Use of XML Schema Definition for the Development of Semantically Interoperable Healthcare Applications

Luciana Tricai Cavalini[1] and Timothy Wayne Cook[2]

[1] Department of Health Information Technology, Medical Sciences College,
Rio de Janeiro State University, Brazil
[2] National Institute of Science and Technology –
Medicine Assisted by Scientific Computing, Brazil
lutricav@lampada.uerj.br, tim@mlhim.org

Abstract. Multilevel modeling has been proven in software as a viable solution for semantic interoperability, without imposing any specific programming languages or persistence models. The Multilevel Healthcare Information Modeling (MLHIM) specifications have adopted the XML Schema Definition 1.1 as the basis for its reference implementation, since XML technologies are consistent across all platforms and operating systems, with tools available for all mainstream programming languages. In MLHIM, the healthcare knowledge representation is defined by the Domain Model, expressed as Concept Constraint Definitions (CCDs), which provide the semantic interpretation of the objects persisted according to the generic Reference Model classes. This paper reports the implementation of the MLHIM Reference Model in XML Schema Definition language version 1.1 as well as a set of examples of CCDs generated from the National Cancer Institute – Common Data Elements (NCI CDE) repository. The set of CCDs was the base for the simulation of semantically coherent data instances, according to independent XML validators, persisted on an eXistDB database. This paper shows the feasibility of adopting XML technologies for the achievement of semantic interoperability in real healthcare scenarios, by providing application developers with a significant amount of industry experience and a wide array of tools through XML technologies.

Keywords: semantic interoperability, electronic health records, multilevel modeling.

1 Introduction

The implementation of electronic health records has been proposed to increase the effectiveness of healthcare, but the expectations in this field are yet to be met. Since 1961, when the first computerized health record system was installed at the Akron General Hospital [1], and over the more than 50 years since that time, software companies of all types have sought the ability to integrate various systems in order to provide a coherent healthcare information platform [2] [3].

J. Gibbons and W. MacCaull (Eds.): FHIES 2013, LNCS 8315, pp. 125–145, 2014.

The challenges related to recording clinical information in computer applications are primarily associated to the fact that healthcare is a complex and dynamic environment. Regarding complexity, it is known, for instance, that the Systematized Nomenclature of Medicine – Clinical Terms (SNOMED-CT), the most comprehensive terminology for healthcare, has more than 311,000 terms, connected by more than 1,360,000 links [4]. The dynamism observed in healthcare information is essentially related to the speed of scientific evolution and technology incorporation, which is a main feature of the field [5] [6].

Furthermore, the healthcare system is by definition hierarchical and decentralized; thus, it is expected that the patients will access the system through primary care settings and then ascend to higher complexity levels of care [7]. For historical and economic reasons, primary care settings are located closer to the user's household, while more complex healthcare institutions (such as hospitals) are usually built in central areas [8]. The functions of primary care and hospitals are clearly different, which determines a high level of variability regarding their architectural format and structure and, in consequence, each healthcare institution will adopt specific workflows that are adapted to its form and function [9]. This process will reflect on the specificity of information collected, stored and processed inside a given facility [10].

However, no healthcare institution is isolated from the others. Because of the configuration of the healthcare system, patients circulate across more than one setting [11]. This is particularly true of patients with chronic conditions that see more than 80 different physicians in the course of their disease [12]. Thus, ideally, every patient's record should be kept longitudinal, since any piece of information might be important at any moment of the patient's life [13].

The achievement of such levels of interoperability between electronic health records still remains as a challenge [14] [15]. Currently, there is a multiplicity of companies and governmental institutions whose mission is to develop healthcare applications, each one of them implementing its own data model, which is specific for that application [16] [17]. Such data models are not only different from system to system, but they are also ever changing as the scope of the applications change, which includes the continuous changes in medical science, insurance company regulations and government policies [18] [19].

This constant change is a costly component of managing healthcare information [20] and creates a situation in which much of the semantic context of the healthcare data is embedded into the structure of the database, as well as in the programming language source code. Thus, when sharing data between healthcare applications is attempted, even in the simplest situation (when the data types are the same), the complete context in which the data was recorded remains unknown to the receiving system. This happens due to the fact that the semantics are locked up in the database structure and the source code of the application [21].

Many solutions have been proposed to the problem of interoperability in healthcare information systems, which include a vast and variable set of knowledge representation models, especially terminologies and ontologies [22] [23]. Nevertheless, the high implementation and maintenance costs of the available electronic health records have slowed down their widespread implementation; even some throwbacks have been observed over the last years [24] [25]. Until this date, the only development method that has achieved semantic interoperability is the multi (or dual)-level modeling approach originally proposed by the *open*EHR Foundation [26] and evolved by two projects based on the same principles: the ISO 13606 family of standards [27] and the Multilevel Healthcare Information Modeling (MLHIM) specifications [28].

Although the ability to achieve semantic interoperability between electronic health records has been already proven in multilevel modeling-based software [29], there are relatively few known implementations of the *open*EHR specifications or the ISO 13606 standards. This can be attributed to the complexity of the *open*EHR specifications [30] or to the fact that the ISO 13606 standard does not provide for data persistence, but only message exchange between systems [31].

Another significant barrier to the wider adoption of the multilevel modeling principles, as implemented in *open*EHR and ISO 13606, is the use of a domain-specific language, the Archetype Definition Language (ADL), for defining the data models. In both approaches, ADL was adopted for the definition of constraints to the information model (known as Reference Model) classes, for each healthcare concept [27]. Some authors have expressed their concerns about the technical barriers of using ADL for the widespread development of applications to run on real healthcare settings, when concepts will have a high level of complexity [32] [33].

Given the fact that semantic interoperability is such a key issue for the successful adoption of information technologies in healthcare, and multilevel modeling is a solution for it, there is a need for making such principles implementable in real life applications. This was achieved in the MLHIM specification by adopting XML technologies for its implementation, which are an industry standard for software development [34] and information exchange. This paper presents the development of a demo application based on version 2.4.2 of the MLHIM specifications.

2 Method

The methodological approach adopted in this study included: (a) the implementation of the basic components of the MLHIM specifications (the Reference Model and the Domain Models) in XML Schema 1.1; (b) the generation of simulated data based on a set of selected Domain Models for demographic and clinical concepts and (c) the demonstration of persistence and querying procedures implemented in two demo applications, using the simulated data produced.

2.1 Overview of the MLHIM Specifications

The MLHIM specifications are published (https://github.com/mlhim) as a suite of open source tools for the development of electronic health records and other types of healthcare applications, according to the principles of multilevel modeling. The specifications are structured in two Models: the Reference Model and the Domain Model.

The conceptual MLHIM Reference Model is composed of a set of classes (and their respective attributes) that allow the development of any type of healthcare application, from hospital-based electronic medical records to small purpose-specific applications that collect data on mobile devices. This was achieved by minimizing the number and the residual semantics of the Reference Model classes, when compared to the original *open*EHR specifications. The remaining classes and semantics were regarded as *necessary and sufficient* to allow any modality of structured data persistence. Therefore, the MLHIM Reference Model approach is minimalistic [34], but not as abstract as a programming language.

The reference implementation of the MLHIM Reference Model is expressed in a XML Schema 1.1 document. Each of the classes from the Reference Model are expressed as a complexType definition, arranged as 'xs:extension' [34]. For each complexType there is also an 'element' definition. These elements are arranged into Substitution Groups in order to assist with the concept of class inheritance defined in the conceptual Reference Model.

The MLHIM Domain Model is defined by the Concept Constraint Definitions (CCDs), expressed in XML Schema 1.1, being conceptually equivalent to the *open*EHR and ISO 13606 archetypes. Each CCD defines the combination and restriction of classes and class attributes of the (generic and stable) MLHIM Reference Model that are *necessary and sufficient* to properly represent a given healthcare concept. In general, CCDs are set to allow wide reuse, but there is no limitation for the number of CCDs allowed for a single concept in the MLHIM ecosystem. Each CCD is identified by a Type 4 Universal Unique Identifier (UUID) [28]. This provides permanence to the concept definition for all time, thus creating a stable foundation for instance data established in the temporal, spatial and ontological contexts of the point of recording. This is a very important concept, in order to preserve the original semantics at the time of data capture so that any future analytics will not be skewed into unknown directions. This is a common problem when data is migrated from one database format to another and source code in the application is modified [35]. Since this is where the semantics exist in typical applications, the data no longer represents those semantics after such a migration.

The key innovation in the MLHIM specifications is the use of complexType definitions in the CCD based on restrictions of the Reference Model types. Giving the fact that the majority of medical concepts are multivariate, for the majority of CCDs, a n ($n > 0$) number of complexTypes will be included. For instance, since it is likely to have a CCD with more than one complexType, each one of them will be also associated to a Type 4 UUID, which is similar to the complete CCD identification process described above [28]. This allows the existence of multiple complexTypes of the same nature (for instance, a CCD may have more than one ClusterType or more than one DvStringType) in the same CCD without a conflict of the restrictions. This approach also enables data query, since it creates a universally unique path statement to any specific MLHIM based data.

CCDs have the capability to accommodate any number of medical ontologies and terminologies [27]. All complexTypes may include links as computable application information ('xs:appinfo'), which can be used to include any amount of specific semantics by linking into any ontology or terminology. These are created as part of the CCD in an 'annotation' element and allow the inclusion of Resource Description Framework (RDF) content for further improvement of the concept's semantics, based on any relevant ontology.

The second key innovation is in the approach in handling missing data or data that is outside the expected range or type. This is not an uncommon occurrence in healthcare applications. All data types in MLHIM (descendants of DvAny) carry an 'ev' element for exceptional value semantics [36]. This approach is similar to what ISO 21090 calls Null Flavours. However, the approach in ISO 21090 is brittle and does not allow for expansion, creating the probability for missing, incomplete or incorrect missing data semantics. MLHIM solves this issue by providing a tree based on the 'ev-meaning' and 'ev-name' elements of the ExceptionalValue complexType, being the values for these elements fixed for each complexType.

For example, with the INVType; 'ev-name' is "Invalid" and 'ev-meaning' is "The value as represented in the instance is not a member of the set of permitted data values in the constrained value domain of a variable"; which are taken from ISO 21090. An example of an extension to ISO 21090 is the ASKRType, representing the prevalent (yet underreported) "Asked But Refused" value. Thus, in addition to the extensions for exceptional values in the Reference Model, any CCD can extend the ExceptionalValue complexType to create context specific missing or exceptional value data semantics with no loss of interoperability.

It is important to note that the MLHIM specifications are concerned with semantic interoperability of all biomedical applications. This means that many application development requirements that are specific to any particular type of application are not included. This includes very important concepts such as; how to persist CCDs in meaningful and useful ways, authentication and authorization, Application Programming Interfaces (APIs) and query processing. These are all outside the scope of the MLHIM specifications. These other requirements are well defined in other industry specifications and standards, and attempts to include them inside MLHIM would only serve to confuse the core issue of semantic interoperability.

2.2 Description of the MLHIM Reference Model

The implementation of the MLHIM Reference Model version 2.4.2 was produced as a single XML Schema Definition (XSD) file according to the XML W3C standards version 1.1 (source code available at https://github.com/mlhim/specs). The implementation approach in XML was based on extensions and substitutions, in order to maintain the hierarchical structure of the conceptual model.

The MLHIM Reference Model data types are defined as the Datatypes package and are originally based on ISO 21090 with modifications to reduce unnecessary complexity and semantic dependency. For any Element of a CCD, the ‘Element-dv’ attribute must be constrained to one of the concrete complexTypes of this package.

The ordered data types from the MLHIM specifications comprise any type of data whose instances can be ordered; such are all complexTypes under the abstract DvOrdered complexType. The DvOrdered children complexType allow the persistence of ordinal values such as ranks and scores (DvOrdinal), dates and times (DvTemporal) and true numbers (all complexTypes under DvQuantified) (Table 1).

Table 1. MLHIM Reference Model: Ordered complexTypes

Parent complexType	complexType	Usage
DvAny	DvInterval ReferenceRange	Intervals of DvQuantitifed data types Normal or abnormal intervals
DvOrdered[a]	DvOrdinal	Ranks or scores
DvQuantified[b]	DvQuantity DvCount DvRatio	Quantities in units Count data Ratios, rates and proportions
DvAny	DvTemporal	Complete or incomplete dates or times Durations

[a.] DvAny child complexType. [b.] DvOrdered child complexType.

The unordered data types from the MLHIM specifications comprise any type of string, Boolean or parsable data. Some of those complexTypes inherit directly from the abstract DvAny complexType and do not have any other inheritance relationship (DvBoolean and DvURI). On the other hand, the DvString and DvCodedString complexTypes defines a data type set that might contain characters (as well as DvIdentifier), line feeds, carriage returns, and tab characters, and the DvEncapsulated children complexTypes define the common metadata and allow persistence of all types of parsable or multimedia data (Table 2). A UML diagram of the Datatypes package is shown in Figure 1. For improved usability, a ZIP compressed package of all UML diagrams of the MLHIM Reference Model, in SVG format, is available at https://docs.google.com/file/d/0B9KiX8eH4fiKQVpHbmNmQ1pZS1U/edit?usp=sharing.

Table 2. MLHIM Reference Model: Unordered complexTypes

Parent complexType	complexType	Usage
DvAny	DvBoolean DvURI DvString	Truly boolean data (e.g. true/false) Uniform Resource Identifiers (URIs) Alphanumeric characters
DvString	DvCodedString DvIdentifier	Controlled vocabulary terms Identities of DemographicEntry
Dv Encapsulated[a]	DvMedia DvParsable	Multimedia types and their metadata Encapsulated parsable strings

[a.] DvAny child complexType. [b.] DvOrdered child complexType.

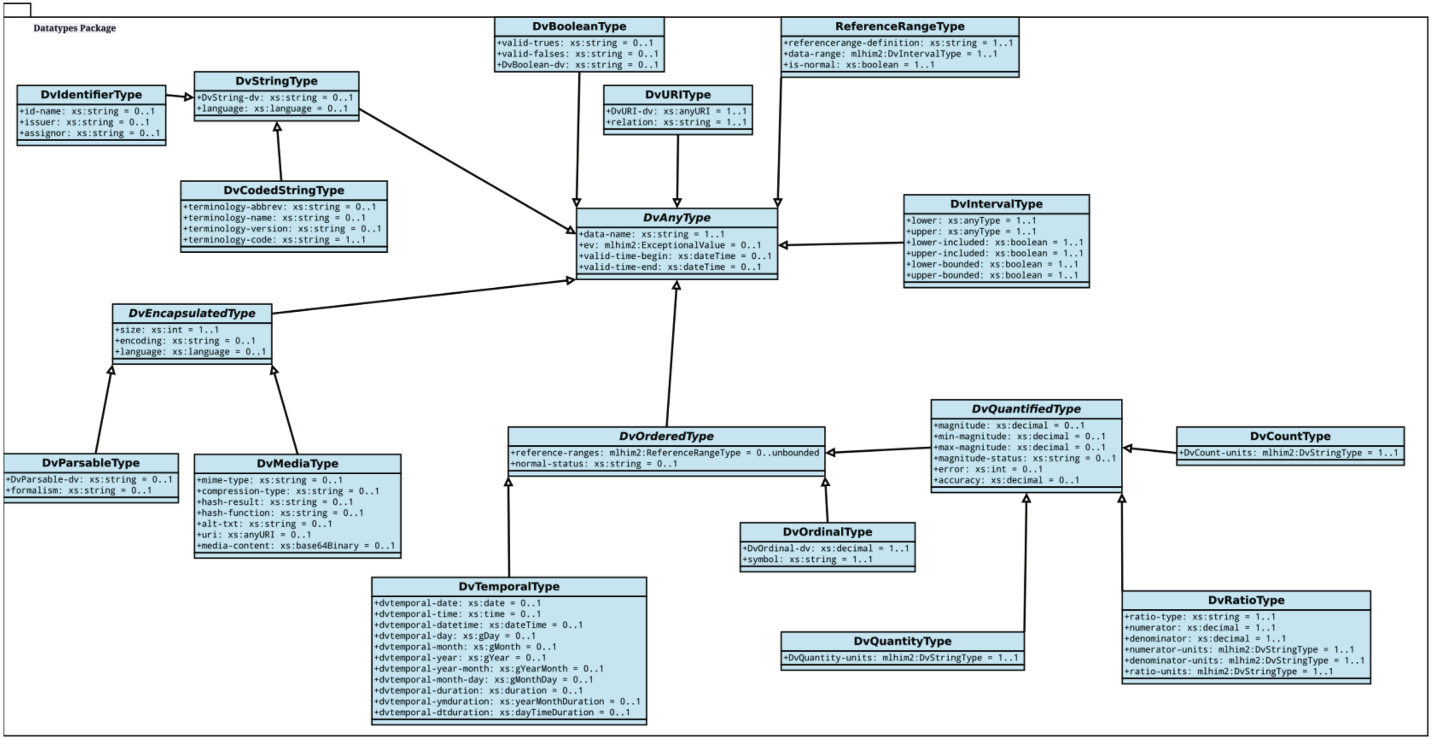

Datatypes Package
DvIdentifierType
+id-name: xs:string = 0..1
+issuer: xs:string = 0..1
+assignor: xs:string = 0..1
DvStringType
+DvString-dv: xs:string = 0..1
+language: xs:language = 0..1
DvCodedStringType
+terminology-abbrev: xs:string = 0..1
+terminology-name: xs:string = 0..1
+terminology-version: xs:string = 0..1
+terminology-code: xs:string = 1..1
DvBooleanType
+valid-trues: xs:string = 0..1
+valid-falses: xs:string = 0..1
+DvBoolean-dv: xs:string = 0..1
DvURIType
+DvURI-dv: xs:anyURI = 1..1
+relation: xs:string = 1..1
ReferenceRangeType
+referencerange-definition: xs:string = 1..1
+data-range: mlhim2:DvIntervalType = 1..1
+is-normal: xs:boolean = 1..1
DvAnyType
+data-name: xs:string = 1..1
+ev: mlhim2:ExceptionalValue = 0..1
+valid-time-begin: xs:dateTime = 0..1
+valid-time-end: xs:dateTime = 0..1
DvIntervalType
+lower: xs:anyType = 1..1
+upper: xs:anyType = 1..1
+lower-included: xs:boolean = 1..1
+upper-included: xs:boolean = 1..1
+lower-bounded: xs:boolean = 1..1
+upper-bounded: xs:boolean = 1..1
DvEncapsulatedType
+size: xs:int = 1..1
+encoding: xs:string = 0..1
+language: xs:language = 0..1
DvParsableType
+DvParsable-dv: xs:string = 0..1
+formalism: xs:string = 0..1
DvMediaType
+mime-type: xs:string = 0..1
+compression-type: xs:string = 0..1
+hash-result: xs:string = 0..1
+hash-function: xs:string = 0..1
+alt-txt: xs:string = 0..1
+uri: xs:anyURI = 0..1
+media-content: xs:base64Binary = 0..1
DvOrderedType
+reference-ranges: mlhim2:ReferenceRangeType = 0..unbounded
+normal-status: xs:string = 0..1
DvQuantifiedType
+magnitude: xs:decimal = 0..1
+min-magnitude: xs:decimal = 0..1
+max-magnitude: xs:decimal = 0..1
+magnitude-status: xs:string = 0..1
+error: xs:int = 0..1
+accuracy: xs:decimal = 0..1
DvCountType
+DvCount-units: mlhim2:DvStringType = 1..1
DvOrdinalType
+DvOrdinal-dv: xs:decimal = 1..1
+symbol: xs:string = 1..1
DvTemporalType
+dvtemporal-date: xs:date = 0..1
+dvtemporal-time: xs:time = 0..1
+dvtemporal-datetime: xs:dateTime = 0..1
+dvtemporal-day: xs:gDay = 0..1
+dvtemporal-month: xs:gMonth = 0..1
+dvtemporal-year: xs:gYear = 0..1
+dvtemporal-year-month: xs:gYearMonth = 0..1
+dvtemporal-month-day: xs:gMonthDay = 0..1
+dvtemporal-duration: xs:duration = 0..1
+dvtemporal-ymduration: xs:yearMonthDuration = 0..1
+dvtemporal-dtduration: xs:dayTimeDuration = 0..1
DvQuantityType
+DvQuantity-units: mlhim2:DvStringType = 1..1
DvRatioType
+ratio-type: xs:string = 1..1
+numerator: xs:decimal = 1..1
+denominator: xs:decimal = 1..1
+numerator-units: mlhim2:DvStringType = 1..1
+denominator-units: mlhim2:DvStringType = 1..1
+ratio-units: mlhim2:DvStringType = 1..1

Fig. 1. UML diagram of the MLHIM Reference Model – Datatypes package

The Reference Model Structures package contains the Item class and its children complexTypes; Element and Cluster. Clusters are structural containers of any Item child complexType (including other Clusters), which allows the definition of any size or shape of data model for a given healthcare concept. Elements are the finest granularity of the MLHIM Reference Model structure, where data types are assigned for each variable of a healthcare concept.

The complexTypes that compose the Structures package are used to model the data structure of the Entry children complexTypes, which are defined in the Reference Model Content package: CareEntry, AdminEntry and DemographicEntry.

An Entry is the root of a logical set of data items. It is also the minimal unit of information any query should return, since a whole Entry (including sub-parts) records spatial structure, timing information, audit trail definition and contextual information, as well as the subject and generator of the information, required for complete semantic interoperability.

Each Entry child complexType has identical attribute information. The subtyping is used to allow persistence to separate the types of Entries, which is primarily important in healthcare for the de-identification of clinical information.

The CareEntry complexType defines data structure, protocol and guideline attributes for all clinical entries. The AdminEntry complexType is used for recording administrative information that sets up the clinical process, but it is not clinically relevant itself, such as admission, episode, ward location, discharge and appointments. The DemographicEntry complexType is used to record demographic information, such as name structures, roles, and locations. It is modeled as a separate Entry child complexType in order to facilitate the separation of clinical and non-clinical information, and especially to support de-identification of clinical and administrative data.

Finally, the Constraint package is composed of the CCD complexType, which has one element named 'defintion', which must be constrained to any of the Entry child complexTypes (Table 3). A UML Diagram of the Content, Constraint and Structures packages are shown in Figure 2.

Table 3. MLHIM Reference Model: Content and Structures packages

Parent complex-Type (Package)	**complexType**	**Function**
Item (Structures)	Element Cluster	The leaf variant of Item class, to which a data type instance is attached The grouping variant of Item class, which may contain further instances of Item in an ordered list
Entry (Content)	CareEntry AdminEntry DemographicEntry	Container of healthcare data Container of administrative data Container of demographic data
CCD (Constraint)	CCD	Defining the further constraints on the Reference Model for a given healthcare concept

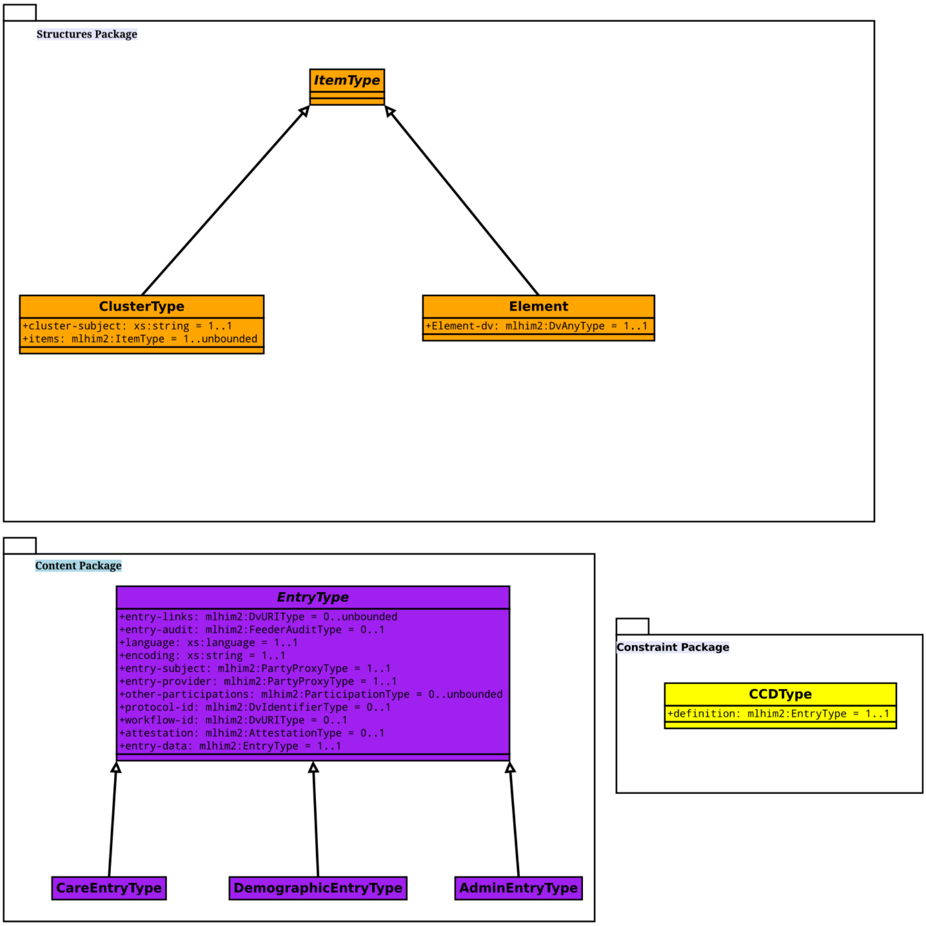

Structures Package
ItemType
ClusterType
+cluster-subject: xs:string = 1..1
+items: mlhim2:ItemType = 1..unbounded
Element
+Element-dv: mlhim2:DvAnyType = 1..1
Content Package
EntryType
+entry-links: mlhim2:DvURIType = 0..unbounded
+entry-audit: mlhim2:FeederAuditType = 0..1
+language: xs:language = 1..1
+encoding: xs:string = 1..1
+entry-subject: mlhim2:PartyProxyType = 1..1
+entry-provider: mlhim2:PartyProxyType = 1..1
+other-participations: mlhim2:ParticipationType = 0..unbounded
+protocol-id: mlhim2:DvIdentifierType = 0..1
+workflow-id: mlhim2:DvURIType = 0..1
+attestation: mlhim2:AttestationType = 0..1
+entry-data: mlhim2:EntryType = 1..1
CareEntryType
DemographicEntryType
AdminEntryType
Constraint Package
CCDType
+definition: mlhim2:EntryType = 1..1

Fig. 2. UML diagram of the MLHIM Reference Model – Structures, Content and Constraint packages

Table 4. MLHIM Reference Model: Common package

Parent Type	complexType	Usage
PartyProxyType	PartySelfType PartyIdentifiedType	Representing the subject of the record Proxy data for an identified party other than the subject of the record
xs:anyType	ParticipationType AttestationType FeederAuditType FeederAuditDetailsType	Modeling participation of a Party in an activity Recording an attestation of item(s) of record content by a Party Audit and other meta-data for software applications and systems in the feeder chain Audit details for any system in a feeder system chain
ExceptionalValueType	Please refer to [36]	Please refer to [36]

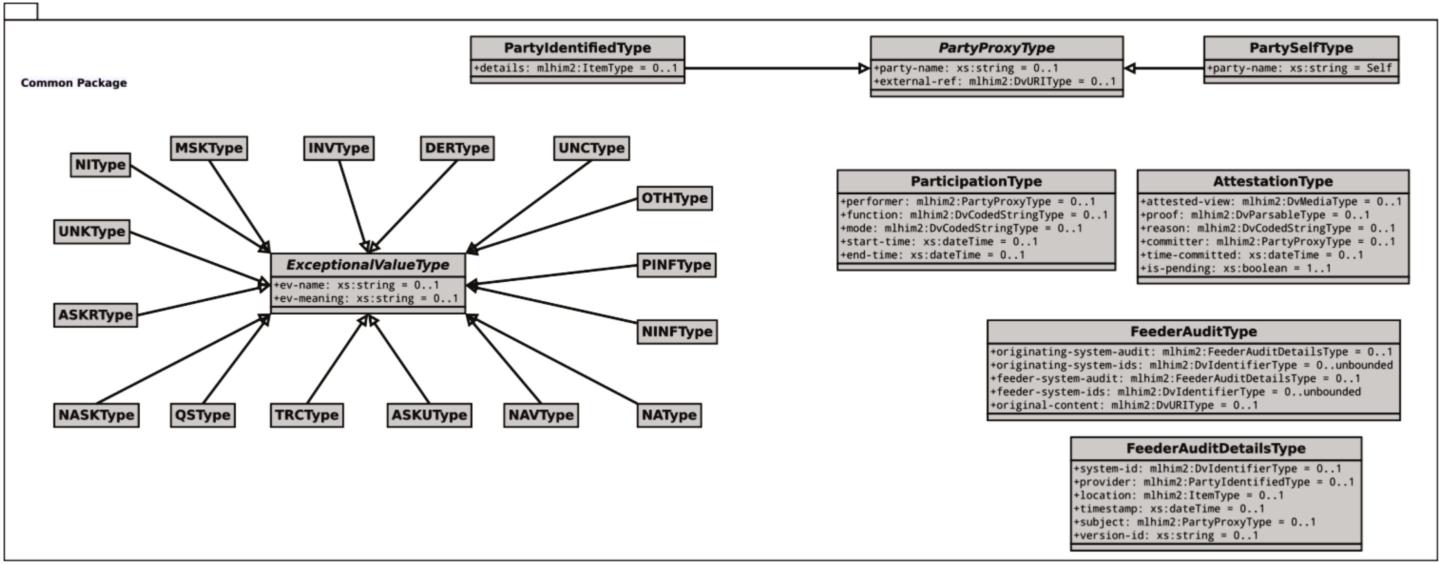

Common Package
PartyIdentifiedType
+details: mlhim2:ItemType = 0..1
PartyProxyType
+party-name: xs:string = 0..1
+external-ref: mlhim2:DvURIType = 0..1
PartySelfType
+party-name: xs:string = Self
ExceptionalValueType
+ev-name: xs:string = 0..1
+ev-meaning: xs:string = 0..1
NIType
MSKType
INVType
DERType
UNCType
OTHType
UNKType
PINFType
ASKRType
NINFType
NASKType
QSType
TRCType
ASKUType
NAVType
NAType
ParticipationType
+performer: mlhim2:PartyProxyType = 0..1
+function: mlhim2:DvCodedStringType = 0..1
+mode: mlhim2:DvCodedStringType = 0..1
+start-time: xs:dateTime = 0..1
+end-time: xs:dateTime = 0..1
AttestationType
+attested-view: mlhim2:DvMediaType = 0..1
+proof: mlhim2:DvParsableType = 0..1
+reason: mlhim2:DvCodedStringType = 0..1
+committer: mlhim2:PartyProxyType = 0..1
+time-committed: xs:dateTime = 0..1
+is-pending: xs:boolean = 1..1
FeederAuditType
+originating-system-audit: mlhim2:FeederAuditDetailsType = 0..1
+originating-system-ids: mlhim2:DvIdentifierType = 0..unbounded
+feeder-system-audit: mlhim2:FeederAuditDetailsType = 0..1
+feeder-system-ids: mlhim2:DvIdentifierType = 0..unbounded
+original-content: mlhim2:DvURIType = 0..1
FeederAuditDetailsType
+system-id: mlhim2:DvIdentifierType = 0..1
+provider: mlhim2:PartyIdentifiedType = 0..1
+location: mlhim2:ItemType = 0..1
+timestamp: xs:dateTime = 0..1
+subject: mlhim2:PartyProxyType = 0..1
+version-id: xs:string = 0..1

Fig. 3. UML diagram of the MLHIM Reference Model – Common package

The Common package is composed by complexTypes that inherit directly from the 'xs:anyType' from the XML Schema 1.1 specifications, containing all of the components required for all CCDs such as subject of care, provider and other participants as well as audit trail and exceptional value information (Table 4). A UML diagram of the Common package is shown in Figure 3.

2.3 Demo Application Development

Two demo applications were developed using the MLHIM Demo EMR, an eXist-db based application development framework of the MLHIM specifications (code available at https://github.com/mlhim/mlhim-emr). The demo application data models were based on a set of selected Common Data Elements (CDE) developed by the National Cancer Institute, available at the NCI CDE Browser (https://cdebrowser.nci.nih.gov/CDEBrowser/), related to Demographic and Vital Signs (Demo 1) and Demographic and Basic Metabolic Panel (BMP) (Demo 2) data. The CDEs were mapped to the admin interface of the Concept Constraint Definition Generator (CCD-Gen), a web-based MLHIM CCD editor (https://github.com/twcook/ccdgen-public), which generated the code for the Pluggable ComplexTypes (PCT) for each CDE. Some CCDs pre-dated the CCD-Gen and were hand developed using an XML Schema editor. A mixture of MLHIM 2.4.1 and 2.4.2 CCDs were used to demonstrate the continued validity of MLHIM based instance over time, even as the Reference Model may be modified for future versions. The CCDs were validated and simulated XML data instances were generated for each CCD by the use of the XML editor oXygen version 14.2 and persisted in the eXist-db database.

A minimalist application design was used on the demo applications in order to demonstrate the interoperability provided by MLHIM and does not represent the industrial implementation of a fully functional, robust Electronic Medical Record (EMR) or other healthcare application. Two instances of both applications were installed with some instance data from different CCDs in each. The patient record identifiers and demographics were identical, since it was not the purpose of this paper to address the issue of patient record linking. Again, that is outside the scope of semantic interoperability. The CCDs used are available from the Healthcare Knowledge Component Repository at: http://hkcr.net/ccd_sets/mlhim_emr_demo.

3 Results

The achievement of semantic interoperability between the two demo applications was based on two core elements: the data model definitions as CCDs, and the backwards validation chain, from the data instance to the CCD schema, the MLHIM Reference Model Schema and finally to the W3C XML specifications.

3.1 Data Modeling

The ‘Demographics’ CCDType was constrained to DemographicEntry complexType, which contained a ClusterType including ElementTypes for person details and address data. The ‘Vital Signs’ and ‘BMP’ CCDTypes were constrained to CareEntry complexTypes. The ‘Vital Signs’ CCD included blood pressure, heart and respiratory rate and temperature measurements; the ‘BMP’ CCD defined the data model for the recording of sodium, potassium, glucose, urea and creatinine measurements.

The data modeling process defined the type of each data element, according to the MLHIM Datatypes package, as defined in Tables 1 and 2 and Figure 1. For instance, for the definition of the data element ‘Gender’, DvStringType was chosen, and restrictions were made to its correspondent ‘enumeration’ facet, to constrain the permissible values to ‘Male’, ‘Female’, ‘Unknown’ and ‘Unspecified’. The same process was repeated to each one of the data elements included in the CCDs by following the specific requirements of each data type defined on the MLHIM specifications.

After the definition of the data types for all ElementTypes, they were combined into a ClusterType (see Table 3). In the CCD-Gen, the procedure requires the selection of the ElementTypes that will compose a given ClusterType. For this demo application, one ClusterType was defined for each one of the CCDs, which included all correspondent ElementTypes as seen on Table 5.

The ClusterType that contains all the ItemTypes is associated to an EntryType that corresponds to demographic (DemographicEntry), administrative (AdminEntry) or clinical (CareEntry) data (Table 3). In the CCD-Gen, this association is made by the selection of the containing Cluster that will be included in the chosen Entry child type as the value for the 'entry-data' element. In this example, the Demographic CCD was modeled as a DemographicEntry, and Vital Signs and BMP were modeled as CareEntry types. To complete the generation of the CCD, Dublin Core Metadata Initiative (DCMI) information was included in the correspondent section of the CCDs.

The Demographic, Vital Signs and BMP CCDs defined the simulated XML data instances for 130 fictitious patients, each of them with one Demographic data instance and n (n = 1, 2, 3...) Vital Signs and BMP data instances, resulting, as an example, in 1,531 data instances of Diastolic Blood Pressure from the Vital Signs CCD. All data instances were valid according to the correspondent CCDs, and those were valid according to the MLHIM Reference Model Schema (either 2.4.1 or 2.4.2), which is valid according to the W3C XML Schema Definition 1.1 and to the W3C XML Language specification; thus, the MLHIM specifications achieved a complete backwards validation chain, from the data instance to the W3C XML specifications. That was repeated for all data instances, with a success rate of 100%. Figure 4 shows an XQuery performed on the database using the web-based XQuery IDE eXide.

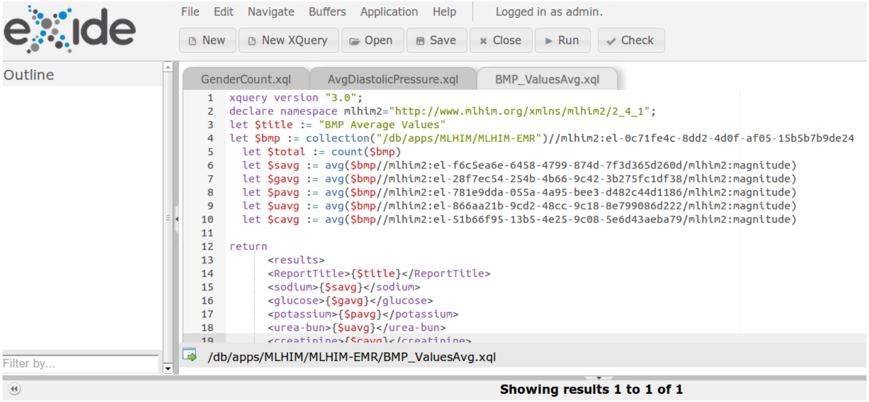
Table 5. Results of the data modeling for the concepts of Demograhics, Vital Signs and Basic Metabolic Panel (BMP) as MLHIM CCDs

CCD	Data Element	Data Type
Demograhic	Gender	DvString with enumeration
	Zip Code	DvIdentifier
	State	DvCodedString
	City	DvCodedString
	Driver License no.	DvIdentifier
	Social Security no.	DvIdentifier
	Phone no.	DvString
	Email address	DvURI
	First Name	DvString
	Last Name	DvString
Vital Signs	Systolic Pressure	DvQuantity
	Diastolic Pressure	DvQuantity
	BP Device Type	DvString with enumeration
	Cuff Location	DvString with enumeration
	Patient Position	DvString with enumeration
	Heart Rate	DvCount
	Respiration	DvCount
	Body Temperature	DvQuantity
	Temperature Location	DvString with enumeration
	Temperature Device	DvString with enumeration
BMP	Sodium	DvQuantity
	Potassium	DvQuantity
	Glucose	DvQuantity
	Urea	DvQuantity
	Creatinine	DvQuantity

The Basic Metabolic Panel CCD based on RM 2.4.2 (id=ccd-f8dada44-e1e9-4ea9-8e7e-46af767ccc66) also demonstrates the use of ‘xs:assert’ elements. These assertions are XPath statements that are added to complexTypes to provide more fine grained control or permissible data such as the requirements for a valid geographical latitude;

```
   <xs:asserttest="matches(mlhim2:DvString-dv,'^-?([1-
8]?[0- 9]\.{1}\d{1,6}$|90\.{1}0{1,6}$)')"/>
```

as well as to provide a level of built-in decision support. The assertions can function as business rules on one complexType, or across multiple complexTypes in a CCD, to insure that if a certain type of data is chosen for one entry then it may restrict the available entries for another choice. For example, if the Gender was chosen as Male, then it might restrict a selection of test options from including Pap Smear. This is a key benefit of the internal semantics of a CCD. In current application design approaches there is no way to share this concept with other applications. In MLHIM, it is shared by default.



```
xquery version "3.0";
declare namespace mlhim2="http://www.mlhim.org/xmlns/mlhim2/2_4_1";
let $title := "BMP Average Values"
let $bmp := collection("/db/apps/MLHIM/MLHIM-EMR")//mlhim2:el-0c71fe4c-8dd2-4d0f-af05-15b5b7b9de24
  let $total := count($bmp)
  let $savg := avg($bmp//mlhim2:el-f6c5ea6e-6458-4799-874d-7f3d365d260d/mlhim2:magnitude)
  let $gavg := avg($bmp//mlhim2:el-28f7ec54-254b-4b66-9c42-3b275fc1df38/mlhim2:magnitude)
  let $pavg := avg($bmp//mlhim2:el-781e9dda-055a-4a95-bee3-d482c44d1186/mlhim2:magnitude)
  let $uavg := avg($bmp//mlhim2:el-866aa21b-9cd2-48cc-9c18-8e799086d222/mlhim2:magnitude)
  let $cavg := avg($bmp//mlhim2:el-51b66f95-13b5-4e25-9c08-5e6d43aeba79/mlhim2:magnitude)

return
      <results>
      <ReportTitle>{$title}</ReportTitle>
      <sodium>{$savg}</sodium>
      <glucose>{$gavg}</glucose>
      <potassium>{$pavg}</potassium>
      <urea-bun>{$uavg}</urea-bun>
      <creatinine>{$cavg}</creatinine>
```

```
<results>
   <ReportTitle>BMP Average Values</ReportTitle>
   <sodium>139.9633914863924</sodium>
   <glucose>99.51531577110978</glucose>
   <potassium>4.489028262386586</potassium>
   <urea-bun>13.443119330076774</urea-bun>
   <creatinine>1.001613747383116</creatinine>
   <Total>5732</Total>
</results>
```

Fig. 4. eXide XQuery for BMP average (detail)

3.2 The Proof of Concept

The proof of concept of the achievement of semantic interoperability across the MLHIM-based demo applications developed for this study is shown in the ability to exchange instance data between applications and those instance data components having the ability to point to the specific semantics for the concept, as well as adhere to the exact syntactic constraints that were designed for those semantics at CCD modeling time. The demos are quite small, but this proof of concept was kept small so that the entire system can be seen at one time without the analysis being too arduous.

Since the unit of exchange is the concept as defined by a CCD and the representation in XML is available across all platforms, it is therefore proven that any type of healthcare information application can be accommodated in the MLHIM ecosystem.

4 Discussion

This study presented the process of development of an open source, industry-standard based multilevel modeling specification. The results have shown that the adoption of XML technologies implemented in a multi-level approach, allowed the establishment of a backward validation chain from the data instance to the original W3C XML specifications.

The real advantage of adopting XML technologies for the development of the MLHIM specifications is the potential of having semantically interoperable applications being developed for real healthcare settings completely independent of the application size or use. Since XML is a universal industry standard and every major programming language has binding tools for XML Schema, this allows developers to work in their preferred language, using their preferred persistence models and yet not build data silos [37]. MLHIM-based applications can persist data on native XML and other types of NoSQL databases as well as SQL databases. It is also common to generate GUIs through XForms tools and other language specific frameworks as required for each application.

Using XML technologies also allows use of emerging semantic web tools and technologies by allowing CCDs to be marked up with common use RDF and other tags for semantic reasoning across conforming instance data [38]. The uniqueness in the MLHIM approach is to *not markup the instance data*, but the CCDs that the instance data refers to for its syntactic and semantic constraints. This approach reduces the size and overall overhead of data querying and exchange processes.

The knowledge modeling process adopted in this study was based on the MLHIM specifications. The process of modeling CCDs was a simple task for the domain expert, only responsible for selecting the NCI CDE concepts and defining their data types according to the MLHIM Datatypes package, and then defined the constraint for each variable on the CCD-Gen. It is important to notice that there have been reports in literature that found the elaboration of openEHR archetypes quite complex [30], but that has not being the case for MLHIM CCDs.

It is important to note that systems that use MLHIM concepts do not have to have the Reference Model in source code or even the CCD for that matter. That is why the validity chain is important. It is considered best practice for new applications to be written based on the MLHIM Reference Model; however, this is not required for effective semantic interoperability, which is ensured by the exchange of documents containing valid instance data and their correspondent CCDs. Since MLHIM is based on a widely adopted and well supported industry standard, the XML Schema representation can be used with virtually any application in any programming language. The only requirement will be that the application can import and export valid instance data when compared with the CCD. This provides a complete validation chain that no other approach can provide; from the data instance to the CCD, to the MLHIM RM Schema, to the W3C XML Schema and finally to the W3C XML specification.

4.1 Relationship to Model-Driven Architecture

There are some conceptual similarities between multi-level modeling in MLHIM and Model Driven Architecture, also known as model-driven engineering or metamodeling; however, there are distinct differences. The MDA approach is concerned with the overall architecture and development of a specific software application or specific system of applications developed around a set of requirements. This approach improves software quality and ease of maintenance. These are generally implemented using a domain specific language (DSL) and a specific technology platform [39].

While MLHIM incorporates those same advantages, it extends the MDA approach to achieve syntactic and semantic interoperability at the concept level, across every development platform. This is accomplished by using XML technologies, because of the ubiquity of XML [40]. While DSLs provide a significant level of power and control when used in a closed environment, this is not the case in healthcare, where a multitude of software and hardware platforms must be accommodated. This has been proven by the *open*EHR Foundation specifications where they initiated the multi-level interoperability concepts but used ADL, a DSL that lacks broad uptake and reusable tooling. After more than 15 years it has achieved very little penetration across the global healthcare community, in spite of also being part of the ISO 13606 standard.

During the development of MLHIM the MDA approach, using the Eclipse Modeling Framework (EMF), was investigated, which showed that the EMF locks the developer into that technology; even the XML Schema export process includes EMF dependencies. In addition to this the Eclipse system did not fully support XML Schema 1.1. This lack of support for multiple substitution groups and assertions negated the ability to export the models into a format that was usable outside of the EMF.

4.2 Relationship to OWL and RDF

As there is often confusion in the purposes of the Web Ontology Language (OWL) and the Resource Description Framework (RDF) in building semantic web applications [41], it is important to address them. Our investigation of those technologies has shown that both OWL and RDF have been extensively used to markup or define a structure for metadata (in the case of OWL) [42] and instance data (in the case of RDF) [43]. However, we did not find implementations where syntactic data model structures were marked up with either to create concept models for interoperability.

OWL is intended as an ontology language for the Semantic Web with formally defined meaning. OWL ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL ontologies can be used along with information written in RDF, and OWL ontologies themselves are primarily exchanged as RDF documents using one of several syntaxes.

Since RDF is an implementation for graph networks of information and can also represent OWL constructs it is useful for MLHIM in having one representation syntax for all MLHIM metadata. A major representation for RDF is XML and therefore can exploit the plethora of XML tools for processing. We decided that it is a natural fit for MLHIM to use RDF/XML to represent CCD metadata and provide the semantic links for that metadata. However, RDF lacks the expressiveness, syntactic structure and completeness as well as the relationship to XPath and XQuery that XML Schema provides. Because of these and other missing features of these two concepts as well as the complexity in expressing relationships in them, a number of syntaxes have evolved for each. This leads to wide open challenge to interoperability.

Therefore, the MLHIM specifications use RDF as it was intended, as a link to expanded semantics, by including the ability to add these links into the CCD so that it represents the semantics of all data instances generated against it, without the requirement to include that code and data overhead in every instance. However, it is important to keep studying those technologies; there is a possibility that, with maturity in the specifications and the tooling, MLHIM 3.x may be developed using OWL semantics, using the RDF/XML representation. At that time there will be tooling that can translate MLHIM 2.x instances (that will remain valid) to MLHIM 3.x instances without loss of semantic integrity.

4.3 Relationship to other Standards

MLHIM can be seen, in a general way, as the harmonization of the Health Level Seven version 3 (HL7v3) standard and the *open*EHR specifications, without the limitations they each introduce. For instance, in *open*EHR, there is a requirement that the entire Reference Model be included in each application, since there is no independent validity chain for *open*EHR; all validation is based on the human eye or internal *open*EHR Reference Model parser or validator. This is not the case with MLHIM because it uses standard XML technologies that are available on all platforms in both open source and proprietary packages.

A similar comparison can be made with the Health Level Seven version 3 (HL7v3) standard. Although HL7v3 is not a restriction-based multi-level model standard, it is also XML-based. The challenge for the achievement of semantic interoperability with HL7v3 is that, since it is not fully restriction-based, there is no validity chain to insure conformance back to a known valid model. The HL7v3 Reference Information Model-based data models are all independently designed as can be seen by the update, simplify and re-expand process that has gone on through its history, which poses maintenance issues for HL7v3-based applications. Although the HL7v3 Common Definition Architecture (CDA) has been partially adopted as a reference document and there is now a tendency to use it as a base reference model, it is too large and has unnecessary requirements for many applications, such as mobile applications or devices using only a push data approach.

MLHIM 1.x began as an XML Schema implementation of the *open*EHR model, to which additional HL7v3 benefits were added, such as closer alignment with ISO 21090, finally having all of the semantics that directly relate to specific applications (such as EMRs) extracted. Also, there is the semantic integrity issue developed in the *open*EHR eco-system by non-reviewed archetypes being created, outside of the centralized control required by the *open*EHR specifications; that creates the risk that multiple archetypes with the same archetype ID (that may actually define different syntactic and semantic structures) to appear in that eco-system. This issue causes instance data to, in the best case, be invalid and, in the worst case, create unknown and undetectable errors. This issue was solved in MLHIM with the CCD identification by Type 4 UUID and by making CCDs non editable. This also resulted in a much simpler eco-system since there is no need to track CCD modifications and versioning.

The Integrating the Healthcare Enterprise (IHE) profiles that actually define data structures are implementable in MLHIM. However, the majority of IHE work is based around standardized work-flow, which is not a semantic interoperability issue, being solved at the application implementation level.

The Standards and Interoperability (S&I) Framework is analogous to the HL7v3 CDA and the NCI CDE initiatives. It can be defined as a top-down, document-centric approach attempting to gain consensus on modeling concepts. The documents available from the S&I Content Browser can be modeled as MLHIM CCDs or collections of CCDs without requiring global consensus, still keeping semantic interoperability among distributed, independently developed applications.

4.4 Disadvantages of the MLHIM Approach

The adoption of any technological solution to a social problem requires trade-offs and the healthcare domain is not different. The first major hurdle for the adoption of a technology such as MLHIM is a shift in the thinking from one level to multi-level modeling. There are anecdotal comments that (within the healthcare informatics domain) this shift is similar to that required from geo-centric to helio-centric awareness in the study of cosmology. This is a challenge for many software developers that have been taught how to develop one-level modeled systems.

Another challenge is the complexity of the XML Schema reference model implementation and the rules around CCD development. Many of the one-level model XML experts are not familiar with this innovative use of the XML Schema specifications, which is similar to the geo-centric versus helio-centric debate described above.

The last and likely the most difficult issue is the need for domain experts to participate in the development process. Though there is enough evidence showing healthcare providers should be included in the design process, this is not yet regarded as a formal part of the work for most of the healthcare professionals. In order to overcome this obstacle, there is a need for the emergence of a new area of expertise in biomedical sciences: knowledge modeling. A healthcare knowledge modeling expert should be specifically trained to take the domain knowledge from healthcare providers and turn it into computer-readable concept models such as MLHIM CCDs.

5 Conclusion

The results of this study showed that semantic interoperability in healthcare information systems is achievable by the adoption of the multilevel modeling approach, which is implementable by the XML technology-based MLHIM specifications. While the broad goal of the MLHIM specifications is to foster long-term, semantic and syntactic interoperability across all healthcare related applications on a global scale, even self-contained applications can benefit from MLHIM technologies (e.g., a software company, a locality, a state). Even in such cases, it is possible to build applications that are already interoperable and require less maintenance overhead as the science of healthcare changes, in the temporal, spatial and ontological dimensions.

A key concept in any interoperability solution is that there is an eco-system that must grow and permeate the industry. As long as the information technology businesses benefit from the lack of interoperability, government policies and user requirements must demand it, since the technological solution exists. The MLHIM eco-system model approach has learned from decades of research on a global basis. The MLHIM approach allows developing application requirements capability, at any level, to suit the local needs, across all time; along with maintaining interoperability and freedom for developers' choices.

Ongoing and future work requires improved tools to engage domain experts. As the functionality of the Eclipse Modeling Framework matures it may be suitable to use for a more solid model-driven engineering approach.

References

1. ACMI, electronic medical records, `http://www.youtube.com/watch?v=t-aiKlIc6uk` (last accessed: April 1, 2013)
2. De Leon, S., Connelly-Flores, A., Mostashari, F., Shih, S.C.: The business end of health information technology. Can a fully integrated electronic health record increase provider productivity in a large community practice? J. Med. Pract. Manage 25, 342–349 (2010)
3. Javitt, J.C.: How to succeed in health information technology. Health Aff. (Millwood) (2004); Suppl. Web Exclusives:W4-321-4
4. U.S. National Library of Medicine. 2011AA SNOMED CT Source Information, `http://www.nlm.nih.gov/research/umls/sourcereleasedocs/current/SNOMEDCT` (last accessed: April 28, 2013)
5. Maojo, V., Kulikowski, C.: Medical informatics and bioinformatics: integration or evolution through scientific crises? Methods Inf. Med. 45, 474–482 (2006)
6. Fitzmaurice, J.M., Adams, K., Eisenberg, J.: Three decades of research on computer applications in health care: medical informatics support at the Agency for Healthcare Research and Quality. J. Am. Med. Inform. Assoc. 9, 144–160 (2002)
7. Preker, A., Harding, A.: The economics of hospital reform from hierarchical to market-based incentives. World Hosp. Health Serv. 39, 3–10 (2003)
8. Harris, N.M., Thorpe, R., Dickinson, H., Rorison, F., Barrett, C., Williams, C.: Hospital and after: experience of patients and carers in rural and remote North Queensland, Australia. Rural Remote Health 4, 246 (2004)
9. Zusman, E.: Form facilitates function: innovations in architecture and design drive quality and efficiency in healthcare. Neurosurgery 66, N24 (2010)
10. Ward, M.M., Vartak, S., Schwichtenberg, T., Wakefield, D.: Nurses' perceptions of how clinical information system implementation affects workflow and patient care. Comput. Inform. Nurs. 29, 502–511 (2011)
11. Jung, M., Choi, M.: A mechanism of institutional isomorphism in referral networks among hospitals in Seoul, South Korea. Health Care Manag (Frederick) 29, 133–146 (2010)
12. Hoangmai, H.P., O'Malley, A.S., Bach, P.B., Saiontz-Martinez, C., Schrag, D.: Primary care physicians' links to other physicians through medicare patients: the scope of care coordination. Ann. Intern. Med. 150, 236–242 (2009)
13. Sittig, D.F., Singh, H.: A new sociotechnical model for studying health information technology in complex adaptive healthcare systems. QualSaf Health Care (suppl. 3), i68–i74 (2010)
14. Hyman, W.: When medical devices talk to each other: the promise and challenges of interoperability. Biomed. Instrum. Technol. (suppl.), 28–31 (2010)
15. Charters, K.: Home telehealth electronic health information lessons learned. Stud. Health Technol. Inform. 146, 719 (2009)
16. Raths, D.: Shifting away from silos. The interoperability challenges that hospitals face pale in comparison to the headaches plaguing State Departments. Healthc Inform. 27, 32–33 (2010)

17. Achimugu, P., Soriyan, A., Oluwagbemi, O., Ajayi, A.: Record linkage system in a complex relational database: MINPHIS example. Stud. Health Technol. Inform. 160, 1127–1130 (2010)
18. Metaxiotis, K., Ptochos, D., Psarras, J.: E-health in the new millennium: a research and practice agenda. Int. J. Electron. Healthc 1, 165–175 (2004)
19. Hufnagel, S.P.: Interoperability. Mil. Med. 174, 43–50 (2009)
20. Kadry, B., Sanderson, I.C., Macario, A.: Challenges that limit meaningful use of health information technology. Curr. Opin. Anaesthesiol. 23, 184–192 (2010)
21. Blobel, B., Pharow, P.: Analysis and evaluation of EHR approaches. Stud. Health Technol. Inform. 136, 359–364 (2008)
22. Rodrigues, J.M., Kumar, A., Bousquet, C., Trombert, B.: Using the CEN/ISO standard for categorial structure to harmonise the development of WHO international terminologies. Stud. Health Technol. Inform. 159, 255–259 (2009)
23. Blobel, B.: Ontologies, knowledge representation, artificial intelligence: hype or prerequisites for international pHealth interoperability? Stud. Health Technol. Inform. 165, 11–20 (2011)
24. National Health Service Media Centre. Dismantling the NHS national programme for IT, `http://mediacentre.dh.gov.uk/2011/09/22/dismantling-the-nhs-national-programme-for-it` (last accessed: May 15, 2013)
25. Lohrs, S.: Google to end health records service after it fails to attract users. The New York Times (June 24, 2011), `http://www.nytimes.com/2011/06/25/technology/25health.html?_r=3&.` (last accessed: April 28, 2013)
26. Kalra, D., Beale, T., Heard, S.: The open EHR Foundation. Stud. Health Technol. Inform. 115, 153–173 (2005)
27. Martinez-Costa, C., Menarguez-Tortosa, M., Fernandez-Breis, J.T.: Towards ISO 13606 and open EHR archetype-based semantic interoperability. Stud. Health Technol. Inform. 150, 260–264 (2009)
28. Cavalini, L.T., Cook, T.W.: Health informatics: The relevance of open source and multilevel modeling. In: Hissam, S.A., Russo, B., de Mendonça Neto, M.G., Kon, F. (eds.) OSS 2011. IFIP AICT, vol. 365, pp. 338–347. Springer, Heidelberg (2011)
29. Dias, R.D., Cook, T.W., Freire, S.: Modeling healthcare authorization and claim submissions using the openEHR dual-model approach. BMC Med. Inform. Decis. Mak. 11, 60 (2011)
30. Kashfi, H., Torgersson, O.: A migration to an open EHR-based clinical application. Stud. Health Technol. Inform. 150, 152–156 (2009)
31. Eichelberg, M., Aden, T., Riesmeier, J., Dogac, A., Laleci, G.: A survey and analysis of electronic healthcare record standards. ACM Comput. Surv. 37, 277–315 (2005)
32. Yu, S., Berry, D., Bisbal, J.: Clinical coverage of an archetype repository over SNOMED-CT. J. Biomed. Inform. 45, 408–418 (2012)
33. Menezes, A.L., Cirilo, C.E., Moraes, J.L.C., Souza, W.L., Prado, A.: Using archetypes and domain specific languages on development of ubiquitous applications to pervasive healthcare. In: Proc. IEEE 23rd Int. Symp. Comput. Bas. Med. Syst., pp. 395–400 (2010)
34. Cavalini, L.T., Cook, T.: Knowledge engineering of healthcare applications based on minimalist multilevel models. In: IEEE 14th Int. Conf. e-Health Networ. Appl. Serv., pp. 431–434 (2012)
35. Sanderson, D.: Loss of data semantics in syntax directed translation. PhD Thesis in Computer Sciences. Renesselaer Polytechnic Institute, New York (1994)
36. Cook, T.W., Cavalini, L.: Implementing a specification for exceptional data in multilevel modeling of healthcare applications. ACM Sighit Rec. 2, 11 (2012)

37. Lee, T., Hon, C.T., Cheung, D.X.: Schema design and management for e-government data interoperability. Electr. Je.-Gov., 381–391 (2009)
38. Daconta, M.C., Obrst, L.J., Smith, K.T.: The Semantic Web. Wiley, Indianapolis (2003)
39. Rutle, A., MacCaull, W., Wang, H.: A metamodelling approach to behaviouralmodeling. In: Proc. 4th Worksh. Behav. Mod. Foundat. Appl., vol. 5 (2012)
40. Seligman, L., Roenthal, A.: XML's impact on databases and data sharing. Computer, 59–67 (2001)
41. Fenton, S., Giannangelo, K., Kallem, C., Scichilone, R.: Data standards, data quality, and interoperability. J. Ahima (2007); extended online edition
42. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer, 2 edn., http://www.w3.org/TR/owl2-primer/#Modeling_Knowledge:_Basic_Notions (last accessed: October 17, 2013)
43. Manola, F., Miller, E.: RDF Primer, http://www.w3.org/TR/rdf-primer/#dublincore (last accessed: October 17, 2013)