MEDINFO 2015: eHealth-enabled Health
I.N. Sarkar et al. (Eds.)

doi:10.3233/978-1-61499-564-7-45

Towards the Implementation of an openEHR-based Open Source EHR Platform (a vision paper)

Pablo Pazos Gutiérrez[a]

[a] *openEHR en Español, Asociación Chilena de Informática en Salud, CaboLabs*

Abstract

Healthcare Information Systems are a big business. Currently there is an explosion of EHR/EMR products available on the market, and the best tools are really expensive. Many developing countries and healthcare providers cannot access such tools, and for those who can, there is not a clear strategy for the evolution, scaling, and cost of these electronic health products. The lack of standard-based implementations conduct to the creation of isolated information silos that cannot be exploited (i.e. shared between providers to promote a holistic view of each patient's medical history).

This paper exposes the main elements behind a Standard-based Open Source EHR Platform that is future-proof and allows to evolve and scale with minimal cost. The proposed EHR Architecture is based on openEHR specifications, adding elements emerged from research and development experiences, leading to a design that can be implemented in any modern technology. Different implementations will be interoperable by design. This Platform will leverage contexts of scarce resources, reusing clinical knowledge, a common set of software components and services.

Keywords:

Electronic Health Records; Open Source Software; openEHR; Service Oriented Architecture; Data Collection.

Introduction

In this vision paper we present the basic components of an Open EHR Platform that will be used in different scenarios as a shared EHR core for country wide, federated, hospital and clinic EHRs. The term "EHR core" will be use to reference an implementation of the Platform using a specific technology stack.

The main motivation behind this EHR Platform is to create an open alternative to the closed/proprietary solutions currently offered by EHR vendors. But also to create an EHR Platform that is based on open standards and good design practices, searching for a generic, reusable, scalable, extensible, web-based, knowledge-driven, and future-proof solution.

This Platform is "open" in three ways:

1. open specification accessible for anyone.
2. open source tools and components.
3. free to use, adapt and extend.

This will help developers to create new tools and services without the hassle of dealing with e-Health standards and clinical knowledge (time consuming and costly tasks of any EHR development), and enable developments in contexts of scarce human & financial resources (small software companies, dev teams in hospitals, etc.).

In the next sections the following topics will be presented: design principles, architecture, components, services, and ideas pertaining to Platform development and use by client applications. Developers using services provided by the EHR Platform, for shared data storage, querying and Clinical Decision Support (CDS), will create these applications.

Current Status

Although this paper is about the vision and conception of a new kind of EHR systems, we and other colleagues [1][2] have been working in this area, including research, development and training, mainly to understand the problem, delineate a good solution, and validate it against functional prototypes of the EHR Platform. The outstanding component we created is the EHRServer [3]. It provides services of clinical data storage and querying for clinical information, and this system can actually be deployed and used. For Knowledge Management, the Clinical Knowledge Manager [4] of the openEHR Foundation is being used. There are also some proof-of-concept developments around a Demographic Server and a Rule Engine that will soon be released.

Methods

This paper presents the core components of the proposed EHR Platform, which is an extension/specialization of the openEHR EHR Computing Platform [5], defined by the openEHR specifications [6]. The extensions are based on research & development, project development, integration of EHR, HIS and other clinical information systems, and preparing training materials and case studies.

Design Principles of the Open EHR Platform

- Provide a minimum/common set of generic services
  - Reuse: must support a wide range of client applications with the same set of components and services, leveraging available resources.
  - Pareto principle: the platform will provide the 20% of the functionalities of a full blown EHR system to support 80% of the requirements for any EHR system, keeping the Platform small and manageable. The rest of the functionalities should be added by developing client applications, and those might be customized for each application (i.e. won't suit as core functionalities).
  - Scope: The services will be focused on data storage and querying, permitting data interpretation and manipulation to client applications.
  - Standardization: the EHR Platform will implement open standards to reach interoperability, so developers do not need to do that work.

- Modifiability and Extensibility: the metadata-based configuration of the Platform will allow new data structures, data queries and rules to be added without modifying the database structure or source code.
- Clinical data consolidation:
  - The storage services of the Platform collect and consolidate clinical information from different client applications into a loosely centralized platform, thus facilitating a unique EHR, per patient, that is complete and ubiquitously accessible.
  - Consolidated data will be on a Cloud-ready Platform using a distributed approach, avoiding monolithic systems and one-point-of-failure.
- Secure:
  - Clinical and demographic/identification data are physically separated, allowing secondary use of anonymized clinical data.
  - All communications between client applications and the EHR Platform will be encrypted. On each communication, the client will be authenticated and permissions will be verified. Client apps should comply with security rules, and good security practices will be encouraged.
  - Modifications to the data in the EHR will be done only by authenticated and authorized client applications, and those should ensure correct authentication and authorization for their users (this item should be included in the Platform rules of use).
  - No data is actually modified or deleted; amendments, corrections or deletions all generate new versions of the data.

Knowledge Management (KM)

The KM process is grounded on four main artifacts: 1) clinical record structures, 2) terminology, 3) rules, and 4) processes. Those are defined as computable expressions that can be created and updated, using modeling tools [7], by Domain Experts (physicians, nurses, and allied health professionals). These experts, who are closer to the end users of the EHR, will play the role of Clinical Knowledge Modelers that are in responsible for defining the structure and behavior of the EHR, and how it will evolve over time.

The openEHR specifications [6] define two kinds of artifacts: archetypes and templates. Archetypes are units of content (data structures, constraints and terminology) that specify to one health concept, in a processable and self-contained artifact, that has global validity (e.g. Blood Pressure, Problem/Diagnosis, Clinical Evolution, Glasgow Coma Scale, Body Mass Index, etc.). Templates defined by specific clinical documents, valid in local contexts, refer to many Archetypes, and can use all the Archetype structure or just part of it.

Open EHR Platform Architecture

The Open EHR Platform Architecture is constituted by components with well-defined responsibilities, and provides services to other and client applications. This is based on the openEHR architecture [8].

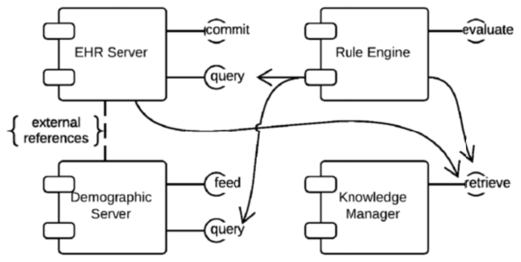

Figure 1 - Open EHR Platform Architecture

Four main components are defined to manage information and provide services for: 1) clinical data (EHR Server), 2) demographic and identification data (Demographic Server), 3) rules to enable CDS (Rule Engine), and 4) Clinical Knowledge (Knowledge Manager). The latter is where all of the metadata, authored by Domain Experts, is stored and made available for the rest of the components: clinical document and administrative data structures, terminologies, rules and workflows.

Due to the narrowed scope of the Platform, a User Interface (UI) for the end-user ought to be provided by client applications developed over the EHR platform. This is why the term "EHR platform" rather than "EHR system" may better suit this work.

Services

Each sub-system of the EHR Platform architecture will provide a set of services related to their responsibilities. Below we describe the major services and internal components of each sub-system.

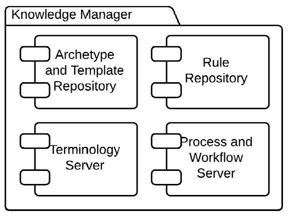
Knowledge Manager
Archetype and Template Repository
Rule Repository
Terminology Server
Process and Workflow Server

Figure 2 - Knowledge Manager Component

The KM sub-system will handle the artifacts created by the KM process. These artifacts are the heart of the whole EHR Platform, if some of those change or new artifacts are added, the data and behavior of the Platform will change, giving the Platform the "future-proof" label. There are four main components:

- Archetype and Template Repository: stores artifacts that define the data structure and low-level constraints of each clinical document in the EHR.
- Terminology Server: provides services around terminologies, classifications, thesauri, dictionaries, and mappings/relationships between them. Archetypes and Templates will use these terminologies.
- Rule Repository: manages rules that will be executed by the Rule Engine.

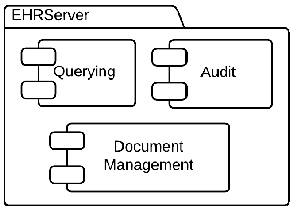

Figure 3 - EHR Server Components

The **EHR Server** is responsible for the safekeeping of clinical records and providing services to change the EHR and query for clinical data.

Document Management: Provides services to validate, create and update clinical information in the EHR of each patient, allowing the management of the internal organization of the clinical records in a directory structure (e.g. by episode or health problem).

Querying: Enables client applications such as EMRs and mHealth apps to access clinical information from the EHR. Queries are specified in an abstract declarative way, based on structures defined by openEHR archetypes [9]. Queries do not depend on specific database technologies, so there is the need to implement a crosswalk between EHR Server queries and database query language such as SQL or XQuery. In this component, queries will be created by Domain Experts and end-users [10], exported & imported and executed against a specific DB technology. So EHR queries can be shared between different implementations of the EHR Server. The results of a query will be returned in standard open formats such as XML and JSON, and organized in different ways: full clinical documents or data points, grouped by data type or by clinical document type, contextualized to time points or intervals.

Audit: Enables the management, evaluation and control of changes to the EHR of a patient, allows the detection of problems such as wrongly organized, deleted or modified clinical information.

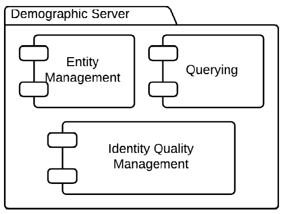

Figure 4 - Demographic Server Components

The Demographic Server is responsible for the management of varying entities such as persons, organizations, groups, devices, systems, their relationships and roles. All of these entities will be referenced from the EHR records, and all referenced entities should be stored in the Demographic Server. This sub-system will act as a Master Patient Index and Human Resource Index (but will not include information about contracts, payments, etc. that will be handled by third party applications). The Identity Quality Management includes functionalities to help detect errored, corrupted and duplicated entities in the demographic repository, and manage the repository audit to evaluate and control changes made by external applications.

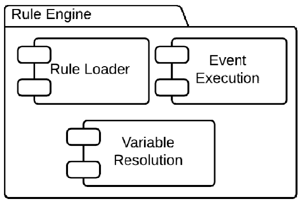

Figure 5 - Rule Engine Components

Rules are declarative units of logic that are evaluated against data, that execute actions based on conditions, and can return values. The rule execution will be triggered by events such as a user clicking a button or selecting a value from a list (e.g. a medication to create a prescription) or system events such as batch jobs executed at certain times. The data to evaluate the rules will come from the EHR Server (clinical data) or the Demographic Server (age range, sex, geographic location), and can be input data (provided by the triggered event), or rule variables (resolved when the rule is executed, e.g. querying the EHR for data). There is a proposal from the community to express rules and guideline-based workflows using a similar syntax to the one used in archetypes [11], this is called the Guideline Definition Language (GDL) [12].

The **Rule Engine** is in charge of loading rules from the Knowledge Manager, Resolve Referenced Values (e.g. querying the EHR Server), and Rule Execution that controls the rule logic and return values.

Customization and Extension: a better way

Since our goal is to design a low cost environment to create EHR systems, the Platform we envision will have an extreme level of modifiability, and this is a design requirement. To reach this objective we need to forget some of the traditional practices currently used to develop software for healthcare. In the traditional way we might be able to create an EHR, but we might not be able to maintain it. So, considering that the costlier and longer phase in software development is maintenance and evolution [13], and EHRs are long-term projects, thus for really effective EHR Systems we need to cut the maintainability cost to the minimum.

The Platform architectural design and the use of metadata to describe the system data structures (archetypes, templates, terminology) and behavior (rules, processes) allows a huge level of modifiability to adapt the EHR to new requirements, while maintaining the stability of the core EHR system. This is achieved because there is no need to modify source code or database schemas to make changes to the Platform, so no new bugs are introduced at the code level. To implement this, a mix of open standards, a Service Oriented Architecture (SOA), current technologies and techniques (Model-View-Controller, Object-Relational Mapping and dynamic languages) should also be applied.

Making changes and adding new features for end-users will be done over client applications, e.g. improving EMR UI or adding a feature, such as tagging patient records, so the core of the system, the EHR Platform, is not affected by these changes.

For example, on the EHR Server, extensions will be added in three main forms:

1. New archetypes and templates will define the structure of new clinical records,
2. New queries will define new ways of accessing the data stored in the EHR,

3. New rules will add behavior, to execute actions under certain conditions, using the data from the EHR.

Deployment

The proposed platform can be offered by a "Software as a Service" (SaaS) model, or software vendors and medical institutions can create their own local deployments. Either way, the platform is focused on providing high availability and data/metadata backup, two basic requirements for any shared EHR system. To provide this, some redundancy is needed; therefore more than one instance of each sub-system should be deployed.

The platform is designed to support this by providing synchronization services between different instances of the same sub-system. E.g. two instances of EHR Server will synchronize data when one instance receives a commit of a clinical document, so queries can be executed in both instances, also providing better performance. This also works when scalability is needed. When the EHR Platform needs to support more applications and users, store and retrieve more data, etc., the service level of the platform should neither deteriorate nor disturb other users. Adding more instances of the Platform sub-systems will let it to scale horizontally without huge investments.

Thinking about long-term viability of a project of this kind, offering SaaS as a hosted solution for software development companies can help to fund the EHR Platform maintenance and evolution. This strategy is implemented by a lot of Open Source projects, accompanying that with paid extensions and value-added services.

Developing Applications

The core set of services provided by the Open EHR Platform will enable developers to focus on creating client applications without starting from scratch, and solve the same problems again and again. Moreover, they get standardized data structures using communication protocols and syntaxes they are familiarized with like REST/SOAP, JSON/XML, without the hassle of implementing the standards themselves, so they create out-of-the-box interoperable applications.

Client applications can be classified into three categories: data input apps, data display apps and mixed. Vital Signs monitors and study result reporting are examples of "data input" applications, because they function to send/commit data to the EHR. Reporting tools are an example of "data display" applications that mainly will query the EHR. EMRs [14] are examples of mixed input/output. It is worth mentioning that when a deployed platform supports several applications, all the clinical documents committed to the EHR by the "data input" apps will be available to the "data display" apps with permission to query the EHR Platform, enabling the implementation of a truly shared EHR.

To help developers adopt the EHR Platform and the methodology to create client applications proposed in this paper, we visualize the creation of helper tools such as Software Development Kits, libraries for different programming languages and learning materials. Some work around this area has already been done. The first is a framework to develop EHR systems called EHRGen [15]. We have also worked in a multi-level methodology and a set of tools to automatically generate User Interfaces for Clinical Information Systems over different technologies (HTML5, Java, .Net), from openEHR Archetypes and Templates [16].

As an example, using the services provided by the EHR Platform, client applications can implement features like: browsing through the patient's medical history, generate clinical and public health reports about population's health with different aggregations (by age range, sex, geographic location, etc.), calculate for different indicators (e.g. number of births per year), sending notifications under certain circumstances to the patient, family, health care team, management team or healthcare authorities (e.g. execution of rules that notify when new test results are available, or when a diagnosis of a certain decease was added to the EHR of a patient), Clinical Decision Support features like alerts, recommendations, reminders, reference material, etc. (rules are executed, and based on context data, a correspondent result is returned, e.g. give an alert if the patient is allergic to the medications that is about to be prescribed).

Validation of the proposed architecture

The validity of the proposed architecture can, and should, be evaluated against standards and specifications, first against international ones, to satisfy generic requirements, methodologies, needs and rules, and then against local compliances that define more specific requirements. Since the architecture is the most generic and abstract definition of a complex system, it should not be evaluated against requirements related to low level designs, with a bigger amount and more specific requirements), and the use of specific technologies. After the architecture is adapted to a low level designs and mapped to a specific technology stack (this is called Implementation Technology Specification, ITS for short), those more specific artifacts can be evaluated to more specific requirements and rules.

Considering the aforementioned ideas, it would be useful to define some guidelines about how to validate the proposed architecture. The actual validation would be a good area for further investigation. Presented below is a list of varied standards and specifications that should (*) or might (+) be used as validation guidelines. In some cases just part of the standard would apply because it is too specific and the validation of the architecture should be done in a very broad, generic and abstract way, because that is the nature of a Software Architecture. After these, local standards and specifications might also apply.

Introduction, Terminology, Requirements and Architecture:

(*) ISO 18308: Requirements for an electronic health record architecture.

(*) openEHR Specifications [6]

(+) IHE IT Infrastructure Profiles [17]

(+) ISO 20514: Electronic health record -- Definition, scope and context

(+) ISO 14292: Personal health records -- Definition, scope and context

(*) ISO 14639: Capacity-based eHealth architecture roadmap -- Part 1: Overview of national eHealth initiatives

(+) ISO 14265: Classification of purposes for processing personal health information

(+) ISO 21298: Health informatics -- Functional and structural roles

(+) ISO 22790: Health informatics -- Functional characteristics of prescriber support systems

Healthcare record definition and sharing:

(*) ISO 27790: Document registry framework

(*) ISO 13128: Clinical document registry federation

(+) ISO/HL7 27932: Data Exchange Standards -- HL7 Clinical Document Architecture, Release 2

(+) ISO 12773-1: Business requirements for health summary records -- Part 1: Requirements

(+) ISO 12773-2: Business requirements for health summary records -- Part 2: Environmental scan

(+) ISO 13119: Clinical knowledge resources -- Metadata

Healthcare Records and Healthcare Data Communication:

(*) ISO 13606-1: Electronic health record communication -- Part 1: Reference model

(*) ISO 13606-2: Electronic health record communication -- Part 2: Archetype interchange specification

(*) ISO 13606-3: Electronic health record communication -- Part 3: Reference archetypes and term lists

(*) ISO 13606-4: Electronic health record communication -- Part 4: Security

(*) ISO 13606-5: Electronic health record communication -- Part 5: Interface specification

Conclusion

We are confident that creating a truly Open EHR Platform will enable e-Health developments and long term EHR projects, that are impossible today due the lack of resources. The proof-of-concept prototypes already developed are attracting attention from the academy and the industry, and we are very close to the delivery of EHR Platform sub-systems in a production-ready state. As time passes, we are also designing and developing tools that will benefit software developers in the creation of openEHR-based applications that will be compatible with the Platform. Such applications could produce a highly scalable backend for shared EHR systems, thus allowing their applications to share a common/stable set of services and easily evolve through time.

References

[1] Koray Atalag, et al., Assessment of Software Maintainability of openEHR Based Health Information Systems - A Case Study in Endoscopy, eJHI 2012; Vol7(1):e5

[2] Koray Atalag, et al., A Standards-based Approach to Development of Clinical Registries - Initial Lessons Learnt from the Gestational Diabetes Registry, HiNZ 2014.

[3] EHRServer: open source, openEHR based, Shared EHR System project. https://github.com/ppazos/cabolabs-ehrserver (accessed on Dec 22nd 2014)

[4] Clinical Knowledge Manager by Ocean Informatics http://ckm.openehr.org/ (accessed on Dev 22nd 2014)

[5] What's openEHR? http://www.openehr.org/what_is_openehr (accessed on Dec 22nd 2014)

[6] openEHR Specifications http://www.openehr.org/programs/specification/releases/1.0.2 (accessed on Dec 22nd 2014)

[7] openEHR Modeling Tools http://openehr.org/downloads/modellingtools

[8] Beale, T., Heard, S., openEHR Architecture Overview, openEHR Foundation, 2008. http://openehr.org/releases/1.0.2/architecture/overview.pdf

[9] Chunlan Ma, Heath Frankel, Thomas Beale, Sam Heard, EHR Query Language (EQL) – A Query Language for Archetype-Based Health Records, MEDINFO 2007, pp.397-401 (2007)

[10]Shelly Sachdeva, Subhash Bhalla, Visual Query Language for Archertype-Based Electronic Health Records Databases, Journal of Information Processing, Vol.20 No.2, pp.438-450, 2012.

[11]Thomas Beale, Sam Heard, Archetype Definition Language, openEHR Foundation 2008. http://www.openehr.org/releases/1.0.2/architecture/am/adl.pdf (accessed on Dec 22nd 2014)

[12]Rong Chen, Iago Corbal, Expressing and Sharing Clinical Decision Support Rules Using openEHR Archetypes, MEDINFO 2013.

[13]Schach, R. (1999), Software Engineering, Fourth Edition.

[14]Electronic Medical Record definition http://www.healthit.gov/providers-professionals/electronic-medical-records-emr (accessed on Dec 22nd 2014)

[15]Pablo Pazos, EHRGen: Generador de Sistemas Normalizados de Historia Clínica Electrónica Basados en openEHR, Congreso Argentino de Informática y Salud 2012.

[16]Arianne Palau, Laura Cuadrado, Pablo Pazos, Generación automática de interfaces de usuario para sistemas de información clínicos basados en una metodología multi-nivel, INFOLAC 2014.

[17]IHE IT Infrastructure Profiles http://wiki.ihe.net/index.php?title=Profiles#IHE_IT_Infrastructure_Profiles

Address for correspondence

Pablo Pazos Gutiérrez
Clinical Informatics Consultant
pablo.pazos@cabolabs.com