Design of an Electronic Healthcare Record Server Based on Part 1 of ISO EN 13606

Tony Austin*, Yin Su Lim, David Nguyen, Dipak Kalra

Centre for Health Informatics and Multiprofessional Education, UCL, London, UK

Submitted July 2010. Accepted for publication February 2011.

ABSTRACT

ISO EN 13606 is a newly approved standard at European and ISO levels for the meaningful exchange of clinical information between systems. Although conceived as an inter-operability standard to which existing electronic health record (EHR) systems will transform legacy data, the requirements met and architectural approach reflected in this standard also make it a good candidate for the internal architecture of an EHR server. The authors have built such a server for the storage of healthcare records and demonstrated that it is possible to use ISO EN 13606 part 1 as the basis of an internal system architecture. The development of the system and some of the applications of the server are described in this paper. It is the first known operational implementation of the standard as an EHR system.

1. INTRODUCTION

Electronic health records (EHRs) are now part of many national eHealth programmes, aiming to support the migration away from paper records, the detailed capture of relevant clinical information, its life-long storage and the communication of EHR information between systems in order to create a cradle-to-grave record for every citizen.

An essential component of achieving such challenging aims is the use of standards to represent and communicate EHR information. Such standards need to cater for the complexity and richness of such information, recognising the diversity of needs between care settings and information needs: from community nursing to acute coronary care, from paediatrics to medicine for the elderly, from psychiatry to cancer genomics. Health information is often contextual: individual observations or inferences need to be interpreted with an understanding of the overall patient picture at the time. The information also needs to be managed in ways that ensure it is well governed, enabling the EHR to meet ethical and legal requirements.

* Corresponding Author: Tony Austin, Centre for Health Informatics and Multiprofessional Education, University College London, 4th Floor, Holborn Union Building, The Archway Campus, Highgate Hill, London N19 5LW England. Phone: (+44) 20 7288 3372. Fax: (+44) 20 7288 3322. E-mail: tony.austin@ucl.ac.uk. Other authors: y.lim@ucl.ac.uk; d.nguyen@ucl.ac.uk; d.kalra@ucl.ac.uk.

The challenge of representing the EHR has been the subject of significant international research. This includes European framework programme projects such as the Good European Health Record [1], Synapses [2], SynEx [3] and EHR-SupA [4] from which the ISO EN 13606 standard ultimately derived. The requirements for EHR architectures are well documented in the outputs of the research projects mentioned above, as well as national projects [e.g., 5], academic work ([e.g., 6]), and the international standards themselves [7, 8]. The last of these, presently in final draft form, will be the most complete internationally agreed set of requirements that any interoperability standard and any implemented EHR server needs to meet.

The earliest European interoperability standard, ENV 12265 [9] was published in 1995 and defined the core information properties of an EHR. ENV 13606 [10] (published in 1999) was the first complete information model standard, drawing on the research results of the projects mentioned above. The authors used this standard as the basis for a previous generation of EHR server that was implemented, deployed and technically evaluated by one of the authors [11].

Experience gained across Europe from vendors who had implemented ENV 13606, as well as new requirements from clinical and governmental stakeholders, have contributed to the latest EHR communications standard, ISO EN 13606. This standard has five parts, published progressively between 2007 and 2010, and defines:

- a generic reference model of an EHR “extract” [12];
- a knowledge representation for the clinical organisational structure of EHRs: clinical “archetypes” [13];
- a set of vocabularies and reference archetypes to ensure a consistent mapping between EN13606 and other key standards such as HL7 [14];
- a framework for communicating the EHR disclosure wishes of patients and for communicating the audit history of who has accessed a particular patient’s health record [15];
- interfaces between requesting and responding processes or systems to enable EHR communication [16].

Although conceived and published as an interoperability standard to which existing EHR systems and services will transform legacy data through interfaces, the requirements met and architectural approach reflected in this standard also make it a good candidate for the internal information architecture of an EHR server. The objective of the present work is to demonstrate this.

2. METHODS

ISO EN 13606 emphasises what it calls the “twin model approach”. Recognising that it is impossible to describe the whole of clinical medicine but that it is possible to definitively establish certain features of sensible server design, two models are offered to implementers that inform different aspects of the process.

The reference model defined in Part 1 of the standard establishes the set of “global truths” that are not dependent on the clinical domain being modelled. For example, it does not matter if the application is for the management of asthma or diabetes, all

entries in either domain must be dated and recorded with the specifics of the clinician who observed them.

In contrast, the knowledge that a thing called a “Blood Pressure” always includes further things called “Systolic” and “Diastolic” values is unique to clinical domains related to the circulatory system. Each of these individual clinical domain knowledge aggregates is what part 2 of ISO EN 13606 refers to as an “archetype” [13].

The previous record server from UCL [11] also followed the twin model approach. However, it predated many if not all of the modern turnkey Object-Relational Mapping (ORM) frameworks and a large proportion of the Web application development toolkits that are now mature. This new version of the server was designed to take advantage of more modern ORM and Web options, and in particular:

1. was to use toolkits wherever possible;
2. was to emphasise programmer productivity when writing applications to use the server.

2.1. Infrastructure

The server was built using open-source components starting at the lowest tier with Java™ [17]. Further components are shown in Figure 1.

The open-source container JBoss [18] was chosen as the Application Server, and the record services are deployed in JBoss as a collection of Enterprise JavaBean 3 [19] “stateless beans”. Persistent record entities are simply “Plain Old Java Objects” (POJOs) that are annotated with Java class file metadata describing the persistence methodology so that at run-time, the built-in Hibernate [20] ORM layer can manufacture suitable tables and columns in the database.

Figure 1. The infrastructure components

Although we selected the open-source PostgreSQL [21] database for our in-house deployments, the use of Hibernate ensures that any of a range of databases can be used instead. In fact however, all of our user sites have been satisfied with PostgreSQL in terms of performance and support.

JBoss also comes equipped with a version of Tomcat [22], the reference implementation of the Java Servlet standard [23] from the Apache group, and this we augmented with a distribution of Apache Pluto [24] which forms the reference for the Portlet standard [25] (also known as JSR-168). This pair underpins the Web applications that call the record server and is only very loosely coupled to it, access being provided only via a Java interface looked up in a directory service.

2.2. New Components

The record server itself comprises a number of individual modules constructed by us. A simple deployment metadata change determines whether these services are provided only to the Web applications running in the same JBoss container, or more widely to other remote callers. The services are shown in Figure 2.

Broadly, the server is constructed so that non-core server functions are contained in a separate extensions package. One such extension (“Important Dates”) is shown in Figure 2, which allows application users to record dates during which appointments with patients should not be made (for example, during national holidays or staff absence).

Figure 2. Record server components

The core server comprises the “Users” package that manages user log-in, the “Record” package that manages storage of health records themselves, with a complementary “Demographics” package to manage the evolution of patient demographics data, and finally “Audit”. Actually, very few audit functions are exposed to requesters directly and this package is mostly employed by other modules to note when an action has taken place. Use is made of the “façade” design pattern [26] to hide the implementations of the services behind the regular Java interfaces that are provided to callers.

2.2.1. Archetypes

Two further major components are not diagrammed in Figure 2 because they don’t have a data storage implication. The first is a “validations” package that examines class file metadata and determines whether the values stored in a class instance meet the criteria. “Archetypes” define the clinical data structures that an EHR system should anticipate and be able to incorporate. The system therefore needs to validate candidate instances of EHR data against the structure and constraints of the archetypes to which these instances indicate conformance. Only instances that conform should be stored in the repository.

For example, a theoretical class for a Systolic Blood Pressure might include the following stipulations:

```
@Quantity(Property.PRESSURE)
@PossibleMin(0)
public class Systolic extends Element {
    public getSystolicValue() {
        ...
    }
}
```

Given this definition, the validation toolkit would police that the Systolic value provided at run time by a user did in fact meet the physiological minimum criteria specified. Since the validation engine does not itself cause data to be committed, it is usually possible to apply it on both the server and client sides to any data committed as part of the same ISO EN 13606 Composition. Some validations do however require data to be extracted from the server and compared, and this would require a live connection.

There is a new Java standard for the validation of class properties known as “Bean Validation” (JSR-303) [27]. Although the latter standard was formalised long after the activities described in this paper took shape, the new standard does follow a very similar approach and a future task may be to align the vocabularies of this and the new work.

2.2.2. Rendering

The second component not shown in Figure 2 is a major toolkit introduction for the client. Hibernate and other ORM toolkits provide a convenient way to introspect a piece of data and determine automatically (or almost automatically) how it should be persisted in a database. However, class file metadata is sufficient to describe how a screen should be consistently presented as well. To our knowledge, no effective toolkit had yet been written to take advantage of this as part of the archetype expression, although other toolkits do exist that use archetypes with a second separate formalism [28].

2.3. Security

Obviously for a development of this nature, a critical issue is the security model under which the server is expected to operate. This includes the security inherent in the implementation as well as that of the information retained. Most access to the server occurs through Web applications, and it goes without saying that these are secured with transport-level security using Secure Sockets Layer (SSL) [29]. However, the server has inherent security features that remain in place even when it is not accessed from a server-local Web tier.

2.3.1. Log-In

The model for user log-in is shown in Figure 3. Broadly, a “User” has one of five fixed “ApplicationRights” which determine which patient data they are permitted to see, if

Figure 3. The model for users

any. “Accounts”, or administrative care domains, group users and patients so that on log-in a user selecting an account may view the records of any patient with whom they share that account. A “Patient” meanwhile, may only view his/her own record irrespective of his/her account. An “Administrator” is permitted to see the demographics of any patient in the system across all accounts but may not see records at all (this enables administrators to bulk-assign patients to accounts when a server is being introduced). A “Researcher” is in some sense the inverse of an administrator, able to see aggregated query results from records across all patients but denied access to demographics with which they would otherwise be able to determine who the results relate to. Finally a “Carer” is a special type of user who must be named by a patient explicitly and be able log-in to the appropriate account, in order to view a record.

Even as a result of having ApplicationRights of User or Carer, a professional does not automatically get to see all the available data in a record. A user may simultaneously have more than one “Role” (for example “General Practitioner” or “Researcher in HIV”) and must nominate which role he or she assumes when logging-in. The selected role and account is indelibly added to an audit of access whenever records are returned, even in aggregate form.

Each role that may be selected, and each record instance, has an associated sensitivity rating. The default value is “Clinical Care”, indicating a sensitivity suitable for use by healthcare professionals delivering any aspect of care but not for more widespread dissemination. The archetype model from which record instances are derived can modify the default value for creation, modification or access of instances and the sensitivity may be varied still further per-instance by an authoring clinician.

Patients or users may vary the sensitivity associated with named roles accessing their record (see Figure 4), with respect to accessed archetypes or to individual record compositions. When a user selects a role on log-in, he or she assumes the sensitivity rating that role entails and every action is compared to the sensitivity that should be needed in order to perform it. If the role does not have sufficient rights, the action is precluded. In such a way, access to records may be denied even to logged-in users with otherwise appropriate roles.

2.3.2. Function Security

The JBoss Application Server is one of a range of servers capable of “Aspect-Orientation” [30]. This is a feature that enables cross-cutting concerns such as security

Consent	
subjectOfCareIdentifier:	String [1]
role :	Role [1]
accessSensitivity:	Sensitivity [0..1]
amendSensitivity:	Sensitivity [0..1]
createSensitivity:	Sensitivity [0..1]
dateCreated:	Timestamp [1]
dateDeprecated:	Timestamp [0..1]

Figure 4. Patient consent preferences

and audit to be modelled functionally separate from the feature that it is securing or auditing. A coarse-grained example of this is the “RolesAllowed” feature in JBoss which polices that certain functions are not available to certain ApplicationRights. For example in the following code, only a logged-in user with rights of “Administrator” is allowed to create a new account to associate with patients:

```
@RolesAllowed({ "ADMINISTRATOR" })
public void addAccount(Account acct) {
    ...
}
```

However the function security can be much more sophisticated than this, allowing for fine-grained policing of role-based activities. For example, the “getUser(username)” function is available to all users irrespective of their ApplicationRights, but only in the case of administrators are details for those other than the calling user available to view.

2.3.3. Audit

Auditability will be a widely exploited aspect of healthcare record systems in the future as patients and governments demand more accountability and transparency in the access to and use of personal health data. Research shows that users who know their use of a system is being monitored are more likely to behave appropriately [31]. A number of actions are audited by this system, and crucially, the audit trail is available immediately to view (see Figure 5).

2.4. Record Access

The service whose responsibility it is to return records is called the “FeederDatabaseSource”. As for all services, it is built as an EJB 3.0 Stateless Session Bean [19] in Java that is container-managed.

The previous server envisaged archetypes as a set of constraints on generic objects specified in the reference model. Bearing in mind the desire to simplify development as much as possible for programmers, this version of the FeederDatabaseSource expects archetypes to be represented more conventionally with classes that inherit from those

Audit View

Please fill in select one or more search parameters to perform a search.

Username:

Patient ID:

Role Name:

Purpose:

Archetype:

Date from: Date to:

There are 2 audit(s) found.

Date Audited	Purpose	Actor	Patient ID	Record Component ID	Archetype	Function Name	Notes
2010-02-08 12:04:05	RECORD_COMPONENT_RETRIEVED	tomy/Clinical Care/CHME	Smith, John Henry (January 19, 1956)	00000000-4628-2a75-0000-000000000001	TherapyPlan	getRecordComponent()	
2010-02-08 12:04:16	RECORD_COMPONENT_RETRIEVED	tomy/Clinical Care/CHME	Smith, John Henry (January 19, 1956)	00000000-4628-2a75-0000-000000000001	TherapyPlan	getRecordComponent()	

Figure 5. An audit of “Therapy Plan” record access

of the reference model. For example, a BloodPressure could be represented as an ISO EN 13606 Cluster which has properties set to those expected of a BloodPressure:

```
Cluster bp = new Cluster();
bp.setName("BloodPressure");
bp.setArchetypeIdentifier("...");
...
```

Nevertheless, a better alternative from a developer's point of view just has the BloodPressure as a regular class which sets the values itself:

```
BloodPressure bp = new BloodPressure();
...
```

This more conventional approach to instance creation reduces the complexity associated with using an ORM tool for automated persistence, but more particularly makes for a query expression using the tool that most developers will already be familiar with. If the underlying database is accessed directly for reporting purposes or compliance assessment, then rather than an unconventional approach using ISO EN 13606 tables (or even a single "RecordComponent" table), the use of "proper" classes within the ORM layer means that users see tables and columns that have meaning to them in the clinical domain of interest. The classes themselves are almost entirely "boilerplate" code, and can readily be generated from the original archetype definition.

As well as ensuring that all components of a record are dated and stored with the user credentials under which they were established, the FeederDatabaseSource maintains a history of every version of every record component from the first to the most recent. It also marks import of data from a different system distinctly from data whose primary definition is local. For local definitions, a site-wide identifier generator is available that is guaranteed unique if used. Of course, table columns containing these identifiers themselves have uniqueness constraints, in case the provided generator is not used.

3. RESULTS

The record server has been employed to provide record services for four application domains so far. Each of the applications builds on the basic framework of record storage and visualisation. In principle, to create a new application, a clinical panel describes the data to be captured and the associated layout of screens. Any reports or bespoke application features are also described. The engineering team then generates clinical classes corresponding to the screen design and implements the bespoke features, including colour palette and logo. Relevant data entry form objects are incorporated to capture data of the type specified for each archetype node, including free text, coded values, quantities, dates and times, and multimedia objects (e.g. digital images, which may be uploaded).

A new application usable for data capture may be generated for end user testing in just a few days. An interesting by-product of this increased efficiency in rapidly mapping clinical end user requirements to archetypes and then to testable complete applications is

The screenshot shows the 'HeartBeat' application interface. On the left is a sidebar with a tree view containing categories like 'Users', 'Patients', 'Administration', 'Anticoagulant Control', 'Reports and Letters', and 'Additional Links'. The main content area is titled 'Anticoagulation Clinic Contact' and contains a form with various input fields and checkboxes. Fields include 'Present Health Status', 'INR Test Result', 'Treatment Controller' (set to 'MAINTENANCE'), 'Days of Anticoagulant Control', 'Recommended Dosage' (with sub-fields for 'Accepted' and 'Dosage'), 'Recommended Interval', 'Appointment Date' (set to '28 Jan 2009 11:17'), and 'Clinic Name' (set to 'CHIME'). There are also checkboxes for 'Unknown value' and 'Accepted' next to several fields. A message at the bottom states 'There are no previous Clinic Contacts, so only INITIATION could offer support'. At the bottom right are 'OK' and 'Cancel' buttons.

Figure 6. The server used for cardiovascular health

that the rate-determining step in application development has become the time taken to gain clinical consensus. This process has been found to take longer than might be anticipated (e.g., months, not weeks) because of simple logistic constraints such as the time taken to schedule meetings at which busy senior clinicians can discuss the more complex aspects of the application. (The authors have found that e-working and paper document exchanges frequently fail to deliver the required consensus.)

The applications are summarised in the following sections.

3.1. Cardiovascular Health

A distributed anticoagulation service [32] (Figure 6) in the Whittington Hospital in London is the principal clinical reference site for UCL's research and development on EHRs. The demonstrator setting now includes full support to practitioners across the north London geographical region (five Primary Care Trust areas) including General Practices and secondary prescribers and delivers care to almost 10,000 patients. The system incorporates queries and reports to support the clinical governance of this distributed service.

In summer 2008, a distributed heart failure service became the second cardiovascular health area to be added to the application. Plans for further areas such as atrial fibrillation management are also in progress.

Most recently, the north London anticoagulant service has been recognised personally by the British Prime Minister as delivering the highest level of customer service excellence.

3.2. Athletic Training Support

The SESAME project [33] (Figure 7), which commenced in mid-2006, is acquiring novel data from elite athletes (including British Olympic candidates) from worn body sensors and observations made by high-speed 3-D video, to provide coaches with new insight into ways to improve performance and avoid injury. The project entails

Figure 7. The server used for athlete training

compiling an electronic profile and training record for each athlete and augmenting this with monitoring data that have been analysed using innovative bio-mechanical models.

3.3. Mental Health

A memory management clinical application (figure 8) is a third clinical area, being funded by the Royal Berkshire NHS Trust in the UK, and to be deployed initially to support the outpatient management of approximately 1,500 patients with Alzheimer's Disease. This system is being deployed with the capacity to support greater patient

Figure 8. The server used for mental health

The screenshot shows the GReYTRIALS web application. The header includes the logo and a welcome message: "Welcome to Greytrial, Dr. Tony Austin, Your patient is Stevens, Theophilus March 15, 1937". A left sidebar contains navigation links for Users, Patients, Clinical Data, Reports and Letters, Participant Search, and Additional Links. The main content area is titled "Patient Consent" and includes a note "Fields marked * are mandatory". It contains a "Patient Consent" section with a "Study Recruitment Information" field, a "No Contact For:" dropdown, a "Capacity to Consent:" dropdown (set to YES), a "Current Stage of Consent:" dropdown (set to AUTOMATICALLY_ACCEPTED), a "Consented For Tissue Banking:" checkbox, a "Consent Signed By:" field, an "Approached or Consented By:" field, a "Copy to GP:" checkbox, a "Copy to Patient/Carer:" checkbox, and an "Original Filed in notes:" checkbox. At the bottom right are "OK" and "Cancel" buttons.

Figure 9. The server used for trials recruitment

numbers, as it is anticipated that other Trusts will be interested in collaborating in shared care underpinned by a common EHR.

3.4. Trials Recruitment

The Dementias and Neurodegenerative Diseases Research Network (DeNDRoN) is a national research network funded by the UK Department of Health initially for five years from 2006. The server is to be used to underpin two national patient registries for dementia and for motor neurone disease, respectively. These registries are a core part of the strategy to increase national recruitment to clinical trials. The first disease registry, for dementia, will shortly begin use, and there is every intention for the research and development partnership to continue to embrace additional neurodegenerative diseases during the five year period.

That four such distinct information and knowledge scenarios could be successfully underpinned by one database and visualisation architecture, helps establish the validity of the standard and the implementation, and the value of a dual model approach that allows for such rapid customisation.

3.5. Summary of the Results

The authors have built a complete record server based on the ISO EN 13606 standard for medical records exchange. The middleware comprises key services for authentication, demographics management, records retrieval and audit as well as some extended services for clinic management. It incorporates many features designed to facilitate the secure curatorship of a longitudinal health record including role-based access control and patient consent preferences. The server can faithfully record any required clinical data and includes typical record server features such as versioning and audit. Meanwhile, it is generic at run-time, meaning that it could potentially be extended while in operation with new data capture options. It exhibits these properties while at the same time maximising the familiarity for coders developing with the server, and for database administrators extracting data for independent reporting.

Several application domains have been successfully modelled using the ISO EN 13606 approach and deployed in practice using this server. Around 20,000 patients are now managed across the deployment scenarios. Each application is physically housed on a different server so that sponsoring organisations can legally claim to “own” the data, but in fact all of the independent records could be merged into a single longitudinal record for a patient at any time. In fact, a demonstration server for all four applications does indeed connect to one single repository.

4. DISCUSSION

While the use of the ISO EN 13606 standard seems reasonable, given its modelling focus and outward simplicity, developing an entire server that embodies its intent is not itself a simple endeavour. This is the first known complete and practically usable implementation of ISO EN 13606 as a persistent architecture, and although not open-source in itself, some portions are expected to be published to inform other international activities. Other research progressing towards this goal includes the work of Brass et al. [34]

4.1. Clinical Domain Modelling

A question occasionally asked, particularly in the light of major development work at a national or trans-national level, is what relevance a server of this scale has when millions of pounds are poured into much larger efforts. We have found that often the larger efforts, driven by the short term imperative to deliver a particular application domain, lack the semantic power to integrate longitudinally with other domains or query effectively across them. In contrast, servers based on archetypes retain the ability to merge across an arbitrary number of domains.

Archetypes, considered narrowly as a constraint description on a reference model rather than in its wider connotation of clinical domain modelling generally, are not actually required because of the inability of systems to completely model the clinical domain. Many domains cannot be completely modelled at the point of release. A computing toolkit for graphical user interfaces for example, could not also include every possible window content that could be constructed with it. This is the problem that inheritance is designed to solve.

Clearly, clinical domain modelling generally is appropriate and the work described here demonstrates that at least in Java, an approach based on inheritance of clinical classes rather than supplying values to a part 1 superclass works perfectly well for the purpose. Not every implementer will choose Java of course, and a description formalism that can be readily employed across executable engineering environments would be sensible. Based on our findings, we would advocate OWL [35] or its predecessor RDFS [36], or a simple property list for the purpose. The necessary metadata statements are in fact relatively simplistic, and the property list approach makes it possible to represent them in a relational database for collaborative development [37].

4.2. ISO EN 13606 Part 1

With or without archetypes, there are health record and clinical document formalisms available that are more complicated than the one embodied by ISO EN 13606. The need

for such complexity seems reasonable to question based on the purpose of the standard. Since ISO EN 13606 represents the limit of what is possible to request from an EHR (because it is the exchange standard), there is, by extension, no point in persisting anything else that might require a more complicated solution.

ISO EN 13606 part 1 is a printed standard and, unlike most contemporary technical standards, requires no reference implementation. It's not unexpected on that basis that there are oversights, typographical errors, and so forth. It is, in short, unimplementable without modification. However, it is certainly possible to correct the more trivial issues (for example, where variable names change throughout the document but it is clear the intent was that they be the same), and even some more complicated ones under certain assumptions about usage (for example, where classes that describe Null could themselves be Null). We understand that these corrections, influenced by our findings, will be published as a technical revision in due course.

The document-orientation of the standard sits well with those used to the paper record, and translates well into document-centric engineering environments like XML. But it succeeds less well in a typical relational database where, for example, data values occupy a single column rather than a whole other table simply because they inherit from complex data. Actually, because the model is generic, a suitable database layout often involves one large table that stores all the values in order to facilitate polymorphic queries. In reality, individual element entries almost never need attribute data that is at variance with the container and a typical hand-crafted representation in relational terms could dispense with that portion of the super-structure altogether.

It is not always clear when a model attribute should be used, and when an archetype. For example, when determining the length of a therapy plan, should there be an actual start and actual end date archetype, or should the observation begin and end time ("obs_time") attribute be used? The fact that the question might even be asked, suggests that at some point a modeller may do the "wrong" thing (in this case we would elect to use an archetype, because of the regularity of there being an "intended" date as well, not provided as an attribute).

At present, ISO EN 13606 does not provide good guidance about which data types to adopt in a candidate record. Although the standard includes some types (for example, for Coded Text), these are only so that the standard is self-contained. That is, they are not sufficient for describing an actual record, only for describing the structure of the transmission container. A new standard 21090 promulgated by ISO was intended to replace the minimal set in the original 13606 but is so large as to be unusable for this task. A new reduced "EHR profile" of 21090 is in development and should rectify the problem.

4.3. Implementation Technologies

In order to avoid the overhead of writing a parser for a query language, the FeederDatabaseSource contains a lightweight set of functions that enable various types of data retrieval based on the archetype identifier. These use Enterprise JavaBeans Query Language (EJB-QL) [38]. One of the reasons EJB-QL was chosen was to enable database implementations to be changed according to the preferences of a potential

deployment environment. To the contrary, all deployment partners have so far installed PostgreSQL instead to replicate our in-house development environment. Since that removes the need for database independence, the development could default to native SQL which is easier to test with the database's own supplied tools.

The persistence and the Web tier are cleanly separated and deployed individually in the container. Unfortunately, the libraries that each depends on are provided as part of a JBoss installation and are difficult to upgrade individually without upgrading the container as a whole. For example, a recent feasibility experiment looked at upgrading to version 2 of Java Server Faces (JSF) [39]. This would deliver a number of benefits in simplification and enhanced functionality. However, it is not easy to upgrade JSF by itself in version 4.0.4 of JBoss. In order to upgrade JSF, it is necessary to upgrade the container as a whole. However, that entails an upgrade to a later version of Hibernate in which a query syntax used by the FeederDatabaseSource has been deprecated. It is of course, just as difficult to downgrade Hibernate in a later container as it is to upgrade JSF in an earlier one.

In this version of the system, users with ApplicationRights Administrator are able to administer any account that has been defined. It would increase scalability if the scope of that could be controlled. One way of doing so is to provide a new "SuperAdmin" ApplicationRight whose function is simply to define logical groupings of accounts and attach administrators to them. SuperAdmins need not have access to either patient records nor demographics. Administrator users would then retain their present purpose but within their defined account groupings only.

The Web applications came in two phases with non-clinical components in the first phase (for example, user and patient creation and management) and the rendering toolkit developed subsequently. Unfortunately, the two phases were written using two different technologies, Java Server Pages (JSP) [40] and JSF. While JSF is considered the "better" way to develop Web applications of significant size, it was introduced into a container in which JSP was already established. This complicated style-sheet development and made it very difficult to get pages to formally validate against a Document Type Definition.

4.4. Performance

A well-known Web application maxim is that no individual screen population should take longer than 10 seconds. Beyond that timeframe, a user's mental map of his/her own intentions breaks down. Even on aged machinery with software disk encryption, the server exhibits round-trip times significantly under this amount. For example, viewing a list of recent clinic contacts for a patient takes approximately 5 seconds for lists with hundreds of such contacts. Since a simple list of contacts requires very little information (usually just an identifier), but viewing the most recent contact requires a lot (because the complete ISO EN 13606 part 1 hierarchy from the Composition through to the data values must be fully instantiated), it is instructive to note that 4 seconds are required even when there is only one contact in the list. This means that the majority of time is spent on walking the hierarchy implied by the model, and performance-oriented simplifications at both the 13606 model level and at the archetyped domain model level would be best directed at hierarchy reduction. Entering a single record entry also takes

approximately 5 seconds. This again reflects the complexity of hierarchy building, but this time including permissions checking on the basis of patient consent preferences and role-based sensitivity.

Obviously, timings are variable and based on the type of machine, the available random-access memory, and the speed, number and configuration of disk drives, as well as the complexity of the healthcare record being manipulated. A modern server-class device would in general be expected to offer response times significantly reduced from those described above.

5. CONCLUSION

This research has demonstrated the use of ISO EN 13606 as the architectural basis for the implementation of an EHR server comprising a repository that conforms to Part 1 of the standard and middleware components that process 13606-conformant record instances, utilises 13606-2-based archetypes as the basis for data validation and utilises RECORD_COMPONENT sensitivity according to Part 4 as part of its security architecture. Although developed as a standard for communication (i.e., for an interoperability architecture), this research has demonstrated that it is possible to use ISO EN 13606 as the basis of an internal system architecture. It is the first known operational implementation of the standard within an EHR system.

There are a number of re-usable outputs from the work. The most important is a set of technical corrections to the ISO EN 13606 standard which will be provided as input to a future technical revision. The second is a set of publishable artifacts such as XML Schema and interface specifications. A final output is experience of the design of EHR servers, which is partly achieved with this publication and partly by others planned that focus more on the engineering approaches adopted.

REFERENCES

- [1] Lloyd D, Kalra D, Beale T, Maskens A, Dixon R, Ellis J, Camplin D, Grubb P, and Ingram D (eds). The GEHR Final Architecture Description. European Commission, Brussels; 1995; The Good European Health Record Project: Deliverable 19. 250 pages. available from <http://www.chime.ucl.ac.uk/HealthI/GEHR/EUCEN/del19.pdf> (last accessed July 2010)
- [2] Grimson J, Grimson W, Berry D, Stephens G, Felton E, Kalra D, Toussaint P, and Weier O.W. A CORBA-based integration of distributed electronic healthcare records using the synapses approach. *IEEE Trans Inf Technol Biomed.* Sep 1998; 2(3):124-38
- [3] Sottile PA, Ferrara FM, Grimson W, Kalra D, Scherrer JR. The holistic healthcare information system. *Toward an Electronic Health Record Europe '99.* Nov 1999; 259-266.
- [4] Dixon R, Grubb PA, Lloyd D, and Kalra D. Consolidated List of Requirements. EHCR Support Action Deliverable 1.4. European Commission DGXIII, Brussels; May 2001. 59 pages. Available from http://www.chime.ucl.ac.uk/HealthI/EHCR-SupA/del1-4v1_3.PDF (last accessed July 2010)
- [5] Introducing computer based patient records: prerequisites and requirements. Swedish Institute for Health Services Development (SPRI), Sweden; 1998; Report number 477; ISSN 0586-1691.
- [6] Kalra D. Clinical Foundations and Information Architecture for the Implementation of a Federated Health Record Service. PhD Thesis. University of London, 2002. Available from <http://eprints.ucl.ac.uk/1584/> (last accessed July 2010)
- [7] Schloeffel P, (ed). Requirements for an Electronic Health Record Architecture. ISO TS 18308. International Organisation for Standardisation, Geneva, 2002.

- [8] Kalra D, (ed). Requirements for an Electronic Health Record Reference Architecture. ISO FDIS 18308 Final Draft International Standard. ISO, Geneva, 2010
- [9] Hurlen P, (ed). Electronic Healthcare Record Architecture. ENV 12265. CEN, Brussels, 1995.
- [10] Kay S and Marley T (eds). EHCR Communications: Part 1 Electronic Healthcare Record Architecture. ENV 13606. CEN, Brussels, 1999.
- [11] Austin T. The Development and Comparative Evaluation of Middleware and Database Architectures for the Implementation of an Electronic Healthcare Record. PhD Thesis. University of London, 2004.
- [12] Kalra D (ed). Electronic Health Record Communication Part 5: Interface Specification. ISO/EN 13606-5. ISO, Geneva, 2010.
- [13] Kalra D (ed). Electronic Health Record Communication Part 3: Reference Archetypes and Term Lists. ISO 13606-3. ISO, Geneva, 2009.
- [14] Kalra D (ed). Electronic Health Record Communication Part 4: Security. ISO TS 13606-4. ISO, Geneva, 2009.
- [15] Kalra D, Beale T, Lloyd D, Heard S (eds). Electronic Health Record Communication Part 2: Archetype Interchange Specification. ISO 13606-2. ISO, Geneva, 2008.
- [16] Kalra D, Lloyd D (eds). ISO 13606 Electronic Health Record Communication Part 1: Reference Model. ISO 13606-1. ISO, Geneva, 2008.
- [17] <http://www.javasoft.com> (last accessed July 2010)
- [18] <http://www.jboss.com> (last accessed July 2010)
- [19] <http://java.sun.com/products/ejb> (last accessed July 2010)
- [20] <http://www.hibernate.org> (last accessed July 2010)
- [21] <http://www.postgresql.org> (last accessed July 2010)
- [22] <http://tomcat.apache.org> (last accessed July 2010)
- [23] <http://java.sun.com/products/servlet> (last accessed July 2010)
- [24] <http://portals.apache.org/pluto> (last accessed July 2010)
- [25] <http://jcp.org/en/jsr/detail?id=168> (last accessed July 2010)
- [26] Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: Elements of reusable object-oriented software. Addison Wesley. ISBN: 0201633612
- [27] <http://jcp.org/en/jsr/detail?id=303> (last accessed July 2010)
- [28] van der Linden H, Austin T, Talmon J. Generic screen representations for future-proof systems, is it possible? There is more to a GUI than meets the eye, Comput Methods Programs Biomed. Vol 95, 2009, p 213-26.
- [29] <http://tools.ietf.org/html/rfc5246> (last accessed July 2010)
- [30] <http://jboss.org/jbossaop> (last accessed July 2010)
- [31] Safran C., Rind D., Citroen M., Bakker A.R., Slack W.V., and Bleich H.L. Protection of confidentiality in the computer-based patient record. MD Computing 1995; 12:187-92.
- [32] Coleman B, Martin C, Barber N, Patterson D. An evaluation of the safety and acceptability of an anticoagulation clinic in a community pharmacy setting – a pilot study. The Pharmaceutical Journal 2004, 273: 822-824.
- [33] L. Cheng, G. Kuntze, H. Tan, D. Nguyen, K. Roskilly, J. Lowe, I. N. Bezodis, T. Austin, S. Hailes, D. G. Kerwin, A. Wilson, D. Kalra, "Practical Sensing for Sprint Parameter Monitoring", accepted to be published in the Proceedings of the 7th IEEE Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Boston, Massachusetts, USA, June 2010.
- [34] Brass A, Moner D, Hildebrand C, Robles M. Standardized and flexible health data management with an archetype driven EHR system (EHRflex). Stud Health Technol Inform. 2010;155:212-8.
- [35] <http://www.w3.org/2004/OWL> (last accessed July 2010)
- [36] <http://www.w3.org/TR/rdf-schema> (last accessed July 2010)

- [37] Lea N, Austin T, Hailes S, Kalra D. Expression of Security Policy in Medical Systems for Electronic Healthcare Records. Proceedings of ICMISE 2009, Tokyo.
- [38] http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/EJBQL5.html (last accessed July 2010)
- [39] <http://java.sun.com/javaee/javaxserverfaces> (last accessed July 2010)
- [40] <http://java.sun.com/products/jsp> (last accessed July 2010)