Framework For An Intensive Care Unit Digital Infrastructure With FHIR and Ventilation Data

By

Andy Pham
THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Health Informatics

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Dr. Nick Anderson, Chair

Dr. Brooks Thomas Kuhn

Mark Carroll

Committee in Charge

2020

Table of Contents

Table of Figures

Abstract

As the modern intensive care unit (ICU) becomes more crowded and management of disease become more complex, the burden upon intensivists and nurses to care for their patients increases substantially. Advancements in processing power, standardization development, and medical device capabilities are leading to the development of a "smart ICU", centralized around the idea of a connectivity envelope that all medical devices can feed their data into and that an interoperable data system is possible as a result. This study outlines the necessary building blocks of this framework using the Health Level 7 (HL7) Fast Healthcare Interoperability Resources (FHIR) standard as its core data model and proposes a prototype using waveform ventilator data. We built a backend consisting of a Tomcat server running PostgreSQL as a database and using the HAPI FHIR package to establish FHIR compliance and successfully hosted on Amazon Web Services. The data source consists of historic waveform data pulled from patient ventilators and transformed into the FHIR data model.

However, using FHIR model requires that the data points be mapped to a specified medical standard. The Institute of Electrical and Electronics Engineers (IEEE) 11073-10101a codes are a set of standardized medical informatics codes for dealing specifically with point-of-care medical and personal health care devices. They were used to translate the waveform ventilator data into valid FHIR resources. The major obstacles towards getting the framework setup were finding the right code system for waveform data and gaining the web architecture knowledge to host a server with an application file. However, the ease of implementation once the initial knowledge building is finished is fairly high, and the system has the potential for others to construct their own interfaces using the FHIR backend and API as a data source. Future work in using this framework includes the creation of a client interface to complete the framework, the addition of security protocols, and inclusion of more code systems to

widen the range of data types that the system can take in. Further research is needed to quantitively assess the effectiveness of the system through interoperability and usability metrics.

1. Introduction

In the past, intensive care units (ICUs) were designed with the concept of an isolated place to hold and treat the critically ill.[1] They were small rooms in hospitals and generally restricted to both visitors and staff. However, with the rapid development of processors and technology, the infrastructure of the ICU evolved to become more focused around collecting high-resolution data and utilizing suites of medical devices. Furthermore, these medical devices are becoming more portable and user-friendly. Connectivity between patients, nurses, intensivists, and physicians is vastly improved. Clinical decision support and alert systems help prevent adverse events and augment decision-making. Currently, it is common to see a patient connected to multiple machines and a network of wires streaming from the bed. According to the Society of Critical Care Medicine[2], more than 5.7 million patients are annually admitted to ICUs, often with combinations of cardiac, respiratory, and neurological conditions. Aging populations and higher severity of illness create an increasing demand for quality ICU care. However, there is a shortage of nurses, critical care physicians (intensivists), and qualified staff that can deliver high quality care. Only 37% of all ICU patients are attended by intensivists in the United States and a 35% shortfall of intensivist hours is predicted to occur by 2020.[2,3] The amount of beds in the ICU and the average length of stay are both increasing, leading to a higher ICU to hospital bed ratio.[1] Therefore, the combination of increasing demand for better quality care, shortage of work force and widespread availability of beneficial technology has led to a significant opportunity for creating a connectivity envelope of medical devices that can enhance and support ICU activities.

By developing interoperability across devices, we can create a ripe environment for developing an infrastructure that can utilize this data to its fullest potential. With the internet of things permeating throughout the design and development of smartphones, tablets, laptops, and wearable devices, the difficulty of data collection has been minimized. Increases in processing power allow data to be collected in vast amounts at high speeds and create multiple opportunities to perform data-driven analysis. In critical care, the practice of data science has the potential to apply to many applications, such as physiological waveform analysis and predictive modeling.[4] In a research design proposal, Halpern reflects the change in the data landscape for critical care in his vision of the Smart ICU.[5] By using "smart" or "intelligent" technology, the patient data generated from medical devices can be turned into practical information.[5] By converting data into information in real-time, the smart ICU can better enhance and support the work of intensive care staff. Intensivists and clinicians can act on accurate, timely information, making decisions that are based on real-time context and constantly updating views of the patient. Nurses can be given a better holistic view of the patient, even as hospital conditions change, as data coming from different devices are contextualized and presented in the most efficient way possible by the system. Constructing a connectivity envelope that is patient-centric is the foundation of a smart ICU. In Halpern's paper, 5 steps are outlined in creating the connectivity envelope.[5] They are:

1. Installation of a robust wired and wireless infrastructure
2. Integrating connectivity hardware that can communicate with all data sources in patient rooms
3. Putting automatic identification tags on all data sources
4. Attaching additional hardware to facilitate the transmission of data between medical devices and the receiving system
5. Installing middleware, which include servers and applications, into hospital and ICU networks

Within these steps, Halpern emphasizes several concepts. Generated data should be associated with each patient's unique identifier. Achieving data alignment between medical devices and a receiving system helps with interoperability and temporal synchronization. Middleware (i.e., alarm systems and third-party devices) act as a supporting tunnel to facilitate the transfer and utilization of data.[5] Due to the advances of medical device technology, most hospitals already have a stable wired and wireless framework with varying levels of connectivity. Thus, most of the work remains with implementing steps 3 through 5, which is all about interoperability. Once a high level of interoperability is achieved, the system paves the way for ICU data to inform the functionality and creation of applications to enhance the work of healthcare professionals. Intensivists and clinicians can quickly build their own tools without having to worry about the accuracy or the variability of the data coming from medical devices and leverage their tools to increase their knowledge of patient care. Therefore, a shared learning environment is constructed, providing continuous knowledge building. The National Institute of Medicine dictates that programs created for improving patient quality of care must be safe, timely, effective, efficient, equitable and patient-centered.[6] The smart ICU reflects that by focusing on patient-generated data, a shared learning environment, and continuous evaluation. Applying the concepts of the smart ICU design, a digital environment is constructed that is both secure, interoperable and adaptable. However, the lack of projects involving data driven systems like the smart ICU present a formidable barrier to implementation.[4] The cause for this deficiency involves the current state of interoperability of medical devices in the ICU.

2. Background

Interoperability, as defined by the Healthcare Information and Management Systems Society (HIMSS), is the "…ability of different information technology systems and software applications to communicate, exchange data, and use the information that has been exchanged."[7] Data should have the capability to be ubiquitously shared across all members of the health care spectrum, regardless of the application.[7] For developing systems, there are 4 types of interoperability: technical, syntactic, semantic, and organizational.[8] Technical and semantic interoperability refer to the data exchanging process and tools required for the system to be interoperable. Semantic and organizational interoperability deal with data interactions with the users and between different systems. Since this study is focused on developing a new system, only the technical and semantic portions will be assessed. There have been many cases where interoperability would have prevented medical errors: incorrect conversion of units has led to critical dosage errors; and misconstrued documentation of artifactual signals are also other examples.[9] Another large issue that interoperability could help solve is manual data entry, which often varies from 2 minutes to 6 hours and often results in errors due to poor handwriting or readability.[10] If data is exchanged accurately digitally between diverse data sources and displayed, it will make the manual annotation of measurements stemming from different medical devices obsolete since the data can viewed clearly on a screen with the correct clinical context irrespective of the device itself. In an environment where access to trustworthy data is vital, there is a lack of connectivity between the devices that provide this data. As technology and IT capacity improves, medical devices have become increasingly interrelated and require standards to sort out the complexity. Currently, vendor and manufacturer perspectives lead the development of these devices, instead of clinician perspectives. The interests of all stakeholders involved in the ICU care team need to be addressed and reflected in the usages of these devices. Interoperability provides a way to implement these interests and increase the quality of ICU care.

2.1 The Current State of Medical Device Interoperability

Historically, medical devices have been designed to measure various patient characteristics and nothing more. Today, medical devices in the ICU are sophisticated informatics platforms.[5] They come with connectivity, alerting software, and data management middleware.[5] As such, they are valuable sources of data generation and collection. With multiple devices connected to one patient and multiple patient beds in the ICU, there is a ripe environment for developing an interoperable infrastructure that can utilize this data to its fullest potential. Hospitals and researchers are already starting to build this connected network of devices using the technology available today. In 2016, the University of Missouri Health center implemented a project to connect their medical devices to the EHR.[10] Using a combination of Health Level 7 (HL7) interfaces, vendor data conversion software, and inherent device capabilities, they managed to connect a multitude of devices ranging from ventilators to dialysis machines to their EHR, effectively creating a central hub of communication. They dubbed these high-tech rooms, the "Smart Rooms". Third-party systems or vendors offer built-in device interoperability and EHR integration. Cerner's CareAware and Iatric System's Accelero Connect systems establish pipelines for standardized data transmission between most of their devices and the EHR, including support for outside devices.[11] Bedmaster XA, developed by Excel Medical Electronics, Jupiter and Florida, connects various device data through the hospital local area network and is compatible with both GE and Philips technologies.[12] One of the largest projects for medical device interoperability has been the Medical Device Plug-and-Play (MD PNP) Interoperability program headed by the Massachusetts General Hospital and the Center for Integration of Medicine and Innovative Technology.[13] The program was focused on constructing and developing a multitude of components and standards that model the "Integrated Clinical Environment" (ICE), a platform that centers around open standards for device interoperability to achieve better patient care. Using the American Society for Testing and Materials standards for integrating equipment and data transmission, MD PNP has created scenarios, use cases, an open-source ICE platform for integration, and interfaces for external data transfer.[13] Similarly, the University Hospitals Case Medical Center and Case Western Reserve University developed their own architecture for real-time data acquisition and device integration called the Integrated Medical Environment (tIME).[12] tIME uses a local database to extract and import waveform and parametric data from various devices using an HL7 data transfer protocol while a critical care informatics middleware analyzes the data flowing through the system. There is a large amount of work and development across multiple institutions being done to facilitate better communication between medical devices and the hospital system. The common element amongst these examples is standards, which play an integral role in data transfer and collection. However, there are a variety of standards to choose from.

2.2 Medical Device Communication Standards

As medical devices became standard fare for ICUs, many standards development organizations formed to provide guidance and structure for dealing with device data. Initially, standards were focused around the general safety and build of medical device hardware, with the International Electrotechnical Commission (IEC) 60601-1 codes and International Standards Organization (ISO) 14971 codes acting as the main leads.[9] IEC 60601-1 describes the requirements for preventing electromagnetic disturbances and performance tests for electronic medical equipment.[14] ISO 14971 covers the standards for medical device risk management, providing 3 ways for manufacturers to build safety into the devices. When the EHR became ubiquitous and these instruments became powerful processing machines, a more informatics-based standard needed to be developed. Three notable groups came to the foreground to pioneer the effort. ISO/IEEE 11073, a collaboration between ISO and the Institute of Electrical and Electronics Engineers (IEEE), provides data models and nomenclature for point-of-care medical device communication.[9] This standard uses a manager/agent principle, where the agent provides the data and the manager reacts to the agent's input. Both a manager and an agent can switch roles as well as form hybrid systems. ISO/IEEE 11073 includes nomenclature that can be linked back to Object Identifier codes and specific device specializations, such as those for pulse oximeters and ventilators.[15] Another alternative standard is the Integrating the Healthcare Enterprise (IHE) Patient Care Device domain, also known as IHE-PCD 01.[16] IHE is an initiative involving both providers and vendors to improve the way upon medical device communication using Integration profiles. According to the 2018 IHE Handbook, each profile has actors-- a system or device responsible for certain tasks-- which can perform a certain set of actions to communicate with other actors. The Device Enterprise Communication Profile is responsible for transmitting data from patient care devices to enterprise applications, such as the EHR or clinical information systems.[17] It uses a combination of IEEE 11073, HL7 v2.6, and standardized nomenclature for units of measure to manage semantic interoperability between devices. A terminology bridge called the Rosetta Terminology Mapping ties together the nomenclature between the different standards and ensures easy, efficient implementation of each IHE profile.[17] The last major standards organization is Health Level 7 (HL7), an American National Standards Institute (ANSI)-accredited group focused on defining the standards for the exchange and sharing of electronic health information. HL7 Fast Healthcare Interoperability Resources (FHIR) is a framework structured around health information data models and centered around using web services to connect health devices.[18] It centers around using a set of structured data items and values called resources, which can be categorized into identifiers such as 'Patient' or 'Observation'. These resources are a result of combining the best components of all the HL7 standards and the ability to map medical codes to the data, giving HL7 FHIR the flexibility to integrate with any device, regardless of vendor, as long as the right resources are configured.[18] While this is a strong start for medical device connectivity, there are still several hurdles to overcome to achieve full interoperability. The lack of adoption, requirement of significant IT overhead, too little choices, and inflexible workflows are all issues that define the gaps in interoperability.

2.3 Current Gaps and Issues in Medical Device Interoperability

In a survey done by HIMSS[12], over 90% of the hospitals used 6 or more types of medical devices, but only a third of those devices could communicate with each other or with the EMR. While standards development organizations have made considerable headway in developing ways to standardize communication between medical devices, the lack of adoption in ICUs remains a large obstacle. According to De Georgia *et al*.[12] (2016), acute medical care devices are not designed for interoperability, but instead follow their own protocols for transmitting data. For standards like ISO/IEEE 11073, which rely on configuring the devices to communicate the right data, ICU devices are problematic. Many of the devices are vendor-specific and are locked from customization because of hardware specifications. Any modifications would have to go through a formal process of contacting the vendor, rebuilding the software in the device, and paying for the changes. The difficulties are further exacerbated if there are different versions of the devices being used, especially if customized hospital interfaces rely on certain devices staying consistent with coded in data models across upgrades. The cost and resources required for IT staff to implement the standards could outweigh the benefits gained. Furthermore, upgrades or software modifications could undo the standardization process unintentionally. Weininger *et. al.*[9] (2016) notes that device manufacturers can have differing interpretations of the meaning of the standards requirements and test methods. Thus, even if the device is built for a certain standard, the end result may be incompatible with it. Therefore, standards like ISO/IEEE that center around enforcing new practices for devices see a low rate of adoption. Another issue with implementing interoperability arises due to the lack of choices in standards that are developed enough to be credible. Hospitals using legacy devices may not be able to take advantage of modern standards. If a device is not compatible with the limited selection of standards, then it will cause a serious delay in hospital workflows. Even the standards themselves can be incompatible with each other for certain devices. Rodriguez *et. a*l's[19] (2018) systematic analysis on data collection and integration of two large clinical studies in the United States and Europe reflects all these flaws. One of the major weaknesses observed was the variability and availability of measurements from a range of source devices at each site. Due to a lack of standardized nomenclature, data streams coming from different sites followed dissimilar labelling guidelines and were recording at different resolutions.[19] Furthermore, the lack of standards for data archival led to physiological data being stored in custom user-defined formats.[19] Some devices required significant IT resources to connect with the hospital network and were subject to manual data upload.[19] The results of the systematic analysis uncovered critical gaps in interoperability. Finally, there are no set guidelines for which standard is optimal for each device. This can result in situations where institutions could spend long hours having to research and deduce the best standard among a large collection. With these barriers, the road to interoperability is unclear.

2.4 HL7 FHIR: A Growing Solution

There is a standard that has the potential to solve the flaws plaguing medical device interoperability. HL7 developed its own modern framework called FHIR, taking the best characteristics from previous versions, like the HL7 v2 and the HL7 Clinical Document Architecture. FHIR is built upon RESTful architecture, using HTTP protocols to facilitate data transfer and storing data in popular data formats like JSON and XML.[20] Resources are the driving force for FHIR, acting as entities for meaningful information representation and as transaction items for data exchange.[20] They are represented as structured concepts, malleable enough to allow for a wide range of contextual information, but specific enough to allow users to concretely identify what the resource is being used for. The resources can also map to commonly-used terminologies such as LOINC or to other resources.[18] Thus, FHIR possesses an enormous amount of flexibility and usage that other standards don't have. Since configuration is achieved through HTTP, devices from different vendors can adopt the standard as long as they have an internet connection. The implementation itself is easy as well, with a comprehensive array of documentation and test servers on the HL7 FHIR website.[18] Some implementations have been done in as little as 2 weeks. These factors lower the IT and device hardware requirements, which have been critical barriers in preventing standards from being adopted. In terms of comparing to other standards, FHIR outperforms them in terms of information reuse and flexibility. Lee and Do's 2018 study involving the ISO/IEEE 11073, IHE PCD-01, and FHIR standards proves this notion. By comparing the data size, content, and processing among the three standards, the authors found that FHIR exceeded the other two standards for most of the measured variables.[16] This is because ISO/IEEE 11073 is limited in its abilities to only send fixed messages and IHE PCD-01 was too constrictive in its data representation.[16]

Furthermore, FHIR is backed by HL7, giving it credibility. For the circumstances surrounding implementation in the ICU and for long-term interoperability, FHIR is one of the best contenders.

2.5 Previous Related Works

There are several related works that have been done in the realm of interoperability and decision support for the ICU. Kogure *et. al.*[21] integrated a remote patient monitoring telemedicine system that could transmit real-time patient data to a Java-enabled mobile phone. Ji *et. al*.[22] created a mobile application for the early detection of delirium within the ICU that nurses preferred to use rather than the traditional detection method. The closest example is Murphy and Kincaid's ventilator weaning application using HL7 FHIR as the standard and the Cerner API.[23] Using FHIR to organize and access the data, an application was built to monitor a patient's readiness to wean off the ventilator. The data accessed from the ventilator to the FHIR interface included parameters like tidal volume, respiratory rate and plateau pressure. A FHIR prototype was constructed in 10 days and clinical mappings were established using SNOMED codes. The project is currently in the process of building a FHIR application team and has yet to proceed outside the trial. Other prominent projects using FHIR involved using i2b2--an open-source research platform that houses deidentified patient cohort data that can be used for testing and analysis--as a platform for exposing patient data to FHIR data models and establishing a mapping layer to translate i2b2 observations to FHIR resources.[24, 25, 26] Much of the focus of these projects was on the data mapping and translation between the i2b2 cells and FHIR resources. While these previous works set a precedent for the feasibility of integrating FHIR into patient and hospital data systems, they did not assess interoperability or usability of the infrastructure. Additionally, the projects used FHIR as a bridge to access data from outside the hospital, but few looked at the use of FHIR from within the hospital infrastructure.

3. Objective

Implementing enterprise-wide medical device interoperability in the ICU is a difficult task requiring the cooperation of professionals across many different disciplines and extensive testing and integration. For the scope of this thesis, the best course of action to move towards building this interoperability is to focus on one type of device and its data, with the vision that the results could be extrapolated to future implementations and research. Currently, the best device and data for usage that I have access to is the patient ventilator.

Mechanical ventilation is a life-saving, but nuanced and potentially dangerous intervention frequently used deployed in ICUs. Over half of the patients are mechanically ventilated within the first 24 hours of admission, many due to direct lung injury.[27] In these situations, the breath a patient wants can actually be dangerous, therefore there can be conflict between patient demand and ventilator delivered breaths, a concept referred to as patient-ventilator asynchrony (PVA). PVA can lead to patient discomfort, longer time on mechanical ventilation, and even ventilator-induced lung injury.[28] Thus, ventilator-induced lung injury and patient-ventilator asynchrony are serious problems that need to be addressed within the ICU. Jason Adams *et. al.*[28] developed a multi-algorithmic software called ventMAP that collected and wirelessly transmitted patient ventilator data from Raspberry Pi microcomputers to a remote server. After collecting the breath metadata, the software classified the breaths based on adverse breathing events like asynchrony. It has been shown in Ramirez *et. al.*[29] that the identification of PVA using waveform analysis relies neither on years of experience working in the ICU nor the specific profession of the intensivist. However, it was concluded that specialized training on mechanical ventilation can significantly increase the chances of PVA identification.[29] Therefore, the data provided by the ventilator is extremely crucial in clinical decision-making. Due to the availability of the software and the wide use of mechanical ventilators, the ventMAP dataset will be used in the prototype of the environment.

Given the availability of health data, increasing support towards interoperable data systems, and lack of ICU data interfaces, I am proposing a framework of an ICU digital infrastructure built on FHIR standards with recommendations, steps, and limitations, using the experience of building a test system based on ventilator data as a foundational basis.

To measure the feasibility of this proposed infrastructure, an assessment can be categorized into two factors: usability and interoperability. I will be utilizing the Center for Virtual Care and setting up several simulated patients. There will be two run-throughs where physicians will keep track of the patients with and without the FHIR interface. The variables tested will be the time taken to detect PVA in a patient, and the overall usability of the application in terms of workflow, data presentation, accuracy, and interactivity. Physicians will be given a post-usability questionnaire after running each scenario. Interoperability will be measured using a interoperability assessment model based on the different layers of interoperability.

4. Theoretical Framework

In building the technical requirements and components of the proposed infrastructure, the major pieces are:

- Implementing a FHIR server that is compliant with FHIR standards
- Mapping ventilation data to FHIR resource data elements correctly
- Setting up a data exchange layer supporting RESTful API protocols to the FHIR server
- Creating a user interface for interacting and visualizing the FHIR data through the FHIR API

Having a compliant FHIR server running is the pivotal piece of this design because it will act as a data model to read in the ventilator information and as a back-end unit for communicating with the web browser. The minimalistic requirement for a server to be FHIR-compliant is to define and represent resources.[18] Resources are FHIR's building blocks and the central data model, composed of metadata and human readable elements. They represent concepts in healthcare, ranging from administration to questionnaires to clinical. Each resource shares a common map of required data elements but can be expanded or combined in different ways for a diverse range of uses. Resource specifications are public and hosted at the official FHIR website. However, a minimalistic implementation would not be sufficient to support the exchange of data between systems. In order to integrate RESTful API protocols between the FHIR server and the data interface, FHIR servers must have a conformance layer. A conformance layer is a set of resources that defines how resources are exchanged and imposes rules for how resources should be used in a particular context. It's composed of the structure definition, capability statement and value set resources. The structure definition resource establishes the schema of the FHIR server, defining data types, resource constraints, and what resources are accepted. The capability statement resource describes the functionality of the server itself, including involved software, web interaction endpoints, and server metadata. Value set resources set up the server's data mapping and representation capabilities, since they are collections of coded values, like the International Classification of Diseases (ICD) or Snomed, and their properties. Combining all three resources into a package is called an implementation guide, and these implementation guides form the foundation of the conformance layer. FHIR compliancy reduces undetected risks when the server is going through review and vetting as well as setting expectations for outside systems to communicate with.

After establishing a compliant FHIR server, it is important to choose a terminology or coding system to map data elements to. In many of the resources, the values are connected to a coded element, sourced from a code system declared within the system. These code systems can contain medical codes, standardized terminology, and related metadata that the FHIR server can use to provide context to concepts and resources. Some notable code systems accepted by FHIR are LOINC, SNOMED, and ICD.[18] Implementors can choose either to use existing internal FHIR code systems or import their own. As long as the minimum metadata for each of the coded concepts is present, FHIR will accept customized code systems. Ideally, the imported FHIR code systems would thoroughly represent all of the data attributes present in the targeted data source. After importing a code system, the targeted data will be transformed into the necessary FHIR resource format.

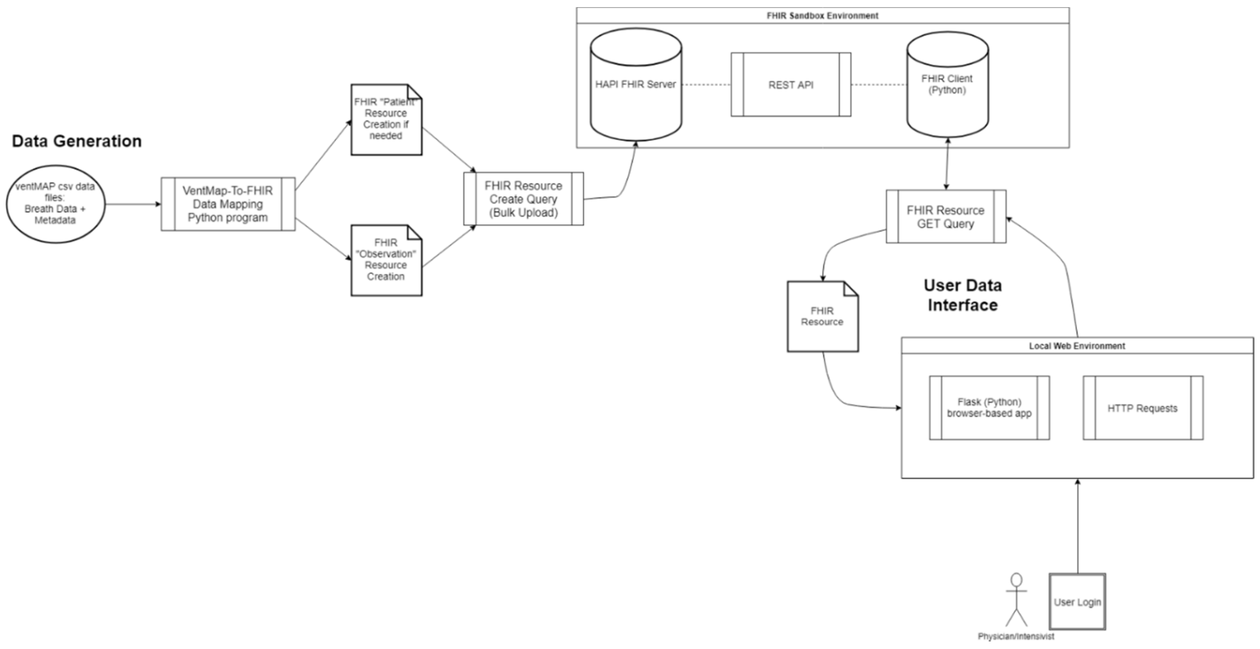
FHIR supports RESTful operations and has resource types that are catered for uploading and reading data. One of these resource types is called a Bundle, and it allows for the manipulation of multiple resources in one batch transaction. A Bundle acts as a list of resources, and it can take different resource types. CRUD operations can be coded into the Bundle resource, and it will apply to all the resources packaged within the Bundle. Furthermore, FHIR resources from the server are received as Bundle resources. Therefore, developing a data exchange layer between the FHIR server and the data source must include functions that can parse Bundle resources.

When creating the data exchange layer, it would ideally contain separate functions for reading, updating, inserting and deleting resources. When reading in data from the source to the FHIR server, there are two approaches. One approach is to read the data in real-time, with updates happening to the FHIR server every time new breath information is collected on the ventMap system. This provides the most timely data, but requires the system to always have processing memory ready to use. The second approach is to separate the data loads into batches over a set time interval. The data would always be delayed by the time interval, but it would take less processing power on the system as a whole. The most suitable approach will depend on the interfaces built to bring the data to health practitioners. For example, nurses changing shifts could benefit from a time-delayed visualization of historic ventilator data to accurately inform them of previous events and get a holistic view of the patient. Intensivists would benefit from real time data on all the patients they’re overseeing in one view, instead of having to walk from patient bed to bed.

The last step in the process is to create a user interface that can visualize the ventilator data in an effective way that is beneficial to the user's workflow. This requires meeting with various clients to gather the necessary business needs and features. Since the data is stored on a centralized database, any implementor can initialize their own FHIR client server, connect to the database, and build customized tools that use the data. This opens many opportunities for a variety of specialized tools and interfaces to be created and built off each other, resulting in a learning digital system.

A visual interpretation of the theoretical framework is given below.

Figure 1: Data Flow of a HAPI FHIR Backend and Frontend Structure


FHIR Sandbox Environment
HAPI FHIR Server
REST API
FHIR Client (Python)
Data Generation
ventMAP csv data files: Breath Data + Metadata
VentMap-To-FHIR Data Mapping Python program
FHIR "Patient" Resource Creation if needed
FHIR "Observation" Resource Creation
FHIR Resource Create Query (Bulk Upload)
FHIR Resource GET Query
FHIR Resource
User Data Interface
Local Web Environment
Flask (Python) browser-based app
HTTP Requests
User Login
Physician/Intensivist

5. Measuring Interoperability and Usability

There are a multitude of different ways to measure interoperability. In a review by Rezaei *et al.*,[8] 8 interoperability models were assessed and compared according to their depth and method of measurement. For this study, I will be using the Levels of Information Systems Interoperability (LISI) model presented in the review, supplemented by a few quantitative measurements for connectivity measurements and data transfer frequency.[8] LISI was developed by the Department of Defense as a reference model for the standardized assessment of the interoperability of information systems.[8] Within this model, there are 5 levels of interoperability that the target system can be categorized into: isolated, connected, functional, domain-based, and enterprise-based. As the levels increase, the system becomes more interoperable. Additionally, LISI measures 4 attributes for each of these levels, covering aspects that inform the design of the system, such as infrastructure and procedures. Forming a 5 x 4 matrix, the LISI metric can assign a system an interoperability grade based on where the various attributes of the system land within the 5 levels. Other models built for more specific systems, such as the ISIMM[30] model for government systems interoperability, build upon the concepts presented in LISI. While there are several models to choose from, LISI has the most flexibility in terms of the type of system it can measure and its focus on technical interoperability fits well for measuring the initial implementations of new systems. LISI has been used in a previous study for assessing the interoperability between an obstetrics-gynecology department, several radiology departments, an analysis laboratory and a general practitioner's office in a hospital information system in 2012.[31] For quantitative measurements, Leite proposed a series of interoperability assessment models for measuring components such as information flow and data capacity.[8] Many of the models deal with testing the limits of system information transfer, such as system capacity and overload. As this research is focused on establishing the feasibility and requirements of an interoperable system, gathering the variables to use most of these models would require extensive work outside of the research scope. The only model that can be used given the circumstance is the connectivity index model. It quantifies the system's flexibility to send and receive data. It only requires gathering the number of nodes and paths in the system to calculate. The other quantitative measurement used will be the receiving frequency of the breath data points, as it will have to be close to or match current ventilator frequencies for effectiveness.

For usability, choosing the right type and method of evaluation based on the testing environment and resources available is crucial for gaining the most insight on improvements to healthcare interactable systems. In a comparative review done by Jaspers,[32] three major types of usability evaluations were detailed and contrasted based on characteristics such as required resources, data generated, and actionable outcomes. The three usability methods were heuristic evaluation, cognitive walkthrough and the think-aloud method. The first two were expert-based, requiring the participation of experts and competent evaluators to analyze the system. The think-aloud method was the only one that is user-based, meaning the end-user is the primary input of the testing, and is the chosen method for this study. This is because gathering expert evaluators would take extensive time, and there are end-users readily available at the UC Davis Medical Center for testing. The think-aloud method involves users performing a series of tasks on the testing environment given to them by the researcher and verbalizing their thoughts either during or after the activity. [32] This usually involves audio recording or video taping of the subjects. Think-aloud evaluations result in a rich source of data to analyze and the identification of severe usability problems not usually detected by the other two methods. [32] There are many instances of this methodology being utilized to test interactable health technologies, such as clinical decision support systems and physician data query tools.[33,34,35] For the purposes of this study, the users will be asked to perform interactive tasks with the ventilator data interface connected to the FHIR server and verbalize their thoughts during the session. After the testing session, subjects will take a post usability survey based on Nielsen's 10 usability heuristics for further analysis and evaluation.[36]

6. Methods

6.1 FHIR Library and Server

The HAPI-FHIR library was used as a guide and foundation for setting up a server to meet FHIR compliancy. HAPI-FHIR is an open-source implementation package written in Java that provides the setup and resources for integrating FHIR onto web applications.[37] All the code written for the package is readily available on Github for customization and implementors can customize the package towards their preferences. The package also includes a pre-built starter application that could be self-deployed on a local environment or loaded on a web server. Setting up a FHIR server becomes quick and achievable for implementors of any level using the starter package. Using it, all FHIR resources are defined with the most current version of FHIR and are setup using HAPI FHIR's conformance layer. After a data system has been well-defined in its processes and data flow, implementors can customize the conformance layer to manipulate the behavior of resources and any interactions with them. For the purposes of this framework, HAPI's default settings are suitable enough to allow for a wide variety of testing and data. HAPI FHIR's Java foundation allows for the flexibility and scope needed to meet the requirements to setup this framework for any operating system. For the server, Apache Tomcat 8.5 was used because of its popularity, ease of use, and detailed documentation. Apache Maven 3.6.0 with Java 8 was run on the pre-built starter files to create a WAR (Web Application Resource) file that can be deployed on Tomcat. To provide endpoints for the data exchange, both a standard Amazon EC2 instance and a locally hosted server was established as containers. The configuration file supplied by the starter package was customized to connect to each respective container.

6.2 Database

The HAPI FHIR package defaults to using Derby as the database but can be configured to connect with other databases compatible with Hibernate. PostgreSQL was chosen as the backend to store the data in because of its wider usage rate across different operating systems and programming environments and my familiarity with the database structure. Both a local and Amazon Relational Database Service PostgreSQL database were set up, for testing and querying. The Amazon PostgreSQL database was password-protected and ran on engine version 10.6. The PGAdmin tool was used to access and view the data within the database for validation purposes. The HAPI FHIR package was configured to point to the Amazon PostgreSQL database location, changed to use the PostgreSQL 9.4 database dialect for automated schema setup, and included the database account information.

6.3 Data Source

The data source for ventilation data was provided by the authors of the ventMAP software and paper.[28] It contained genuine breath data and metadata on 37 patients, whose identifiers were masked. The attributes contained in data are listed in the Appendix B. The data was given in a zip file of folders, with folder each representing a subject. Each of these folders contained 3 CSV files detailing the annotations, breath metadata, and raw breath data sampled at 50 Hz. Each row in the annotation and breath metadata files corresponded to a complete breath. The raw breath data file contained the pressure and flow measurements for a set number of breaths. The breath metadata and raw breath data for each patient were read into a Python script for mapping to FHIR resources.

6.4 Ventilation Codes

The 2015 IEEE 11073-1010a point-of-care medical device communication standards for health informatics was used as the underlying code system for mapping the ventilation data to FHIR resources. These codes were chosen after consulting FHIR implementors on the Zulip communication platform and researching other standardized coding systems. The data dictionary was downloaded off of the IEEE Standards Association website in XML format.[38] It consisted of a list of medical device nomenclature and terms specific to vital sign and ventilator devices. Each entry contains an IEEE reference ID, description of the term, and a context-free numerical ID in the Rosetta Terminology Mapping format.[39]

6.5 Ventilation Data FHIR Mapping

The R4 version of FHIR was used as a reference for the creation of the resources required for the framework. Consulting with one of the ventMAP authors for validity, the ventMap data attributes were mapped one-to-one to the codes where it was appropriate. The IEEE codes were converted into a custom CodeSystem resource for the FHIR server through a FHIR CodeSystem resource builder website called Snapper.[40] With a Python script, the original code files were converted from XML into a comma-delimited format with data attributes matching the required input for Snapper. The ElementTree library was used to parse out XML elements and read them into a dictionary data structure for processing. Each type of measurement detailed in the data source was coded to an existing entry within the IEEE codes. The comma-delimited output file was then imported into Snapper, and after filling in additional metadata, it was transformed into a CodeSystem resource JSON file. The CodeSystem resource file was successfully validated through the public HAPI FHIR testing server and loaded accordingly into the target

FHIR server. The mapping dictionary to convert ventMap data attributes to IEEE codes is displayed in Table 2.

Table 2: Mapping dictionary to convert ventMap codes to IEEE codes.

VentMap Attributes	Common Name	ISO Common Term	ISO REFID	ISO Code
METADATA				
ventBN	VentMap relative breath number	N/A		
BS	Breath start time	Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	67985
BE	Breath end time	Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	67985
IEnd	Inspiratory breath end (equal to rel_time_at_x0)	Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	67985
I:E ratio	Inspiratory Expiratory Ratio	Ratio inspiration expiration time	MDC_RATIO_IE	151832
iTime	Inspiratory Time	Inspiratory Time	MDC_TIME_PD_INSP	152608
eTime	Expiratory Time	Expiratory Time	MDC_TIME_PD_EXP	152612
inst_RR	Spontaneous respiration rate	Spontaneous respiration rate (preferred)	MDC_RESP_BTSD_PS_RATE	151674
tvi	Inspired tidal volume	Inspired Tidal Volume	MDC_VOL_AWAY_TIDAL_INSP	152660
tve	Expired tidal volume	Expired Tidal Volume	MDC_VOL_AWAY_TIDAL_EXP	152664
tve:tvi ratio	Expiratory inspiratory tidal volume ratio	Ratio inspiration expiration time	MDC_RATIO_IE	151832
maxF	Max flow	Ventilator airway flow	MDC_VENT_FLOW	151940
minF	Min flow	Ventilator airway flow	MDC_VENT_FLOW	151940
maxP	Max Pressure	Maximum airway pressure	MDC_PRESS_AWAY_MAX	151793
PIP	Peak inspiratory pressure	Maximum inspiratory airway pressure (peak inspiratory pressure)	MDC_PRESS_AWAY_INSP_MAX	151817
Maw	Mean airway pressure	Mean airway pressure	MDC_PRESS_AWAY_MEAN	151795
PEEP	Positive end expiratory pressure	Applied PEEP	MDC_VENT_PRESS_AWAY_END	151976
ipAUC	Inspiratory Area Under the Curve	N/A		
epAUC	Expiratory Area Under the Curve	N/A		
rel_time_at_x0	Inspiratory breath end	Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	67985
rel_time_at_BS	Breath start time	Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	67985
rel_time_at_BE	Breath end time	Relative-Time-Stamp	MDC_ATTR_TIME_STAMP_REL	67985
min_pressure	Minimum pressure	Minimum ventilation pressure	MDC_VENT_PRESS_MIN	151958
ANNOTATIONS				
vent BN	Ventilator breath number	N/A		
rel time	relative ventilator time	N/A		
TVi	Inspired tidal volume	Inspired Tidal Volume	MDC_VOL_AWAY_TIDAL_INSP	152660
TVe	Exspired tidal volume	Expired Tidal Volume	MDC_VOL_AWAY_TIDAL_EXP	152664
TVe/TVi ratio	Expiratory inspiratory tidal volume ratio	Ratio inspiration expiration time	MDC_RATIO_IE	151832
dbl	Double Trigger Asynchrony	N/A		
mt	Multi Trigger Asynchrony	N/A		
fa	Flow Asynchrony	N/A		
co	Cough	N/A		
su	Suction	N/A		
vd	Ventilator Disconnect	N/A		
aNOS	asynchrony not otherwise specified	N/A		
RAW DATA				
Left column	Flow	Ventilator airway flow	MDC_VENT_FLOW	151940
Right column	Pressure	Airway pressure	MDC_VENT_PRESS	151956
S:####	Ventilator breath number	N/A		

In addition to the ventMap to IEEE mapping dictionary, an IEEE code to FHIR Observation resource mapping table was constructed to be used as a reference in the code to automatically assign the various FHIR Observation elements to the correct IEEE data attributes. This mapping dictionary is displayed in Table 3.

Table 3. Mapping dictionary to convert IEEE codes to FHIR resources.

identifier	text	code	display	valueType	unit	system	origin	period	dimensions
bs_time	The time this breath started	67985	MDC_ATTR_TIME_STAMP_REL	valueQuantity	s	data:text/plain;charset=utf-8,iso11073-10101a			
be_time	The time this breath finished	67985	MDC_ATTR_TIME_STAMP_REL	valueQuantity	s	data:text/plain;charset=utf-8,iso11073-10101a			
i_end_tim	The time the inspiratory breath finished	67985	MDC_ATTR_TIME_STAMP_REL	valueQuantity	s	data:text/plain;charset=utf-8,iso11073-10101a			
i_time	The time interval for an inspiratory	152608	MDC_ATTR_TIME_STAMP_REL	valueQuantity	s	data:text/plain;charset=utf-8,iso11073-10101a			
e_time	The time interval for an expiratory	152612	MDC_ATTR_TIME_STAMP_REL	valueQuantity	s	data:text/plain;charset=utf-8,iso11073-10101a			
ie_ratio	Inspiratory time divided by expiratory	151832	MDC_RATIO_IE	valueQuantity		data:text/plain;charset=utf-8,iso11073-10101a			
inst_rr	Spontaneous respiration rate, the rate of breaths or inspiratory gas flow initiated by the patient where flow and/or volume is determined by the patient and are delivered with the intention that the breath will be terminated by the patient	151674	MDC_RESP_BTSD_PS_RATE	valueQuantity	breaths/minute	data:text/plain;charset=utf-8,iso11073-10101a			
tve	Expired Tidal Volume: Volume of expired gas for all breath and inflation types, reported individually	152664	MDC_VOL_AWAY_TIDAL_EXP	valueQuantity	mL	data:text/plain;charset=utf-8,iso11073-10101a			
tvi	Inspired Tidal Volume: Volume of inspired gas during each breath, breath	152660	MDC_VOL_AWAY_TIDAL_INSP	valueQuantity	mL	data:text/plain;charset=utf-8,iso11073-10101a			
maxF	Maximum airway flow	151940	MDC_VENT_FLOW	valueQuantity		data:text/plain;charset=utf-8,iso11073-10101a			
minF	Minimum airway flow	151940	MDC_VENT_FLOW	valueQuantity		data:text/plain;charset=utf-8,iso11073-10101a			
maxP	Maximum airway pressure	151793	MDC_PRESS_AWAY_MAX	valueQuantity	cm[H20]	data:text/plain;charset=utf-8,iso11073-10101a			
pip	Peak Inspiratory Pressure	151817	MDC_PRESS_AWAY_INSP_MAX	valueQuantity	cm[H20]	data:text/plain;charset=utf-8,iso11073-10101a			
maw	Mean airway pressure	151795	MDC_PRESS_AWAY_MEAN	valueQuantity	cm[H20]	data:text/plain;charset=utf-8,iso11073-10101a			
peep	Positive end expiratory pressure	151976	MDC_VENT_PRESS_AWAY_END_EXP_POS	valueQuantity	cm[H20]	data:text/plain;charset=utf-8,iso11073-10101a			
min_press	Minimum ventilation pressure	151958	MDC_VENT_PRESS_MIN	valueQuantity	cm[H20]	data:text/plain;charset=utf-8,iso11073-10101a			
flow	Ventilator airway flow	151940	MDC_VENT_FLOW	valueSampledData		data:text/plain;charset=utf-8,iso11073-10101a	0	20.0	50
pressure	Airway pressure	151956	MDC_VENT_PRESS	valueSampledData		data:text/plain;charset=utf-8,iso11073-10101a	0	20.0	50

Using the two mapping dictionaries for reference, the Python package glob for reading data in bulk, and the Python package Pandas for excel data processing, Observation resources were mapped with the corresponding breath metadata attributes and an assigned breath number. 18 total data elements were converted into Observation resources from the ventMap data. Unique identification values were created for each Observation consisting of an MD5 hash of a combination of patient id, breath number, and the name of the data attribute. The Smart-On-FHIR Python package was used to support the transformation of the raw ventMap data structure into the FHIR resource. The package contained an Observation resource class that read in dictionary elements and formatted them into the

FHIR resource JSON structure. Resources were validated using Smart-On-FHIR's built-in FHIR parser. Patient resources were created and assigned identifiers based on the folder names provided by the ventMAP data source. Fake names generated by the Python package Faker were used to populate the name components of the Patient resources. Observation resources were referenced to their respective Patient resources by the Patient resource identifiers.

6.6 Resource Uploads to FHIR

Using the RESTful API of the HAPI FHIR servers, Patient and Observation resources were uploaded in batch through a Python script. The resources are validated for errors using the FHIR server's capability statement. The amount of resources that can be uploaded per transaction depends on the server's memory limit, which varied according to environment. The local host environment allowed for larger transactions, while the Amazon AWS server crashed if more than a few resources were read in.

6.7 Interviews with Intensive Care Unit Health Practitioners

Two interviews were conducted with hospital employees working within the UC Davis Intensive Care Unit. The first interview was with Dr. Brooks Kuhn, an assistant professor of clinical medicine in pulmonary and critical care medicine. The interview focused around gathering the requirements that a data system collecting waveform data would need to be usable in the clinical environment. The crucial requirements presented were that the data system should be capturing breath waveforms at the ventilator's sampling rate, the waveform data is time-stamped accurately to an absolute time, and that the information should be presented in a focused context with a clear goal. The data system should be built to store and visualize data with a contextualized goal, such as capturing point-of-care data at the bedside in real time or analyzing delayed breath data for certain breath events. There was also some discussion on the basics of waveforms and the components of a ventilator. The second interview was with Serena Fazio, a critical care nurse and PhD post-doctorate in the ICU. She is involved in patient care management, and responsible for continuous assessment of patients. The interview revolved around targeting the user interface elements that would be useful to a critical care unit and the limitations of how ventilator data is displayed in the Epic EHR. She also described the workflow of a nurse in relation to the patient care management team. A questionnaire was prepared in advance to guide the conversation for Serena's interview.

6.8 Interoperability Assessment

A questionnaire for assessing degree of interoperability within a data system was formed based on the Levels of Information Systems Interoperability (LISI) model developed by the Department of Defense.[8] There are 4 sections stratified by layers of interoperability:

- Policies and Procedures
- Application: System Requirements and Purpose
- Infrastructure
- Data

Each section has 1 – 3 questions asking about the data system's capabilities and purpose. An interoperability metric ranging from 0 – 4 will be calculated based on the answers. Each metric corresponds to an environment where a level interoperability would exist. As the numbers increase, the interoperability complexity increases and the system becomes closer to achieving complete interoperability. The 5 environments are:

- Isolated
- Connected
- Functional
- Domain-based
- Enterprise-based

The questionnaire was formatted as a Google Form, which is referenced in the Appendix A.

6.9 Usability Assessment

A post-test usability questionnaire was developed to measure the effectiveness of the client interface layer. There are 11 questions and a Likert scale from 1 – 5 is used for each question. The questions largely cover difficulty levels in interacting and navigating the interface. Each question is inspired by the Nielson's 10 usability heuristics.[36] The usability questionnaire is referenced in the Appendix A.

7. Results

Of the 44 ventilator data elements given on 37 patients, 18 data elements were successfully validated and transformed into FHIR resources using Python code. All 37 patients were given fake generated names and identifiers for transformation into FHIR Patient resources. The 2015 IEEE 11073-1010a health communications standards were converted into a FHIR code system for reference. Two environments were constructed, one on an Amazon Web Services EC2 instance and one on a local laptop. Both environments used Apache Tomcat 8.5 to host the FHIR web application container and shared a PostgreSQL backend hosted by Amazon Relational Data Services. The Patient resources were uploaded first onto the FHIR servers. The primary data flow consisted of transforming Excel-formatted ventilator data into FHIR Observation resources for each patient through Python, linking the Observation resources with their respective Patient resources and then uploading all the transformed data into the FHIR servers through a bulk upload. If more than 4 FHIR resources were uploaded to the AWS FHIR server, it would crash due to a Java out of memory error. Data was communicated to and from the server using the FHIR API configured by the HAPI FHIR starter package. An interoperability and usability assessment were written for domain experts to analyze and test the system. The implementation of the data system had a steep learning curve for me because of the variety of publicly available FHIR servers to choose from, an overwhelming amount of documentation for FHIR, and the difficulty in finding an adequate medical code system for ventilator data. However, after resolving the code system issue and getting the basics of how FHIR resources work, the difficulty of implementation became considerably lower.

8. Discussion and Recommendations

8.1 Establishing System Prerequisites

This study aims to identify the steps and pieces needed to construct an interoperable data collection and display system based on FHIR standards. It is done primarily to set the starting blocks towards building an ICU digital infrastructure that can communicate with a myriad of medical devices. Previous studies have explored and implemented similar digital infrastructures, but none have yet touched upon bringing a FHIR-centralized data infrastructure to the ICU and applying it to high output medical devices such as ventilators.

One of the most important pieces to consider and solidify before diving into the technical implementation of the FHIR server is to establish a standardized coding system for the data read in. This allows other systems with similar data but in a different format, such as medical devices belonging to different vendors, to read their data in with the same context. Each Observation resource requires a link to a code from any of the terminology systems listed on the HL7 FHIR website. These systems range from well-known externally published code systems to internal FHIR terminology created by FHIR workgroups. However, there are few coding systems that focus around medical device terminology and those that do didn’t have the specific codes needed for thoroughly covering ventilator-produced measurements. A majority of the medical device terminology focused around vital signs and classifying the device components. After researching various coding systems and asking for assistance on the Zulip FHIR implementors chat group, the 2015 IEEE 11073-10101a codes were chosen because of their extensive coverage of breathing and ventilator terminology. However, there were certain measurements, such as inspiratory and expiratory breath times, that weren't addressed by the terminology. Most notably, classifications of ventilator events, such as double trigger asynchrony, were missing. A possible reason for the lack of coverage could be due to the wide variety of proprietary medical device vendors that are used in an ICU, preventing an agreement of established nomenclature, as bias towards one device could lead to other devices becoming incompatible. Another possible reason is that the data collected for these measurements is not easily available enough or provide enough analyzable content to justify the creation of the terminology. Ventilator data is difficult to capture accurately and the overall classification of breath events still requires an expert intensivist's eyes to judge.[20] For data in which codes are difficult to find or one code system doesn't cover thoroughly, it can be worth concatenating several code systems together or asking the FHIR community on social forums like Zulip. Getting the right CodeSystems imported into the FHIR server should be one of the first steps to fulfill to proceed with the rest of the setup.

8.2 Constructing the Backend

The default configured database was set to Apache Derby. Due to unfamiliarity with Derby and the high output of ventilator data, a database was needed that could scale well with increasing amounts of data and could be easily configured for optimal use. Two structures, document and relational, were considered. Document databases were made to store unstructured data and follow a design that reflects the FHIR resource structure similarly. Its JSON-like data structure allows for dynamic changes in data models. Since it is a non-relational database, factors like schema structure and constraints don't need to be considered when constructing a backend design. Given more time, I would test the efficiency and usefulness of using a document database in comparison to a relational database, but due to my unfamiliarity with it and the time it would take to figure out the technicalities, I chose to go with something more well-known. For the purposes of this project, the input data is structured enough for a relational database to be effective.

FHIR servers take a substantial amount of time to set up and configure, but due to FHIR's increasing popularity, many online services offer free pre-configured FHIR servers for testing. I opted to use one of these pre-configured FHIR servers, due to my inexperience with FHIR. There were several specifications that drove the choice to use HAPI FHIR. As the backend database was chosen to be a PostgreSQL server, the premade FHIR server had to be able to configure to a SQL database. Additionally, the FHIR server had to be able to support create, read, write and update operations for manipulating resources. It was also crucial that the FHIR server had persistent independent storage, meaning that data wasn't cleared after a certain interval and the data uploaded wouldn't be restricted by resource content. Many servers required OAuth security protocols to be setup and an app to be registered before starting. However, because of the uncertainty as to how the resources should be structured for ventilator data, it was more prudent to build up the backend FHIR pieces before coding for an application. Another benefit of using HAPI FHIR was that the configuration package automatically performs the schema creation and design for SQL databases, giving less experienced implementors a small leeway in taking on this step. In searching for the right FHIR server for testing, options that fit the specifications such as Aidbox and the HSPC Sandbox were considered and tested. However, Aidbox required developers to pay for full control over the implementation and its Docker-based setup excluded certain versions of the Windows operating system from running it.[39] The HSPC Sandbox had an interface for interacting with the FHIR server, but the options for configuring the server were too limited. Eventually, a combination of Apache Tomcat as a container and HAPI FHIR as the implementation was used as the backend framework, as it provided the most customization while skipping the initial configuration work. Having an open-source Java implementation meant that coding in additional resource structure manipulations could be accomplished and the system could run across multiple environments. Running my own server instance meant that I could control the data flow, and it provided the most straightforward route in getting FHIR to run in the shortest amount of time.

It should be noted that the HAPI FHIR starter package comes with a configuration file where users can customize where the FHIR server should retrieve its information from and where the server end points reside. However, it is recommended that implementors should have a small amount of web architecture experience, such as knowing what database dialects are and how to get a basic Tomcat server running. The HAPI FHIR documentation isn't detailed in instructing how to customize the package for non-default usage. For implementors dealing with commonly used and available medical data, it would be more beneficial to utilize more user-friendly public FHIR servers for testing, such as Cerner's FHIR Sandbox. A list of public FHIR servers and their capabilities are listed at the HL7 FHIR Wikipedia page, under the title "Publicly Available FHIR Servers for testing".[41]

8.3 Creating the Data Exchange Layer

Python scripts were developed for reading data out of the ventMap files, transforming the data into a FHIR resource data model, and uploading the FHIR resources to the FHIR server using the API. Initially, custom Python classes were developed to represent the Observation and Patient resource data structures, but I learned that the approach wouldn't scale well if other types of resources were implemented. Therefore, the Smart on FHIR Python library was installed and used to build the resource structure. The functions would automatically format the provided data into the right JSON format for each specified resource, saving code space and time. A benefit it provided was a validation check when creating the resource objects that would error out if the JSON formatting was incorrect or if any necessary elements were missing. Python functions were modularized based on data processing steps in preparation for automation. Data loading from the ventMap files, transformation of raw data to FHIR resource, and upload of FHIR resources to the FHIR server were separated into functions.

The IEEE codes did not cover all of the data attributes measured in the ventMap outputs. One of these data attributes was breath number. To create an effective visualization and provide clinical context, breath number is crucial. Therefore, the unique identifier for each Observation resource was composed of a combination of the associated breath number, the patient identification value it originated from, and the name of the data attribute. Making the identifiers in this format ensures uniqueness and stores the breath data in an easily parsed format.

8.4 Ease of Implementation

While the initial technical learning requirements of implementing a FHIR server backend with waveform data may be daunting, the amount of actual work required to set it up is far less than expected. Essentially, any implementor with a beginner's amount of web development experience can have a FHIR server up and running within a day. Furthermore, the variety of preset options and helpful FHIR community lessens the learning curve. Most of the major workload comes from searching for the right medical codes and verifying that the data is correctly mapped in the right clinical context. Once a FHIR backend is running, any developer can spin up their own FHIR instance and connect to the FHIR backend to retrieve the data in FHIR format. This flexibility gives way to many opportunities for a variety of health professionals with low to moderate technical experience to create specialized tools that can access FHIR waveform data.

9. Limitations

There were several caveats to consider when performing this study due to time, inexperience, and resources. The aim of the study was to provide the draft of an interoperable data system that could collect ventilator data in the ICU. However, the data source itself is made of clean, static data pulled from another project. It has been pre-formatted for analysis and does not accurately represent what the raw data would look like when pulled out from the ventilator. The IEEE codes used do not represent a widely accepted standardization of ventilation terminology. There is not currently an agreed set of terminology codes for ventilator data, and the IEEE codes were the best representation available. Furthermore, not all of the data attributes were represented in the IEE codes. Calculated attributes such as inspiratory area under the curve and breath event classifications like multiple trigger asynchrony did not have corresponding values in the IEEE codes. Since Observation resources require that the data is linked to a medical code, certain data attributes had to be omitted. Additionally, ventilator breath numbers were not represented, so a workaround was established by including it as part of the Observation unique identifiers, but they are not officially mapped to any Observation resource. In terms of security, both online and local servers are open-ended and do not have any authentication barriers attached. The time cost in learning to work with the various technologies like HAPI FHIR and Apache Tomcat ended up steering the project in a direction where the focus was to get a prototype running. This meant having to omit the implementation of OAuth authentication protocols or security barriers. Future implementors will need to consider using a form of OAuth authentication in their FHIR data systems. Another crucial factor to consider is the server memory. Uploading resources to the AWS FHIR server is limited by the server memory allotted and memory outages were frequently encountered. Any bulk uploads consisting of more than 4 FHIR resources crashed the AWS server. Getting the server back online after various crashes took a significant amount of time because of the lack of documentation on the interactions between online servers and FHIR. Therefore, the local server hosted on my laptop was used as the primary server to take in batch uploads and taking server memory out of consideration. Since it is not an online server, the proposed data system prototype will not accurately reflect the hospital environment, which have servers with varying amounts of memory Implementors will need to consider this factor when building out a system in a real environment. Due to timing constraints, the user interface was not able to be constructed for the framework. Developing an adequate interface requires user experience and design experience as well as a certain technical level of web development, which I do not currently possess. Furthermore, the usability and interoperability evaluation of the system was not able to be carried out because of a missing interface. With more time and development experience, the usability and interoperability questionnaires would be utilized to measure the effectiveness of the system with users and domain experts in the ICU respectively.

10. Conclusion

The rapid increase in the capabilities of medical devices, push towards interoperability and higher patient load of the present-day ICU presents an opportunity and motivation to create a digitized system that incorporates all medical devices to provide an evolving data infrastructure. With the ratio of ICU health professionals and patient beds skewing towards patient beds[2,3], it becomes necessary to supplement manual work with automated processes made available with an interoperable data system. Halpern's vision of a connectivity envelope provides a guide and a foundation in constructing the intricacies of this system.[5] The growing community of standards and HL7 FHIR provides the system the tools it needs to support interoperability. The realm of ICU waveforms have been a seldom visited area for implementing an interoperable system, due to medical device vendor disagreements and lack of waveform standardization.[12]

This study aimed to lay down the design and framework of a FHIR standardized data system built around the data flow of the ICU ventilator. The main criteria for building this data system was established and the foundational pieces were laid out. These criteria included the backend design, data exchange layer, and waveform data mapping to FHIR resources. A theoretical draft of the infrastructure was detailed and planned based off of the surrounding literature. Both a usability and interoperability assessment were created to test the efficacy of a prototype system. The FHIR enabled system was found to be difficult to build initially due to technical inexperience, but easy to implement following growing familiarity. The most difficult portions were found to be identifying the correct coding system for the data to be mapped to and learning the intricacies behind starting a Tomcat server that supported FHIR. Due to the time sink caused by these steps and the amount of debugging time dealing with various server failures, the framework was not able to be fully realized. Only the FHIR backend and data exchange layer was built, with the client interface still untouched. Future research would aim to develop a prototype of the client interface and utilize the assessments to evaluate the system. However, a smart ICU data system centered around FHIR shows a huge amount of promise in providing a platform for users to develop their own tools while keeping the data standardized. Future research should also aim to explore a learning system stemming from developing specialized tools and its ease of implementation.

References

1. Vincent J. Critical care--where have we been and where are we going? *Crit Care*. 2013;17 Suppl 1(Suppl 1):S2. doi:10.1186/cc11500.
2. Society of Critical Care Medicine. Critical Care Statistics. Website. http://www.sccm.org/Communications/Critical-Care-Statistics. Accessed August 23, 2018.
3. American Society of Anesthesiologists. *The Principles of Critical Care Medicine*.; 2017.
4. Sanchez-pinto LN, Luo Y, Churpek MM. Big Data and Data Science in Critical Care. *Chest*. 2018;154(5):1239-1248. doi:10.1016/j.chest.2018.04.037.
5. Halpern NA. Innovative designs for the smart ICU: Part 3: Advanced ICU informatics. *Chest*. 2014;145(4):903-912. doi:10.1378/chest.13-0005.
6. Committee on Quality of Health Care in America. Crossing the quality chasm: A new health system for the 21st century. Washington DC: National Academy Press; 2001.
7. HIMSS. What Is Interoperability? Website. https://www.himss.org/library/interoperability-standards/what-is-interoperability. Accessed November 21, 2018.
8. Rezaei R, Chiew T, Lee S. A review of interoperability assessment models. *J Zhejiang-University Sci C*. 2013;14(9):663-681. doi:10.1631/jzus.C1300013.
9. Weininger S, Jaffe MB, Goldman JM. The Need to Apply Medical Device Informatics in Developing Standards for Safe Interoperable Medical Systems. *Anesth Analg*. 2016;13(11):2395-2402. doi:10.1016/j.celrep.2015.11.047.Long.
10. Bliven B, Bragg M, Long B. Medical device connectivity case study. *J Clin Eng*. 2016;41(2):E1-E11. doi:10.1097/JCE.0000000000000144.
11. Cerner. CareAware. Website. https://www.cerner.com/pages/careaware. Accessed October 20, 2018.
12. De Georgia MA, Kaffashi F, Jacono FJ, Loparo KA. Information technology in critical care: Review of monitoring and data acquisition systems for patient care and research. *Sci World J*. 2015;2015. doi:10.1155/2015/727694.

13. Goldman JM, Whitehead SF. *Enabling Medical Device Interoperability for the Integrated Clinical Environment*. Boston. Massachusetts General Hospital. ; 2015. Accessed October 27, 2018.
14. International Electrotechnical Commission. IEC 60601-1-02. https://webstore.iec.ch/publication/2590. 4th ed. Published February 2014. Accessed November 16, 2018.
15. IEEE. IEEE 11073 Personal Health Devices. Website. http://11073.org/. Accessed November 5, 2018.
16. Lee S, Do H. Comparison and analysis of ISO/IEEE 11073, IHE PCD-01, and HL7 FHIR messages for personal health devices. *Healthc Inform Res*. 2018;24(1):46-52. doi:10.4258/hir.2018.24.1.46.
17. Integrating the Healthcare Enterprise. IHE Patient Care Device User Handbook. 2018.
18. HL7. FHIR. Website. https://www.hl7.org/fhir/overview.html. Published 2017.
19. Rodriguez A, Smielewski P, Rosenthal E, Moberg D. Medical Device Connectivity Challenges Outline the Technical Requirements and Standards for Promoting Big Data Research and Personalized Medicine in Neurocritical Care. *Mil Med*. 2018;183:99-104. doi:10.1093/milmed/usx146.
20. Bender D, Sartipi K. HL7 FHIR: An agile and RESTful approach to healthcare information exchange. *Proc CBMS 2013 - 26th IEEE Int Symp Comput Med Syst*. 2013:326-331. doi:10.1109/CBMS.2013.6627810.
21. Kogure Y, Matsuoka H, Akutagawa M, Shimada Y, Kinouchi Y. The Applications of Remote Patient Monitoring System using a Java-enabled 3G Mobile Phone. *IFMBE Proc*. 2007;14:3658-3661.
22. Ji M, Wu Y, Chang P, Yang X, Yang F, Xu S. Development and Usability Evaluation of the Mobile Delirium Assessment App Based on Confusion Assessment Method for Intensive Care Unit (CAM-ICU). *Stud Health Technol Inform*. 2015;216:899. doi:10.3233/978-1-61499-564-7-899.
23. Murphy RE, Kincaid LA. Get to Green: A FHIR-enabled application for ventilator weaning using Cerner Ignite API. 2017.
24. Pfiffner PB, Pinyol I, Natter MD, Mandl KD (2016) C3-PRO: Connecting ResearchKit to the Health System Using i2b2 and FHIR. PLOS ONE 11(3): e0152722. https://doi.org/10.1371/journal.pone.0152722
25. Boussadi A, Zapletal E. A Fast Healthcare Interoperability Resources (FHIR) layer implemented over i2b2. *BMC Med Inform Decis Mak*. 2017;120(17):1-12. doi:10.1186/s12911-017-0513-6.
26. Wagholikar KB, Mandel JC, Klann JG, et al. SMART-on-FHIR implemented over i2b2. 2017;24(2):398-402. doi:10.1093/jamia/ocw079.
27. Kirton, Orlando. Mechanical Ventilation in the Intensive Care Unit. Website. http://www.aast.org/GeneralInformation/mechanicalventilation.aspx. Accessed September 17, 2018.
28. Adams JY, Lieng MK, Kuhn BT, et al. Development and Validation of a Multi-Algorithm Analytic Platform to Detect Off-Target Mechanical Ventilation. *Sci Rep*. 2017;7(1):1-11. doi:10.1038/s41598-017-15052-x.

29. Ramirez II, Arellano DH, Adasme RS, et al. Ability of ICU Health-Care Professionals to Identify Patient-Ventilator Asynchrony Using Waveform Analysis. *Respir Care*. 2017;62(2):144-149. doi:10.4187/respcare.04750.
30. Staden S Van, Mbale J. The Information Systems Interoperability Maturity Model (ISIMM): Towards Standardizing Technical Interoperability and Assessment within Government. 2012;(October):36-41. doi:10.5815/ijieeb.2012.05.05.
31. Vida M, Stoicu L, Bernad E. Measuring the Interoperability Degree of Interconnected Healthcare Information Systems Using the LISI Model. 2012:76-80.
32. Jaspers MWM. A comparison of usability methods for testing interactive health technologies : Methodological aspects and empirical evidence. *Int J Med Inform*. 2009;8:340-353. doi:10.1016/j.ijmedinf.2008.10.002.
33. Peute LWP, Keizer NF De, Jaspers MWM. The value of Retrospective and Concurrent Think Aloud in formative usability testing of a physician data query tool. *J Biomed Inform*. 2015;55:1-10. doi:10.1016/j.jbi.2015.02.006.
34. Kushniruk AW, Borycki EM. Low-Cost Rapid Usability Engineering: Designing and Customizing Usable Healthcare Information Systems. 2004.
35. Richardson S, Mishuris R, O'Connell A, et al. "Think Aloud" and "Near Live" Usability Testing of Two Complex Clinical Decision Support Tools. *Int J Med Inf*. 2017;106:1-8. doi:10.1016/j.ijmedinf.2017.06.003.
36. Nielsen, J. 10 Usability Heuristics for User Interface Design. NNG Nielson Normal Group Website. https://www.nngroup.com/articles/ten-usability-heuristics/. Updated April 24, 1994. Accessed April 9, 2019.
37. University Health Network. HAPI FHIR Website. http://hapifhir.io/ Updated May 30, 2019. Accessed January 12, 2019.
38. IEEE Standards Association. Health informatics—Point-of-care medical device communication—Part 10101: Nomenclature—Amendment 1: Additional Definitions. https://standards.ieee.org/standard/11073-10101a-2015.html. Updated December 9, 2015. Accessed January 3, 2019.
39. Rosetta Table. RTM Management Service Website. https://rtmms.nist.gov/rtmms/index.htm#!rosetta. Updated December 22, 2016. Accessed June 13, 2019.
40. The Australian e-Health Research Centre. Snapper. Website. http://ontoserver.csiro.au/snapper2-dev/index.html?#/. Accessed June 20, 2019.
41. Health Level 7 International Wiki. Publicly Available FHIR Servers for testing. Website. https://wiki.hl7.org/Publicly_Available_FHIR_Servers_for_testing. Updated August 20, 2019. Accessed July 18, 2019.

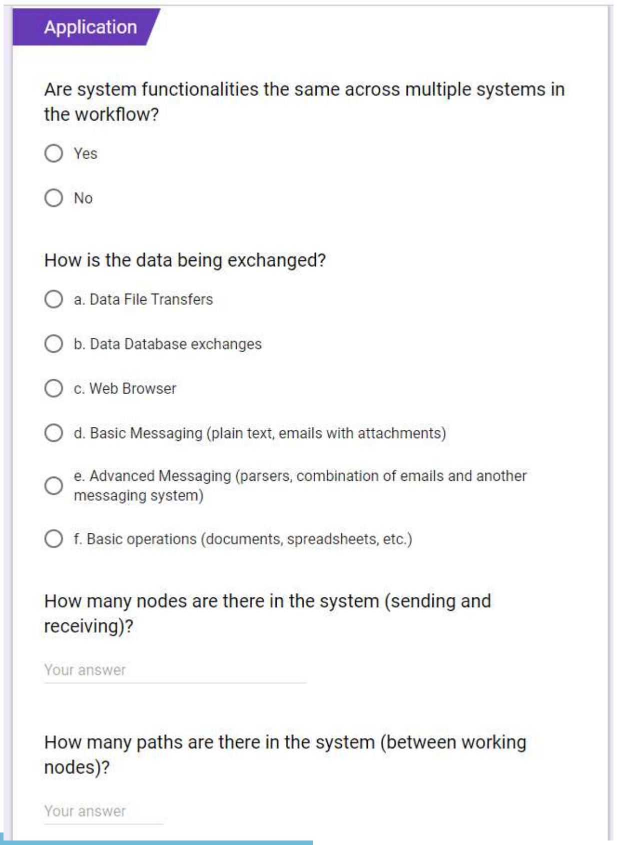
Appendix A

Usability and Interoperability Questionnaires

Nielson-inspired post-test usability survey

Answers will be on a Likert scale from 1 – 5:
Very hard/hard/medium/easy/very easy

1. Difficulty in recognizing feedback from clicks or interactions with the data.
2. Difficulty in understanding the current settings of graphs or visualizations
3. Difficulty in customizing settings to get desired outputs
4. Difficulty in understanding the terminology used to represent breath data
5. Difficulty in understanding the use of interactive elements used to modify or change data visualizations
6. Difficulty in undoing unintended actions
7. Difficulty in navigating from section to section of the website
8. Difficulty in understanding error messages
9. Difficulty in resolving error messages
10. Difficulty in understanding functions of icons and buttons
11. Terminology used accurately reflects the clinical reality
    a. Will have:
       Strongly Disagree/Disagree/Don't Know/Agree/Strongly Agree as choices


Application

Are system functionalities the same across multiple systems in the workflow?

- ◯ Yes
- ◯ No

How is the data being exchanged?

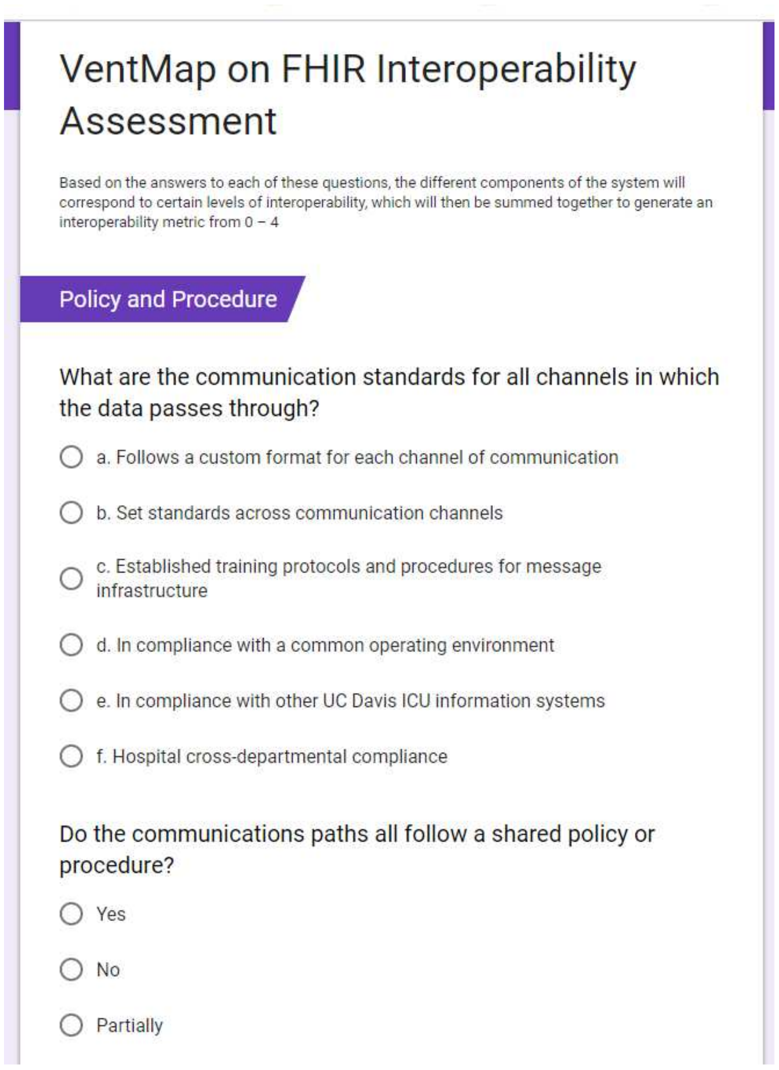
- ◯ a. Data File Transfers
- ◯ b. Data Database exchanges
- ◯ c. Web Browser
- ◯ d. Basic Messaging (plain text, emails with attachments)
- ◯ e. Advanced Messaging (parsers, combination of emails and another messaging system)
- ◯ f. Basic operations (documents, spreadsheets, etc.)

How many nodes are there in the system (sending and receiving)?

Your answer

How many paths are there in the system (between working nodes)?

Your answer



VentMap on FHIR Interoperability Assessment

Based on the answers to each of these questions, the different components of the system will correspond to certain levels of interoperability, which will then be summed together to generate an interoperability metric from 0 – 4

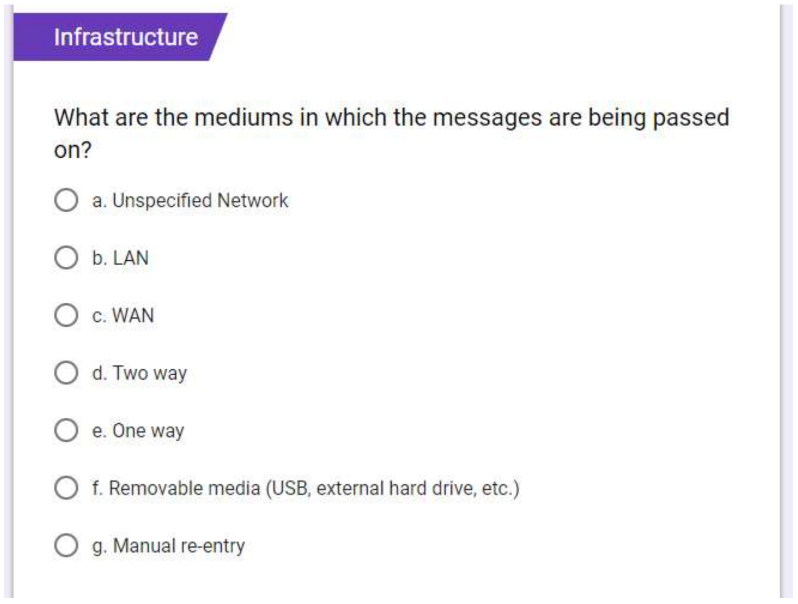
Policy and Procedure

What are the communication standards for all channels in which the data passes through?

- ○ a. Follows a custom format for each channel of communication
- ○ b. Set standards across communication channels
- ○ c. Established training protocols and procedures for message infrastructure
- ○ d. In compliance with a common operating environment
- ○ e. In compliance with other UC Davis ICU information systems
- ○ f. Hospital cross-departmental compliance

Do the communications paths all follow a shared policy or procedure?

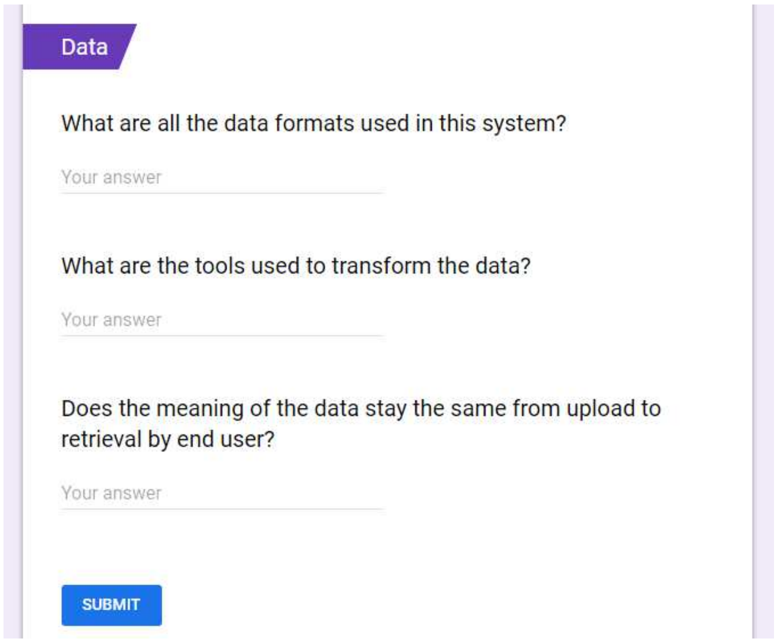
- ○ Yes
- ○ No
- ○ Partially



Infrastructure

What are the mediums in which the messages are being passed on?

- ○ a. Unspecified Network
- ○ b. LAN
- ○ c. WAN
- ○ d. Two way
- ○ e. One way
- ○ f. Removable media (USB, external hard drive, etc.)
- ○ g. Manual re-entry


Data

What are all the data formats used in this system?

What are the tools used to transform the data?

Does the meaning of the data stay the same from upload to retrieval by end user?