The New Navigators: from Professionals to Patients
R. Baud et al. (Eds.)
IOS Press, 2003

PropeR revisited

Helma van der Linden, Huibert Tange, Jan Talmon, Arie Hasman

University Maastricht, Medical Informatics, NL

Abstract

The PropeR EHR is a multidisciplinary EHR system that is built using existing open source components. This paper discusses the implementation of this system and the advantages and challenges encountered when using open source components.

Keywords

Multidisciplinary EHR, open source, components, OMG HDTF, PIDS, COAS, archetypes

1. Introduction

The PropeR project in the Maastricht region of the Netherlands studies the effect on the quality of care when decision support is used in combination with an electronic health record (EHR). The PropeR project studies this effect in two settings. One of the settings is a primary care setting where patients suffering from a cerebral vascular accident (stroke) receive rehabilitation in their own home from a multi-disciplinary team of primary care providers (PropeR Transmuraal).

In the current situation, the team of care providers uses conventional communication methods like fax, telephone and paper based forms and messages. Therefore an EHR should be introduced first before introducing decision support.

In a previous contribution [1] we discussed the architecture of the EHR we intend to build. One of the requirements was to develop software that is robust and flexible. The software should not only be easily modifiable following changing user needs, but it should also have a life cycle that extends beyond the PropeR project.

Another requirement was to avoid reinventing the wheel and to work with components already available. We wanted to use open source components, not only because of their reduced costs, but also because of the possibility to modify the software according to our needs.

In this paper we want to discuss how this architecture is implemented, the advantages and disadvantages of using open source software, whether this results in a jumpstart of the implementation and in the robustness and flexibility that was required.

2. Architecture and requirements

As stated in [1], we required the PropeR EHR to be based on publicly available, international standards. Reuse of existing software components that implement these standards would reduce the time necessary to write our own software code. We also expect that these components provide better quality software, since testing and trials have already been done.

We decided to use the specifications published by the Object Management Group Healthcare Domain Taskforce (OMG HDTF) [2]. These specifications are defined by a large group of experts coming from international companies. The OMG will only accept a specification as standard if it is accompanied by at least one reference implementation. This ensures that the specification can be implemented in software.

Since the OMG HDTF specifications are not intended to build a complete EHR, but merely offer a standardized interface to distributed collaborating systems, we needed

another building block that defines the relevant medical data structures.

Research showed that an increasingly more accepted approach is a two-model meta-architecture that separates the database model from the data definition itself, thus avoiding the constant modification of the database when changes in the data model are necessary. Thomas Beale of openEHR [3] has elaborated on this in the description of his archetypes [4].

3. Implementation

We used the Person Identification Specification (PIDS) and the Clinical Access Observation Specification (COAS) of the OMG HDTF [5, 6]. These are already implemented in OpenEMed [7], which is commonly seen as a reference implementation and which is available as open source.

As mentioned before, a separate specification of data structures is becoming the preferred approach. We therefore decided that the PropeR EHR would also include a separate data model. Ongoing harmonization efforts between CEN TC251 and HL7 will also include Beale's concepts of archetypes and the accompanying models. This made us decide to implement archetypes in our EHR [4]. Integration of this archetype architecture in the OMG HDTF should be done using the Terminology Query Service[1] (TQS) [8] as a generic interface to the archetype repository.

In summary, our EHR system consists of PIDS servers for handling demographic information, a COAS server for handling medical data, an implementation of the openEHR archetypes, which we will refer to as the Archetype Engine, and several serverside components that interact with the client on the one hand and the PIDS and COAS servers on the other hand. The Archetype Engine will retrieve the archetypes from an Archetype Repository through a TQS server. Below we will go into these servers in detail.

3.1. PIDS servers in the PropeR EHR

The PropeR EHR is concerned with two different populations: patients and care providers. Our original plan was to store information of both populations in a single PIDS server, allowing for the theoretical case where a provider can also be a patient. This would also keep the setup as simple as possible.

We decided against that for three reasons: first, keeping the populations separated on two different PIDS servers would enhance security and privacy. The number of participating providers is small, so the disadvantage of having to enter a provider twice (once as provider, once as a patient) is minor.

Second, this would also give us the opportunity to study the behaviour of a system with two PIDS servers.

Third, there had to be a relationship between the patient and the members of his care team. The OpenEMed software didn't explicitly support this type of relationship, which meant we had to add or extend the existing code.

Before deciding on the implementation of this relationship, we looked at the most appropriate way to do this, given the OMG specifications. This resulted in a choice for the Party Management Facility Specification [9]. This specification describes the management of parties, people and the relevant relationships, which seems to be most appropriate for this case. However, we were unable to find any implemention of this specification that could be used in our system and implementing it ourselves is not feasible given the

[1] TQS is the new name for Lexicon Query Service (LQS).

timeframe of the PropeR project.

Since the OpenEMed software uses HL7 data concepts, we decided to use an appropriate HL7 concept that more or less describes the same relationship: the Patient Relation Information. This concept consists of a set of four items, which in terms of the PIDS would be a composite trait. The OpenEMed PIDS servercode initially didn't support composite traits, which made an extension to the software necessary. We added this code to the server in close collaboration with the OpenEMed authors.

3.2. COAS server in the PropeR EHR

The project provides a multidisciplinary EHR to the care team. They should be able to enter and update data as well as view data provided by the other team members. The COAS specification provides the view functionality. The OpenEMed implementation of the COAS specification already added update functionality, which provided us with enough functionality to use the OpenEMed COAS server as is.

Initially we decided to use various COAS servers to accommodate a connection to the various EHR systems of every discipline but closer study of the data set of the multidisciplinary EHR showed there was virtually no overlap with the EHR systems in use by several of the care providers, which allowed us to simplify the design of the multidisciplinary EHR to one COAS server handling all the relevant information.

3.3. Archetype Engine

To paraphrase a commonly used analogy: when COAS-structures (Observation Data Structure) are considered LEGO© blocks, which can be used to build any structure, archetypes can be considered as the building plans for meaningful LEGO© structures. To take the analogy further: the specifications of the LEGO© blocks show that there are only limited ways to connect two blocks, but allow every structure to be build. The building plan shows how to connect the blocks to build a meaningful structure, e.g. a house. Transposing this to COAS-structures and archetypes: archetypes define medical concepts using COAS-structures to build them. In their simplest form archetypes can be considered as templates for data entry with data validation rules integrated. Filling in such a template generates a COAS-structure containing validated data.

Archetypes refer to ontologies, structuring the domain terminology, for the relevant terms, thus increasing the standardisation of the archetypes.

For this we needed an Archetype Engine, preferably built in Java, the language used for OpenEMed and our own components, which would handle the retrieval of the appropriate archetype, use it for data entry and data validation as well as the retrieval of the requested data. Archetypes should be stored in an Archetype Repository that amongst others provides version management, to allow for various versions of the same archetype due to changed insights in the knowledge of the underlying concept.

Studying the OMG specifications, we concluded that the Terminology Query Service is the most appropriate interface to this Archetype Repository. Research showed that there is currently no implementation of a TQS server available that meets our requirements and since experts in the field generally acknowledge TQS as a very elaborate specification that will be extremely difficult to fully implement, we decided to implement only a downscaled version of the TQS.

3.4. Other Serverside Components

Informal interviews with future users revealed that their computer knowledge varied from almost computer illiterate to an experienced user. This made us decide to implement the client-side as a webapplication using servlets on the backside and Java Server Pages for the

display and data entry. This is implemented using an Apache webserver with a Resin servlet engine communicating with the user over SSL. The servlets communicate with the PIDS and COAS servers through a CORBA ORB (we currently use IONA's Orbacus).

4. Using Open Source

Up front in our requirements for the PropeR EHR system, we decided that we did not want to reinvent wheels. We applied this viewpoint not only to specifications (resulting in the use of the OMG HDTF specifications and the openEHR Archetype Model), but also to the reuse of available software components (OpenEMed). Using open source software would also allow us to modify the software to our needs.

4.1. Advantages

One of the major advantages of using open source software is the price. Usually there is no or merely a nominal fee attached to usage. Open source software is distributed under various licenses. A list with available licenses and their details can be found on http://www.opensource.org/licenses/osl.php. The OpenEMed license[2] states:

"Permission for the redistribution and use of this software, in source and binary forms, with or without modification, is made provided that the following conditions are met. [...] If software is modified to produce derivative works, such modified software should be clearly marked, so as not to confuse it with the original version available from the University of California."

Open source software is mainly based on the idea that the quality of software will improve when others can comment and elaborate on it, resulting in bugfixes and extensions. It has therefore become common in open source communities to return modifications to the authors of the original software, although it is not obligatory. We decided to contribute bugfixes and generic extensions, while modifications only useful in the PropeR context are kept seperate.

This shows another advantage of open source: it is possible to work in close collaboration with the original authors of the software, exchanging ideas and discussing the best implementation of a certain feature. This close collaboration usually resulted in quick turnaround times where bugfixes or extensions were concerned.

It is very common for authors of open source software to be very responsive to suggestions and bugfixes, which increase the motivation to keep using the software.

Another advantage, in this case, is the fact that the original authors are focusing on the implementation of security and authorisation techniques, which allowed us to focus on the building of the parts specific for the PropeR EHR.

4.2. Challenges

Using open source also raises some challenges that may not have occurred when developing all software personally. Some of these challenges include the fact that some features were not implemented, because the original authors did not need them for their own purposes. Also, especially in this case, the code is sometimes difficult to comprehend, not only because the coding style is different and has to be studied, but the combination of Java[3], CORBA and CORBAmed specifications made the learning curve even steeper. Finally, not

[2] The text of the license is part of the OpenEMed source code distribution found at http://www.OpenEMed.org

[3] Prior to this project there was virtually no working knowledge of Java available, thus increasing the complexity of the code.

all the code is as elaborately documented as needed, which was usually solved in an email discussion with the authors.

A final challenge is the continuous update of the original software. This requires keeping the local code in sync with the published version, while carefully retaining the local modifications. This had to be done by hand.

5. Conclusion

Currently the PIDS servers (one for patient information and one for care provider information) are up and running providing retrieval as well as add, update and query functionality. To keep in line with the original PIDS specification we kept the servers fully independent of each other. The Patient Relation Information is stored with the provider information, but a separate component actually creates and follows the link from patient to provider and vice versa.

The question "Did open source software help in jumpstarting the implementation of the PropeR EHR?" can be answered positively. Using the OpenEMed software, even when facing the challenges mentioned earlier, not only reduced the amount of time necessary to develop the software, but also improved the quality of the software and the adherence to the OMG specifications. Informally it could be stated that the PropeR project team was extended with the knowledge and participation of the OpenEMed authors.

This software showed its robustness during the development and use since there has never been any point in time when the PIDS servers nor the COAS server crashed due to software errors. In fact, the only reboots were performed due to external conditions. The Apache webserver has already proven its stability and robustness due to its large market share in worldwide use of webservers.

The flexibility of the software is demonstrated by the fact that we have built an entirely different EHR based on the same components as the original authors of OpenEMed with only minor extensions to the software that were generic enough to be incorporated into the original code. Furthermore, the original authors have already built several EHR systems for several domains based on the same components.

The archetype model, as implemented in the PropeR system, allows for easy adaptation of the EHR system to other domains than stroke, thus providing further flexibility.

6. Acknowledgments

The authors wish to express their gratitude to David Forslund of OpenEMed and Thomas Beale and Peter Schloeffel of OpenEHR for their efforts in clarifying the various concepts.

7. References

[1] H. van der Linden, J.L. Talmon, H. Tange, G. Boers, A. Hasman, An Architecture for a Virtual Electronic Health Record, in: G. Surján, R. Engelbrecht, P. McNair (Eds.) Proc Medical Informatics Europe, Budapest, 2002, pp. 220-225.

[2] anonymous, OMG Healthcare DTF, http://www.omg.org/healthcare, Last accessed: 15 January 2003

[3] anonymous, OpenEHR, http://www.openehr.org, Last accessed: 4 November 2002

[4] T. Beale, A. Goodchild, S. Heard, Design Principles for the EHR, 2.4,

http://www.openehr.org/downloads/design_principles_2_4.pdf, 2002, Last accessed: 4 November 2002

[5] anonymous, Person Identification Service (PIDS) Specification, Version 1.1, http://cgi.omg.org/docs/formal/01-04-04.pdf, 2001, Last accessed: 15 January 2003

[6] anonymous, Clinical Observations Access Service Specification, Version 1.0, http://cgi.omg.org/docs/formal/01-04-06.pdf, 2001, Last accessed: 15 January 2003

[7] anonymous, OpenEMed, http://www.openemed.org, Last accessed: 4 November 2002

[8] anonymous, Lexicon Query Service Specification, Version 1.0, http://cgi.omg.org/docs/formal/00-06-31.pdf, 2000, Last accessed: 15 January 2003

[9] anonymous, Party Management Facility Specification, 1.0, http://cgi.omg.org/docs/formal/01-02-68.pdf, 2001, Last accessed: 4 November 2002

Address for correspondence

Helma van der Linden
University Maastricht
Medical Informatics
POBOX 616
6200 MD Maastricht
The Netherlands
h.vanderlinden@mi.unimaas.nl