Approaches to Learning openEHR: a Qualitative Survey, Observations, and Suggestions

Erik Sundvall[a,b], Dominique Siivonen[a], Håkan Örman[a]

[a]*Department of Biomedical Engineering, Linköping University, Linköping, Sweden*
[b]*Region Östergötland, Linköping, Sweden*

Abstract

Approaches such as ISO 13606 and openEHR aim to address reusability by defining clinical data structures called archetypes and templates, based on a reference model. A problem with these approaches is that parts of them currently are rather difficult to learn. It can be hard to imagine what an archetype-based clinical system combined with modern terminology systems will look like and what consequences different modeling choices have, without seeing and experimenting with an operational system.

This paper reports findings from a survey among openEHR learners and educators combined with observations of related openEHR mailing list discussions. The paper ends with an opinion piece, where we discuss potentially fruitful ways to learn, explore, and extend archetype-based EHR systems using visualization and examples.

The findings highlight potential stumble blocks and solutions and should be of interest for both educators and self-learners.

Keywords: Electronic Health Records; Software; Learning; Standards; openEHR; archetypes

Introduction

Electronic health record interoperability approaches such as ISO 13606 and openEHR aim to address reusability by defining small clinical data structures called *archetypes* and *templates*, based on a *reference model* that can be used as building blocks in different clinical systems. With openEHR being used in domains as diverse as methadone treatment in general practice [1], biobank information management [2], and geriatric home care [3], a growing number of learners have invested time to learn—at variable depth—how archetype-based systems work. Future development and maintenance of such systems at a large scale will require many clinicians and developers familiar with various aspects of these frameworks. The aim of this paper is to explore approaches to teaching and learning openEHR and to suggest ways to make it easier for new-coming system developers and clinical content developers to become productive.

A crucial feature in openEHR is the two-level modeling approach, which separates technical infrastructure concerns and clinical concerns [4]. The technical **Reference Model (RM)** provides the foundational, general building blocks that are then combined, named and used in tree-like data structures according to rules and constraints defined in archetypes and templates. The aim of the RM is to provide common structures for general data that are useful in many clinical settings: configurable data fields, units, time-points, user participations, versioning, etc.

An **archetype** in openEHR and ISO 13606 contains a set of names, rules and constraints describing how to use the RM building blocks to create a data structure that tries to cover all possible aspects (maximal dataset) of a specific well-bounded clinical concept, such as the recording of blood glucose measurements or body weight (including details of measurement method, amount of clothing etc) [4]. These archetypes, expressed using the **Archetype Model (AM)**, can then, for example when used for data entry, be combined into larger structures.

An archetype can also contain language translations so that structured data entered using labels from the archetype in one language can be displayed in another language. (The content of unstructured free-text fields will not be automatically translated by this.)

A **template** in the openEHR sense of the word is used to combine several archetypes into a larger structure intended for a specific use case, for example to be used as the basis for a data entry form in a certain EHR system [4]. A template can also constrain, hide, or set default values in the archetypes and the reference model it builds upon. Templates do not "add" new clinical concepts; they use and constrain concepts defined by existing archetypes.

Every part (node) of an archetype-based data structure in an EHR is addressable and thus retrievable by a **path** containing, among other things, a concatenation of the used archetype IDs

and subsequent node-IDs. Combinations of paths and values can be used in queries to extract and display data in the EHR system [5].

The split between archetypes and templates is primarily for practical and pedagogical reasons; they have different purposes. Archetypes, due to their maximal dataset nature, are supposed to be reusable and created for example nationally or internationally [6]. Templates are intended for more specific and local use cases, for example reflecting local terminology usage, and don't add data with any other paths than the ones available in the archetypes [4]. Thus, data originating from systems using different templates but the same archetype can be retrieved using the same query.

Advanced terminology systems like SNOMED CT and many other biomedical ontologies also aim to address reusability, for example by providing internally cross-linked structures that allow data entry at fine granularity that can be re-represented and interpreted also by other users later using a coarser granularity. This allows shifts of perspective between entry and retrieval [7]. Archetype and template nodes, including fields and field values, can optionally be bound to external terminologies [4], thus adding another level of possibilities but at the same time complex relationships that need thoughtful design and maintenance.

If you by now feel a bit puzzled and confused about the role and usage of RM, archetypes, templates, paths, queries, terminology systems and what difference they all make in practice, then you sense a bit of the commonly occurring learning difficulties that meet people trying to understand and get started with these systems.

Before diving further it's important to realize that the average EHR user does not need to understand more of the system "under the hood" than for current "classical" systems. The RM, archetypes, templates and terminology systems don't need to be shown in user interfaces. Those who need to understand more about the underlying possibilities and constraints of the different system levels are for example software developers, interaction designers, and clinical "super-users" wanting to develop and modify the system and the clinical models used. Policy makers may also need to understand the options and implications of design decisions.

Materials and Methods

This study was conducted as an international qualitative survey via email. One set of questions was sent to eight well-known educators who were actively involved in technical and clinical openEHR development and were all teaching a mixture of audiences from technical, academic, and clinical background. Four educators responded. Another set of questions was sent to the two main openEHR mailing lists (for technical and clinical discussions, respectively), where another four people responded. The learners that responded were from both clinical and technical background but all had some software development experience.

The discussion section combines the survey results with observations from actively following openEHR mailing lists discussions since 2006 and being involved in education, research, and development of related systems.

Results

The **bold** texts below show the questions, and the bullet lists show (spell-checked and shortened) reply samples. Queries to learners are *in italics* and marked with (L). Related questions to educators and learners follow each other when possible.

When teaching, how do you describe the structure and semantics of openEHR?

- I usually start by dividing the specs into RM and AM. Then drill down into different parts of RM and AM, but I think for non-developers it will be too much information in the beginning. A graphic overview of all specs would be very useful with zooming possibility to look at details etc.
- I describe it as a 3 layer model: reference model, archetypes, templates.
- Describe the 2 level model—clinical vs. technical domains; clinicians driving the clinical domain. Classes described, with practical examples.

What activities did you do to learn the structure and semantics of openEHR? (L)

- Looked at example program code
- Partially read/browse the docs on the website, lurk on mailing lists. Engaged in software implementation.
- Read openEHR specs, was involved in producing views of openEHR sample data and that familiarized me with the most common openEHR RM classes.
- Studied the reference documents and existing presentations

If you are using metaphors to describe the structure and semantics of openEHR, describe the metaphors.

- Lego bricks!! I also talk about the need to 'evolve interoperability' whilst maintaining 'bio-diversity' of local practice and content.
- Composition Class aligning with a piece of paper in a paper record. Section aligning with headings on a Word document. Entry Classes of archetype aligning with the clinical tasks done by the clinician. Templates aggregate archetypes and allow them to be constrained to be 'fit for use'.
- We have used the following metaphors to describe archetypes. **Language**: the reference model is like a dictionary of words—not meaningful on its own; archetypes are like meaningful sentences. **Lego**: the reference model is like Lego bricks; archetypes are like Lego model designs (which you see on the paper that comes with the Lego)
- Lego bricks, for example. I also compare to existing EHR systems.

What metaphors and/or examples were used to explain the structure and semantics of openEHR (also describe the metaphors)? (L)

- None, I invented a few myself, for example, a car-dealer/repair keeping track of his sold cars.
- Lego bricks for putting together archetypes into a template, the Lego round thingies being the openEHR archetype slots, (but I don't like this metaphor for more than a very basic understanding of what an archetype is and what it's for). In describing archetypes, I personally like the metaphor of a sculptor with a block of marble taking parts of the maximal dataset (original marble block) away to make a domain concept (sculpture of person).
- **1:** Archetypes as models of USE, needing their own models, next to models of Documentation/Archiving (EN13606-1/openEHR) and models of Knowledge. **2:** The patient system and types of ENTRY classes.

Which are the internal components and processes in the structure and semantics of openEHR that are important to know for understanding how openEHR functions?

- The main process behind the Entries—observation, evaluation, instruction, and action.
- From a clinical perspective, the separation of technical infrastructure and clinical content in archetypes and templates. The way that terminology is used within openEHR.
- Classes of archetypes and how to differentiate between their uses. Features of each class and how they are expressed in the tooling; how they express the clinical content in various example scenarios

Which are the tools you use today to teach the structure and semantics of openEHR? What was the reason that you choose those tools?

- Right now very limited. Design specs in PDFs, a few archetypes in editors, and UML diagrams/PPT slides.
- PowerPoint, practical demos using an archetype editor, template editor
- Tools from Ocean Informatics: Archetype Editor, Template Designer, Terminology Service, Clinical Knowledge Manager (CKM).
- As much visualization of difficult concepts as possible.

What tools and/or resources did you make use of when you learned about the structure and semantics of openEHR? (L)

- Archetype-editor, archetype workbench, example code, and documents
- An archetype editor, now the CKM. Docs/specs on the openEHR.org website
- OpenEHR specs, sample archetypes, even mails on openEHR tech/implementers mailing list to some extent
- Tools by openEHR, Ocean Informatics and LINK-EHR

Which parts of the structure and semantics of openEHR do you think are easy to teach, and which do you think are hard to teach?

- Data types and structures are quite easy to grasp since they are common in other computing platforms. The distributed versioning and participation model is much harder since it's a difficult topic.
- From a clinical perspective, the relationship between archetypes and templates is easy to teach. It is difficult to teach about some aspects of the technical reference model that they do need to understand, e.g., the time attribute in the ACTION class. The difference between state, protocol and data can be difficult, especially when complex timings are involved. The relationship between INSTRUCTION, ACTIVITY and ACTION is complex and difficult to teach. The use of PARTICIPATIONS is also difficult to understand. Overall, understanding exactly how the data are finally recorded is difficult to teach, especially in complex cases as above.
- It is all hard to teach—it is very abstract for non-technical clinicians to grasp, especially archetype development. Templates are easier as taking a 'concrete' archetype and aggregating/modifying is an easier concept and the outcome is related to their clinical experience.

Which parts of the structure and semantics of openEHR do you think are easy to learn, and which are hard to learn for the learners?

- It mirrors the teaching difficulties.
- Observation and Evaluation are hard to learn and still very difficult to apply in real-life examples.
- Difficult: showing how archetypes work in real systems (we have used animated slides to explain it)

What parts of openEHR were difficult to learn? (L)

- Initially, to grasp properly the two levels of modeling. Then to realize that domain models are modeled by constraints rather than by adding data points. Hard for me to put this into words still!
- I found it hard to switch **from** an object-oriented approach where a simple base class is extended more and more to obtain resulting classes that are used in concrete programs, **to** a 2-model approach where one starts with a giant all-encompassing model (archetype) and cut away unneeded properties resulting in concrete classes that can be used in programs.
- I found the openEHR architecture overview document very hard to read; sometimes it wants to explain too much all at the same time, and sometimes terms are used without having been introduced properly (if I recall correctly, things like ENTRY, COMPOSITION, CONTRIBUTION) or without a concrete example. Referring to other documents (which does not work nicely in PDF). Often it is quite abstract and or too mathematical: e.g., as a relative outsider, I hate the use of the word ontology: it does not ring a bell to me at all.
- When to use what type of ENTRY archetype. How to deal with Instruction/Action? How to use the same documentation patterns as much as possible modeling archetypes

What were your strategies to learn the difficult parts of openEHR? (L)

- Never give up
- Ignore them; wait until they become clearer via discussions on mailing lists. It would be so much nicer to have one or two use cases (or patient's travels as they are called for the connect-a-thon) described on various detailed levels and introduce the openEHR concepts while discussing them. Using way more pictures/diagrams.
- Just a lot of work reading. I can't recall specific strategies. The RM class diagrams in the openEHR specs are certainly great to print out while still learning.
- Produced a 130-page document with all lessons learned and thoughts about improvements

Which properties in the structure and semantics of openEHR do you think are the most important to teach (e.g. if time is limited)?

- The process of obs/eval/instruc/action is important to communicate.
- 1) Two-level modelling with archetypes. 2) The documentation process.
- For a clinical audience, an understanding of the relationships between archetypes and templates, and the importance of the 'maximal dataset' approach.
- The two-level model—clinical vs. technical domains; clinicians driving the clinical domain.

If you could design your own tool to teach the structures and semantics of openEHR, what would it do and how would it look?

- The assistance from the tool should be context-specific depending on what in the model you are working with. Some examples from current archetypes would be good to have.
- Be able to show clinically relevant reference model attributes, interaction between archetypes and templates, relationship of INSTRUCTIONS and ACTIONS, and to be able to show how data are actually committed in a clinician-friendly way.
- A tool for engineers (who like the X-ray view of things) would just be a fancier version of the ADL Workbench, or the Valencia teams' tool. For clinical people and teaching, it would be something that visualizes each main Entry type in an intuitive way. Apart from EVALUATION, the other types have a time concept that needs to be visualized (the current Archetype Editor does this in a very simple way—by at least separating it from the structure data part).

Which properties of the structure and semantics of the openEHR would you wish that you could describe with the help of a graphical tool?

- Selecting appropriate RM classes and AM constraints, and a quick overview of the complexity of the models. If the tool can indicate the density of used RM classes by archetypes in a public repository, it will be useful information.
- Participations, Instruction, Actions, History class.
- The instruction state machine could obviously be visualized much more effectively. Also, the data/state/protocol pattern could be visualized. E.g., think of a picture of a human body and an instrument measuring a datum, and then you can imagine how to guide the user to correctly classify the various bits of information. E.g., in OGTT, the glucose value comes from the machine, the fact '1hr post glucose challenge' relates to the subject's body and the brand and other details of the machine are part of the protocol.

Comments

- "I'm not sure if you realize how hard your questions are to answer ;-) The scope of openEHR is huge and complex. The training has to be designed to reflect the audience as I'm sure you'll agree. These answers are brief as it is all I have time for but I hope will help a little—it is not easy to distill the essence of openEHR into even a couple of pages. The real answer is probably much, much longer than that."

Comments (L)

- "I hope you succeed in cutting the vast amount of concepts to learn into sizeable chunks, e.g. via a series of tutorials containing lots of visualizations!"

Discussion

The openEHR platform is a complex framework of design specifications, tools, and clinical models. Not surprisingly, responses indicate that there are easier as well as tougher parts to teach and learn. Tools designed for experienced users are also employed in the learning environment. This may work in some cases, but there is also need for more tailored learning tools.

Approaching openEHR as a beginner

Based on the survey responses and experiences gained in educational and research development projects, we have made some observations of recurring issues and learner reactions and questions.

It is hard and takes time to understand and learn!

It does take time to learn the inner workings of any big EHR system. A system designed to deal with the semantics at the scale of a nationally interconnected system of EHR systems may take even more time [8]. Survey participants indicated that they needed considerable time to learn the openEHR approach. For Swedish technical students doing their Masters thesis, it often takes around 10 weeks to properly get into openEHR fundamentals and then into the subparts of openEHR they need for the thesis using currently available specifications and teaching materials. The amount of time needed for a clinician to get productive in archetype authoring is shorter in most projects, and is usually shortened by the help of being closely guided by an informatician knowing the system very well. Guided training has evolved over the years, but the need for self-study materials and tools does not yet seem to

be adequately met regarding openEHR. This makes the hard task of learning on your own even harder.

Do archetypes allow me to model anything any way I like?

A common beginner misconception is that you safely can model anything you like anyway you like using archetypes. It is true that technically parts of the archetype formalism allow you to build arbitrarily big and complex clusters (tree structures) using building blocks from the RM with nodes that you can name any way you like. However, this does not mean you necessarily should do that.

If you need total freedom, then for example unrestricted object-oriented programming would allow users to model anything any way they want. The drawback is that there is a risk that many users will model similar things in very different ways and thus get incompatible systems. In openEHR, the RM is aimed to capture commonly occurring things in standardized ways in order to minimize unnecessary variation leading to incompatible systems.

Another issue we have seen on openEHR mailing lists (see Appendix) and in projects (including national eHealth activities) is that things such as certain time-points and agent participations that already have well-defined places in the RM have been re-modeled a second time in yet another place as free-form clusters in archetypes. One reason, which was also confirmed by the survey, seems to be that a common entry point for openEHR beginners is to look at archetypes and archetype editing software. It is then easy to think that what you see in an archetype editor corresponds directly to what an entry form in the EHR system will contain.

Archetypes only contain rules and constraints for exactly the part of the RM they are modifying. Every other part of the RM is invisible in an archetype, and in many archetype editor programs. If the archetype is "silent" about something in the RM, only the RM specification "speaks" the rules for that part. Since the RM specifications are technical, detailed, and long, it is not surprising that many parts of them are unknown to novice archetype authors. As a result, users build their mental models of the systems and their possibilities primarily based on example archetypes and oversimplified archetype editors. It is easy to fall into the trap of believing that the structure that you view or create in an archetype should contain every field you will have in a user interface form.

Providing users with tools that support and not mislead is of utmost importance for the openEHR community. For example, such tools would help beginners see archetypes in proper contexts, for instance with traceable RM parts included—parts that in the perspective of a plain archetype would have been hidden and thereby at risk of being remodeled.

Is It Unnecessarily Complicated?

Some consider the modular multilayered approach overly complicated, and if the purpose is to build a fairly static system for a specific setting where data reuse in other contexts is a non-issue, they are most likely right [8]. If, on the other hand, the purpose is to design a national eHealth platform for cooperation, they are likely to need something of considerable complexity [9]. The openEHR approach is developed to be suitable for example

- in a setting where data is reused and possibly aggregated in other contexts than the entry context
- in a system of systems where independent systems need to use and update shared information
- when clinical models change regularly and the system needs to update accordingly. (An archetype-based system can require considerably fewer man-hours to update than traditionally built purely object-oriented systems [10].)

The reusability problem and the benefits of maximal dataset approach with well-defined semantics and paths may not be obvious until one tries to reuse data or use software operating on that data. Writing algorithms to safely convert between different entry formats with mismatched semantics can be very complicated, if at all possible.

This does not mean that openEHR is at a minimal necessary complexity level for its purpose yet, and there are likely still parts that can be simplified without sacrificing functionality. For example, the model behind templates has been simplified and shrunk over time and integrated into the archetype model [11]. There are discussions regarding simplifications of other structures.

Suggestions for Learning Environments and Prototyping Approaches

Technical specifications and UML diagrams are the cornerstones of an archetype-based system, but they easily become obstacles for the novice. To make learning and experimenting with archetype-based systems easier, we need to find alternative ways for those who do not find them a useful, fast, or simple enough way to get started with openEHR and ISO 13606.

Using XML Representations of EHR Instance Data

In the survey, some participants asked for clear clinical examples like some complete "patient journeys". One related learning strategy that has worked rather well for master students and in tutorials for people with XML knowledge is to look at XML structures with openEHR-based EHR content. These show serialized instances of hierarchies of RM objects including references to used archetypes, and show all RM parts used in that particular example. Names of the archetypes and their nodes used for naming and constraints also show up in the XML EHR data. However, RM parts that are not used in that particular clinical example are not visible, which make UML diagrams or specifications needed as a complement. All the alternative constraints and possibilities of the used archetypes are not seen in that XML EHR data either, so access to the archetypes used are needed as a complement.

Graphical Representations of EHR Instance Data

Some openEHR specification documents such as the EHR Information Model [12] contain graphical EHR data instance examples like the one in Figure 1. While these diagrams lower the entry barrier by removing the need to read and understand XML, the same issues of just showing the parts of the RM and archetypes used remain. The available illustrations are useful

tools, but people risk missing these gems in lengthy specification documents.

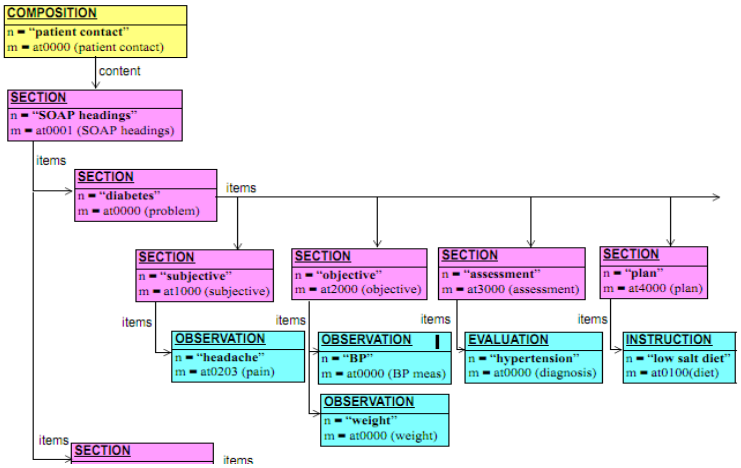

Figure 1- Parts of an explaining diagram from the openEHR specifications (part of figure 15 in the EHR Information Model [12]) © Copyright openEHR Foundation 2001-2016. All rights reserved. www.openEHR.org Reproduced by kind permission of the openEHR Foundation

Interactive Graphical Representations of EHR Instance Data, Allowing Browsing and Manipulation

In order to keep the official openEHR XML serialization (mentioned above) more compact, for example for messaging purposes, unused RM parts are—and should be—omitted and archetype information kept to a minimum. Graphical representations like the one in Figure 1 are also appropriately simplified and show selected parts of the chosen RM pieces. There are several reasons to simplify things, for example to fit models on available paper and to avoid information overload caused by details irrelevant to what one wants to illustrate at the moment.

Showing the entire RM together with archetypes and templates with equal emphasis for an entire example patient's EHR all at the same time risks overloading the user with information and is not likely to be useful. Instead, the visual information-seeking mantra [13] shortened as "overview first, zoom and filter, then details-on-demand" (with five out of seven main steps in bold below) could be applied to browsing archetyped EHR data examples in an interactive learning environment:

- **Overview**: Gain an overview of the contents of an EHR
- **Zoom**: Allow several steps of zooming into the EHR, for example drilling down via Folders to Compositions to Sections to entries like Observations.
- **Filter and details-on-demand**: Allow selection of perspective to filter out or de-emphasize some information. An RM perspective could put emphasis on RM class names (like "OBSERVATION") and also show unused optional RM attributes in the hierarchy. An archetype perspective on the other hand could put emphasis on names derived from the archetypes used (like "headache") and also illustrate unused options available in the archetypes and templates. The hierarchical nature of the EHR data provides for details-on-demand, for example via collapsing and expanding sub-trees.
- **Relate**: There are many relationships in EHR data that may be interactively explored. EHR data is created in relation to certain archetypes and templates which in turn may be related to other archetypes and templates via compositional or specialization relations. Objects can be related via several folders. Different versions of objects are of course related to each other. All this cannot be easily shown at the same time, but could be interactively explored a few at a time.

We have begun (but not published) design of such an interactive browsing environment to be combined with our Educational EHR Environment LiU EEE [14].

Understand Paths, Queries, and Reuse in Model Construction and Data Retrieval

A learning environment should encourage learning about different kinds of reuse enabled by an archetype-based approach.

Capturing clinical requirements for EHR systems can be daunting also in traditional systems not based on archetypes. Constructing maximal datasets like archetypes is even more demanding since it involves collaboration (often international) between different kinds of users in different contexts in order to catch different requirements. As of this writing, the Clinical Knowledge Manager (CKM), the major international collaboration system for archetype development, contains about 300 archetypes in different stages of maturity. A learning environment should encourage the user to reuse as much as possible from existing archetypes as a basis when creating local use-case specific templates (and only create new archetypes for missing things). This kind of reuse can reduce the total work needed to create a usable system—the requirements gathering work is broader than when focusing on a specialized system, but more people are involved to share that load. Reuse is also valuable in the emerging perspective of quality assurance [15].

An archetype-based approach also creates reusable paths that can be used to retrieve data for GUI construction, decision support rule engines, statistical queries, etc. A learning environment should encourage the user to explore and use path-based retrieval, for example via modifying and extending example queries and overviews. UML diagrams of the RM (and the technical specifications to a large extent) are useful when developers are building systems—basing learning in queries and paths instead puts the focus on how to use the system, including extending and modifying it for clinical needs.

Understand the Loose but Necessary Coupling between Interaction Design, User Interface, and the Underlying Semantics

When defining archetypes, focus should primarily be on use and reuse of clinical information, with reuse possibly in another context than that of data entry. However, it is not always necessary to manually fill out all things defined by the archetypes at the point of care, for example things that are obvious from the context of use. Such things could be set as default in templates or as sets of common presets by the system.

In an archetype-based system, the entry form in the user interface is not necessarily equivalent to the underlying semantic model, so it is important to prevent establishment of the false mental model "archetype equals entry form". The semantics of the underlying models will indeed affect data entry, and a change in one is likely to affect the other. Such dependencies have been explored and used by Kashfi [16] in the combination of a user-centered design (UCD) process and the process of archetype-based concept design.

Limitations

This study is limited in several ways. First, the number of respondents is small, and especially a higher number of responding learners could have made the picture more complete. Second, the study is based on spontaneous responses, and we do not know if participants are representative of the community. Third, the questionnaires were sent in January 2010, so answers may not reflect the current situation for example regarding available tools.

Conclusion

The learner and educator experiences reported in the survey can guide newcomers and those who develop supporting software in what to spend extra energy on and some learning traps to avoid. Further effort is needed from the openEHR community to reduce recurring learning issues.

If archetype editors–that often hide parts off the model–continue to be the most accessible tools available to learners then they will likely continue to be a common entry point. Risks associated with that approach include remodelling of existing structures–thus "re-inventing the wheel".

As the number of freely accessible examples of openEHR systems keeps increasing some learning issues are likely to be reduced. Showing the relations between archetypes, templates, queries, user interfaces and patient data instances using examples seems like a promising approach.

An even more streamlined start of the learning process may come from future visualization based learning tools that conveniently allow perspective shifts between for example RM focus, archetype focus and patient data instance focus.

Acknowledgments

We thank all survey responders. We also want to thank the named email list contributors for allowing publication of the excerpts in the Appendix.

References

[1] Xiao L, Cousins G, Courtney B, Hederman L, Fahey T, Dimitrov BD. Developing an electronic health record (EHR) for methadone treatment recording and decision support. BMC Medical Informatics and Decision Making. 2011 Feb 1;11:5. doi: 10.1186/1472-6947-11-5

[2] Späth MB, Grimson J. Applying the archetype approach to the database of a biobank information management system. International Journal of Medical Informatics 2011 Mar;80(3):205-26. doi: 10.1016/j.ijmedinf.2010.11.002

[3] Hägglund M, Chen R, Koch S. Modeling shared care plans using CONTsys and openEHR to support shared homecare of the elderly. Journal of the American Medical Informatics Association. 2011 Jan-Feb;18(1):66-9. doi: 10.1136/jamia.2009.000216

[4] openEHR Architecture Overview. Release-1.0.3 London: The openEHR Foundation; 2015 Available from: http://www.openehr.org/releases/BASE/Release-1.0.3/docs/architecture_overview/architecture_overview.html

[5] Archetype Query Language (AQL), London: The openEHR Foundation; 2015 [cited 2016 Mar 21] Available from: http://openehr.org/releases/QUERY/latest/docs/AQL/AQL.html

[6] Garde S, Knaup P, Hovenga E, Heard S. Towards semantic interoperability for electronic health records. Methods of Information in Medicine. 2007, 46:332-43. doi: 10.1160/ME5001

[7] International Health Terminology Standards Development Organisation. SNOMED CT® User Guide. July 2012 International Release [Internet]. Copenhagen: The Organisation; 2012 Jul 31 [cited 2013 Jan 7] Available from: http://ihtsdo.org/fileadmin/user_upload/doc/download/doc_UserGuide_Current-en-US_INT_20120731.pdf

[8] Arikan S. Is openEHR hard? 2014 Oct 5 [Internet] http://serefarikan.com/2014/10/05/is-openehr-hard/

[9] Beale T. The Health Record - why is it so hard? In: Haux R, Kulikowski C, editors. IMIA Yearbook of Medical Informatics 2005. Stuttgart: Schattauer; 2004. p. 301-4.

[10] Atalag K, Yang HY, Warren J. Assessment of Software Maintainability of openEHR Based Health Information Systems – A Case Study In Endoscopy. Electronic Journal of Health Informatics. 2012; 7:12. Available from: http://www.ejhi.net/ojs/index.php/ejhi/article/view/156

[11] Beale T, editor. Archetype Object Model 2 (AOM2) Specification. Issue 2.0.6. London: The openEHR Foundation; 2016. Available from: http://www.openehr.org/releases/AM/latest/docs/AOM2/AOM2.html

[12] The openEHR Reference Model: EHR Information Model. Release-1.0.3. London: The openEHR Foundation; 2015. Available from http://www.openehr.org/releases/RM/Release-1.0.3/docs/ehr/ehr.html

[13] Shneiderman B. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: Proceedings of the IEEE Symposium on Visual Languages. Washington: IEEE Computer Society Press; 1996. p 336-43.

[14] Sundvall E, Nyström M, Karlsson D, Eneling M, Chen R, Örman H. Applying representational state transfer (REST) architecture to archetype-based electronic health

record systems. BMC Medical Informatics and Decision 2013; 13:57 DOI: 10.1186/1472-6947-13-57

[15] Kalra D, Tapuria A, Austin T, De Moor G. Quality requirements for EHR archetypes. Studies in Health Technology and Informatics. 2012;180:48-52.

[16] Kashfi H. Applying a user centered design methodology in a clinical context. Studies in Health Technology and Informatics. 2010;160(Pt 2):927-31.

Address for correspondence

Erik Sundvall, PhD, Linköping University, erik.sundvall@liu.se

Appendix

Example snippets from mailing lists indicating the value of not hiding too much in archetype editing tools etc. Quoted with permission. The complete conversations are available in the list archives at:

http://www.openehr.org/community/mailinglists

Shortened text is indicated with [...]

From: Peter Gummer
Date: Tue, Jan 18, 2011 at 12:55
Subject: Re: Use of Identifiers in archetypes
To: openEHR technical discussions

[...]

Each LOCATABLE has an attribute called 'feeder_audit', of type FEEDER_AUDIT. Within the FEEDER_AUDIT class, there are lists of DV_IDENTIFIER where systems can store ids generated by the originating system and other systems. The FEEDER_AUDIT also has an attribute called 'original_content', where an image or a reference to the image would be stored.

Because COMPOSITION inherits from LOCATABLE, an obvious place to set the 'feeder_audit' attribute might be on the composition. You could of course prefer to set it on, say, the imaging exam OBSERVATION.

This is an excellent example of something that is already catered for in the reference model, and so it probably shouldn't be modelled in archetypes. Unfortunately, current tools don't make the feeder_audit attribute visible visible to modellers, so they are likely to "reinvent the wheel", unaware that it's already available. (They're designing "wheels" for the "car", but the car already has wheels.)
This is a problem to modellers: an important part of the model that they are designing is to all intents and purposes invisible to them in the archetype.
[...]

From: Peter Gummer
Date: Tue, Jan 18, 2011 at 23:12
Subject: Re: Use of Identifiers in archetypes
To: openEHR technical discussions

> Generally, about FEEDER_AUDIT, it's something I had missed, so
> I'll go and review it, but how does it manifest in the archetype editor?

FEEDER_AUDIT isn't shown in the Archetype Editor at all. It's one of many parts of the reference model invisible within the tools, and so easily overlooked by modellers. As Ian said, there's growing recognition that future tools need to rectify this.
- Peter

From: Heath Frankel
Date: Fri, Feb 22, 2008 at 00:05
Subject: RE: Understanding XML archetypes..
To: openEHR technical discussions

> - The fact that the current tools do not expose or use these
> attributes, is a design decision made by the people writing
> the tools.

Well probably often a "decision" in lack of time/resources or (less likely) lacking ideas of good/useful ways to present them. A tool exposing the RM has to deal with both RM and AM in detail and thus takes more time building than dealing with AM only.

Actually I think it was more to try to keep the task of archetyping simple as it is a task targeted at Domain Experts (Clinicians) without them requiring to know about the RM (well so we thought). Unfortantly, hiding some attributes that are commonly required by the clinician forces them to put it in the archetype so they can see it. We are also finding more and more RM attributes that we want to archetype other than just data structures such as participations.

The challange is to find a visualisation of the archetype that is still simple but can also expand out to include relevant RM attributes.

In Ocean's next generation of tools, mainly inspired by the requirements of the Archetype Query Builder where criteria on RM attributes is common, we will have a configurable tree view of templates where individual RM attributes can be turned on or off, right down to the data type attributes if needed. We are also looking at alternate visualisation of archetypes for the next iteration of the Ocean Archetype Editor.

From: "Erik Sundvall"
Date: Tue, 22 May 2007 09:24:15 +0200
Subject: Re: Point in time 2
To: openEHR clinical discussions

Hi!

On 2007-05-22, Heather Leslie wrote:
Perhaps the apparently 'hidden' reference model stuff should perhaps even be displayed, in an uneditable format, in the Archetype Editor and Template Designer - to make this design process more transparent and help bridge the clinical/technical divide just a little.

This very much matches my point of view. Ideally archetype editors etc should be delivered with a built in mini-EHR system for simple testing purposes (security, scalability etc would not be in focus then). I think such a solution will come from somewhere eventually.
[...]

Included for more context; On 2007-05-22, Heather Leslie wrote:
From my clinician point of view, the average clinical archetyper can only imagine that what they see in the archetype will be what can possibly be displayed on their User Interface. It would be ideal if we can work to make the 'unseen' magic that comes from the reference model clearer, as the UML diagram is (almost) totally unintelligible to others, like me, and even if it can be understood, they may not neccessarily be able to make the leap from the diagram to how it will work in practice (ie a UI).

End of appendix.