Health Informatics Journal

Original Article

Health Informatics Journal
1–15
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/14604582211043920
journals.sagepub.com/home/jhi
SAGE

An HL7 FHIR and GraphQL approach for interoperability between heterogeneous Electronic Health Record systems

Suresh Kumar Mukhiya
Western Norway University of Applied Sciences, Norway

Yngve Lamo
Western Norway University of Applied Sciences, Norway

Abstract
Heterogeneities in data representation and care processes create interoperability complexity among Electronic Health Record systems (EHRs). We can resolve such data and process level heterogeneities by following consistent healthcare standards like Clinical Document Architecture (CDA), OpenEHR, and HL7 FHIR. However, these standards also differ at the structural and implementation level, making interoperability more complex. Hence, there is a need to investigate mechanisms that can resolve data level heterogeneity to achieve semantic data interoperability between heterogeneous systems. As a solution to this, we offer an architecture that utilizes a resource server based on GraphQL and HL7 FHIR that establishes communication between two heterogeneous EHRs. This paper describes how the proposed architecture is implemented to achieve interoperability between two heterogeneous EHRs, HL7 FHIR and OpenMRS. The presented approach establishes secure communication between the EHRs and provides accurate mappings that enable timely health information exchange between EHRs.

Keywords
authorization server, health information exchange, HL7 FHIR, interoperability, OpenMRS, resource server

Corresponding author:
Suresh Kumar Mukhiya, Western Norway University of Applied Sciences, Høgskulen på Vestlandet, Postbox 7030, Bergen 5020, Norway.
Email: skmu@hvl.no

Introduction

Interoperability is acknowledged as an essential factor for the success of HIS. We can achieve interoperability by using uniform standards, outlining the meaning of the information being exchanged.[1] `CDA`, `openEHR`,[2] and `HL7 FHIR` are the common standards for Health Information Exchange (HIE).[1] As shown by the literature study,[1,3–6] `HL7 FHIR` has received interests by both industry and academia. Hence, we chose to use `HL7 FHIR` in our system. `HL7 FHIR` supports REST architecture and SOA for information exchange. However, it inherits the RESTful approach's inflexibility and complexity, such as over-fetching and under fetching.[1] In over-fetching, a REST endpoint provides more data than required. The latter, under-fetching, does not return all the data necessary. In these situations, an application has to make an additional request to fetch all the required data. This addition request is referred to as `n+1` request. To reduce the problems mentioned above, Facebook proposed `GraphQL`[7] query language in 2012. After `GraphQL`'s public release in 2015, companies from diverse areas, including technology (GitHub, GitLab), entertainment (Netflix), marketplace (Airbnb), travel (KLM), finance (PayPal), and others, added `GraphQL` to their development stack. Consequently, there is a rise of architectures, design patterns, development paradigm, experimentation, and frameworks to leverage the benefits of `GraphQL` and support interoperability. A notable characteristic of these architectures and studies is, the communication between systems (for simplicity, consider two systems) mainly fall in one of the following scenarios:

1. *Scenario 1*: A single EHR system that follows a client-server architecture and has several end-users (For example, an independent clinical EHR system).
2. *Scenario 2*: Both EHR systems follow the same health standard (For example, organization A follows `HL7 FHIR` version R4, and organization B follows `HL7 FHIR` version R4).
3. *Scenario 3*: EHR systems follow the same standards but different versions (For example, organization A follows `HL7 FHIR` V3, and organization B follows HL7 version R4).
4. *Scenario 4*: EHR systems follow different health standards (For example, organization A follows `OpenMRS`, and organization B follows `HL7 FHIR`).

Most of the existing studies concentrate on the problem associated with scenario 1 and 2 from different perspectives. Interoperability can be achieved when one or more EHR systems follow the same healthcare standard (as in scenario 1 and 2). However, it becomes challenging when EHR systems[6] differ in healthcare standards (scenario 3 and 4). As a solution to scenario 3 and 4 problems, we propose the exploitation of `GraphQL`-based architecture to achieve interoperability among the heterogeneous system. This paper presents a `GraphQL` based architecture for interoperability among several EHR systems following different healthcare standards. To justify our approach, we developed a prototype where we communicate between OpenMRS and `HL7 FHIR` server. Furthermore, our investigations are accompanied by a non-empirical evaluation based on software quality metrics ISO/IEC 25000.[8]

The study's primary aim is to: (a) establish technical interoperability between two different EHR systems following different healthcare standards, `HL7 FHIR` and `OpenMRS`, (b) report architecture for exploiting `GraphQL` for achieving communication between distinct EHR systems. We report the following novel contribution in this paper: (a) We outline the architecture for exploiting `GraphQL` to achieve interoperability between EHR systems following different healthcare standards, and (b) We present an open-source `GraphQL` powered resource server using the recent version of `HL7 FHIR`.

Background and motivation

The term *interoperability* has a subjective interpretation based on people context and experience.[9] We use the term in context to the definition provided by the IEEE 1990 as:

Interoperability is achieved when two or more systems or the systems' components can interchange data/information and utilise the data/information being interchanged. (IEEE 1990)[10].

The definition incorporates both the *technical interoperability*, that is, the interchange of information and *semantic interoperability*, which is the ability to utilize the information being interchanged. In addition to this technical and semantic interoperability, Tolk et al. proposed seven interoperability layers, including no interoperability, technical interoperability, syntactic interoperability, semantic interoperability, pragmatic interoperability, dynamic interoperability, and finally, conceptual interoperability.[11] Gibbons et al.[12] introduced *clinical interoperability*, which refers to the actual usage of information in the clinical process workflow. In this study, we mainly focus on achieving technical interoperability (by following `HL7 FHIR` and `GraphQL`) and semantic interoperability (by following common healthcare IT standards `SNOMED CT,` and `LOINC`).

Why do we need common healthcare IT standards for interoperability?

Healthcare IT Standards are a set of agreed-upon representations of data and methods for communication between Healthcare Information Systems (HIS). These standards incorporate agreement for communication among HIS. Such agreement includes: (a) mechanisms for enforcing secured communication, (b) structure of format of data structure, (c) definitions of common terminologies, (d) description and architecture of the communication protocols, and (e) comprehensive documentation. Examples of such Healthcare IT standards include `HL7 FHIR` and `OpenEHR`. We need such healthcare IT standards for the following reasons:

- *Lack of regulator*: There is no one to regulate with power to ensure healthcare deployment happen systematically. Consequently, a custom version of EHR systems and healthcare standards are followed. The obstacle with these custom standards is not that there are so diverse to pick from, but we have failed to adopt the currently available ones.[13]
- *Complex standards*: The available standards have been complicated and expensive to implement, which lead to the development of `HL7 FHIR`.
- *Volume of data*: There are massive events in the healthcare ecosystem, including patient admission, prognosis, pre-assessment, diagnosis, monitoring, follow up and others. These events show the characteristics of big data. Hence, it requires proper attention to be processed, stored, and accessed.
- *Interfacing with other EHRs*: The number of API links required to connect `n` EHR systems increments according to the formula[13]:

$$Number\ of\ \ API\ interfaces = \frac{n\,(n-1)}{2} = \binom{N}{2}$$

Hence, connecting only two EHRs require a single API interface. Linking 5 EHR systems would require ten distinct API interfaces and connecting hundred nodes needs 4950 interfaces.[13] Such growth in connecting different EHRs is referred to as the combinatorial explosion.

- *Translation*: Interoperability at the root level transfers data/information from system A (sender) to B (receiver). If both sender and receiver adhere to the same standard, no translation is required. However, EHR systems have been used before these standards came into existence. Many of these EHR systems are wrapped in health providers protected firewall and follow traditional or custom standards. In order to communicate with the external EHR systems, the sender data has to translate to the format understood by receiver and vice versa. `HL7 FHIR` provides such a common standard to avoid such translation.

REST versus GraphQL

Both REST and `GraphQL` are the framework for developing *Web Services (WS)*. Traditionally, there are two WS-protocols: *SOAP*[14] and *REST*.[15] The SOAP protocol requires specification of all available methods, including input and out data structures in a schema. In contrast, the REST uses information encoded in the `URL` and `HTTP` method to describe the service interface. The main disadvantage of SOAP is seen in its complexity and the overhead imposed through the explicit schema.[4] REST, on the other hand, is more lightweight and relies on the developer discipline. `GraphQL` is an alternative approach to WS, developed by Facebook. Every server interface using `GraphQL` has to define its underlying domain model in a schema representing a graph. Such a definition is given a textual *Schema Definition Language* (SDL). As aforementioned, `GraphQL` improves the problems[1] associated with the REST approach.

HL7 FHIR versus OpenEHR

Readers must note that OpenMRS is an EHR system, whereas OpenEHR is a healthcare IT standards. `HL7 FHIR` is one of the popular healthcare standards created for web standards like `XML, JSON, HTTP`, and `OAuth`. The standard incorporates a modular set of components referred to as Resources such as Patient, Observation, Careflow, Organization, and others. `HL7 FHIR` supports a coding system that is defined by `SNOMED-CT, LOINC, RxNorm, ICD 10` family, and others.[16] `HL7 FHIR` supports RESTful communication and SOA architecture. OpenEHR[17] is an alternative to `HL7 FHIR` that uses over 300 more complex Archetypes created to yield a maximal number of data items.

Related works

There is not much work done to establish interoperability between heterogeneous system. However, to support interoperability, the OpenMRS community are developing an `HL7 FHIR` module that can utilize `HL7 FHIR` resources and communicate with the OpenMRS.[18] However, the FHIR modules are not complete and do not support all the `HL7 FHIR` resources yet. Consequently, there is a need to build a method to communicate between OpenMRS and other EHR systems. In the study done by Kasthurirathne et al.,[18] the authors introduce the FHIR module recently developed for OpenMRS to show how the newly developed endpoints can be used to communicate with `HL7 FHIR` based resource server. Gavrilov et al.[19] presents a design of a healthcare warehouse to achieve technical and organizational interoperability. Similar to our proposed architecture, their framework communicates over SOA. However, unlike their study, we present open-source RS based on `HL7 FHIR` and present architecture to establish communication between two or more EHR systems following a completely different healthcare standard.

In the study by Mukhiya et al.,[1] the authors proposed architecture for communicating between EHR systems following the same standard. Besides, we presented an empirical evaluation of the use of the REST approach with `GraphQL` endpoints. The result indicated that `GraphQL`-based endpoints has better performance, offers scalability, and are economical compared to REST. This paper extends experimentation of the work by Mukhiya et al.[1] Unlike their study, we aim to establish communication between two heterogeneous EHRs.

Solution

This section outlines the architecture of the proposed system for establishing communication between heterogeneous EHR systems. An obvious question would be, *"why do we need to communicate between OpenMRS and* `HL7 FHIR` *resource server?"* The answer is a continuous evolution. All the current EHR records are in the OpenMRS server, and we envision supporting interoperability by using `HL7 FHIR` resource server as most of the healthcare systems are adopting `HL7 FHIR`.[1,3,4] As aforementioned, the lack of regulation gives healthcare providers the freedom to choose their EHR system based on their technical, economic, legal, operational, and system feasibility study. So, the need for such inter EHR system communication is a common problem. Hence, in this architecture, we maintain the existing system (OpenMRS) and establish communication with the `HL7 FHIR` Resource server. `HL7 FHIR` Resource server is also responsible for sharing endpoint with other EHR systems in case of external provider communication.

Architecture

The multi-level model illustrated in Figure 1 outlines the communication between OpenMRS and Resource server based on `HL7 FHIR`. The top left block in Figure 1 represents an RS based on `HL7 FHIR`, which consists of `GraphQL` endpoint guarded by Authorizations Server. Each `HL7 FHIR` resources (like *Patient, Encounter, Observation*, and others) have their associated service and strategic resolvers for intercepting the valid requests and translating the resources from OpenMRS format to `HL7 FHIR` format. The top right block in Figure 1 represents an instance of the Resource server.

The bottom blocks of Figure 1 represents OpenMRS layers and its instance. As aforementioned, OpenMRS has a built-in Authorization server. The `GraphQL` endpoint captures any requests made to the Resource Server. The Authorization Server guards the endpoint for authentication and authorization purposes. The valid requests are then forwarded to associated Resource services. For example, for a valid request to write patient resource (POST HTTP request), `PatientService` captures the requests and forwards to `PatientStrategyResolver` which translates the requests body (patients attributes) from OpenMRS to `HL7 FHIR` format. The translated format is then used by request to communicate with the database service.

Experimental setup

This section explains the main components of the recommended architecture, as depicted in Figure 2. In addition, we outline how these components communicate with each other.

1. *OpenMRS server*: OpenMRS[17] is an open-source, configurable EHR system programmed in Java, JavaScript and several open-source components including `MySQL`, Apache Tomcat, Hibernate, and others. It represents data in `XML` and `XML` Forms format and uses a relational database with `SQL`. The architecture of OpenMRS is presented in Seebregts et al.[20] for the interested readers.

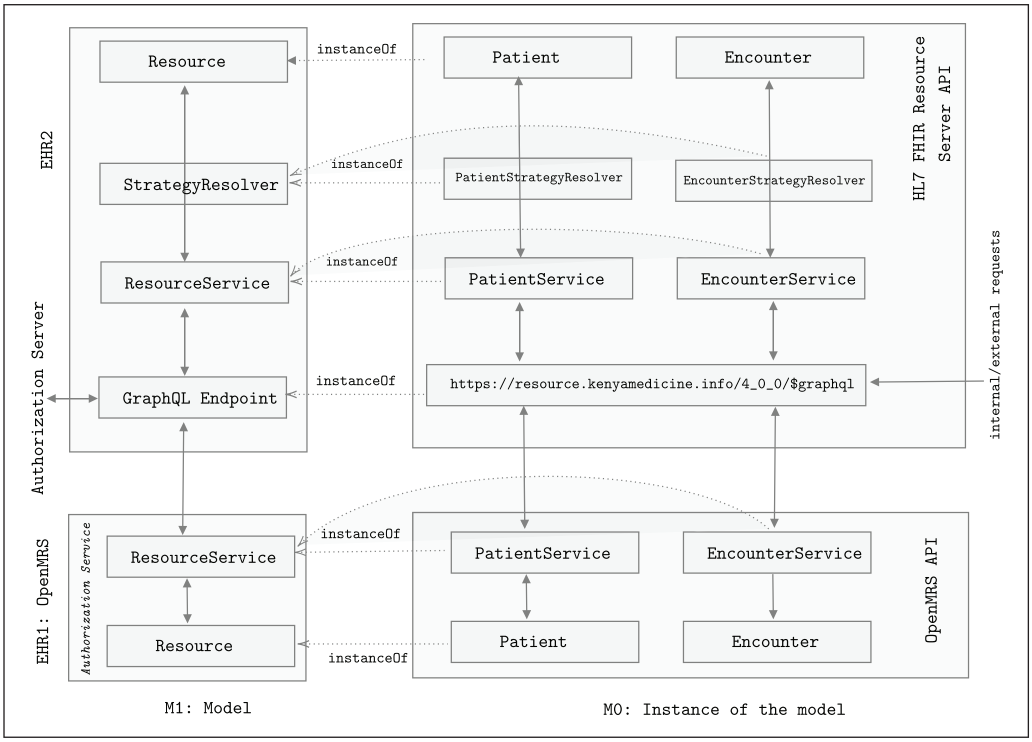

Figure 1. Multi-level model illustrating communication between OpenMRS API and HL7 FHIR Resource server.

2. *Resource Server*: It is a web server complying with `HL7 FHIR` healthcare standard and can communicate using REST API. The communication is established when an authorization Server grants a valid `access tokens`. In SOA, terms, a resource server is a service provider.[21] The available open-sourced resource servers are REST-based, and some non-open sourced are proprietary to the best of our knowledge. We intend to leverage the benefit of `GraphQL` based approach, and hence we initiated an open-source resource server powered by `HL7 FHIR` and `GraphQL`. The resource server is available for use under MIT license. The GitHub links for downloading the source code for all the components are given in supplementary resources.
3. *Authorization Server*: The authorization server is a web server based on `OpenID connect`[22] and has a function to authenticate and authorize users. Besides, it is responsible for managing the clients' scope, permission, and introspect token for its validity. Our prototype consists of a standalone authorization server; however, any the architecture is compatible with any third-party authorization server provided they are based on `OpenID connect` and can grant valid `access tokens`.
4. *FHIR patient app*: FHIR patient app can be mobile-application or web application that assist in data interchange between wearable sensors and resource server. Moreover, we can utilize such a patient app to create a user interface for the patient to provide information or accumulate information. In SOA, both mobile and web application corresponds to the service requester/consumer components.[21] In our implementation, the FHIR patient

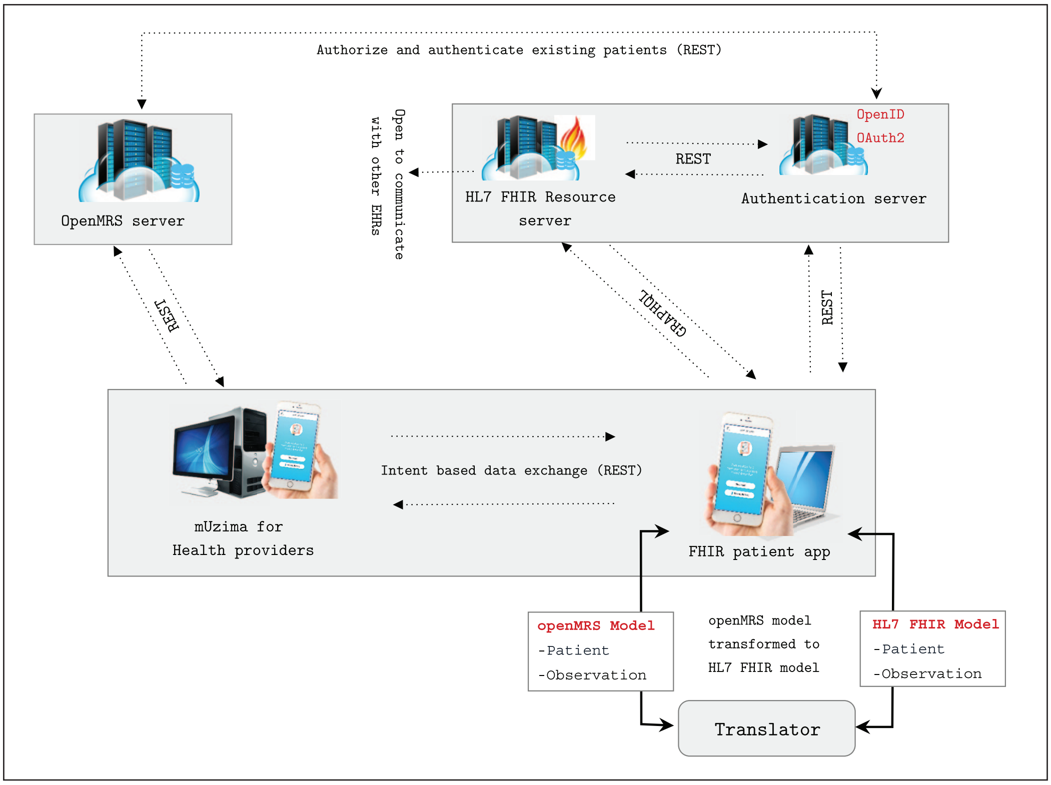
Authorize and authenticate existing patients (REST)
OpenMRS server
Open to communicate with other EHRs
HL7 FHIR Resource server
REST
OpenID
OAuth2
Authentication server
REST
GRAPHQL
REST
mUzima for Health providers
Intent based data exchange (REST)
FHIR patient app
openMRS Model
-Patient
-Observation
openMRS model transformed to HL7 FHIR model
HL7 FHIR Model
-Patient
-Observation
Translator

Figure 2. Our set up establishing communication between OpenMRS and resource server based on HL7 FHIR and GraphQL.

application uses a middleman service to translate OpenMRS entities to HL7 FHIR entities and vice versa. We can delegate such translation to a separate service similar to the broker-architecture pattern.[23] However, we chose to implement a middleman service for translation inside the FHIR patient mobile application for the initial prototype.

5. *mUzima for health providers*: mUzima for health providers[24] is a client that presents interfaces authentication, and authorization to the healthcare provider. The muZima health provider application provides abilities to create resources such as patients, observations, clinical process, and others for their users and present them in a dashboard, associated with their progress, and activities.[25]

Communication flow

In our architecture, there are two types of communication: (a) between OpenMRS server and HL7 FHIR Resource server, (b) among HL7 FHIR Resource server, authentication server, and FHIR patient app.

Communication among HL7 FHIR Resource server (RS), authentication server, and FHIR patient app: The communication among HL7 FHIR Resource server, authorization server and FHIR client follow the same workflow as mentioned in the study.[21] This communication follows contextless handshaking recommended by SMART on FHIR[26] and involves the following steps:

1. The FHIR patient app makes conformance `HTTP` request call to the RS.
2. The RS responds back by providing valid conformance endpoints (for example,/token,/authorize, /manage).
3. The FHIR patient app attempts to authenticate by sending valid scopes, headers, and payloads to the authorization server.
4. Once the FHIR patient app is successfully authenticated with the Authorization server; they are redirected back to the app. In this process, the FHIR patient app receives an `Access Token`, a long string of characters that serve as credentials in the next step.
5. The FHIR patient app then creates a request to the RS with access tokens provided by the Authorization Server. An example of the `HTTP` request is shown below (for readability purpose, the token has been shortened):

```
GET/Observation?subject=Patient/987654

Accept: application/fhir+json

Authorization: Bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6
```

6. The RS makes token introspection to verify the users' scope, grants and permissions. The request is only successful, if the user requesting the information has valid access, grant types and more importantly, correct access token. The following is an example request made by an `HL7 FHIR` Resourcer server to an Authorization Server. Note the header `Authorization` whose value is the string `Bearer` followed by the `Access Token`.

```
POST/oauth/token/introspect

Accept: application/json

Content-Type: application/x-www-form-urlencoded

Authorization: Bearer ImlhdCI6MTYwMDcwMDczOX0.e95nuufbE7lBrWkpRrNuJu
PAsW5

token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6
```

7. Provided the request has valid payloads; the authorization server returns with scopes, permissions, token type and token expiration. It throws an exception, otherwise. The following is an example response made by an Authorization Server that indicates the user has been approved for `observation` read scope, the expiration date of the access token, and other meta information.

```
{

"active": true,

"scope": "observation/*.read online_access openid profile",

"client_id": "growth-chart",

"token_type": "Bearer",

"exp": 1316269159,
"iat": 1316269109,

"iss": "https://auths.kenyamedicine.info"
```

8. With valid scope and permission, the RS performs a DB query.
9. Assuming DB has been configured, and correctly connected, it returns with asked resources.
10. Finally, the RS sends the requested resources to the FHIR patient app.

Communication between OpenMRS server and HL7 FHIR Resource server: OpenMRS comes with an inbuilt authorization server. The mUzima for health provider application[24] communicates with the OpenMRS server via a REST interface and the inbuilt authorization server. Here is a point to note: OpenMRS server comes with REST communication and suffers from the challenges mentioned earlier. To overcome such challenges, we envision moving toward the `GraphQL` approach, and therefore this architecture is used. To comprehend the communication between these components, let us say a patient needs to create an observation resource in the `HL7 FHIR` Resource server. In such a case, there are two actions:

1. A patient data model in the OpenMRS and `HL7 FHIR` is not the same. There can be the following situations: (a) some attributes present in the OpenMRS data model is not present in the `HL7 FHIR` Resource server and vice versa, (b) different naming for the same attributes, (c) different data types and structure for the same attributes. To solve this situation, we introduced translator service inside the FHIR patient app that translates one model to another.
2. Once the patient is validated, the observation resources need to be created in the FHIR Resource server.

The Authorization Server authenticates the FHIR patient app, making a REST call to the OpenMRS authentication service to check the patient details. If the patient is valid, the authorization server returns the requested resources to the FHIR patient app. The app then sends requests to mUzima for the Health provider app for the patient detail and gets the patient details back as the response. The patient model returned by the mUzima health provider app has the OpenMRS patient model and is translated to `HL7 FHIR` by the translator service inside the FHIR patient app. The translated data is then sent to the `HL7 FHIR` RS with the valid access token. With a valid access token, the `HL7 FHIR` resource server saves the patient information inside its database. Once the patient information is successfully saved into the `HL7 FHIR` Resource server, the following action is to create the observation resources. The observation information is encoded into the `HL7 FHIR` format. The request is made to the RS using the flow mentioned in the previous section, that is, contextless handshaking recommended by SMART on FHIR.[26]

Experiment results

The primary artifact of this paper is the software architecture model, which envisions establish communication between the OpenMRS server and the Resource Server. To evaluate the architecture, we conducted an experiment where we sent several queries to read, write, update, and delete (CRUD) `HL7 FHIR` resources. This section presents 11 queries (Q1, Q2), each indicating two types of communications: (a) C1: Query by the mobile client to the RS and (b) C2: Query by the mobile client to the OpenMRS server. We executed each query 10 times and recorded the response time and response size of the request. The reported response time and size are the average values

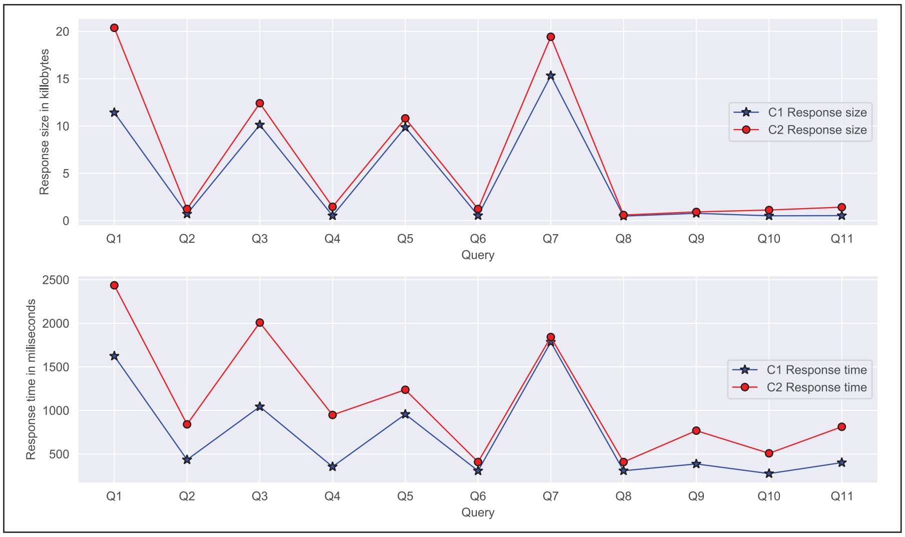

Figure 3. The figure compares the response size (top) and response time in millisecond (bottom) with 11 different queries for two communications, C1: query by the mobile client to the resource server, C2: query by the mobile client to the OpenMRS server.

Table 1. The table outlines the expected and actual response for each of the queries. Resources: HL7 FHIR resource, [resources]: array of resources, resource: response in object format.

Query	Description	Expected response	Actual response
Q1, Q3, Q5, Q7	Read all resources	Status code: 200 Response body: [resources] content-type: JSON	Status code: 200 Response body: [resources] content-type: JSON
Q2, Q4, Q6, Q8	Read one resource	Status code: 200 Response body: {resource} content-type: JSON	Status code: 200 Response body: {resource} content-type: JSON
Q9, Q10	Create and update a resource	Status code: 200 Response body: {resource} content-type: JSON	Status code: 200 Response body: {resource} content-type: JSON
Q11	Delete a resource	Status code: 200 Response body: {id} content-type: JSON	Status code: 200 Response body: {id} content-type: JSON

taken for each request (see Figure 3). The expected response and the actual response from each of these queries are outlined in Table 1. To keep the experiment result homogeneous, we hosted all the servers locally on Asus Predator Triton 500 with 32 GB of RAM, 1 TB SSD drive, 2070 RTX GPU, and Ubuntu 20.14.

Apart from the communication mentioned earlier (C1, C2), we executed `HTTP` requests to read resource from OpenMRS and write to the RS (C3). Thus, this communication involved two phases (a) query to read resource (Q2, Q4, Q6, and Q8) and (b) query to write the resource. Finally, we evaluated the communication to be successful by manually checking if the database of the RS. It is imperative to note that `HTTP` requests are subjected to network bandwidth, connectivity, and server

Table 2. The table presents an interoperability scenario between HL7 FHIR-based Resource Server (RS) and OpenMRS. Here, the FHIR patient app attempts to get Observation resource from OpenMRS and write to HL7 FHIR based RS.

Scenario	Interoperability scenario
Source of stimulus	FHIR patient app
Artifact	OpenMRS, HL7 FHIR Resource server, Authorization server, mUzima for providers, and FHIR patient app communicating over SOA.
Environment	Read Observation resource from OpenMRS and write to HL7 FHIR Resource server
Response	FHIR patient app gets valid token, sends requests to OpenMRS, OpenMRS sends valid resources to FHIR patient app, it sends write requests to HL7 FHIR RS, RS writes the observation in its database.
Response measure	Valid requests and responses

execution. Hence, one must expect to get different response time and response size on their set-up. Furthermore, this paper aims to advocate the feasibility of communication between heterogeneous EHR systems rather than performance measures.

Evaluation

We performed a non-empirical evaluation by comparing the software quality metrics of our prototype based on ISO/IEC 25000 and expert evaluation discussed in the next section. In addition to this, we share our experience regarding the maturity of `HL7 FHIR` and challenges faced during the study.

Software quality metrics

Following are the lists of software quality metrics, based on ISO/IEC 25000 standards, associated with our prototype.

1. *Interoperability*: To enforce interoperability, we use `HL7 FHIR` as the underlying communication standard. Since our principal aim is to establish interoperability between heterogeneous EHRs, we used scenario-based evaluation as recommended by the SAAM.[27] The result of the scenario-based evaluation is presented in Table 2.
2. *Security*: Both our authorization server and the `HL7 FHIR` RS communicate over `TLS`[28] and follow the contemporary practices as mentioned in the study.[21,26,28] As per the `TLS` community's recommendation, the authorization server generates the short-lived access tokens and provides a way to revoke access tokens when security conflict is noticed.
3. *Modifiability*: Several studies outline that SOA-based architecture[21,29] supports the modifiability. In SOA, the components are not dependent on vendors, technologies, brands, or underlying technological stack. Consequently, we can modify individual components without affecting others. For example, in our case, we can upgrade the database of OpenMRS without affecting the database of `HL7 FHIR` RS.
4. *Reusability*: The proposed architecture uses SOA that supports a great extent of reusability. For example, we can use the authorization server for handling authentication and authorization for several services. Besides, any third-party authorization server can communicate with the resource server, as long as they meet communication protocol standards.

5. *Scalability*: Since our architecture is based on SOA; scalability comes as in inherent attributes. The modular architecture makes it easy to scale each module if required. For example, we can increase RS's storage capacity when there are more resources to store, query, update, or delete.

Expert evaluation

The code for all the components of the prototype has been evaluated by seven developers (front-end and back-end) to verify their agreement under ISO/IEC 25000[8] software product quality requirements, and presence of anti-patterns.[30] The review team were presented the RA and the open-source framework. An interview followed up the review to determine their reaction toward the architecture, interoperability communication, and components. In addition to these questions, we had open-ended questions for feedback, reviews, and improvements. We used these feedbacks and thoughts to the enhancement of the architecture and prototype.

Discussion

Maturity of HL7 FHIR implementation: Interoperability is complex due difference in standards and representsion.[31] However, it is essential for the successful realization of healthcare. Interoperability using `HL7 FHIR` is adopted speedily. It is evident when Apple company, which is considered vital in the mobile industry, declared to adopt FHIR in iOS devices.[32] Similarly, powerful companies like Amazon, Google, Microsoft, Salesforce, IBM, and Oracle signed a joint pledge in August 2018 to support true health data interoperability by using standards like `HL7 FHIR`.[33] While `HL7 FHIR` is open-source,[34] specifications for `HL7 FHIR` are open, keeps 80/20 rule, is suitable for mobile apps and abides by the standard, well-known web technologies with specific challenges. This section addresses some of the challenges we discovered during this study.

- *Essence of rigorous validation and testing*: Interoperability with `HL7 FHIR` is in its nascent stage and hence not all EHR systems are compatible with it. Therefore, the implementer must guarantee that the evaluation tools (functional testing, platform testing, conformance testing, load and performance testing) work as specified in the `HL7 FHIR` specification.
- *Data conformance challenges*: One of the prevalent issues during the study was data matching. Data in EHR may not be the same structure and data types. For example, the `patient` data model in OpenMRS is different from `HL7 FHIR` patient model. Such mismatch scenario is common in vendor-specific EHR systems and hence makes interoperability practically challenging. Patients are the core user of any EHR systems, and patient matching could be challenging if there is a mismatch in either structure, representation, or data types. For example, a small error in the social security number (SSN), such as a dash, spelling mistake, can cause a problem.[32]
- *HL7 FHIR standards are not backwards compatible*: `HL7 FHIR` relies on 80/20 rules meaning that resources defined by `HL7 FHIR` cover 80% of the data elements used in existing EHR systems. The outstanding 20% are exceptional use cases that can be dealt with as `HL7 FHIR` extensions. Extensions are supplementary resources designed at specific companies to include data not handled by the core `HL7 FHIR` profiles. Consequently, it provides various ways for implementors to achieve identical action. Such a difference in implementation backfires when `HL7 FHIR` practitioners extend `HL7 FHIR` differently.[32]
- *FHIR does not address infrastructure*: Current healthcare solutions need to be secured and reliable. To achieve such security and reliability, it needs to follow the current infrastructure

system backed up by cloud service, Kubernetes platform, and other IAAS features. HL7 FHIR does not address these infrastructure questions. Furthermore, HL7 FHIR is not meant to address the security issue; hence, one must build HIPPA technical safeguards[35] themselves.

Limitations

The presented prototype and architecture do not address issues related to patient privacy. They do not evaluate the security to set up communication between heterogeneous EHR systems for gradual migration from OpenMRS to HL7 FHIR resource server. HL7 FHIR does not address issues associated with security and privacy. However, it provides guidelines to support Oauth2.0 and OpenID Connect for authentication. In addition to this, we do not evaluate the usability perspective of the user interface. After successful qualitative communication between the system, the prototype requires design evaluation and usability evaluation. Nevertheless, even with these restrictions, this study's proposed architecture, prototype, and communication approach will significantly benefit the healthcare industry and research communities to obtain interoperability in EHR systems following different healthcare standards.

Conclusion

We architectured and developed a system to communicate between OpenMRS and Resource server based on HL7 FHIR. This paper sought to demonstrate how the proposed architecture could be generalized to establish communication between one or more heterogeneous healthcare systems. The proposed approach can grow or keep communication alive between legacy healthcare systems. As a validation of the proposed architecture, we blended OpenMRS with a Resource server based on HL7 FHIR and GraphQL, thereby illustrating interoperability between heterogeneous healthcare systems. This study's architectural communication paves the way for innumerable attractive novel opportunities and a notable architectural mindset innovation. Nonetheless, the lessons uncovered by this study will help the OpenMRS research community by familiarizing them with how to allow the OpenMRS healthcare standard to evolve to achieve better interoperability.

Abbreviations

CDA Clinical Document Architecture
EHR Electronic Health Record
FHIR Fast Healthcare Interoperability Resources
HIS Healthcare Information Systems
HTTP Hypertext Transfer Protocol
IAAS Infrastructure-as-a-service
ICD International Classification of Diseases
JSON JavaScript Object Notation
REST Representational State Transfer
RS Resource Server
SAAM Software Architecture Analysis Method
SOA Service-oriented Architecture
SOAP Simple Object Access Protocol
SQL Structured Query Language
TLS Transport Layer Security
URL Uniform Resource Locator
XML Extensible Markup Language

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This publication is a part of the INTROducing Mental health through Adaptive Technology (INTROMAT) project, funded by the Norwegian Research Council (259293/o70). INTROMAT is a research and development project in Norway that employs adaptive technology for confronting this issue.

ORCID iD

Suresh Kumar Mukhiya https://orcid.org/0000-0002-1813-7633

Supplementary materials

The following links contains source codes for all the components used in the experimental setup, and are open-sourced under MIT license. https://github.com/sureshHARDIYA/phd-resources/tree/master/Papers/SAGE

References

1. Mukhiya SK, Rabbi F I, Pun VK, et al. A graphql approach to healthcare information exchange with hl7 fhir. *Proc Comput Sci* 2019; 160: 338–345.
2. Kalra D, Beale T and Heard S. The openehr foundation. *Stud Health Technol Inform* 2005; 115: 153–173.
3. Rezaei R, Chiew TK, Lee SP, et al. Interoperability evaluation models: a systematic review. *Comput Ind* 2014; 65(1): 1–23.
4. Stünkel P, von Bargen O, Rutle A, et al. Graphql federation: a model-based approach. *J Object Technol* 2020; 19(2): 18:1–21.
5. Gulden C, Blasini R, Nassirian A, et al. Prototypical clinical trial registry based on fast healthcare interoperability resources (fhir): design and implementation study. *JMIR Med Inform* 2021; 9(1): e20470.
6. Saripalle R, Runyan C and Russell M. Using HL7 FHIR to achieve interoperability in patient health record. *J Biomed Inform* 2019; 94: 103188.
7. Team F. GraphQL — a query language for your API. https://graphql.org/.
8. ISO/IEC 25000. ISO/IEC 25000: 2012 Software Engineering - Software product quality requirements and evaluation (SQuaRE) - guide to SQuaRE, 2012. https://www.iso.org/obp/ui/#iso:std:iso-iec:25000:ed-2:v1:en
9. Healthcare Information & Management. *HIMSS dictionary of health information and technology Terms, Acronyms and Organizations*. 2019. Boca Raton: Productivity Press.
10. Geraci A, Katki F, McMonegal L, et al. *IEEE standard computer dictionary. A compilation of IEEE standard computer glossaries*. 1991. New York City: IEEE.
11. Wang W, Tolk A and Wang W. The levels of conceptual interoperability model: Applying systems engineering principles to M&S. In: *Spring simulation multiconference 2009 - co-located with the 2009 SISO spring simulation interoperability workshop*. San Diego, CA, USA. 0908.0191.
12. Gibbons P, Artz N, Burke-Beebe S, et al. *Coming to terms: scoping interoperability for health care*. Health Level Seven EHR Interoperability Work Group, Washtenaw Avenue, 2007.
13. Benson T and Grieve G. *Principles of health interoperability: SNOMED CT, HL7 and FHIR*. 2013. New Your City: Springer International Publishing.
14. (W3C) WWWC. Simple object access protocol (soap) 1.2, 2007. https://www.w3.org/TR/soap12/ (2007).

15. Fielding RT and Taylor RN. Principled design of the modern web architecture. *ACM Trans Internet Technol* 2002; 2: 115–150.
16. International HLS. Hl7 FHIR SMART app launch, http://hl7.org/fhir/smart-app-launch/ (accessed 28 October 2020).
17. Inc O. Openmrs community, https://openmrs.org/ (accessed 28 October 2020)
18. Kasthurirathne SN, Mamlin B, Kumara H, et al. 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.
19. Gavrilov G, Vlahu-Gjorgievska E and Trajkovik V. Healthcare data warehouse system supporting cross-border interoperability. *Health Informatics J* 2020; 26(2): 1321–1332.
20. Seebregts CJ, Mamlin BW, Biondich PG, et al. The openmrs implementers network. *Int J Med Inform* 2009; 78(11): 711–720.
21. Mukhiya SK, Rabbi F, Pun KI, et al. An architectural design for self-reporting e-health systems. In: *Proceedings of the 1st international workshop on software engineering for healthcare. SEH '19, Piscataway, NJ, USA, 27–27 May 2019*, pp. 1–8. IEEE press.
22. Sakimura N, Bradley J, Jones M, et al. Final: OpenID Connect core 1.0 incorporating, 2014. https://openid.net/specs/openid-connect-core-1_0.html
23. Schmidt DC, Stal M, Rohnert H, et al. *Pattern-oriented software architecture, patterns for concurrent and networked objects*. Hoboken: John Wiley & Sons, 2013. Vol. 2.
24. Team M. mUzima for health providers v2.7.0 release – Muzima. 2021. https://www.muzima.org/blog-news/muzima-for-health-providers-v2-7-0-release/.
25. Katarahweire M, Bainomugisha E and Mughal KA. Authentication in selected mobile data collection systems: current state, challenges, solutions and gaps. In: *2017 IEEE/ACM 4th international conference on mobile software engineering and systems (MOBILESoft)*, Buenos Aires, Argentina, 22–23 May 2017, pp. 177–178. IEEE.
26. On FHIR S. Smart on FHIR, http://www.techterms.com/definition/social_media (accessed 28 November 2020)
27. Kazman R, Bass L, Abowd G, et al. SAAM: a method for analyzing the properties of software architectures. In: *Proceedings - international conference on software engineering*, Sorrento, Italy, 16–21 May 1994. IEEE.
28. (IETF) IETF. Recommendations for secure use of transport layer security (TLS) and datagram transport layer security (DTLS). *RFC 7525 2015*. https://datatracker.ietf.org/doc/html/rfc7525
29. MacKenzie CM, Laskey K, McCabe F, et al. *Reference model for service oriented architecture 1.0. OASIS standard*. 2006. Woburn: OASIS Open.
30. Ambler S. *Process patterns building large-scale systems using object technology*. 1998. Cambridge: Cambridge University Press.
31. Benson T and Grieve G. Why interoperability is hard. In: *Principles of health interoperability*. New Your City: Springer, 2016, pp.19–35.
32. Cabot. Aren't you on the HL7 FHIR bandwagon yet?. https://www.cabotsolutions.com/arent-you-on-the-hl7-fhir-bandwagon-yet (accessed 16 February 2021).
33. Council ITI. Cloud health care pledge agreements, https://www.itic.org/public-policy/CloudHealthcarePledge.pdf (accessed 13 November 2020)
34. Foundation F. Index - FHIR v4.0.1. https://www.hl7.org/fhir/ (accessed 16 February 2021).
35. Miaoulis WM. Hipaa security overview-retired. *J AHIMA* 2013. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html