

Cooking Up An Open Source EMR For Developing Countries: OpenMRS – A Recipe For Successful Collaboration

Burke W. Mamlin, M.D.^{†‡}, Paul G. Biondich, M.D., M.S.^{†‡}, Ben A. Wolfe[†],
Hamish Fraser, M.B. Ch.B., M.Sc.[§], Darius Jazayeri[§], Christian Allen[§],
Justin Miranda[§], William M. Tierney, M.D.^{†‡}

[†]Regenstrief Institute, Inc., [‡]Indiana University School of Medicine, [§]Partners In Health

ABSTRACT

Millions of people continue to die each year from HIV/AIDS. The majority of infected persons (>95%) live in the developing world. A worthy response to this pandemic will require coordinated, scalable, and flexible information systems. We describe the OpenMRS system, an open source, collaborative effort that can serve as a foundation for EMR development in developing countries. We report our progress to date, lessons learned, and future directions.

INTRODUCTION

Our world continues to be ravaged by a pandemic of epic proportions. Over 40 million people are infected with or dying from HIV. The vast majority of these people (up to 95%) are in developing countries. In 2005, over 3 million people died from AIDS. The brutality of this pandemic demands rapid and coordinated efforts toward prevention and treatment.

In 2004, we (BWM & PGB) began as consultants, asked to scale up an MS Access®-based system in western Kenya. Our response was to design and develop the AMPATH Medical Record System (AMRS).¹

One year later, we began a collaborative effort between teams at Regenstrief Institute in Indianapolis and Partners In Health (PIH) in Boston. Regenstrief was developing the AMRS for an HIV/AIDS project in western Kenya and PIH was supporting existing projects in Peru and Haiti initially focused on TB but expanding into HIV/AIDS.² We quickly learned that our approaches and data models shared more similarities than differences and embarked on a collaborative effort to build a foundation which could be shared by both efforts and, hopefully, fuel a larger collaboration throughout developing countries. We coined the name OpenMRS for our collaborative project.

OPENMRS

OpenMRS (openmrs.org) represents our earnest attempt to create the foundation for collaborative EMR development within developing countries. When we began our work on this project in early 2004, we evaluated other work in this arena. We

believe the overwhelming need for basic clinical data management (often to provide data to funding agencies) along with the need for rapid response in the face of limited technical resources led to many disparate, "stovepipe" efforts which often stored non-coded values and rarely scaled well.

To combat these challenges, OpenMRS aims to provide the foundation and "building blocks" from which fledgling implementations can begin constructing health information systems to meet specific needs. Admittedly, as a fledgling effort, we're just another stovepipe; but we hope that by using freely available tools, employing modular design techniques, and sharing our work, we can seed something bigger. We're excited about our progress to date and the hunger for collaboration that we have found among other upstart initiatives in the field. There are many components which compose a basic OpenMRS implementation.

THE COLLABORATION

OpenMRS is more than just software or a data model. One of the more powerful and exciting aspects of OpenMRS is the collaboration we've enjoyed, initially with PIH and, more recently, with teams like KwaZulu-Natal in South Africa. These relationships have been nurtured, in part, through support from the WHO, the Rockefeller Foundation, and the President's Emergency Plan for AIDS Relief (PEPFAR).

We worked collectively to construct a core application programming interface (API) around the data model and build up a framework for a web-based application running above the API. We leveraged weekly conferences calls, mailing lists, a wiki site, a code versioning system, and project tracking software to manage the collaboration. We have also tried to keep the core group of developers relatively small, especially at the early stages of collaboration.

DATA MODEL

We designed an enterprise-quality data repository modeled upon the lessons learned in the 30-year history of the Regenstrief Medical Record system. The data model is patient-centric and conforms fairly

well to standard HL7 representations of observations, encounters, etc. It incorporates room for internationalization and is tightly constrained to guard against invalid data. At the heart of the data model is a concept dictionary with flexible semantic relationships and significant context-dependent metadata that is used in various ways throughout the application. The OpenMRS data model has been described elsewhere and is available in more detail online at openmrs.org.¹

API

A tightly constrained and scalable relational data model is inherently complex. OpenMRS hides much of the complexity within an API, which greatly reduces the upfront barriers inherent in writing code to realize transactions within the database. Developers need not worry about all of the constraints and details required for manipulating the data model, but rather can reference database query functions such as: “getObservations(*patient*)”.

The OpenMRS API is built in Java with an eye on service-oriented architecture. Its development has been aided by the use of the Eclipse (eclipse.org) IDE for Java, Subversion (subversion.tigris.org) for version control, and Hibernate (hibernate.org) as an object/relational persistence layer between Java and the data model.

WEB APPLICATION

We created a web-based front end for access to and management of the database. This includes a full gamut of content and data management utilities needed to build and maintain the repository – e.g., concept dictionary management, clinical data management, user management, role/privilege management, and form management.

The web application is written in Java, JSP, and HTML, using the Spring Framework (springframework.org) for MVC design and internationalization and DWR (getahead.ltd.uk/dwr) for AJAX-based bridging between client-side JavaScript and server-based Java.*

STANDARDS SUPPORT

OpenMRS uses HL7 as the primary mode of transmitting data between external applications and the repository. This promotes re-usability and interoperability. The system also supports and stores mappings between local concepts and existing standards, such as LOINC, ICD-10, SNOMED, and CPT.

* Asynchronous JavaScript And XML (AJAX) is a web development technique for creating interactive web applications. See <http://en.wikipedia.org/wiki/AJAX>.

DECISION SUPPORT AND REPORTING

The tragic scale of the HIV pandemic demands that increasingly more care be delivered by providers with less training – even, at times, by fellow patients.³ The ability to deliver quality guidance to providers through decision support has never been more critical than in this setting.

We are building a full v2.5 Arden syntax rule builder and interpreter/compiler engine which will be applied through the repository to drive not only alerts and reminders, but also complex concept definitions and research tools. Our goal is for these rules to become a fluid feature of the API – e.g., concepts such as “PEAK FLOW” and “ASTHMATIC” are requested through the same API call, yet “PEAK FLOW” returns a simple observation while “ASTHMATIC” calculates a *derived* concept that represents an amalgam of existing diagnoses, peak flow measurements, etc.

The decision support logic will most often be called during report generation. OpenMRS provides reporting features through patient-level abstracts and aggregate reports. We use the template engine of Velocity (jakarta.apache.org/velocity/) to provide template-based report definition.

OCC

The OpenMRS Concept Co-op (OCC) is a clearinghouse that manages links between all of the vocabularies of OpenMRS participants. The OCC serves as an important starting point for new projects, looking to get a head start on vocabulary development, and will also serve as the platform for mapping concepts between systems and to standardized vocabularies such as LOINC, ICD-10, SNOMED, and CPT.

The OpenMRS concept management interface will integrate calls to the OCC to aid content developers in concept development. Local implementations will be able to publish any concepts they wish to the OCC, contributing their mappings to the co-operative, benefiting from any mappings performed by others, and simplifying the process of sharing data between implementations.

DATA-DRIVEN FORMS

Of course, reports are only as good as the data from which they are derived. Before we can take advantage of OpenMRS’ reporting mechanism, we must get data into the system. OpenMRS approaches data entry with the flexibility and scalability of data-driven forms. Forms are defined as a collection of data pointers – more specifically concepts and/or database attributes (e.g., patient demographics). This collection serves as a schema – in the case of our first data entry module, an XML Schema – describing the

data to be gathered for the form (the form's "model"). How the form's model is used to gather data is up to the data entry application.

Data-driven forms provide the power to create and design forms without programming. Our goal is to take form design out of the programmers' hands and put it into the realm of content management, much as form-generation tools (like [Ruby on Rails](#) or [Plone's Archetypes](#)) aid the developer in rapidly generating forms.

Our first stab at data entry uses Microsoft® Office 2003's InfoPath® tool for form design and form completion. We found the thicker client interface afforded by InfoPath® to be a better experience that was not yet easily matched by browser-based technologies such as AJAX. We were also attracted by InfoPath's ability to save form data as discrete files – a feature we could use to for remote data entry. In any case, not only InfoPath®, but also several other solutions are rapidly evolving toward a richer client experience for web-based applications. Our options for data entry tools will only increase over time.

OPEN SOURCE

OpenMRS comprises, almost entirely, open source components and is itself available for public consumption through open source licensing. We do not advocate open source as an alternative to commercial products; rather, we have chosen open source technologies where they could meet our needs in order to reduce the barriers to adoption. Our goal is to allow projects to visit our website, download the platform, and get started immediately at little to no up-front cost, helping to foster a collaborative development community.

PROGRESS REPORT

When we described our early work at AMIA 2005, we laid down a series of nine design goals.¹ Here, we cover each of our goals and how we've done in meeting them: the successes and the challenges we've faced.

1. *Collaboration* – *systems need to be developed openly and built upon a common infrastructure, the sharing of "best of breed" modules can best occur within a shared platform*

We switched from Zope (a python-based portal that supported the original AMRS) to Java as our development environment. We decided on Java, primarily because the language works well in a collaborative development environment by aiding good coding practices and normalizing coding styles. It also had some practical advantages (PIH's experience in Java, large existing base of potential Java developers in the community). . The teams

discussed and shaped the data model and then constructed an API that effectively wraps the data model within DAO and Services layers. The initial suite of application modules includes a registration and data entry module (FormEntry) along with a HL7 interface from one team, an Arden Syntax module from another team and a reporting module and messaging/event module from yet another team. There has been little redundancy in efforts, which has made collaboration easier thus far.

The collaboration has been greatly aided by regular communication, including weekly conference calls, a mailing list, a wiki, and a simple, yet elegant project management system (edgewall.com/trac/).

While collaboration was easier at first, we've encountered some challenges while scaling up. Changes to the data model are more difficult to implement once multiple implementations are in use. We've also found increasing energy is required to coordinate efforts and align design goals amongst multiple teams as the infrastructure's scope grows. There has also been a tension between bringing more developers on board and getting overwhelmed by the number of voices.

Thus far, we have enjoyed the process of collaboration and OpenMRS has been much better served for it. Our efforts invested in collaboration have already begun paying dividends.

2. *Scalability* – *the infrastructure must not only handle thousands of patients and hundreds of thousands of observations, but also be scalable to tens of thousands of patients and millions of observations*

The best test of scalability is proven usability in the field. Admittedly, OpenMRS is young. Despite its youth, we have entered thousands of patient encounters with a surprising lack of problems. We owe this, in large part, to the fact that OpenMRS' data model design was fashioned from proven, large-scale systems.

Scaling up comes with its own challenges. Large scale systems require enterprise-quality design. For example, as we adopted Hibernate as a persistence layer, we were initially performing our database transactions within the database layer. As we began scaling up, this proved unsustainable and required a re-design to allow for transactions to occur at an application level, above the service layer.

3. *Flexibility* – *systems must support not only HIV-centered care, but also general medical care, since clinical care is not limited to HIV*

To date, our content has focused only on HIV- and TB-specific care. While our content (initial concepts, forms, and reports) are HIV- and TB-centered,

there's nothing about the code or infrastructure that limits us to these domains.

4. *Rapid form design – data collection needs are a moving target; therefore, form design and deployment must allow for continual change*

In most of our sites, clinicians are completing paper forms which are collected and sent for data entry (or entered on site) before returning to the patient's chart. OpenMRS must keep up with changes to existing forms and allow for newly introduced forms. Also, adding this information to the system must not delay revisions to the paper forms. By using data-driven forms within OpenMRS, changes to forms are made without programming, facilitating a rapid response.

One of the early challenges we faced with form design was simply the need to organize and coordinate efforts so that all affected parties were aware of changes. For example, simply adding version numbers to forms and creating an official "Form Approval" committee has helped smooth the process. More challenges lie ahead. As multiple implementations start using forms, there will likely be a desire to share forms between institutions. Since the forms comprise implementation-specific concepts, these components must be "translated" from one implementation to the next. Our hope is the OCC will facilitate the mappings needed for such a process.

5. *Clinically useful – feedback to providers and caregivers is critical – if the system is not clinically useful, it will not be used*

Our first task after getting data into the system was getting it back into the hands of those who generated the data – that is, the clinicians in the field. This prioritization is driven by three beliefs: (1) clinical summaries and decision support can improve patient care (our primary aim), (2) all other parties – including those looking for aggregate data – will best be served by reliable, coded data, and (3) maintaining quality within the data relies, in large part, on getting data back into the hands of those generating the data – i.e., errors in data entry are most likely to be recognized by the providers caring for individual patients.

The challenge will be keeping all parties happy. Clinicians must genuinely recognize the value of the data they're producing and governmental and funding agencies must be satisfied with their interval reports. We believe our best chance of pleasing everyone is through the collection of highly coded, quality data.

6. *Use of standards – to maximize the flexibility and extensibility of the system*

We have adopted several standards within the OpenMRS system. We rely heavily on XML, XSLT

for representation of data outside of the system and for presentation of data. We have also targeted HL7 as a medium for transmission of data into and out of the system. In fact, we're working on a WHO and CDC-sponsored pilot to demonstrate the feasibility of using HL7 and vocabulary standards to transmit a minimum dataset for HIV care between OpenMRS and another independent system (Care-Ware).

The primary challenge for using standards is the simple faith that the additional energy expended in supporting a standard returns values greater than the cost. In an environment where re-usability and collaboration are the rungs to our ladder, we have no choice but to embrace standards whenever possible.

7. *Support high-quality research – via non-ambiguous, coded data*

Nearly all data collected by OpenMRS is coded. While OpenMRS can accept non-coded observations (e.g., narrative reports), we have avoided these in the early phases of data collection, since our primary source of data is checkboxes on paper forms.

Getting quality coded data into the system is highly dependent on the quality of data collected in the field. One of our early challenges was trying to convert ambiguous questions on the existing paper forms into non-ambiguous coded concepts. We used clinicians, researchers, data managers, and informaticians to refine our paper encounter forms, focusing on reducing ambiguity, increasing quality, maximizing the efficiency of data collection. Once the questions and answers on the paper encounter forms were disambiguated, created coded concepts for them was relatively straightforward.

OpenMRS helps content managers avoid ambiguous concepts by providing simple tools during concept development that search for similar concepts or look up definitions online. We expect the OCC to aid in non-ambiguous concept development as well. To aid in research, OpenMRS allows data managers to define cohorts of patients and select observations of interest which can then be exported into a flat file for analysis and reporting purposes.

8. *Web-based with support for intermittent connections – developing countries do not always have reliable power or internet connections, but when available, internet-based technologies offer increased scalability*

While the OpenMRS database and surrounding API do not rely on a web-based front-end, the core application and all modules to date are managed through a web-based interface. All content and interaction is served through SSL connections for security. So, with reasonably fast internet connections, remote data entry can be performed

today over any distance. Support for remote data entry with intermittent or absent internet connectivity is a work in progress. The FormEntry module lends itself to a disconnected environment, since InfoPath provides a mechanism for saving form data directly to a file, but there is still a need for keeping remote patient lists synchronized with the central repository. Mechanisms for doing this are in development.

9. Low cost – if the system is to be widely available and adaptable in developing countries, cost must not prohibit adoption. Ideally, the nuts and bolts of the system should be downloadable for free.

OpenMRS comprises, almost entirely, open source components and is itself available for public consumption through open source licensing. Everything from the database to the web server we use can be downloaded off the internet and used for free. The only aspect of the current system that is not open source and freely available is InfoPath®, which was chosen for features not yet matched within the open source community. Since InfoPath® is a component of Microsoft® Office, it does not pose a significant barrier to adoption.

Although the system is freely available, the cost of content management and expertise needed for adapting the system for various implementations is not trivial. While we do not believe that a turnkey solution is viable, we hope that, as OpenMRS matures, the barriers to adoption will continue to decrease, and the costs to implementation will decrease as well.

LESSONS LEARNED

NOTHING INFORMS DEVELOPMENT MORE THAN LIFE ITSELF

In the initial implementation of the FormEntry module, we had patient searches restricted to either a name lookup or exact identifier. Once we tried using this algorithm with actual encounter forms, we discovered that ~25% of the patients could not be properly identified within the system using data from the paper form. After adjusting the search algorithm, we were quickly able to reduce the failure-to-find rate to near zero. In fact, we were pleased on the first day of training when data entry assistants could not only find the patients easily, but also were able to identify transcription errors in the patient identifier based on search results within the new system.

USE LIBRARIES WITH A LARGE USERBASE

We have found powerful efficiencies through the adoption of popular open source development tools such as Subversion, Eclipse, Hibernate, Spring Framework, and MySQL. With an adequately large user base, online support and documentation is

extended through active user forums and published books.

PARTNERSHIPS CAN CHANGE LIVES

Indiana University and Regenstrief Institute have been immeasurably rewarded by the partnership with Moi University in Eldoret, Kenya and through working with our Kenyan colleagues.

There's an immediate need for partners that have a deeper level of informatics expertise. We need help building and maintaining this infrastructure. We've been asked to scale the early pieces of this project into Tanzania and Uganda, but, as you can imagine, once these sites get in the habit of collecting quality structured data, they're going to want to maximize their workflows by building and customizing multiple applications that re-use the data to serve many different uses and constituencies. Our groups alone cannot scale this effort appropriately. In fact, we believe that each country implementing OpenMRS should have a companion informatics team in a developed country with significant informatics/programming expertise to help dive into the bowels of the OpenMRS source to tailor the system. We're not building a turnkey system by design, because we don't consider it feasible in these types of resource-poor environments.

FUTURE PLANS

The OpenMRS system has been deployed into Eldoret, Kenya and is planned for use in Tanzania and Uganda. PIH is adapting OpenMRS to their needs to serve Rwanda. A team in KwaZulu-Natal is working to implement OpenMRS within South Africa.

Our work has just begun.

ACKNOWLEDGEMENTS

We are indebted to the World Health Organization and PEPFAR for their generous support, and to Dr. Joseph Mamlin for his leadership and sperm. This project was supported by a grant from the Rockefeller Foundation and grant number 1-D43-TW01082 from the Fogarty International Center, National Institutes of Health.

REFERENCES

1. Mamlin BW, Biondich PG. AMPATH Medical Record System (AMRS): Collaborating Toward an EMR for Developing Countries. AMIA Annual Symposium; 2005; 2005.
2. Fraser H, Biondich P, Moodley D, Choi S, Mamlin B, Szolovits P. Implementing electronic medical record systems in developing countries. Informatics in Primary Care. 2005;13:83-95.
3. Wools-Kaloustian K. Community Care Coordinator Protocol. *Unpublished work*. Indiana University; 2006.