FHIRing up OpenMRS: Architecture, Implementation and Real-World Use-Cases in Global Health

Bacher I[1] MA, PhD, Goodrich M[2] BS, Kimaina A[3] BS PhD, Seaton M[2] MS, Faulkenberry G[4] MD MS, Vaish S[5] BS, Flowers J[6] MS, Fraser HS[7] MBChB, MSc

1 Brown Center for Biomedical Informatics, Warren Alpert Medical School of Brown University, Providence, RI, USA
2 Partners In Health, Boston, MA, USA
3 Moi University, Eldoret, Kenya
4 Children's Hospital Of Pennsylvania, Philadelphia, PA, USA
5 Indian Institute of Technology, Gwalior, India
6 iTech, University of Washington, WA, USA
7 Department of Health Systems, Policy and Practice, Brown University School of Public Health, Providence, RI, USA

Abstract

HL7 FHIR was created almost a decade ago and is seeing increasingly wide use in high income settings. Although some initial work was carried out in low and middle income (LMIC) settings there has been little impact until recently. The need for reliable and easy to implement interoperability between health information systems in LMICs is growing with large scale deployments of EHRs, national reporting systems and mHealth applications. The OpenMRS open source EHR has been deployed in more than 44 LMIC with increasing needs for interoperability with other HIS. We describe here the development and deployment of a new FHIR module supporting the latest standards and its use in interoperability with laboratory systems, mHealth applications, pharmacy dispensing system and as a tool for supporting advanced user interface designs. We also show how it facilitates date science projects and deployment of machine leaning based CDSS and precision medicine in LMICs.

Key words

FHIR, EHR, OpenMRS, Global Health Informatics

Introduction

Health information systems (HIS) are increasing central components of successful health systems and provision of high quality care. Over the last 2 decades there has been a rapid growth in the development and deployment of a wide variety of health information systems in low and middle income countries (LMICs). These include EHRs, laboratory information systems, pharmacy systems, national reporting systems such as the District Health Information System DHIS2, and systems for logistics and personnel management. A large and growing proportion of deployed systems are classed as mobile health (mHealth). These include not only specific smartphone apps, but also robust platforms that can be rapidly deployed in community settings at scale such a CommCare, Open Data Kit, and OpenSRP. Many of the most widely deployed HIS in LMICs are developed with open source software and are part of a group of "Global Goods" identified and supported by Digital Square[1]. As well as being freely available and typically locally supported, many of these applications are deployed widely with Commcare being used in more than 80 countries, DHIS2 in 90 countries, and OpenSRP supporting the care of over 150 million patients in 2023.

OpenMRS is a free and open source electronic health record system that is used in over 6000 health facilities in more than 45 low- and middle-income countries supporting the care of almost 20 million patients[2,3,4]. It provides an electronic database of medical records that has been successfully deployed for a wide variety of applications, including routine HIV treatment[refs], emergency situations like the 2015 Ebola crisis, Oncology, mental health care, and as a backbone for hospital information systems and health care facilities. However, there are increasing needs for integrating the EHR with other health information systems. From the very beginning, OpenMRS was built with international health informatics standards in mind, supporting a number of technologies aiming at improving interoperability with healthcare ecosystems. This has included support for various terminology systems including ICD10 and 11, SNOMED-CT, and LOINC. It also supports messaging standards including HL7 V2, and more specialised formats like Integrating the Healthcare Enterprise (IHE)'s Aggregated Data Exchange. Additionally, various OpenMRS implementations have included support for radiology-focused systems integration with the DICOM standard, and an earlier iteration of support for HL7 FHIR. However, overall these technologies have been under-used in the OpenMRS ecosystem and in LMICs more broadly. This may result in the perception that EHRs like OpenMRS are "write-only" HIS.

In 2019, several groups within the larger OpenMRS community, including the co-authors of this paper, decided to begin working together on building a robust module that would implement core parts of HL7's latest FHIR specification. This would allow OpenMRS to export the data it held on patients, and accept in-coming data in FHIR formats. Before working on a new implementation of a FHIR module, we closely evaluated the existing module developed by Kasthurirathne et al. [5]. While it was functional and successfully mapped many FHIR resources to corresponding OpenMRS types, that implementation had some limitations. First, because it used a "strategy pattern" implemented by looking up class names in a table at run time, it was difficult to determine from code which classes were used for which purpose. Second, it did not define "reusable" components so, for example, the implementations of the FHIR Condition resource and the FHIR Observation resource had separate logic for translating between an OpenMRS Concept and a FHIR CodeableConcept datatype, even though the logic was very similar. Third, while OpenMRS itself is quite extensible, meaning that it provides a core data model that modules are able to extend and modify as needed, the FHIR module had no obvious mechanisms to allow for new resources or to ease the creation of custom resource handlers. Fourth, it was deeply tied to OpenMRS's existing service layer which meant that the ways in which resources could be queried were quite limited and hard to document.
The overall goal of the project described here was to develop a new FHIR module that took full advantage of OpenMRS's modularity. It also needed to provide a FHIR REST API that would operate in a way that developers with no familiarity with OpenMRS would be able to easily work with. Finally we aimed to ensure that the module provided the appropriate building blocks for more complicated use-cases involving the exchange of data. We describe here the design and implementation of the FHIR module and several applications at different stages of development and implementation including broad use in health care facilities in Ivory Coast. We also discuss the strong potential for wide use of FHIR in Global Health Informatics and several challenges in adapting the approach in low-income settings.

Table 1: Definitions of FHIR and OpenMRS components and functions etc.

<u>OpenMRS</u>
OpenMRS Concept: OpenMRS's representation of a piece of clinical terminology; often these are mapped to reference terminologies like LOINC or SNOMED, but many concepts are implementation-specific, e.g., a specific question on a form.
OpenMRS Concept Dictionary: OpenMRS's tables of concepts and the mappings of those concepts to other concepts and reference terminologies.
OpenMRS's Obs table: OpenMRS representation of a clinical observation which usually consists of a concept, representing what the observation is and a value. For example, an individual lab result is stored as an observation with the concept for the lab test and a value that corresponds to the type of test.
Spring Framework: A Java framework that provides tools for managing the lifecycle of objects in a Java application. It is widely-used across the Java ecosystem and provides a number of functions that are easy to customise and that accelerate the development of application features.
Spring beans: "Beans" are the basic building block of Spring applications. They are specific Java objects that Spring manages, which means that they are configured using Spring's configuration technology.

FHIR Definitions

FHIR Resource: Resources are the basic building blocks of FHIR. A FHIR Resource is simply a representation of some kind of healthcare entity, e.g., Patient, Practitioner, Observation, MedicationRequest, CareTeam, etc.
FHIR Datatype: Each resource in FHIR contains a series of structured data elements; the allowed data elements are defined by FHIR's datatypes. A datatype explains how a data element is laid out. For example, a FHIR CodeableConcept datatype explains how a codeable concept is represented in FHIR.
FHIR Condition resource: A FHIR resource that represents a clinical condition, problem, or diagnosis.
FHIR Observation resource: A FHIR resource that represents a clinical observation or measurement about a patient or other subject.
FHIR Request pattern: FHIR specifies a few "patterns" for resource's that serve similar types of functions. A request is usually an order and the Request pattern defines fields common to all types of orders, e.g., who is placing the order, who the order is placed on behalf of, the status of the order, etc. These patterns help ensure that FHIR feels "consistent".
FHIR MedicationRequest resource: A FHIR resource that represents an order or request to supply a medication to a patient and instructions for administration.
FHIR CodeableConcept datatype: A FHIR datatype that covers how codeable concepts, such as those from standard terminologies like SNOMED CT or LOINC are represented in a FHIR resource.
HAPI FHIR library: The Java Reference Implementation of the HL7 FHIR standard; it primarily consists of an application programming interface for defining instances of FHIR resources and implementing the FHIR REST API.
FHIR REST API: A core part of the HL7 FHIR standard that describes how to interact with a FHIR Server using the REST architectural patterns.
FHIR Search API: A subset of the FHIR REST API that details specifically how search queries are handled; search queries are any query that uses URL parameters, e.g., */Patient/123* is a request for a Patient with the ID of 123. */Patient/?name=Hernán* is a search query for any patients whose name matches "Hernán".

Methods

One of the principles of the development of OpenMRS is to use existing frameworks and libraries that are standard in a domain as much as we can, so that the "OpenMRS" specific parts of components developed are primarily just the code needed to build a functional EHR. With the new FHIR module, we adhered closely to that principle, making extensive use of the Spring Framework and the HAPI FHIR library to implement the functionality. Since OpenMRS already uses Spring extensively, it was easy to leverage Spring to provide a simpler basis for implementing both a strategy pattern and modularity. Adding a new resource is as easy as creating a couple of relevant Spring beans with appropriate annotations. Overriding the implementation of a resource can be done by simply re-implementing an appropriate interface and adding that class to the Spring context. In essence, this means that working with the new FHIR module is just an extension of working with Spring, which developers working on OpenMRS are almost always familiar with. The HAPI FHIR library made the implementation of the FHIR REST API very straightforward. We simply used HAPI's "plain server" concept to implement the REST facade and plug in the custom logic necessary to translate between FHIR and OpenMRS. The combination of these two libraries meant that the main focus of the module was simply on "translating" between FHIR and OpenMRS.

There are two key ways in which the FHIR module translates between FHIR resources and the OpenMRS data model: first, in the relatively straightforward translation between an OpenMRS object and a corresponding FHIR resource and second, in the somewhat more complicated translation between FHIR's relatively rich search API and queries against the OpenMRS data model. In many cases, the first of these kinds of translations are fairly standard. For example, FHIR has an Observation resource that closely resembles OpenMRS's Observation table. This "translation" was simply a matter of creating a straightforward bidirectional mapping between the FHIR model and OpenMRS's model. In doing these translations, we aim to preserve all of the important attributes of the OpenMRS data model in the corresponding resource and similarly, to try to ensure that any important attributes in the FHIR resource are mapped to the OpenMRS data model, where the data model has an available space for that. For example, OpenMRS does not currently support per-observation reference ranges, and so reference ranges had to be treated as a read-only property.

In other cases, the translation between the OpenMRS data model and the FHIR resource can be quite complex. One key example of this is FHIR's "Encounter" resource. Partially, this derives from FHIR's quite broad notion of an encounter, which is meant to cover most interactions between a patient and a healthcare provider over a variable life-time. For example, in FHIR, an ambulatory visit might be an encounter and a multi-day inpatient stay would also be an encounter. In OpenMRS, we make a distinction between "encounters", which record an interaction between a patient and a health care provider, and “visits” which group one or more encounters. In order to handle this, we needed to support mapping both “visits” and “encounters” to the FHIR Encounter resource. Another issue arises when the OpenMRS data model does not map one-to-one to a FHIR data type. For example, resources following the FHIR “Request” pattern—basically orders—include a required field called “status”, meant to capture the current state of the order or request. OpenMRS also allows tracking the “status” of orders, but this is an alchemy of attributes of the order including: when the order is scheduled to start, when the order is scheduled to end, whether the order has been manually discontinued, and whether a downstream system has updated us as to the status of the order.

These more complicated types of mapping questions were addressed by the team working on the FHIR module through discussion. The developers looked at how things were already working in existing OpenMRS systems, and, where there were differences across implementations and versions, trying to determine how to best accommodate those differences in a robust application.

The second kind of translation was complicated by similar considerations as well as limitations of the OpenMRS system in general. OpenMRS effectively defines dozens of “named queries” which allow the user to specify certain relevant criteria and retrieve results. For example, asking “How many patients are scheduled for Monday?” is generally handled by having an API method that gets the number of scheduled appointments for a given day, and the user just needs to provide the “day” in question. However, the FHIR Search API, as a core part of its purpose, allows a user to express relatively complex relationships. For example, the FHIR query */Patient?given=Hernán* looks for all patient records with the given name of “Hernán” while */Patient?given=Hernán,Emilio* looks for all patient records with the given name of “Hernán” or “Emilio” and */Patient?given=Hernán&given=Emilio* looks for all patient records that have the given name “Hernán” and the given name “Emilio” (In FHIR, the patient’s given name can have multiple values). Similarly, parameters can be chained together so that */Patient?given=Hernán&family=Acevedo* looks for all patient records that have the given name “Hernán” and the family name “Acevedo”. While OpenMRS already had numerous useful queries for searching for patients named “Hernán”, “Acevedo” or even “Hernán Acevedo”, it did not allow the same flexibility that the FHIR search API does.

In order to address this, the new FHIR module includes a data access layer that helps translate FHIR search API queries into queries against the OpenMRS data model. This translation needs to happen in a couple of ways. First, the name of the search parameter needs to be mapped to the corresponding OpenMRS attribute or attributes. Again, these are often straightforward (“given” and “givenName”), but sometimes quite complex like with order status. Second, many search parameters need to be mapped based on their contents. For example, it is common to search for FHIR Observations based on a standard coding system; in FHIR this looks something like *“http://loinc.org|8867-4”*, meaning the code “8867-4” as defined in LOINC. This needs to be decomposed into the appropriate query to find the OpenMRS concept that maps to LOINC 8867-4. Third, any search prefixes need to be taken into account as, for example, the FHIR search *“value-quantity=gt100”* is very different from the *“value-quantity=le100”*. In the OpenMRS FHIR module, these queries are translated into dynamically-generated but safe SQL statements that are then executed against the OpenMRS database. This makes the FHIR Search API into a useful and safe API for running certain ad-hoc queries. This is the basic building block for much of the functionality the FHIR module provides within an OpenMRS application.

Finally, as much as possible, we tried to create helpful abstractions so that adding new resources or translating resources differently can be implemented as simply as possible. This mostly means, for example, that we provide robust, datatype level translations between OpenMRS types and FHIR’s datatypes, e.g., mapping an OpenMRS concept into an FHIR CodeableConcept has a dedicated implementation as does mapping a concept into FHIR’s Coding data type. Earlier work on the OpenMRS FHIR module is described here [6,7,8,9]

Results

Practical Implementation

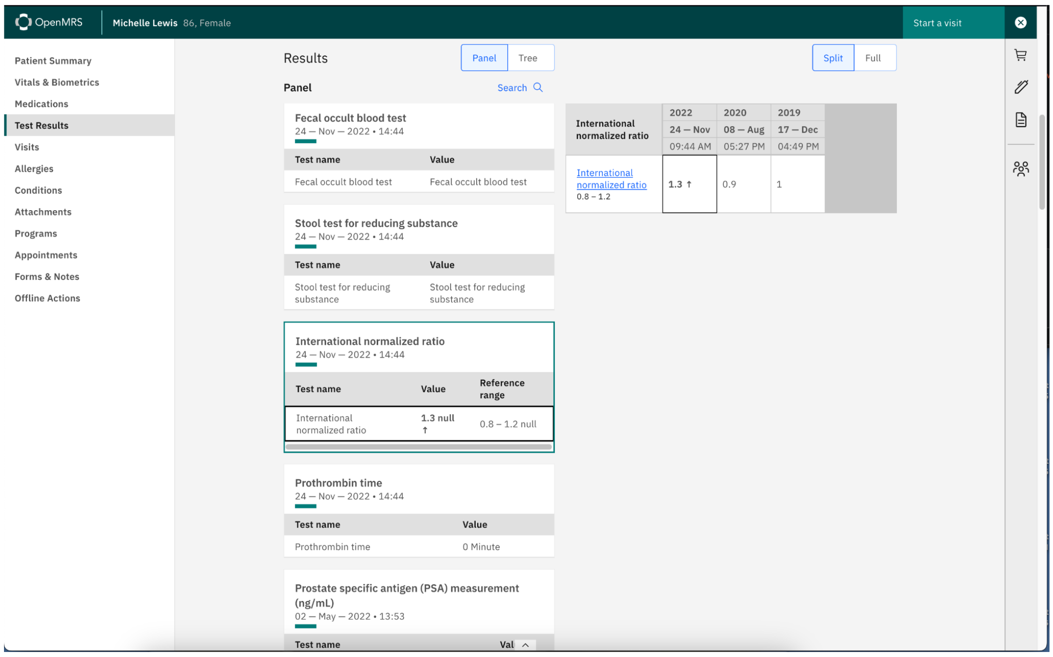
Here we describe three example projects using the new OpenMRS FHIR Module to support aspects of healthcare in LMICs at different strategies of implementation. We also summarise other current and potential uses of this capability in the global health context.

Support for the Redesigned OpenMRS User Interface

Over the past 4 years, the OpenMRS community has undertaken a major project to to deliver a new, more up-to-date interface for the application. Historically, OpenMRS has largely been used as a medical record repository with data entry after the fact. With the new user interface, OpenMRS was also aiming to support clinical data entry and patient management at the point of care. Early on in the process, we decided to make this new user-interface FHIR-first, which typically meant that if we could support a feature through the FHIR API, we would use the FHIR API in preference to OpenMRS's custom API. There were several reasons for this decision. First, FHIR provides a wealth of very clear documentation on the expected data elements, the expected data layout, etc., allowing developers working on the user interface to focus on the features they need rather than worry about API design. Second, the use of FHIR allows a degree of decoupling of the frontend application from the backend, which makes it easier to add certain kinds of extensions to OpenMRS, such as querying secondary FHIR stores.

While the new OpenMRS interface does make extensive use of FHIR and the FHIR API, and while we still aim to develop features FHIR-first, this does not always work out in practice. First, the FHIR documentation, while extensive, is sometimes ambiguous, and intentionally so as some representations are more useful in some contexts than for others. This ambiguity meant that we often still needed to coordinate which data elements can be expected and what values those elements should contain. Second, certain patterns of usage that have grown up in the OpenMRS community over its almost 20 year history are not easily amenable to simple FHIR representations. A common example of this is that OpenMRS uses a concept dictionary to tag the semantic meaning of most data elements stored in the system. This has led to patterns of developing OpenMRS modules that refer to specific concepts that need to be loaded at runtime, e.g., to determine what string to display to a user in their current locale. FHIR does not have a convenient representation of an individual concept, only a concept within either a concept system or a value set. Third, FHIR resources and FHIR queries tend to be very verbose. *This can be a critical issue in Global Health Informatics Projects* as in low-resource settings bandwidth can be at a premium and so concise representations can be essential. While it is possible to create FHIR operations or named queries that can filter the data returned, building out these queries would have added substantially to the time it took to develop features. Similarly, while FHIR has provision for ad-hoc exclusions from returned results via the search API, the syntax for doing so is cumbersome, as it requires specifying every property that should be omitted.

However, while we encountered problems, in several cases, the FHIR API was able to support features that otherwise would have taken more effort. Much of this thanks to the effort we put into building out the FHIR Search API, which allows us to express a number of queries that were not easily expressible in OpenMRS's own REST API. An example of this is a lab results app that we built that dynamically grabs the last 10 results of any lab test performed for a patient. With the OpenMRS REST API, this would require two or more web calls, to load the lab concepts and then load the resulting observations, as well as doing the filtering down to the appropriate observations all in the browser. An OpenMRS native version version was found to take excessive time to display the results. However, the standard FHIR $lastn operation, which retrieves a user-specified n number of operations and uses the FHIR Search API parameters to restrict the returned values turned out to be the perfect fit.



Fecal occult blood test
24 — Nov — 2022 • 14:44

Test name	Value
Fecal occult blood test	Fecal occult blood test

Stool test for reducing substance
24 — Nov — 2022 • 14:44

Test name	Value
Stool test for reducing substance	Stool test for reducing substance

International normalized ratio
24 — Nov — 2022 • 14:44

Test name	Value	Reference range
International normalized ratio	1.3 null ↑	0.8 – 1.2 null

Prothrombin time
24 — Nov — 2022 • 14:44

Test name	Value
Prothrombin time	0 Minute

Prostate specific antigen (PSA) measurement (ng/mL)
02 — May — 2022 • 13:53

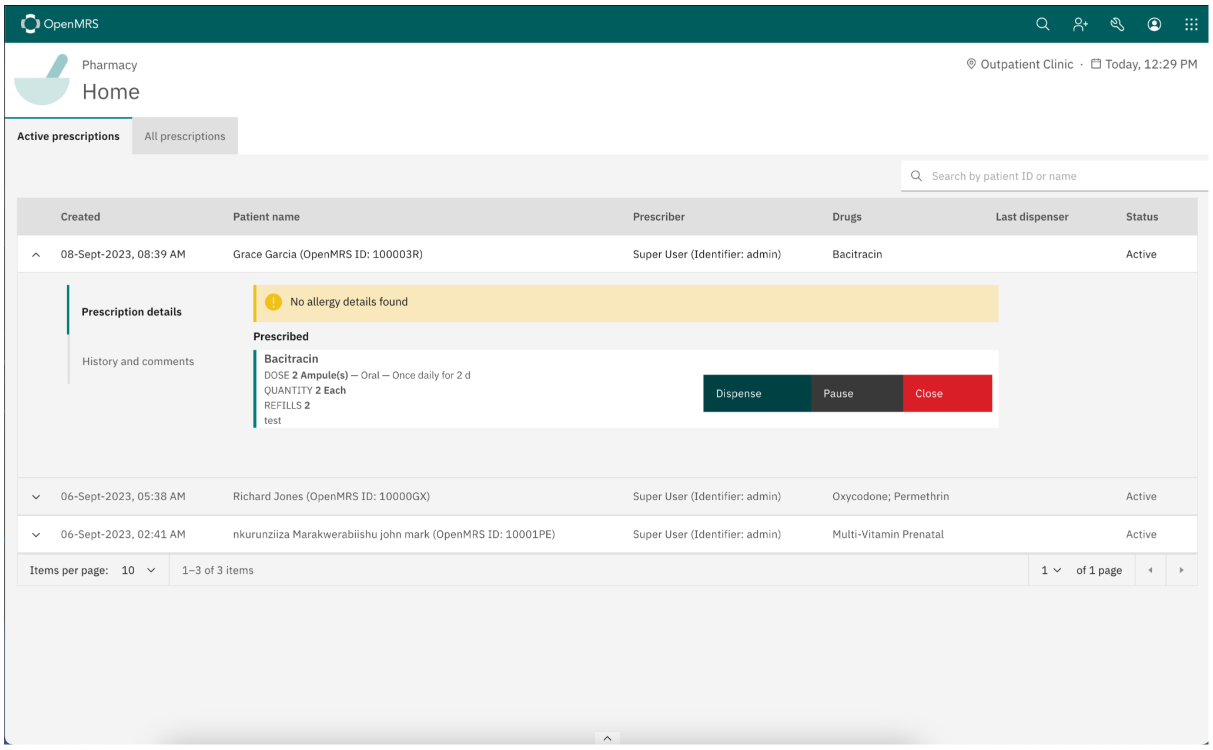
Test name	Val

International normalized ratio	2022 24 — Nov 09:44 AM	2020 08 — Aug 05:27 PM	2019 17 — Dec 04:49 PM
International normalized ratio 0.8 – 1.2	1.3 ↑	0.9	1

Figure 1: The new OpenMRS O3 User interface showing display of lab results through FHIR

Dispensing Application

The OpenMRS Dispensing application was developed to meet the needs of facilities running OpenMRS that have an on-premise pharmacy. Many of these facilities need basic functionality to track the dispensing of medications based on orders but do not require or have the resources to implement a full Pharmacy Management System (PMS). The Dispensing application provides pharmacists with a simple interface that lists all medications ordered in the EMR, allowing them to record and track medications dispensed in response to these orders.



Created	Patient name	Prescriber	Drugs	Last dispenser	Status
08-Sept-2023, 08:39 AM	Grace Garcia (OpenMRS ID: 100003R)	Super User (Identifier: admin)	Bacitracin		Active
06-Sept-2023, 05:38 AM	Richard Jones (OpenMRS ID: 10000GX)	Super User (Identifier: admin)	Oxycodone; Permethrin		Active
06-Sept-2023, 02:41 AM	nkurunziiza Marakwerabiishu john mark (OpenMRS ID: 10001PE)	Super User (Identifier: admin)	Multi-Vitamin Prenatal		Active

No allergy details found

Prescribed

Bacitracin
DOSE 2 Ampule(s) — Oral — Once daily for 2 d
QUANTITY 2 Each
REFILLS 2
test

Items per page: 10 1–3 of 3 items 1 of 1 page

Figure 2: UI of the FHIR enabled dispensing application

When implementing the Dispensing application, we adopted a "FHIR-first" approach. The Dispensing application is written in the new OpenMRS frontend user interface framework, and all communication with the OpenMRS backend (beyond a few system and session data points) occurs via the new FHIR module. Specifically, the Dispensing application requests all OpenMRS Orders as FHIR MedicationRequest resources and submits all dispenses as FHIR MedicationDispense resources. To support this, enhancements were made to the existing FHIR2 Module translator that maps OpenMRS Orders to FHIR Medication Requests. Additionally, OpenMRS Core lacked an existing domain object for dispensing events, so a new Medication Dispense domain object was added to OpenMRS Core, closely modelled after the FHIR MedicationDispense resource. This made it straightforward to add support to the new FHIR module to translate an OpenMRS Medication Dispense domain object into a FHIR MedicationDispense resource.

A key benefit of this approach is that it provides smaller implementations with a simplified dispensing system built directly into OpenMRS while also enabling the development of FHIR-compatible medication request and dispensing support within OpenMRS Core. This facilitates integration with a full PMS for larger implementations that require the rich feature set it provides. Additionally, the FHIR-first approach taken during the development of the Dispensing application offers the potential to retrieve data from non-OpenMRS sources, including other EHRs or mHealth applications with FHIR support. The dispensing system is being implemented by PIH health facilities in Sierra Leone and is planned to be rolled out to other countries, including by separate teams in Ethiopia and Uganda.

Analytics Engine

As described by Kimaina et al.[10], previous work was done to leverage the new FHIR module to support data analytics cases. OpenMRS's data model is very effective for EHR operations, but is inefficient for queries across large numbers of patients. For monitoring and evaluation purposes, we are often able to produce reports, but this reporting process itself can be time consuming. We thus aimed to produce software that would enable both bulk extraction and streaming of data out of OpenMRS and into a FHIR-based data warehouse solution, with the ultimate goal of providing "live" analytics, dashboards, and even CDSS roles.

We ran into several challenges with this effort. First, bulk export was not designed into the new FHIR module. The bulk data extraction had to rely on the standard FHIR Search API, which is not suitable for extracting large volumes of resources. The conversion of OpenMRS data to FHIR Resources in bulk is a time-consuming process and the resulting JSON representations of the FHIR resources were relatively large, both of which made the pipeline slower than required. Unfortunately the open-source analytics technology around FHIR was less actively maintained than expected and we had to rely on FHIR resources in the STU3 format. We needed to use the FHIR Convertor library to convert the R4 resources of the new FHIR module to the older STU3 formats and these translations often involve the loss of key data. More up to date FHIR libraries and resources should improve performance in these examples.

Laboratory Workflow

One of the key goals for the FHIR implementation in OpenMRS is to be able to communicate with other point-of-service healthcare products. A key connection is between an EHR and a laboratory management and information system (LMIS), to automate as much as possible the ordering of lab tests and the timely return of results from the LMIS to the EHR. We worked with the OpenELIS[ref] global team to implement a lab workflow. This took the form of a module for OpenMRS and a HAPI FHIR server that is now a standard part of an OpenELIS install. With this module, we are able to exchange orders as FHIR ServiceRequests from the EHR to OpenELIS, receive back results as FHIR Observations. We can also store those results in the EHR, and track the overall status of an order via a FHIR Task, since a single lab order can result in many actual lab tests. Several strategies were considered for sending and receiving orders and results. Since our initial target for this work was to have it deployed in Haiti in health facilities that often lack reliable internet connections, it was decided to build the service so that each of OpenMRS and OpenELIS would poll for updates from the other and then act on those updates. While this may result in a small delay between placing the order in the EHR and when in appears in the LMIS, in low-connectivity settings it can help to ensure that neither orders or results are missed. This system has been rolled out in Botswana and Côte d'Ivoire. A valuable benefit in these environments is that the FHIR connection makes it possible to set up point-to-point communication between the EHR and LMIS in a matter of hours rather than days it took previously, and that this work can easily be accomplished by on-the-ground teams. Based on this work, a separate team was also able to build out a connection to a different LMIS, Senaite[ref], in a matter of days, despite Senaite not natively supporting FHIR.

FHIR enabled mHealth form tool

As above, supporting FHIR gives OpenMRS the opportunity to leverage existing work using FHIR data, even if it was not specifically designed for OpenMRS. Two such examples have been created as part of the open source project known as FHIR-FLI (FHIR with Flutter Library Integration). Flutter is an open source UI Toolkit from Google written in the programming language called Dart.

The first project is a data capture tool. This tool follows the Structure Data Capture Workflow of FHIR, which falls under the Request Pattern that was mentioned earlier. The first step is actually to create a Questionnaire, which is done using a simple spreadsheet. This lowers the barrier to entry, allowing almost anyone to design a Questionnaire, Survey, or Form, that is then parsed into a FHIR Questionnaire Resource. This Questionnaire can then be distributed to a mobile device or any device that can run a web browser, allowing patients to answer surveys quickly and efficiently. This application has not yet been deployed internationally, but is being used in 2 low resource settings in the United States. Connecticut and New Jersey's Integrated Care for Kids Programs are using this application to screen children that have social care needs, such as homelessness or food insecurity. We believe that this could readily be used by groups such as community health workers for offline, mobile data collection that could then easily integrate into the EHR.

Another example based on the FHIR-FLI architecture, this time done with more of an OpenMRS-Environment in mind focuses on Vaccine Forecasting. Globally it is appreciated how essential immunizations are, but guidance may change country to country (due to factors like following the WHO versus CDC guidelines, or which specific vaccines an organisation has in stock). This application again began with a spreadsheet. The CDC provides spreadsheets with complete immunisation guidelines, but only for the US. However, we were able to parse these spreadsheets into a rules engine that would then accept an immunisation history in FHIR.

Discussion

The initial work on FHIR was mainly exploratory with few active examples of systems being deployed. As we show here the newer work based on the latest FHIR standards has been more successful. OpenMRS provides a good platform for this work as one of the most widely used EHR systems in LMICs. Open source software also has benefits in this work as it allows close inspection of the interfaces and a full understanding of the ways that data is stored and represented. The OpenMRS concept dictionary with it's extensive mapping to coding standards also simplifies semantic interoperability. As described here there are still many challenges to the effective use of FHIR for all applications. We have also identified several areas where the overheads of FHIR and the existing tools can be a major limitation in environments with limited infrastructure and bandwidth.

Use of FHIR in scaling Health Information Systems in LMICs

The work described here shows the progress that has been made in using FHIR to improve the performance, interoperability and robustness of OpenMRS and related HIS. Several other FHIR based GHI projects have recently been deployed. As described in the context of the FHIR enabled mHealth form tool there is a critical need to support seamless data exchange between EHRs and the ubiquitous mHealth applications that support much of the health care provided in LMICs. Previous examples such as CommCare and OpenSRP have used custom APIs to talk to EHRs such as OpenMRS [Motech]. We are now seeing systems like OpenSRP 2.0[ref] natively supporting FHIR. Another major potential benefit of FHIR is in connecting the wide range of innovative point of care tests starting to be deployed in LMICs. At present their potential is not being realized in part because they are not easily linked the patients main health record. In addition recent work has strengthened support for the SMART on FHIR in OpenMRS[ref].

Machine learning based clinical decision support

Although FHIR has been available in stable form for several years, its use in LMICs has been very limited. As the technology matures and there is more experience with its deployment there are other promising developments. Over the last few years, we have seen increasing explorations of using machine learning to help address a number of challenging problems in global health. In these environments health systems need to use what resources they have as efficiently as possible and machine learning can serve as a way to help target them efficiently. For example a key challenge to effective long term HIV treatment is patients who cease care for various reasons. Initial evidence [allan] suggests that machine learning models may be able to successfully predict patients who are likely to experience an interruption in treatment, for example, by analysing patterns in a patient's medication adherence or distance from care.

In order to address such issues, we are working on developing tools for deploying machine learning models that can extract data from OpenMRS in FHIR formats. While raw FHIR representations are unlikely to be the right representation for every machine learning project, having data in a well-documented and well-structured format can help such projects get started and enable us to enrich the EHR data with data from other FHIR-enabled applications. This work complements recent projects extracting data from OpenMRS in the OMOP format[ref] and in future may see the use of FHIR to OMOP converters. Work on the implementation of this system is currently underway in partnership with the AMPATH project and Moi University in Kenya. It will also rely on the FHIR module for communication between OpenMRS and the run time prediction models, and with key UI components. These open source and standards based tools will be available to projects developing and implementing machine learning models in health facilities.

Conclusions

The initial results with the new OpenMRS FHIR module suggest that we will see rapid progress over the next few years in both scaling up existing use cases and development of new and innovative approaches in LMICs. Building more reliable ways of exchanging data and easier ways to establish interoperability should have major impacts on the delivery of care, the development and implementation of machine learning based decision support and the deployment of new point of care testing. These should support improvements in clinical care, precision medicine and data science.

Acknowledgements

This work was funded in part by NIAID R01xxx "Data science for Decision Support in the HIV treatment Cascade". Some of the authors also received funding from USAID and PEPFAR that support clinical care at AMPATH. Work was also supported by a grant from Digital Square.

The authors report no conflicts of interest.

References

1. Digital Square Global Goods Guidebook. 2019, PATH: Seattle.

2. Wolfe, B.A., et al. The OpenMRS system: collaborating toward an open source EMR for developing countries. in AMIA Annual Symposium Proceedings. 2006. American Medical Informatics Association.

3. Allen, C., et al., Experience in implementing the OpenMRS medical record system to support HIV treatment in Rwanda. Studies in health technology and informatics, 2007. 129(1): p. 382.

4. OpenMRS Annual Report. 2021, OpenMRS Inc: Indianapolis, Indiana, USA.

5. Kasthurirathne SN, Mamlin B, Kumara H, Grieve G, Biondich P. Enabling Better Interoperability for HealthCare: Lessons in Developing a Standards Based Application Programing Interface for Electronic Medical Record Systems. *J Med Syst*. 2015;39(11):182. doi:10.1007/s10916-015-0356-6

6. Bacher I, Faulkenberry G, **Fraser HS,** Rasmussen L. Connectathon: Making Open-Source Global Health Systems Talk to Each Other. Workshop, 2021 AMIA Fall Symposium.

7. Bacher I, Mankowski P, White C, Flowers J, **Fraser HS**. A New FHIR-based API for OpenMRS. Poster presented at AMIA Clinical Informatics Conference, May 2021

8. Openmrs 3.0 Documentation https://om.rs/o3docs

9. OpenMRS 3.0 Demo server https://o3.openmrs.org

10. Kimaina A, Dick J, Sadjad B. OpenMRS Analytics Engine: A FHIR Based Approach. *Stud Health Technol Inform*. 2022;290:314-315. doi:10.3233/SHTI220086