A Migration to an openEHR-Based Clinical Application

Hajar KASHFI 1, Olof TORGERSSON
Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Sweden

Abstract. MedView is a suite of clinical applications for recording, retrieving and visualizing patient records, which has been developed and in use for more than ten years. By the introduction of the openEHR architecture, the MedView project started an investigation to migrate from its locally developed framework to openEHR. Issues related to this process, have been discussed in this paper.

Keywords. openEHR, medical application, two-level modelling, archetype

1. Introduction

MedView2 focuses on understanding and implementing the chain “formalize-collect-view-analyse-learn” in oral medicine, in other words, using computing technology to handle clinical information such that clinicians more systematically can learn from gathered data [1]. Several tools have been developed for formalization, acquisition, visualization and analysis and sharing of data. As of January 2009, the main database at the clinic of oral medicine in Gothenburg contains data and digital images from well over 20,000 examinations making it a unique collection of information in its area.

MedView is based on its own locally developed framework. In a time when data sharing is becoming increasingly important, this limits the possibilities for its development and expansion. By the introduction of the openEHR architecture, the MedView project started an investigation to migrate from its framework to openEHR. Motivations, and issues related to this process, have been discussed in this paper.

2. Comparing MedView and openEHR Approaches

openEHR is an EHR development framework that facilitates the level four of interoperability [2]. In a two-level modelling approach, which has been used in openEHR, a system is built from a general model that describes the basics of the system, and provides stable implementation for the parts that do not change often [3]. This stable basic system is then complemented with specialized artefacts that are created from tools or models provided by the basic system.
In openEHR two-level modelling, the first level is the reference model (RM) [2] that describes all basic data structures, overall organization of the EHR, etc. The second level is made up of archetypes, which enables the creation of meaningful domain-models of clinical knowledge from the reference model building blocks. Archetypes specify valid EHR entries, their sequence, and structure. The division of systems into two levels provides for a clear division between the technical and medical parts of the system, and makes it possible to develop the clinical part independently of the system implementation. openEHR-based system architecture consists of four different layers:

- The **Persistence layer** includes EHR, an instance of RM, and is the kernel for openEHR-based system.
- The **Knowledge layer** includes archetypes and templates, which at run time, will be used for data validation and presentation.
- The **Service layer** includes services required for creating, storing and retrieving EHR. Moreover, services for connecting to terminologies and online archetype repositories are in this layer.
- The **Application layer** includes application specific services that access EHR in the persistence layer through the service layer.

![Figure 1. Migration from MedView to openEHR: An architectural perspective](image)

While the reference model of openEHR is rather elaborate including, for instance, details about time and versioning, MedView uses a very simple model. Clinical records are described as being made up of a number of examinations that are connected through a patient identifier. Each examination then consists of a number of sections defining different clinical concepts like “Biopsy” in terms of sets of term/value pairs (see Figure 2). The second level in MedView is made up of the actual definitions of clinical content that is very similar to openEHR conceptually. Clinical content should be defined by clinicians and the medical concepts can change without requiring any modifications to software. MedView architecture consists of these layers (see Figure 1):

- The **Data Layer** includes medical records and related images.
- The **Knowledge Layer** consists of templates\(^3\), terms, and values.
- The **Mediator Layer** mediates between lower layers, and the application layer.

---

\(^3\) To prevent any confusion between MedView templates and openEHR templates, MedView templates are written in Italic.
The Application Layer includes application specific services, which can access medical records through the knowledge and mediator layers.

Motivations for Migration to an openEHR-Based Architecture

In openEHR, archetypes are re-usable maximal models of clinical concepts like blood-pressure examination. An existing archetype can be re-defined or extended through sub-classing. A template can combine several archetypes but also allows modifications like excluding parts of an archetype.

In MedView, a simpler model is used where clinicians define templates by combining agreed upon basic units called terms. For instance, “The type of local anaesthetic” is represented by the term “Anest-type”. When a new notion needs to be introduced, a new term is defined that can then be used in future templates. Figure 2 shows an instance examination entry form generated from a template.

A clear limitation in MedView is that it is impossible to combine several templates to form larger units. In openEHR, the idea of maximal data sets, redefining and extending existing archetypes and combining several archetypes to templates facilitates sharing of clinical data. Further, the quality of data in MedView is not sufficiently high since validation of data is basic. For instance, for the term “Anest-type” two possible values specified in the template are “Citanest-octapressin” and “Xylocian-adrenalin” though clinician at the time of data entry can enter a different value or even leave it empty. That results in data inconsistency, and makes sharing even harder. By using archetypes in openEHR, one can put detailed constraints on data. Data is validated based on corresponding archetypes and the storage of invalid data is not possible.
Finally, MedView combines demographic and clinical information in examinations, which decreases privacy. Hence, we started investigating migration from the current framework to openEHR, which overcomes all deficiencies in MedView.

3. Method and Results of Migration to openEHR-Based MedView

A migration to an openEHR-based MedView consists of several processes (see Figure 1):

- Creating archetypes and templates.
- Translating MedView records to EHRs in openEHR and creating a persistence layer. For this purpose, the most convenient option, XML, is used. Up to now, more than 20,000 clinical examinations have been recorded. Each examination is stored in a text file including term/value pairs. In this translation, a mapping from current terms to nodes in archetypes is required. This has to be created, at least partly, by hand, but the actual translation will be done automatically.
- Developing a service layer to connect applications to the kernel. Java is used for developing required services.
- Extending existing Java-based applications to support openEHR.

4. Discussion and Conclusion

Since MedView is specifically aimed at providing clinicians with useful tools for several tasks, a switch to openEHR must not lead to severe negative effects on any of these tasks. For each application, changes required to make them openEHR-enabled should be investigated.

- **Knowledge Management**: MedView provides tools that are used for creating local terminologies and managing forms for data entry. In moving to openEHR, one could either use the existing openEHR archetype and template editors, or create new ones better adapted to MedView. Because of lack of time, the second option is not a solution for now. However, due to the high complexity of existing openEHR tools, developing more usable editors will probably be considered for the future.

- **Data Collection**: MedRecords automatically generates data entry forms based on XML templates (see Figure 1). In an openEHR MedView, this application should generate GUIs based on archetypes and templates, a non-trivial task. Further, the GUIs must not only be functional, but also have high usability.

- **Clinical Viewing**: MedView makes a clear separation between entering and viewing clinical data. The tool MedSummary, aimed at clinical viewing, uses natural language generation to create a summary of one or more examination records and associated images. The style and contents of the text can be defined by the clinicians themselves, using a special editor. In moving to openEHR, there are no hard technical problems related to using the same kind of presentation. All that is required is knowledge about what kind of data is available, and some software glue retrieving archetype-based data.

- **Analytical Viewing**: In MedView, visual exploration of stored clinical data is possible using mVisualizer [4]. A move to openEHR will require some re-thinking when it comes to visualization tools. This is because mVisualizer
assumes that the basic unit to visualize is sets of term/value pairs. Further investigations will be needed to find a suitable visualization model.

- **Personal Viewing**: In MedView suit, mPhoto helps clinicians to search and browse through images together with their corresponding examination data. A move to openEHR would not mean anything for mPhoto in principle, but a new search mechanism has to be implemented.

### 4.1. Problems Encountered in Using openEHR

During our efforts to migrate to openEHR we have encountered some problems:

- **Complexity of the specifications**: The current stable release of the openEHR specifications is 885 pages. If some relevant specifications are included, the total is 1,016 pages. Additionally, there are UML-diagrams and documentation of code. Getting a grip on this is not an easy task.

- **Complexity of archetypes**: The archetype model is very powerful and allows expressing complicated clinical concepts. It also takes some time to learn and we have not found much documentation aimed at helping clinicians develop archetypes. Further, the existing archetype editors cannot really be said to help inexperienced archetype developers beyond hiding the actual syntax.

- **Lack of implementation**: In our preferred language, Java, the openEHR implementation lacks important parts like templates, persistence, and services.

### 4.2. Conclusion

Our investigation proved that, a migration from traditional MedView to openEHR-based MedView is possible in principle. Moreover, in developing the knowledge layer and persistence layer because of current style of modelling in MedView this migration is straightforward. Nevertheless, more effort is required for the other two layers; especially for the service layer as a result of lack of implementation. Further, for visualization applications a visualization model based on openEHR should be found.

**Acknowledgements.** We would like to thank openEHR community, technical and Java mailing list members for their helpful discussions and supports. The work was funded by the Swedish Governmental Agency for Innovation Systems (VINNOVA), grant 2006-02792.

**References**


