MEDINFO 2021: One World, One Health – Global Partnership for Digital Innovation
P. Otero et al. (Eds.)

doi:10.3233/SHTI220030

OpenEHR Implementation Guide: Towards Standard Low-Code Healthcare Systems

Samuel Frade[a], Thomas Beale[b], Ricardo J. Cruz-Correia[a,c]

[a] Center for Health and Technology and Services Research - CINTESIS, University of Porto, Portugal
[b] Ars Semantica (UK); openEHR International Board.
[c] VirtualCare, Portugal

Abstract

Several open source components have been made available in recent years to help develop full openEHR systems. Still doubts exist if these are sufficient. This paper presents a case study of implementing a low-code openEHR system, investigating the feasibility and challenges of developing a system using these components for each step. The method used consisted in selecting successful examples of implementation case studies, identifying key development steps, and for each step searching for possible open source options. As a result, we had a working low-code openEHR powered EHR, successfully demonstrating the feasibility of the proposed implementation guide. The main available free or open source components used were ArchetypeDesigner and EHRbase, developed by Better and Vita/HighMed respectively. In our opinion, it is possible to build EHR systems using the available open source components, but support is still missing in the front end, specifically for form generation and screen representation.

Keywords:

Electronic health records; Model-Driven Development, Reference Standards.

Introduction

Electronic Health Records (EHR) have a positive impact in quality of care, efficiency and patient safety, by improving the ability of healthcare professionals in enacting evidence-based knowledge management and aiding decision making. To advance towards those visions, it is imperative to gain the trust of the involved stakeholders, doctors and other medical personnel, patients, families, health care providers and regulators, as well as system developers and IT personnel [1]. Even though two critical requirements are interoperability among the various systems involved and involvement of all stakeholders, currently existing solutions are still vertical silos to a large extent and developed and maintained by IT professionals [2].

OpenEHR is an open specification in health informatics that describes the management, storage, retrieval and exchange of health data in electronic health records. It allows the standardization of the EHR architecture following a multi-level modelling approach, which separates information from knowledge [3]. The first level, the reference model (RM), specifies a generic model according to which data will be stored and communicated (e.g.: data types). The second level, the clinical content models, also called archetypes, defines constraints to the reference model that represents data groups on a specific domain topics (e.g.: blood pressure), that are then to be combined and further constrained for specific use, similar to paper documents of old, the templates (e.g.: vital signs). This fact changes the way health information systems are developed. Domain experts define the structure and element types of the domain content (making it possible to create new models or update the current ones on their own), while the system developers can focus managing the concrete data that represent the data according to the RM, creating user interfaces for the templates and overall system functionalities and applications [3].

The openEHR standard is a promising Model-Driven Development (MDD) approach for electronic healthcare records, providing data interoperability and making both developers and clinicians work together. But modelling clinical domains is complicated and the standardization process complex [4]. A process called "rapid prototyping", which has been applied successfully to software engineering and to design, and is part of agile development methodologies [5], can be the solution for efficient modelling. It can provide an initial draft of a usable system and/or already be the first version of the system. The benefits are enabling early visualization of a model functionality, flexibility to make rapid changes, rapid finalization of requirements and a vehicle for communication between all stakeholders. This can be achieved with minimal difficulty with a low-code approach to development, enabled by the MDD basis of openEHR.

The term "low-code", first mentioned by Forrester Research in 2014 [6], states that companies prefer low-code alternatives for fast, continuous, and test-and-learn delivery. Low-code development platforms are ecosystems with which applications can be developed, with minimal manual programming labor, since the platforms are built and prefigured to generate code itself. Low-code development platforms emphasize visual interfaces to enable people, without a technological background, to create and deploy business apps with relative ease. Furthermore, these platforms also offer companies an economical way to fulfil requirements. With the low-code development platforms, programs or apps for mobile or desktop devices can be created maintaining the multifunctionality and high information-management capabilities.

The existing system that will be part of the study is Obsccare. It is a clinical record software designed to be used by obstetrics and gynecology doctors, anesthesiologists, nurses and administrative staff to register inpatient admission and discharge, childbirth and newborn data, as well as surgical procedures, nursing records and gynecological interventions.

This study implements an openEHR based low-code EHR system, using available openEHR open source software components and following a Model-Driven Architecture, aiming to investigate the feasibility and challenges of developing such a system and to present it as a step by step guide.

Methods

In order to identify key development steps, a search was made in the openEHR official website (www.openehr.org) during April 2021. Even though there was no general implementation guide, the architecture and documentation provide hints as to how the development process should occur (figure 1).

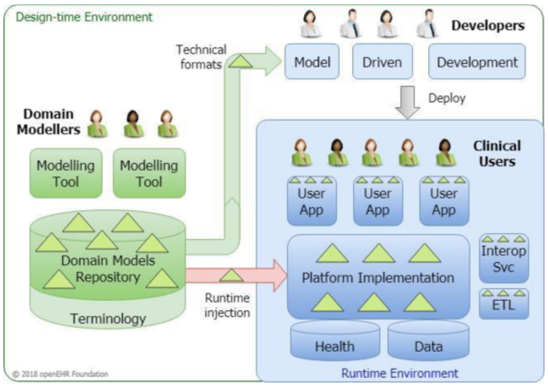

Figure 1- The model-driven openEHR technology ecosystem

Based on the specification, two main distinct development lines were identified that we called: Model Domain and Applicational Domain. This split made it easier to subdivide processes and distribute responsibilities between roles and tools. For each domain, a search was made for successful research projects describing their development process.

The openEHR tools and components to support each step of the process were chosen from the list of options published at the previously mentioned official website, and others of more generic use picked from previous work experience.

Results

Model Domain

In this domain the main artefacts produced were archetypes and templates. These were stored in a local version repository. Templates can be used in runtime, but mainly are to be uploaded to platforms in "Operational Template" (OPT) format. The main stakeholders developing these models were healthcare experts (doctors, nurses, health data analysts). The development process used was [6]:

1. Collect data requirements. For existing systems, this means going through existing forms and for new projects doing roundtables with domain experts, and in both scenarios end up building mind-maps. In this case study, we were aiming for a legacy system upgrade, and it also proved to be a good moment to reorganize and improve data requirements.
2. Search the Clinical Knowledge Manager (CKM). There are already existing archetypes (776 active at April 2021), that mostly match the requirements. CKM provides international governance of the knowledge artifacts, as well as a full life cycle management. This lays the foundation of the interoperability, both semantic and syntactic, capability that openEHR offers, as the same archetype is used in different systems in different countries, all can identify and understand the clinical data since they share the same underlying models.
3. Create new Archetypes or specialize existing. During this step, if no existing archetype was found, a new one was created, and preferably submitted to the CKM to be shared with others. When an archetype partially fulfilled the requirements, it was specialized to accommodate additional fields or constraints, keeping in mind that further constraints can be made when assembling the template.
4. Create/edit Templates. A template is where archetypes are assembled and constrained for context-specific purposes. It will serve as the storage scheme and the basis for the user interface form. For example two templates with the same archetypes can have a different data set (ex: General Practice Visit, Cardiologist Visit), as different parts of the archetypes are constrained in or out of the template.
5. Export templates to OPT format. The operational template format (OPT) is the usable self-contained file (the source .opt template form) only describes references to archetypes and constraints. With this file, the model is complete and ready to be put to use by uploading to an openEHR platform.

The free tools used for each step are described in table 1.

Table 1 – Tools in Model Domain

Step#	Component/Tool	Reason
1	XMind	It is the most popular mindmap tool, free and easy to use
2	CKM https://ckm.openehr.org/ckm/	The international, online clinical knowledge resource
3,4,5	ArchetypeDesigner https://tools.openehr.org/designer	It is the most recent modeling tool, online, better features, free and easy to use

During this study two templates (Birth data and Personal background) were made based on forms of an existing obstetrics EHR system, ObsCare. These forms were chosen because of their different complexity. Birth data contained 99 data inputs from birth date to umbilical cord pH and complications and repeatable sections, and Personal background only 23 simple concepts.

Both were successfully modeled and we then were able to test using the platform developed in the Application Domain.

Application Domain

In this domain, the main focus is to put together low-code development platforms in order to get a functional ecosystem. Apart from specific project related middleware development and component integration, the objective is that after deployment it is possible to build applications based on the template definition. The components used were [8]:

1. openEHR Clinical Data Repository (CDR). This is the main component of an openEHR compliant platform. It is where the clinical records are stored and queried. It should implement the RM and the official REST API, making it possible to be the backbone of the entire ecosystem. The OPT obtained from the model development is uploaded and without further effort, be ready to receive data [9].
2. Middleware development. In this step is where initial development is necessary to make the project specific

ecosystem integrate with each other. For example integrating the openEHR platform with legacy systems, with patient identity source. It is also where additional project specific features can be added or integrated.

3. User interface form generation. This component should be able to generate a usable interface form based on the OPT definition [10].
4. Applications. This is what the users will use in their healthcare setting. In this study there was already ObsCare, which we modified to integrate in this ecosystem.
5. Interoperability services. This component is not openEHR-specific and is often known as "integration bus". It allows the system to communicate with other external systems. In a healthcare setting this communication can be HL7 messages, web services, database access, etc.

Table 2 – Components/tools in Application Domain

Step#	Component/Tool	Reason
1	EHRBase	Good documentation, more compliant REST API, provides SDK for middleware
2	EHRBase Java SDK+*PHP	existing platform client, existing system ObsCare was developed in PHP
3	*PHP	Since no open source component was available we developed our own
4	ObsCare	Existing obstetrics system that provides user authentication and interface for healthcare setting
5	Mirth	Good interface for setting communication channels, relative ease of setup, audit trails, already being used in ObsCare

*programming language used on components developed

The component used as a platform was EHRBase, as it was more compliant with the specification, mainly at the REST API, more lightweight, supports easier deployment through Docker technology and provided open source client SDK.

The middleware developed used the chosen platform Java SDK, and a custom made PHP module to integrate all the components. If our existing platform had been in Java, the SDK could have been used somewhat more efficiently, taking advantage of more of its features, such as runtime object handling.

One of the missing open source components necessary to create a low-code system was a user interface generator. So we developed our own in PHP, parsing the OPT file and generating usable forms for ObsCare. These forms combine HTML with PHP, and can be "themed" with different stylesheets.

For interoperability, we used Mirth, a cross-platform interface engine that was already a part of ObsCare and we also had previous experience in its use.

The end result was a functional integration of these different components and platforms, that allows a domain expert to develop a template, import an OPT, choose a language, from the possible options present at the template, and generate a usable form inside ObsCare. The forms generated are very similar to the previous existing non openEHR forms, in order for existing users to retain familiarity with the interface.

Discussion

The case study allowed us to assess the current reach of the available openEHR open source tools and components, as well as the difficulty implied in their combination for setting up a system that could be used in day-to-day clinical practice.

The model development domain is very well supported, even though it is also less tool demanding. Only three tools are involved in the process, one being optional, while the mind mapping can be done on paper or via other generic tools. Of the two necessary model-related tools, the chosen tool (ArchetypeDesigner) incorporates both archetype and template editing, as well as an integration with a range of versioning repositories (google drive, github, etc).

During the process of developing a template there was a lot of discussion between domain experts, as it is a complex task [4]. Depending on the complexity and detail level of the concept to model, the time consumed varied a lot, but as experience increased, the development also accelerated.

The application development raised more concerns during planning, since most options of available components were developed using Java, and we had an existing system developed in PHP. One of the essential parts of any system is a user interface, however today there is still no open source component in openEHR (i.e. current solutions are all commercial). For the purposes of testing the feasibility of this study, we developed our own. During this development we also felt the need of an artefact that made parsing easier, since the operational template definition is in XML and the structure is complex. A more compact definition in JSON would be easier to handle. A non-essential but very helpful tool would be a visual interface-builder for building the form, since the model is known, a domain expert could build forms block by block to his requirements. This is one of the limitations of our form generator, which is the order in which data is presented. Since the template is built by combining archetypes, the data elements of an archetype are defined next to each other, but at form level it needs to have some flexibility to reorder the inputs to better fit the user experience requirements. Another useful feature is visual query building. This allows domain experts to retrieve data for themselves, for example, or to much easier build queries for specific applications.

There are at least two other useful components we did not find and did not include in our study: Clinical Decision Support (CDS), a component to handle openEHR Task Planning specification providing decision support and process guidance to the ecosystem, and Data Analytics. These will be added to our ecosystem in future work.

The limitations of the study were the lack of testing for scalability and stress due to time constraints.

The proposed guide and the developed system, seems to be a good solution, at least for small projects (with a relatively small number of patients and users) or initial openEHR projects.

Conclusions

In this paper, we demonstrated that it is possible to use the available openEHR open source components to create a low-code ecosystem to build EHR systems. While this form generation approach was successful for the scope of this project, we identified the remaining need for a visual-interface builder tool within the openEHR open source ecosystem, which would enable final visual form design and screen workflow to be specified. Such a tool would be predicated on the kind of generator we developed, but would allow subsequent manual adjustment steps. Even though commercial options are more robust and feature rich, they are also more costly, and currently not easily obtainable for evaluators, small projects or academia. Therefore, an open source option is likely to help the uptake of the platform, and in time can always be replaced without losing any data, thanks to the usage of a standardized openEHR specification.

Secondly, although openEHR has an active supportive community and much detailed documentation, there is currently no general simple step-by-step implementation guide like the one presented in this paper, which we believe will prove useful for concrete application of the openEHR standard to development activities.

Acknowledgements

The authors would like to acknowledge - Project "NORTE-01-0247-FEDER-038393" (ObsCare WCR-Worldwide Maternal and Child Care and Research) is financed by the North Portugal Regional Operational Program (NORTE2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

[1] AHIMA, Assessing and improving EHR Data Quality (Updated), *Journal of AHIMA* ***86*** (2015), 58–64.

[2] C. Meier, A role for data: an observation on empowering stakeholders, *American journal of preventive medicine* **44** (2013), S5–S11.

[3] T. Beale, and S. Heard, Archetype Definitions and Principles (2021), https://specifications.openehr.org/-releases/BASE/Release-1.2.0/architecture_overview.html.

[4] B. Christensen, and G. Ellingsen, Evaluating Model-driven Development for large-scale EHRs through the openEHR approach, *International Journal of Medical Informatics* **89** (2016), 43-54.

[5] S. D. Nelson, G. Del Fiol, H. Hanseler, B. I. Crouch, and M. R. Cummins, Software Prototyping: A Case Report of Refining User Requirements for a Health Information Exchange Dashboard, *Applied clinical informatics* **7** (2016), 22–32.

[6] C. Richardson, and J.R. Rymer, *New Development Platforms Emerge For Customer-Facing Applications*, Forrester: Cambridge, MA, USA, 2014.

[7] L. Min, Q. Tian, X. Lu, and H. Duan, Modeling EHR with the openEHR approach: an exploratory study in China, *BMC medical informatics and decision making* **18** (2018), 75.

[8] B. Haarbrandt, B. Schreiweis, S. Rey, U. Sax, S. Scheithauer, O. Rienhoff, and Et al, HiGHmed - An Open Platform Approach to Enhance Care and Research across Institutional Boundaries. *Methods of information in medicine* **57** (2018), e66–e81. https://doi.org/10.3414/ME18-02-0002

[9] P. Pazos Gutiérrez,. Towards the Implementation of an openEHR-based Open Source EHR Platform (a vision paper), *Studies in health technology and informatics* **216** (2015), 45–49.

[10] G. Kopanitsa, H. Veseli, and V. Yampolsky, Development, implementation and evaluation of an information model for archetype based user responsive medical data visualization, *Journal of biomedical informatics* **55** (2015), 196–205.