

The work reported in this paper is part of a bigger project called LinkEHR whose main objective is the utilization of the dual model methodology for semantic integration and normalization of health data. In LinkEHR we use archetypes to describe the semantics of legacy health data in a manner independent of the particular data organization in the underlying data repositories. This will enable users (mainly health professionals) to view and query data repositories at the level of its relevant semantic concepts. In this paper we describe the archetype editing framework of LinkEHR that includes a formalization of the definition section of archetypes and an archetype editor, called LinkEHR-Ed, which supports multiple reference models. The remainder of this paper is organized as follows. Section 2 provides an overview of the dual model approach. Section 3 deals with the formalization of the definition section of archetypes. Section 4 describes LinkEHR-Ed, a visual multi-reference model archetype editor based on the proposed formalization. Finally, Section 5 discusses and concludes the paper.

2. Background

The most remarkable feature of the dual model approach is the complete separation of information models, such as software models or database schemas, from domain models such as blood pressure measurement, discharge report, prescription or microbiology result. The information model is represented by a stable and small object oriented reference model that models the generic and stable properties of health record information. Only this model is hard-coded in database schemas or software. The possible numerous and volatile domain concepts, represented as archetypes, are modeled separately by domain specialists and their definitions are maintained in a shared repository [14]. Since the software is only bound to the reference model it has no direct dependency on domain concepts. Therefore, when new concepts are introduced or existing ones are altered the system does not need to be updated.

Archetypes are formal definitions of a distinct domain-level concept in the form of structured and constrained combinations of reference model classes. Their principal purpose is to provide a powerful, reusable and interoperable way of managing the creation, description, validation and query of EHRs. From a data point of view, archetypes are a means for providing semantics to data instances that conform to some reference model by assuring that data obey a particular structure and satisfy a set of constraints. The semantic description of domain concepts is achieved by linking the data structures and content to knowledge resources such as terminologies and ontologies [15].

Only those classes of the reference model that define logical building blocks of EHRs can be used to construct archetypes, we call them business concepts. For instance ISO/CEN EN13606 defines six business concepts, namely: Folder, Composition, Section, Entry, Cluster and Element. What is important here is that for each domain concept, a definition can be developed in terms of constraints on structure, types, values, and behaviors of business concepts. Archetypes may be specialised: an archetype is considered a specialisation of another archetype if it specifies that archetype as its parent, and only makes changes to its definition such that its constraints are narrower than those of the parent.

ADL (Archetype Definition Language) [12] is a formal language developed by openEHR for expressing textually archetypes that has also been adopted by EN13606. The most important section of an archetype is its definition tree, where clinical concepts are defined by constraining a particular business class in several different ways. For a thorough explanation of ADL we refer the reader to [12][13]. We also refer the reader to [16], an archetype repository with advanced search capabilities.

3. Methods

Although ADL was designed to be a formal language, the current ADL specification is not precise enough regarding some important issues such as archetype specialization and semantics. The main purpose of archetypes is to provide a mechanism to describe, validate, store and query health data. Since the data instances defined by archetypes are also instances of the underlying reference model, understanding the relationship between reference models and archetypes becomes crucial in order to manipulate archetyped data. We should take into account that ADL is predicated on the existence of object-oriented reference models and the constraints in an ADL archetype are in relation to types and attributes from such a model [12]. But ADL defines hierarchical data structures that mimic the tree-like nature of EHR. Furthermore, archetypes can be specialized. It is not clear if the constrainment of reference models (object-oriented data model) is equal to the specialization of archetypes (hierarchical data model), i.e. if it is possible to apply the same logic to both mechanisms. In our framework both mechanisms are modeled in the same way. As will be explained, this feature eases the development of algorithms for the semantic validation of archetypes and the assignment of types to data. There are also other open questions such as whether a multi-valued attribute can be narrowed down by a single-valued attribute or whether lists have narrower semantics than bags, i.e. a multi-valued attribute with bag semantics can be specialized by an attribute with list semantics.

The aforementioned issues hinder gaining a comprehensive understating of archetypes and beset enrichment of the tooling for archetype authorship. Therefore, we found imperative to define a formal modeling framework as a prerequisite to implement tools and algorithms providing enhanced support for archetypes. Since our ultimate objective is the generation of standardized EHR extracts we focused on the data definition facet of archetypes. It is in the archetype definition section where restrictions on the valid structure, cardinality and content of EHR instances are stated. To put it in other words, the definition section can be viewed as a database or document schema describing a subset of the underlying business concept data instances. In order to provide formal semantics to the definition section of archetypes we have developed:

- A data model for the description of data instances (EHR data).
- A type system that formalizes the definition section of archetypes.
- A subsumption function that models the specialization of archetypes and the relationship between reference models and archetypes.

It should be noted that our purpose is to describe formally the semantics of archetypes independently of the syntax used for their specification rather than proposing an alternative syntax. A complementary explanation of our framework with additional examples can be found in [17].

3.1. Data model and type system

Since archetypes impose a hierarchical structure to the EHR we have chosen a data model based on trees with labeled nodes to formalize the EHR data instances. This model is similar to the models described in [18][19]. A particularity is that nodes can be either ordered or unordered (contrary to XML where the order of nodes is significant), in the sense whether or not there is an order on their outgoing edges. The abstraction of data instances is straightforward. Each data instance is described by a data tree whose nodes represent either attributes or classes. As an example Figure 1 shows the representation of an instance of the openEHR *Element* class. Note the alternation of class names and attribute names in any branch as in ADL.

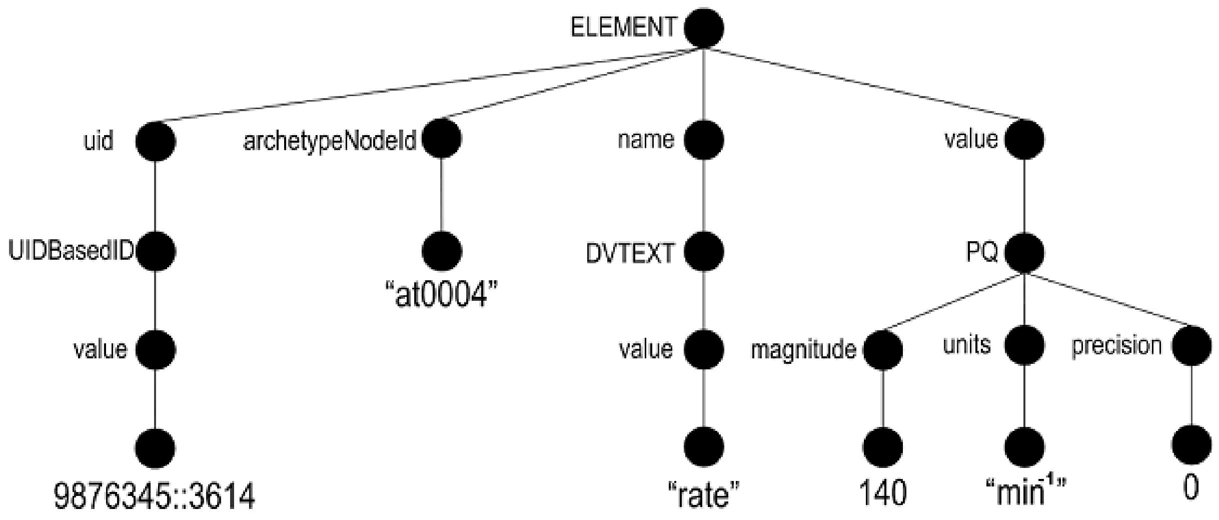

Figure 1. Abstraction of a data instance of the openEHR Element class

For the abstraction of the definition section of archetypes we have developed a type system that models the structural constraints that can be specified by archetypes. This type system allows the specification of sets of data trees, in our context, the sets of data instances of the reference model that satisfy the constraints imposed by archetype. First we need to introduce the concept of constrained multiplicity list, which is the base of our formalization. A constrained multiplicity list (CML) is a language definition expression of the form:

$$\left(t_1^{(l_1:u_1)} \ldots t_n^{(l_n:u_n)}\right)^{|l_0:u_0|}$$

Where $n \geq 1$, l_i is a natural number, u_i is a natural number or ∞, t_i is a type name, $l_i \leq u_i$ and $\sum_{i=1}^{n} l_i \leq u_0$ and $\sum_{i=1}^{n} u_i \geq l_0$.

Intuitively the language generated by a CML is composed of all the words defined by the regular expression $t_1^{(l_1:u_1)} \ldots t_n^{(l_n:u_n)}$ whose length (number of symbols) is between l_0 and u_0 inclusively. As an example, let us consider the CML $(A^{(1:2)}.B^{(1:3)})^{|3..4|}$ whose language is all the words of length 3 or 4 consisting of 1 or 2 A's followed by 1, 2 or 3 B's. Only four words satisfy these conditions:

{ABB,ABBB,AAB,AABB}. In order to make CMLs less verbose, we will drop the length constraints when they do not add extra information, for instance we will write $A^{(0:*)}$ instead of $(A^{(0:*)})^{|0:*|}$.

In our framework an archetype entity (attribute or object) is abstracted by a type that states the set of valid labels and children that the entity can have. We express such type definitions as rules. A rule can have either the form $t := l_t \langle r_t \rangle$ (when the children are ordered) or $t := l_t \{r_t\}$ (when the children are unordered), where *t* is a type name, l_t is a label predicate that describes the valid set of node labels and r_t is a CML that specifies the valid sequence of children. In this setting the definition section of ADL is abstracted by a set of types. A set of types that models an archetype is called a schema in our framework. We need to define the semantics of a schema, i.e. the set of data terms that it defines. Intuitively, a data term D conforms to a schema S if it is possible to assign to every node d of D a type T_i from S, and d satisfies the label predicate and the CML of T_i .

The representation of reference models with the previous type system is straightforward. A class C with attributes $A_1,\ldots,A_n$ can be abstracted by a type definition as follows:

$$Type_C := is\_C\left\{Type_{A_1}{}^{(1..1)} \ldots\ Type_{A_n}{}^{(1..1)}\right\}^{|n:n|}$$

Where $Type_{A_i}$ is the type that models the attribute A_i . In the case of primitive types, i.e. types that are always assigned to leaf nodes of data terms, their CML defines the empty string, denoted by ε. The labels of such nodes are values of primitive types, such as *Integer*, *String*, etc. for which we have defined a corresponding label predicate: *is_Integer*, i*s_String*, etc. For instance, the type definition that abstracts the Integer type is as follows:

$$Integer := is\_integer\{\varepsilon\}$$

Similarly attributes are represented by type definitions as follows:

$$Type_A := is\_A\{CML_A\}$$

where CML_A is a CML over the set of type names that describes the valid values that the attribute can hold.

The wide range of archetype constraints on data instance can be represented with the proposed type systems. For instance, in ADL it is possible to define alternatives for the value of an attribute. Let us consider the following archetype that models a very simple weight observation which can be measured either in kilograms (type QUANTITY[at0005]) or grams (type QUANTITY[at0006]):

```
ELEMENT[at0004] matches { -- weight measurement
        value existence matches {1..1} matches {
                QUANTITY[at0005] matches {--weight in kilos
                        magnitude matches {>0}
                        units matches {"kg"} -- kilograms
                }
                QUANTITY[at0006] matches { --weight in grams
                        magnitude matches {>0}
                        units matches {"grams"} -- grams
                }
        }
}
```

Here the attribute *value* must have one instance of class QUANTITY which can match either the constraints defined by type QUANTITY[at0005] or type QUANTITY[at0006]. The type ELEMENT[at0004] is abstracted by the following schema:

$$t_{ELEMENT[at0004]} := is\_ELEMENT\left\{t_{value\_04}^{(1:1)}\right\}$$
$$t_{value\_04} := is\_value\left\{\left(t_{QUANTITY[at0005]}^{(0:1)} \cdot t_{QUANTITY[at0006]}^{(0:1)}\right)^{|1:1|}\right\}$$
$$t_{QUANTITY[at0005]} := is\_QUANTITY\left\{t_{magnitude\_05}^{(1:1)} \cdot t_{units\_05}^{(1:1)}\right\}$$
$$t_{magnitude\_05} := is\_magnitude\left\{t_{>0}^{(1:1)}\right\}$$
$$t_{units\_05} := is\_units\left\{t_{kg}^{(1:1)}\right\}$$
$$t_{>0} := c\_real\left([0..\infty[, X\right)\{\varepsilon\}$$
$$t_{kg} := is\_kg\{\varepsilon\}$$
$$t_{QUANTITY[at0006]} := is\_QUANTITY\left\{t_{magnitude\_06}^{(1:1)} \cdot t_{units\_06}^{(1:1)}\right\}$$
$$t_{magnitude\_06} := is\_magnitude\left\{t_{>0}^{(1:1)}\right\}$$
$$t_{units\_06} := is\_units\left\{t_{g}^{(1:1)}\right\}$$
$$t_{g} := is\_g\{\varepsilon\}$$

As can be observed in the schema, the CML of the type (t_{value_04}) that models the *value* attribute states that the attribute must have a single value (l_0=u_0=1). Furthermore, the value can either be of type $t_{QUANTITY[at0005]}$ (weight in kilos) or type $t_{QUANTITY[at0006]}$ (weight in grams).

3.2. Archetype specialization

New archetypes can be defined by further constraining other archetypes, i.e. by specialization, in order to obtain a more adequate or fine grained representation of a clinical concept. An archetype is specialized by providing narrower constraints on data. The overall idea is that all data instances that conform to the more specialized archetype also conform to the more general. We abstract the inheritance relationship between archetypes by means of the subsumption relation [19]. Intuitively, an archetype A subsumes an archetype B if A is more general than or identical to B. Since reference models can also be abstracted by the proposed type system, the subsumption relation can also be used to formalize the relationship between business concepts and archetypes. We say that an archetype B specializes a business concept A if A subsumes B. What makes the proposed subsumption relation interesting is that it not only captures the containment relationship between two archetypes, but also captures the structural relationships between node objects from both archetypes.

Given two schemas S and S’, we say that S’ subsumes S under the subsumption relation θ, iff θ is a function from T_S to $T_{S'}$ such that:

1. $\theta(R_S) = R_{S'}$
2. $\forall t_i \in T_S, label_S(t_i) \subseteq label_{S'}(\theta(t_i))$
3. $\forall t_i \in T_S$, if t_i is ordered then $L(h_\theta(CML_S(t_i))) \subseteq lang_{S'}(\theta(t_i))$
4. $\forall t_i \in T_S$, if t_i is not ordered then $\theta(t_i)$ is also not ordered and $\bigcup_{w \in L(h_\theta(CML_S(t_i)))} permutation(w) \subseteq lang_{S'}(\theta(t_i))$

Where R_S and $R_{S'}$ are the root types of S and S’ respectively, $label_S(t)$ is the label predicate of t in S, $CML_S(t)$ is the CML of t in Schema S, $lang_S(t)$ the language defined by the CML of t in S and h_θ is the natural extension of θ to CMLs.

As can be seen from the above definition subsumption is based on defining mappings between types and on inclusion between both label predicates and CMLs. This can be translated to the archetype specialization mechanism. Subsumption mappings specify specialization relationships between the entities (objects and attributes) of the child and parent archetypes. This is compatible with the syntactical rules used in ADL to specify the specialization of objects. In ADL the specialization of node object, e.g. PERSON[at0001], is indicated by using the same identifier, followed by an extension, e.g. PERSON[at0001.1]. Note that this partly defines the subsumption function, e.g. θ(PERSON[at0001.1]) = PERSON[at0001]. Inclusion of label predicates assures that only class and attribute names from the reference model are used as well as the containment of the domains of primitive attribute. Finally, the containment of CML guarantees that specialized archetypes have a valid structure with respect to the structure of parent archetypes. From condition 3 it can be deduced that single-valued attributes can specialized multi-valued attributes and unordered multi-valued attributes can be specialized by making them ordered.

As an example, let us consider the following simple archetype which models a generic numeric measurement:

```
ELEMENT[at0001] matches { -- generic quantity element
        value existence matches {0..1} matches {
                QUANTITY[at0002] matches { --generic quantity data type
                        magnitude matches {*} --any numeric value
                        units matches {*} -- any unit
                }
        }
}
```

This set of constraints may be formalized by the following schema S' having $t_{ELEMENT[at0001]}$ as root type:

$$\begin{aligned}
&t_{ELEMENT[at0001]} := is\_ELEMENT\left\{t_{value}^{(1:1)}\right\}\\
&t_{value} := is\_value\left\{t_{QUANTITY[at0002]}^{(0:1)}\right\}\\
&t_{QUANTITY[at0002]} := is\_QUANTITY\left\{t_{magnitude}^{(1:1)} t_{units}^{(1:1)}\right\}\\
&t_{magnitude} := is\_magnitude\left\{Real^{(1:1)}\right\}\\
&t_{units} := is\_units\left\{String^{(0:1)}\right\}\\
&Real := real\left\{\varepsilon\right\}\\
&String := string\left\{\varepsilon\right\}
\end{aligned}$$

The previous schema subsumes the schema S modeling the previous type ELEMENT[at0004] in section 3.1, Table I enumerates the subsumption function. It can be checked, that this function satisfies all the conditions stated by definition 5. As an example, let us consider the type t_{value_04} which is subsumed by type t_{value}. Both types are unordered and the condition imposed on the label predicates are satisfied: $label\left(t_{value\_04}\right) = label\left(t_{value}\right) = \{value\}$, as well as the containment of the CMLs.

We have also developed an algorithm that takes two schemas as input and calculates the set of subsumption functions between them. It is a generalization of the algorithm described in [20] for the validation of XML documents against regular tree languages. Some interesting properties are exhibited by the subsumption function. Firstly, it is transitive. Secondly, if a data tree D is an instance of a schema A and A is subsumed by A' then D is also an instance of A'. Therefore, we can use the subsumption relation for modeling the specialization of archetypes and the definition of archetypes from business concepts. These features have been explored in LinkEHR-Ed in order to implement an advanced archetype editor supporting multiple reference models.

t	$\theta(t)$
$t_{ELEMENT[at0004]}$	$t_{ELEMENT[at0001]}$
t_{value_04}	t_{value}
$t_{QUANTITY[at0005]}$	$t_{QUANTITY[at0002]}$
$t_{magnitude\_05}$	$t_{magnitude}$
t_{units_05}	t_{units}
$t_{>0}$	$Real$
t_{kg}	$String$
$t_{QUANTITY[at0006]}$	$t_{QUANTITY[at0002]}$
$t_{magnitude\_06}$	$t_{magnitude}$
t_{units_06}	t_{units}
t_{g}	$String$

Table I. An example of subsumption function

4. Results

4.1. LinkEHR-Ed

LinkEHR-Ed (http://www.linkehr.com) is a visual tool implemented in Java under the Eclipse platform which allows the editing of archetypes based on different reference models, the specification of mappings between archetypes and data sources and the automatic generation of XQuery data conversion scripts which translate unnormalized XML data into XML documents which conform to the reference model and at the same time satisfy the data constraints imposed by archetypes. In other words it is a tool for the description and standardization of legacy data.

LinkEHR-Ed uses archetypes as a means to achieve standardization and semantic integration of distributed health data. The main objectives are twofold. Firstly, in the context of data integration [21][22][23], we use archetypes as a semantic layer over the data repositories, whose contents need to be integrated and exchanged, associating them with formal semantics. Secondly, we employ archetypes for making public existing clinical information in the form of standardized EHR extracts. For this purpose, we take advantage of their data definition capabilities that were formalized in the previous section. Thus, it becomes necessary to transform the clinical data held by the un-normalized local sources to meet the data structures defined by archetypes. This problem is known in the literature as the data exchange (translation or transformation) problem [24]. Data exchange at the schema level requires an explicit representation of how the source and target schemas are related to each other; these explicit representations are called mappings [25][26][27]. We use the term integration archetype to denote an archetype for which a mapping specification to a set of data sources has been defined, i.e.: Integration archetype = archetype + mapping specification. An integration archetype can be considered as a view that provides abstraction in interfacing between the data sources that hold the data to be shared and the reference model used to communicate these data in the form of standardized EHR extracts. LinkEHR-Ed is a visual tool for defining integration archetypes.

The heart of LinkEHR-Ed is composed of four modules: (i) a module to manipulate reference models and archetypes, (ii) a module for semantic validation and subsumption checking, (iii) a module to define mappings between archetypes and data source schemas and (iv) a module to generate XQuery [28] transformation scripts. Figure 2 shows an overview of the integration archetype editing process which is divided into four main steps. The first step deals with the importation of reference models. In LinkEHR-Ed a new reference model expressed as a W3C XML Schema [29] can be imported at any time. Obviously, this step only needs to be performed once for each reference model. The second step is the actual archetype editing process. Step 3 is about mapping specification. In step 4, an XQuery expression that encodes the mapping specification is automatically created. The execution of the XQuery expression over a set of data sources yields a XML instance that satisfies the constraints imposed by the archetype and at the same time is compliant with the underlying reference model. Therefore, LinkEHR-Ed helps data engineers write possibly long and complex programs to standardize data. Obviously, all this work should be complemented with methods for the query and visualization of health data [30].

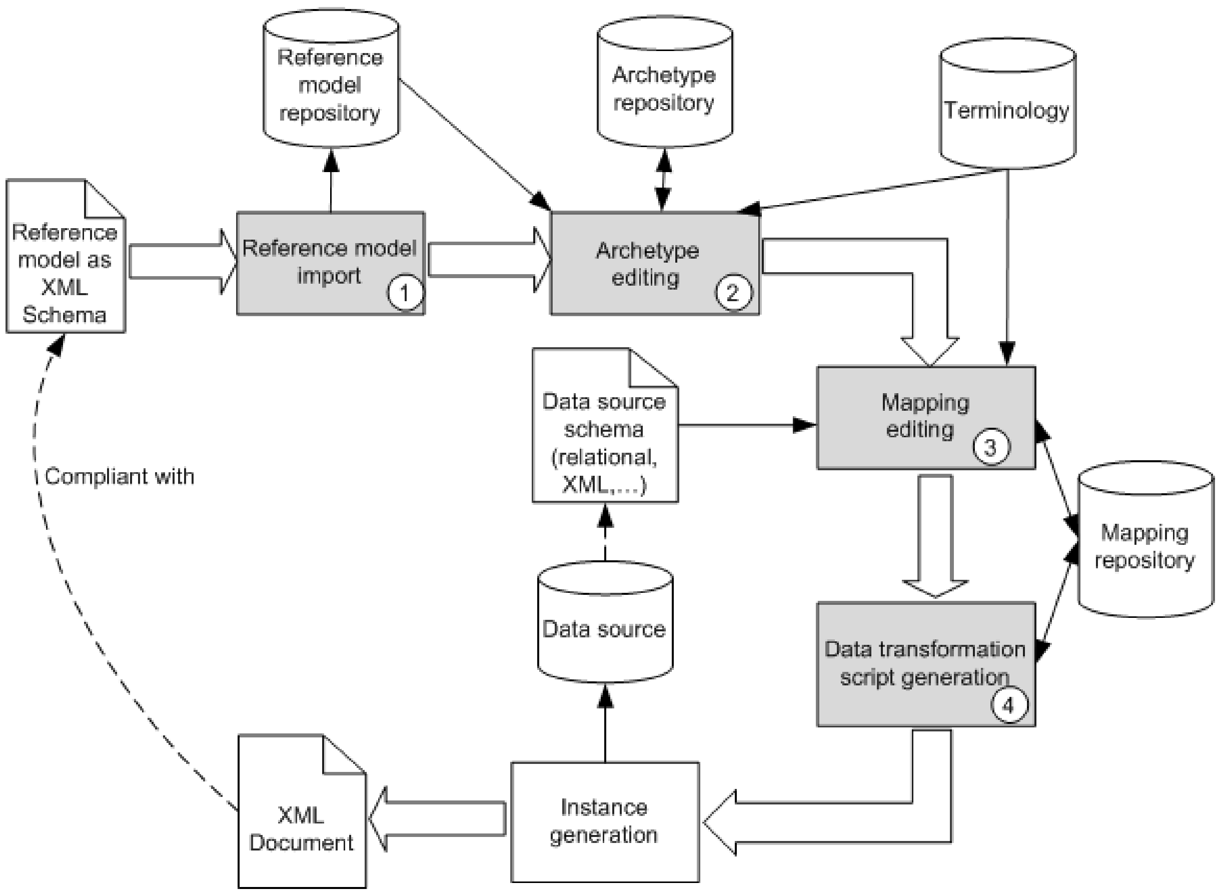

Figure 2. Editing and mapping of archetypes in LinkEHR-Ed. Numbered circles denote major application steps.

In summary, LinkEHR-Ed combines the formal representation of knowledge of a health domain expert in the form of archetypes (steps 1 and 2), with mapping information to clinical data sources for the standardization and semantic description of existing information (steps 3 and 4). Since this paper is focused on archetype editing only the two first steps will be considered in the rest of this section, for a more detailed introduction to archetype mappings we refer the reader to [31].

4.2. Management of reference models

The type system described above allows the formalization of both reference models and archetypes. Therefore, it is possible to apply the same logic both to the specialization of an existing archetype and to the definition of a new one by constraining a business concept. This brings about the possibility of building flexible archetype editors capable of working with several reference models or different versions of the same one. What it is required is the ability to import a new reference model at run time and express its business concepts as archetypes.

In LinkEHR-Ed a new reference model described by a W3C XML Schema can be imported at any time. This feature has been very useful in keeping pace with their evolution without modifying a single line of code. The import process generates:

- An archetype expressed in ADL for each business concept, we call them business archetypes.
- An XML file that enumerates the entities of the reference model providing a place for such entities 1) to be annotated with human-readable information that will be displayed during archetype editing in the form of tool tips 2) to specify complex vocabulary binding for attributes.

A business archetype is an explicit and exhaustive definition of a business class in ADL. Therefore, it contains all the attributes and classes of the reference model that may be used to define the business class. For instance, the business class *Element* of ISO/CEN EN13606 represents the leaf nodes within the EHR hierarchy. Each instance of this class has a single data value (attribute *value*), which is one of a defined set of data types. In the corresponding business archetype all the possible data types are explicitly defined as an alternative for the *value* attribute.

Business archetypes are used for managing the definition and specialization of archetypes and are transparent to users. They represent the most general archetypes that can be defined from a reference model and hence any other archetype must be a specialization of one of them. As a consequence the validation of an archetype with respect to a reference model becomes a problem of finding a subsumption function to a business archetype. Those for which it is not possible to find such functions are considered invalid with respect to the reference model.

We have to face two main problems when importing a new reference model. Firstly, to determine the archetypable classes and secondly to deduce their structure. The selection of archetypable classes cannot be fully automated since it depends on the reference model semantics. LinkEHR-Ed helps users by providing a ranking of classes ordered according their complexity. The complexity is measured by counting the number of primitive attributes that the reference model class includes directly or indirectly (e.g. atomic attributes of aggregated classes). The rationale behind this is that the more complex a class is, the higher the probability for it to be an archetypable class, since archetypable classes utilize non-archetypable ones but not the other way around. The complexity score of some conceptually equivalent business objects (Document, Section and Entry) from ISO/CEN EN13606, openEHR and CDA are shown in Table II. For its calculation the complexity of EN13606 Entry class has been taken as one. As can be observed CDA concepts are by far the most complex whereas EN13606 ones are the simplest. For example, the *entry* concept of CDA is almost eight times more complex that the corresponding concept in EN13606.

reference model	Document	Section	Entry			
EN13606	1.82 (Composition)	1.21	1.00			
openEHR	5.49 (Composition)	4.71	4.54 (Observation)	1.14 (Instruction)	1.22 (Evaluation)	1.76 (Action)
CDA	15.75 (Clinical document)	9.43	7.97			

Table II. Complexity score of some comparable business objects from EN13606, openEHR and CDA where the complexity of EN13606 Entry class has been taken as one. The complexity is measured as the number of primitive attributes that the class references directly or indirectly.

Once the set of archetypable classes is known, the import module mines their structure. Currently, many of the characteristics of W3C XML schemas are supported, such as name spaces, imports and includes (reference models can be defined by using several files) and a wide range of structures such as complex and simple types, elements, attributes, inheritance by extension and restriction, sequence, choice, all, union, attributes, patterns, groups and facets. One important issue to be considered when importing a reference model is the vocabulary binding at the attribute level, i.e. the set of values that an attribute is allowed to hold. Facets (fixed values, patterns, enumerations, etc.) are the main mechanism that XML Schema provides for such bindings. They are used to restrict the set of values a datatype can contain. Their translation into ADL is straightforward except for patterns (a regular expression that a type must match) due to some minor differences between both syntaxes. XML Schema does not support complex enumerations that may be required by the reference model such as ISO/CEN EN13606 which in part III defines a set of complex term list (code + value + description) for several attributes. As stated before the import module generates an XML file that enumerates the entities of the reference model providing a place for specifying such bindings. Once specified they are ready to be used along with business archetypes to guide the editing.

Three reference models have been tested successfully: EN13606, HL7 CDA and openEHR. The XML schema of EN13606 was developed by us from the UML model due to the lack of an official one. In the case of CDA and openEHR the official schemas were used.

4.3. Archetype editing

Although LinkEHR-Ed is oriented to the construction of integration archetypes it may operate as a pure archetype editor. For its development we have used the Java implementation of the AOM and the ADL parser maintained by the openEHR Foundation [32], although several additions have been made in order to satisfy extra requirements such as multi-reference model support and mapping to data sources. In the rest of this section we will mainly focus on the archetype editing process and how the formalization proposed in section 3 has been put into practice, although some important issues about the mapping of archetypes will also be discussed.

As stated before, in LinkEHR-Ed business concepts defined in a reference model are modeled as archetypes capable of being specialized. As a consequence the same logic can be applied both to the specialization of an existing archetype and to the definition of a new archetype by constraining a business archetype. Therefore, in LinkEHR-Ed for the sake of archetype editing, only the logic that guides archetypes specialization which is based on the subsumption relation described above is hard coded in the editor. In other words, LinkEHR-Ed assures that new archetypes are subsumed by the parent archetype (if it exists) or by a business archetype otherwise.

Only those entities (classes and attributes) of the reference model which are actually constrained need to appear in ADL. It is supposed that the constraints imposed on business archetypes are implicit constraints for all of its sub-archetypes. This supposition is consistent with the object-oriented paradigm, where attributes and methods of a superclass are automatically inherited by all its subclasses. The main advantage of this rule is that constraint definitions in ADL are kept simple. For instance, if all the classes and attributes of the reference model were to be included, archetypes would have many constraints (hundreds in EN13606) making the definition unnecessarily complex. This is slightly different for archetype specialization. In a specialized archetype, all the constraints of the parent archetype are included in the child archetype.

These syntactical rules bring about two difficulties in our setting. The first one appears when an archetype is specialized, since it may be necessary to constrain new attributes from the reference model. The second one is related to mapping. When an archetype is mapped to a data source, it may be necessary to define a mapping for an unconstrained attribute and therefore it is not present in the archetype definition. Thus, for the sake of mapping and specialization we are forced to temporally complete the archetype definition with entities (attributes and classes) from underlying business archetype. In LinkEHR-Ed we have defined a merge function that takes an archetype and a business archetype as input and outputs what we call a comprehensive archetype, which contains all the entities defined in the latter but at the same time preserves all the constraints imposed by the former. Put in other words, comprehensive archetypes include all the explicit constraints (those defined by the archetype to be specialized or mapped) and all the implicit ones (those defined by the business archetype) that data instances must satisfy. Figure 3 shows the overall editing schema of LinkEHR-Ed. As can be seen the archetype being edited is subsumed by the comprehensive archetype in order to assure its structural and semantic validity. It should be noted that in the case of the editing of an archetype from scratch, the business and comprehensive archetypes are the same. This guarantees that the new archetype is subsumed by the business archetype.

Comprehensive archetypes are used by a semantic manager to guide the editing process in order to guarantee that the archetype being edited satisfies all the inherited constraints. It calculates the set of entities (either attributes or types) that are allowed at any point of editing. When the user wishes to add a new entity the editor displays the valid entities and the user must select one of them. For instance, the set of valid attributes for the CDA *ClinicalDocument* class is shown in Figure 4. The semantic manager also checks that the constraints on data (cardinality, existence, domain, etc.) are narrower that those specified in the comprehensive archetype. It involves, for instance, testing the containment of constrained multiplicity lists or the containment of regular expression in the case of constraining a string attribute.

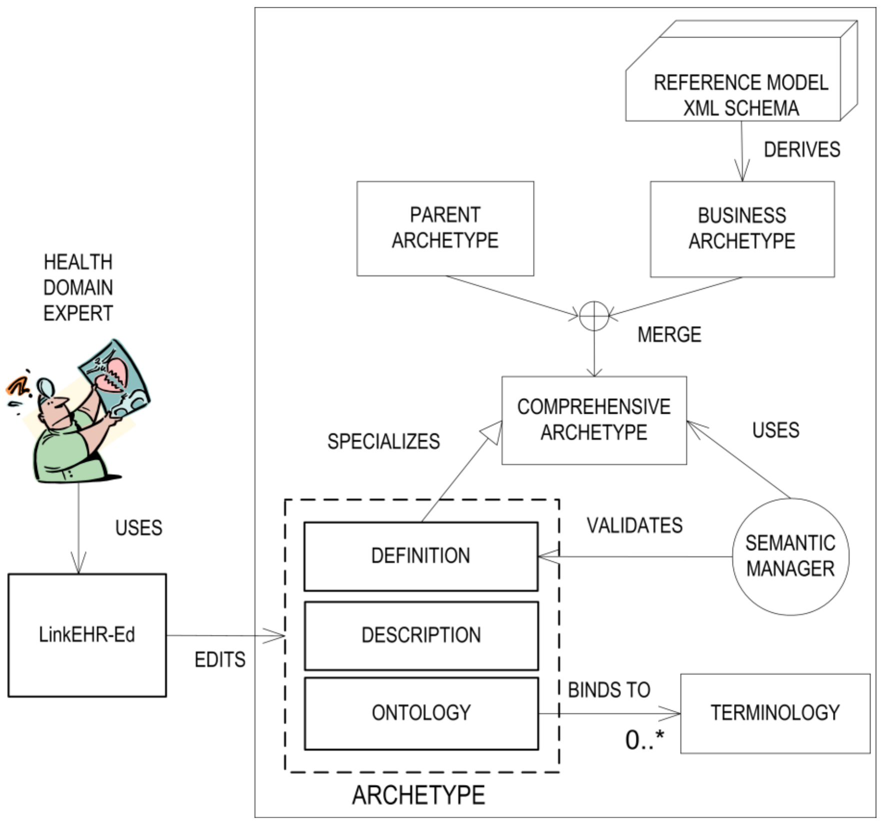

Figure 3. Schema of the archetype editing process of LinkEHR-Ed.

5. Discussion and conclusion

Possibly the most important requirement to support co-operative work among health professionals and institutions is the ability of sharing EHRs in such a way that both the sender and receiver system have a common meaning of the content exchanged, i.e. interoperability at the semantic level. It is widely acknowledged that standardization of data and concepts is a prerequisite to achieve semantic interoperability in any domain. This is even more important in the healthcare sector where the need to exchange health data among professional and institutions is not an exception but the rule. Some of the new generation of international standards for the communication of EHR have adopted a dual model approach for modeling the content of electronic health records. They use archetypes as a means to achieve semantic interoperability between EHR systems. Archetypes facilitate the definition of a semantic layer for common understanding and mutual communication of clinical data structured as a set of formal clinical concept definitions decided by health domain experts, promoting at the same time semantic interoperability among health information systems. But archetypes are also a valid approach for upgrading already deployed systems in order to make them compatible with an EHR standard, considering archetypes as clinical data integration components. This is in line with the utilization of semantic dictionaries or ontologies for semantic heterogeneity resolution and description of the data to be exchanged [33]. As a main advantage, this approach reduces the problem of knowing the contents and structure of many information sources to the problem of knowing the contents domain-specific concepts, which a user familiar with the domain is likely to know or understand easily [34].

Provided that archetypes are regarded as a basic element towards the consecution of semantic interoperability among EHR systems, it seems sensible to compare archetypes and ontologies as representation technologies to discuss whether they can be considered functionally equivalent for such purpose. Both archetypes and ontologies are formal representation technologies. For both technologies, the notions of specialization and evolution are basic, since they define hierarchical structures that may evolve over time. Models are built by means of constraints in both models, although the constraints have a different nature, since the archetype and ontology semantic models are different. Archetypes have a more reduced model, they basically include taxonomic and mereological relations, whereas any type of relation can be defined in ontologies. Archetype constraints are mainly based on constraining data values, whereas the ontological constraint model is richer, since different types of axioms can be defined. For instance, aspects such as disjointedness or equivalency can be modeled in ontological languages such as OWL. To some extent, it might be said that archetype semantics is contained into ontology semantics, for a more detailed discussion on archetypes and ontologies we refer the reader to [35]. Archetypes attempt to harmonize, unify and guide clinical practice by containing consensus knowledge. Ontologies ideally contain all the existing consensus knowledge of a particular domain, being this knowledge recognized and accepted by the community. Therefore, it can be said that both technologies play a similar role. It has proven complicated to agree on standard ontologies since different experts have different points of view on a particular reality, and it is likely to be also a problem for archetypes [36].

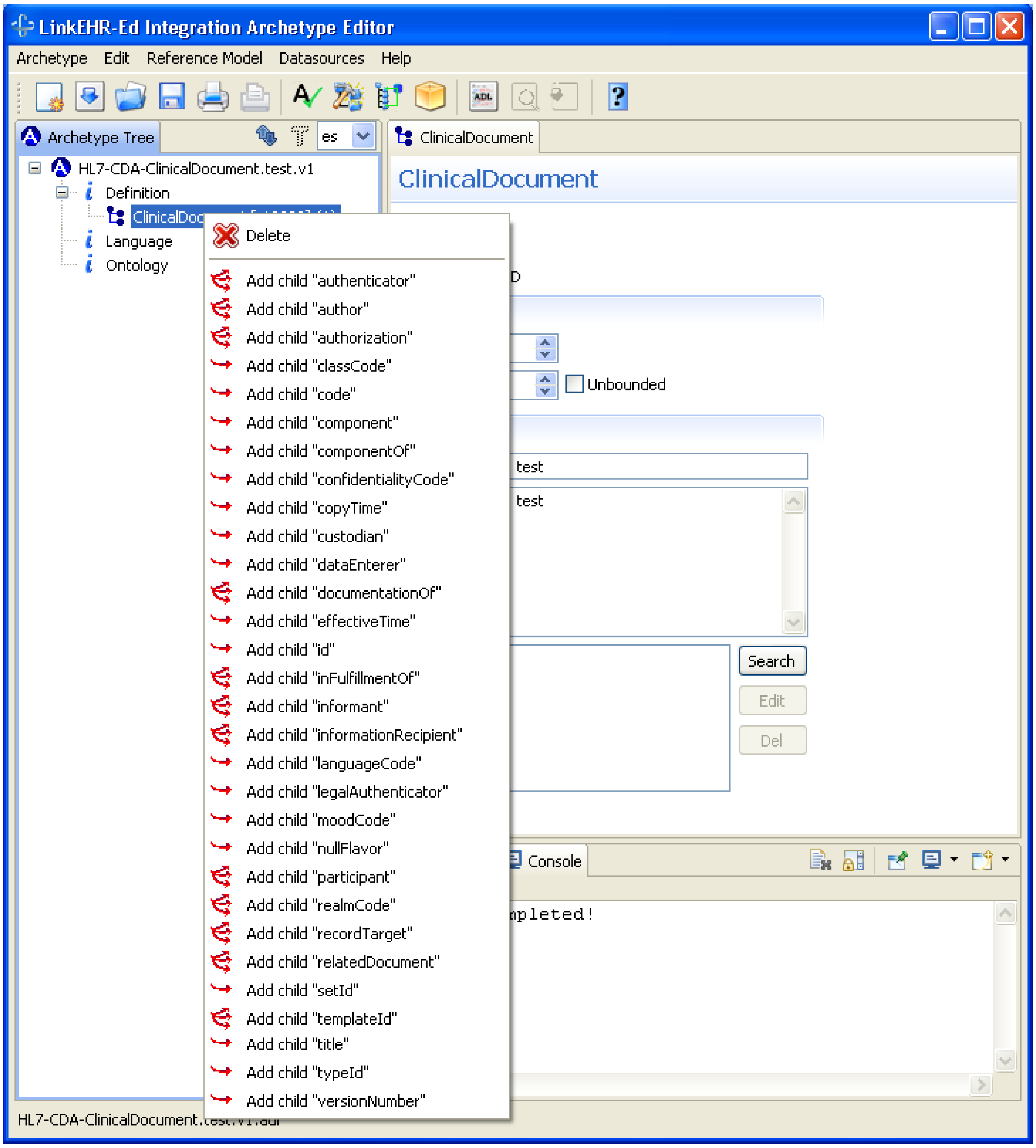

Figure 4. Screenshot of LinkEHR-Ed showing the set of available entities during archetype editing. In this case the set of attributes of class *ClinicalDocument* of CDA.

The work reported in this paper has been developed in the context of a research project aiming at applying semantic web technologies for managing EHRs. Therefore, it is completed with a complementary “top-down” approach that uses Semantic Web technologies to specify clinical archetypes for advanced EHR architectures and systems (see for instance [35]). Both works provide interfaces to different worlds: public external information (OWL archetypes) and internal information (already existing EHR sources). The semantic publication of the contents of the archetypes would be in line with the objectives of the development of the Semantic Web, which targets accessible web contents for both humans and computers so that applications might interoperate semantically in an efficient way.

Since our main concern is the generation of standardized XML documents compliant with a reference model we have chosen a more data-centric approach to model archetypes. The proposed abstraction for the data definition section of archetypes is based on labeled trees which have been extensively used for modeling XML schemas and queries [20][28][37][38]. Archetype specialization is modeled by a subsumption relation. As shown in section 3.4 the proposed subsumption relation is powerful enough to capture the different ways that an archetype may be specialized. Furthermore, it is compatible with the syntactical rules used for node identification in ADL.

In LinkEHR-Ed the archetype editing becomes a process of subtyping by constraints. The rules used to control the subtyping are those specified in the archetype model such as strengthening of domain constraints on primitive attributes or the narrowing of cardinality intervals. These rules are directly implemented in LinkEHR-Ed and are used by a semantic manager to assist the user in the editing process. The editor is then independent of the reference model which in turn is modeled as a set of archetypes (one for each business concept) which can be subtyped to defined proper archetypes. The application of these rules assures that the subtyping by constraints hierarchy is ordered by set inclusion, i.e. the set of instances of an archetype is a subset of the set of instances of all its super-archetypes. This latter is crucial: since any archetype is a subtype of a business class it is guaranteed that the set of instance defined by a sound archetype is a subset of the instances defined by the business archetype. As a consequence, sound archetypes define subsets of reference model instances.

As explained in section 4.1 W3C XML Schema is the language chosen to describe reference models. LinkEHR-Ed analyzes the schema and generates a set of business archetypes expressed in ADL which will be the basis for defining archetypes. Specifically, ADL and XML Schemas have different expressive power for defining tree-structured data. There are data structures that can be expressed with one but not with the other. ADL supports complex enumerations and does not force the unique particle attribution constraint but XML Schema allows more complex content models (roughly speaking the set of children that a node can have). For instance, in XML Schema it is possible to state that a node must have three children and all three must be labeled with "a" or all three must be labeled with "b". This content model cannot be expressed in ADL, and therefore by a CML. The closest representation would be:

```
node cardinality matches {3..3} matches {
        a occurrences matches {0..3} matches {…}
        b occurrences matches {0..3} matches {…}
}
```

But the former ADL extract does not prevent from having both 'a' and 'b' children. Therefore, there may be XML Schemas that cannot be expressed in ADL. When one of such Schemas is found the editor stops the importation process and warns the user. However, reference models are supposed to be OO models which require simple content models.

Currently, there exist two other archetype editors. The first one is the official editor of the openEHR consortium which is implemented in .NET and is maintained by Ocean Informatics (http://www.oceaninformatics.biz/). The second one is the Java-based LiU editor developed by a team from the Linköping University (http://www.imt.liu.se/mi/ehr/tools/). At the time of writing this paper both of them only support the openEHR reference model. These editors can be regarded as "concept-centric" since they are designed to be used by health domain specialists. They hide many details of the reference model making it possible even for users with moderate knowledge of the underlying reference model to construct archetypes. The main drawback of this approach is that the reference model must be hard coded into the editor making it difficult both to keep in pace with its evolution and to support other models.

The fundamental difference from the above efforts is that with LinkEHR-Ed we aim, apart from the editing of archetypes, to provide a tool for the standardization of legacy data by mapping archetypes to data sources. In this setting, archetypes are considered schemas defining a view that combines several un-normalized data structures into a single one that is compliant with the underlying reference model. This feature requires bringing to the front the reference model making the editor "structure-centric" where domain concepts are defined by directly constraining the data structures present in the reference model according to the archetype formalism. Obviously, this approach forces users to have a deeper knowledge of the reference model, but facilitates working with multiple models. In order to make the editor more user-friendly we intend to evolve towards a plug-in based editor. The idea is to consider reference models as plug-ins that will be created using LinkEHR-Ed itself. For instance, plug-ins may contain apart from documentation files, customized visual interfaces or term lists. Plug-ins may then be distributed and used by raw installations in order to obtain a fully operative environment for the reference model.

Authors' contributions

The work presented here was carried out in collaboration between all authors. MR, JAM and JTF defined the research theme. JAM, DM, DB, JTF and MR developed the formalization for archetypes and wrote the paper. All authors contributed to the design and implementation of LinkEHR-Ed. All authors have contributed to, seen and approved the manuscript.

Acknowledgements

This work was supported in part by the Spanish Ministry of Education and Science under grant TSI2007-66575-C02; by the Generalitat Valenciana under grant APOSTD/2007/055 and by the program PAID-06-07 de la Universidad Politécnica de Valencia.

Summary table

The following facts were known before this study: • The dual model approach is postulated as a possible solution to tame the complexity related to EHRs. • Archetypes are suitable for modeling clinical concepts formally and therefore can be used to describe and query legacy data. • The current archetype specification is not precise enough regarding the semantics of specialization and instantiation which hinders gaining a comprehensive understating of the dual model methodology.
This study has added the following to our knowledge: • A formal foundation for archetypes is crucial to develop enhanced archetype processing tools. • A formalization of the semantics of archetypes based on types over trees with labeled nodes and a subsumption function that captures the relationship between archetypes. • The development of an archetype editor capable of handling multiple reference models and with advanced archetype processing capabilities.

Conflict of interest

The authors have no conflict of interest to declare.

References

[1] Blobel, B. Advanced and secure architectural EHR approaches. International Journal of Medical Informatics 2007; 76(5-6):491-496.

[2] Eichelberg M, Aden T, Riesmeier J, Dogac A, Laleci GB. A survey and analysis of electronic healthcare record standards. ACM Computing Surveys 2005; 37(4):277-315.

[3] Kalra D. Electronic health record standards. In Haux R, Kulikowski C, editors. IMIA Yearbook of Medical Informatics 2006. Methods of Information in Medicine 2006; 45 Suppl 1:136-144.

[4] Health Level 7 (last accessed 2009/03/02); available from: http://www.hl7.org.

[5] HL7 Clinical Document Architecture, Release 2.0 (last accessed 2009/03/02); available from: http://www.hl7.org/v3ballot/html/infrastructure/cda/cda.htm.

[6] CEN/TC251. EN13606-Medical Informatics-Electronic health record communication.

[7] CEN/TC251. EN13606-1- Medical Informatics-Electronic health record communication-Part 1 Reference Model.

[8] openEHR consortium (last accessed 2009/02/26); available from: http://www.openehr.org.

[9] International Organization for Standardization. Requirements for an electronic health record architecture. Technical Specification TS 18308.

[10] Rector A. Clinical Terminology: Why is it so hard?. Methods of Information in Medicine 1999; 38(4):239:252.

[11] Beale T. Archetypes: Constraint-based domain models for future-proof information systems. Proceedings of the Eleventh OOPSLA Workshop on Behavioral Semantics 2002; pp. 16-32.

[12] Beale T, Heard S (Ed). Archetype Definition Language ADL 1.4 (last accessed 2009/02/26); available from: http://svn.openehr.org/specification/TRUNK/publishing/architecture/am/adl.pdf.

[13] CEN/TC251. EN13606- 2-Health Informatics-Electronic record communication-Part 2 Archetypes.

[14] Dogac A, Laleci GB, Kabak Y, Unal S, Beale T, Heard S, Elkin PL, Najmi, F, Mattocks C, Weber D, Kernberg M. Exploiting ebXML registry semantic constructs for handling archetype metadata in healthcare informatics. International Journal of Metadata, Semantics and Ontologies 2006; 1(1):21–36.

[15] Qamar R, Rector A. MoST: A system to semantically map clinical model data to SNOMED-CT. Semantic Mining Conference on SNOMED-CT 2006; 38-43.

[16] Clinical Knowledge Manager (last accessed 2009/03/02); available from: http://www.openehr.org/knowledge.

[17] Maldonado JA, Moner D, Tomás D, Angulo C, Robles M, Fernández JT. Framework for clinical data standardization based on archetypes. Proceedings of Medinfo 2007; 454-458.

[18] Beeri C, Milo T. Schemas for integration and translation of structured and semi-structured data. Proceedings of the 7th International Conference on Database Theory 1999; 296-313.

[19] Kuper GM, Simeon J. Subsumption for XML types. Proceedings of the 8th International Conference on Database Theory 2001; 331-45.

[20] Murata M, Lee D, Mani M, Kawaguchi, K. Taxonomy of XML Schema Languages Using Formal Language Theory. ACM Transactions on Internet Technology 2005; 5(4):660-704.

[21] Karasavvas KA, Baldock R, Burger A. Bioinformatics integration and agent technology. Journal of Biomedical Informatics 2004; 36(3):205-219.

[22] Sujansky W. Heterogeneous database integration in biomedicine. Journal of Biomedical Informatics 2001; 34(4):285-298.

[23] Brazhnik O, Jones JF. Anatomy of data integration. Journal of Biomedical Informatics 2007; 40(3):252-269.

[24] Popa L, Velegrakis Y, Miller RJ, Hernández MA, Fagin, R. Translating Web Data. Proceedings of the 28th VLDB Conference 2002; 598-609.

[25] Kolaitis, PG. Schema mappings, data exchange, and metadata management. Proceedings of the 24th ACM SIGACT-SIGMOND-SIGART Symposium on Principles of Database Systems 2005; 61-75.

[26] Bernstein PA. Applying model management to classical metadata problems. Proceedings of the First Biennial Conference on Innovative Data Systems Research 2003; 209-220.

[27] Sun Y. Methods for automated concept mapping between medical databases. Journal of Biomedical Informatics 2004; 37(3):162-178.

[28] Boag S, Chamberlin D, Fernández M, Florescu D, Robie J, Siméon J (Eds), XQuery 1.0: An XML query language. W3C Candidate Recommendation 2005 (last accessed 2009/03/05); Available from http://www.w3.org/TR/2005/CR-xquery-20051103.

[29] World Wide Web Consortium. XML Schema, parts 0, 1, and 2, W3C Recommendation 2001.

[30] Angulo C, Crespo M, Maldonado JA, Moner D, Pérez D, Abad I, Mandingorra J, Robles M. Non-invasive lightweight integration engine for building EHR from autonomous distributed systems. International Journal of Medical Informatics 2007; 76(S3):417-424.

[31] Maldonado-Segura JA, Moner-Cano D, Boscá-Tomás D, Fernández Breis JT, Angulo-Fernández C, Robles- Viejo M. Semantic Upgrade and Normalization of Existing EHR Extracts.

Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 2008; 1466-1469.

[32] Chen R, Klein G. The openEHR Java Reference Implementation Project. Proceedings of the 12th World Congress on Health (Medical) Informatics, 2007; 58-62.

[33] Aslan G, McLeod D. Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution. The VLDB Journal 1999; 8(2):120-132.

[34] Ouksel AM, Sheth A. Semantic Interoperability in Global Information system: A brief introduction to the research area and the special section. Sigmod Record 1999; 28(1):5-12.

[35] Martínez-Costa C, Menárguez-Tortosa M, Fernández-Breis JT, Maldonado-Segura JA. A Model-driven Approach for Representing Clinical Archetypes for Semantic Web Environments. Journal of Biomedical Informatics 2009; 42(1): 150-164.

[36] Garde S, Knaup P, Hovenga EJS, Heard S. Towards semantic interoperability for electronic health records. Methods of information in medicine 2007; 46(2):332-343.

[37] Neven F. Automata Theory for XML Researchers. ACM SIGMOD Record 2002; 31(3): 39-46.

[38] Chidlovskii B. Using regular tree automata as XML schemas. Proceedings of the IEEE Advances in Digital Libraries 2000; 89-104.