Design and implementation of a safe bi-directional document exchange system in health information systems.

by

Aakash Tyagi
B.Tech., Meerut Institute of Engineering and Technology, India, 2016

A Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In the Department of Computer Science

Design and implementation of a safe bi-directional document exchange system in health information systems.

by

Aakash Tyagi
B.Tech., Meerut Institute of Engineering and Technology, India, 2016

Supervisory Committee

Dr. Jens Weber, Supervisor
(Department of Computer Science)

Dr. Neil Ernst, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Jens Weber, Supervisor
(Department of Computer Science)

Dr. Neil Ernst, Departmental Member
(Department of Computer Science)

ABSTRACT

Present-day health care systems are information-dense and progressively relying on computer-based information systems. Unfortunately, many of these information system's use is predominantly restricted to the collection and retrieval of patient records. The failure to digitally communicate and transfer medical record information to other health information systems is undoubtedly a major barrier to efficient patient care and other clinical decision making, and the interoperability between heterogeneous systems still remains a challenge even with decades of investment. The implementation of a safe bi-directional document exchange system is a part of enhancing interoperability by providing a safe messenger system to support some of the core interaction medical documents. Lack of interoperability can also jeopardize patient safety and led to technology-related medical accidents. Hence, the project also focuses on the safety aspect of the system and provides an in-depth hazard analysis along with the implementation of a safety monitoring system to notify users about unexpected system failures.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Jens Weber for his valuable guidance and extraordinary support throughout my master's program. I personally thank him for believing in my skills and providing me the opportunity to learn and contribute to the real world.

Oscar Costa for his support throughout and sharing his knowledge.

My family and my friend **Mona Malik** for motivating me and believing in me.

DEDICATION

This project is dedicated to my parents.
For their endless love, support and encouragement.

Chapter 1

Introduction

Medical practitioners are continuously dealing with large amounts of sensitive patient record data to perform their jobs. The implementation of Electronic Medical Record systems over the past two decades has changed the way clinicians treat patient record data, displacing old paper-only record systems in their practices. Electronic Medical Record systems save time and money, and allows for faster record retrieval, improving clinician's productivity and treating more patients more effectively. With the increase of technology in the health care sector the need to exchange data between heterogeneous systems has also increased and without interoperability, the great promise of Electronic Medical Record systems, meant to assist patients, lead to enhanced coordinated care, and lower healthcare costs, will never be achieved [1].

1.1 Electronic Medical Records

Electronic Medical Records (EMRs) are the digital equivalents of paper records or charts in a doctor's office. EMRs typically provide general information about a patient, such as treatment and medical history. EMR systems are responsible for maintaining and handling patient records. Information such as demographics, prescriptions, allergies, family history, laboratory test reports, patient's age and weight, risks, and billing details are all stored in EMRs. There is a big jump in the number of practitioners adopting EMRs. According to the Canadian Medical Association's (CMA) Workforce Survey [2], EMR use among Canadian primary care physicians is at 85 percent as of 2017 which was at 77 percent in 2014 and 25 percent in 2007[3]. The benefits of using EMRs can be summarized as *"optimizing the documentation of patient encounters, improving the communication of information to physicians, improving access to patient medical information, reduction of errors, optimizing billing and improving reimbursement for services, forming a data repository for research and quality improvement, and reduction of paper"* [4].

However, since practitioners deal with sensitive patient record data, patient safety is now reliant on EMR's reliability and capacity to alleviate any potential failures. EMRs are more than just a replacement for paper records and their true potential cannot be realized if they are just being used as a digital record storage system, EMR systems must also provide some form of "meaningful use" [5].

1.2 OSCAR EMR

The Open-Source Clinical Application and Resource (OSCAR) is a fully featured web-based EMR software program, designed by doctors for doctors, for use in medical offices. The OSCAR project was started by the Department of Family Medicine at McMaster University in 2001 to create a

state-of-the-art web based EMR to support diverse academic and clinical functions. OSCAR is a legacy system that has grown into a comprehensive EMR and billing system and is used by many doctor's offices and private medical clinics in Canada and other parts of the world. Most OSCAR users are in Canada, especially in Ontario and BC. The accurate number of Canadian users is probably higher than expected because OSCAR is a free open-source project and doctors can self-install it [6].

OSCAR is mostly written in Java and JavaServer Pages (JSP) served via Apache Tomcat servlet container. OSCAR relies on the apache maven tool for compilation and packaging. Maven is also used for style checking, running unit tests, running integration tests, generating Javadoc, and other reports [7]. The backend storage is managed by MySQL database and Hibernate is used as an interface layer between Java and MySQL. Several frameworks used in OSCAR web application are Struts, Spring, JPA, Angular and jQuery.

1.3 Interoperability

“The problem is not the systems per se, but rather the difficulty of data exchange between systems, or data incompatibility.” [8]

Interoperability is the ability of heterogeneous information systems to communicate effectively without compromising the content of the transmitted data. The widespread growth of EMRs in Canada has resulted in a complex ecosystem of EMR systems in various regions and provinces. The availability of a large number of different EMR systems, has resulted in a high degree of heterogeneity in terms of communication protocols, supported interfaces, and messaging formats, and many of these systems lack interoperability. As a result, EMR use is primarily restricted to the collection and retrieval of patient records. Although EMRs have advantages over paper-based systems, the failure to digitally communicate and transfer medical record information to other EMR systems is undoubtedly a major barrier to efficient patient care and other clinical decision making.

The absence of interoperability between heterogeneous EMR systems obstructs different workflow tasks and negatively impacts time efficiencies. Zhou et al.'s simulation model shows that high levels of EMR interoperability reduce time spent on tasks such as preparing lab reports, requesting lab orders, electronic prescription of medications, and requesting a consultation from a specialist thus improving the medical practitioner’s efficiency. When different EMRs are interoperable, it becomes possible for health care providers and other stakeholders to share valuable digitized information within and between organizations to care for their patients. Therefore, interoperability is considered a key element of meaningful use [27], [9]. For a patient, medical practitioners generally have to send a ‘referral’ for a consultation to get specialist advice on the matter. The ‘referral’ will contain some sensitive patient information as well. The recipient specialist will schedule one or more appointments with the patient and report back the data to the requesting practitioner. If the systems are not interoperable then each system will have its

own semantic and syntactic understanding and the practitioners will have difficulties sharing data and data sharing will be restricted to very high-level information [10].

1.4 Safety in EMR

Safety has been defined as “freedom from those conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment” [11].
The growing use of EMRs in clinical practice has sparked worries about their safety implications [12] and since large-scale EMR systems are installed across multiple facilities within a healthcare sector, and across a large geographic region, the potential impacts of an EMR failure become far more concerning. Miscommunication and lack of interoperability between different EMR systems may also jeopardize patient safety and can lead to technology-related medical accidents. To identify safety hazards in systems, a variety of hazard analysis methodologies have been proposed. We have implemented the system-theoretic hazard analysis method STPA (*System-Theoretic Process Analysis*) [13] for this purpose.

1.5 Terminology

Before we go any further, let us begin by outlining some terminology used throughout the report.

1.5.1 HL7 and CDA

Health Level Seven (HL7) is a framework for exchanging, integrating, distributing, and retrieving electronic health data. These standards specify how information is packaged and shared from one participant to another, as well as the language, structure, and data types that are required for smooth system integration. HL7 standards are the most widely used in the world and support medical practice, implementation, and assessment of health services [14]. A commonly used exchange format is the Clinical Document Architecture (CDA) which is a flexible markup standard.

CDA is a document markup specification that defines the structure and semantics of "clinical documents" for the purpose of exchange between healthcare providers and patients. A CDA may contain any form of clinical content; for example, a Discharge Summary, Imaging Report, Admission & Physical, Pathology Report, and its most common use is for inter-enterprise data sharing [15].

1.5.2 CDX

The *Provincial Health Services Authority (PHSA)* has collaborated with other health authorities to introduce CDA-based interoperability between clinical information systems, including primary care EMRs. This initiative uses the *Clinical Document Exchange (CDX)* communication system (https://bccdx.ca) [16]. The PHSA has worked with EMR vendors to implement CDX-based interoperability. CDX is a clinical document distribution service developed by Interior Health. CDX

allows for the delivery and exchange of expanded types of clinical documents with the potential for richer clinical content and detail [17]. It also facilitates the EMR-to-EMR data exchange and the vision of CDX is "*to safely and securely enable standards-based meaningful health information exchange.*" [18].

1.6 Outline

This section provides the map of the project report to get a fair idea about the structure of the report.

Chapter 1 introduces the interoperability and safety issues; describes the terminology that is used throughout the report.

Chapter 2 outlines the different functional requirements and quality attributes like usability, safety and, maintainability to be implemented.

Chapter 3 describes the system design and explains each component of the system architecture.

Chapter 4 explains the implementation details and design decisions for the solution.

Chapter 5 discusses the in-depth Hazard analysis of the implemented system and obtained results.

Chapter 6 introduces and explains the design details of the safety notification system.

Chapter 7 discuss the system validation and testing strategy.

Chapter 8 concludes the solution and provides future work direction based on our solution.

Chapter 2

Requirements

2.1 Functional Requirements

OSCAR is an open-source EMR system, used by doctors and other front-line health care professionals. Its main features include patient records, appointment management, prescription module, medication, Eforms and messaging. OSCAR is a legacy system that has grown into a comprehensive EMR and billing system, and even when it is used by many doctor's offices and private medical clinics in Canada and other parts of the world, it does not have any tool or feature to support safe EMR to EMR data exchange for some of the core interaction types. A safe messenger system was required in OSCAR EMR to support the exchange of clinical documentation and basic clinical workflow. The project was focused on the implementation of a CDX-based messenger system in the OSCAR to support safe EMR to EMR clinical document exchange for some of the core document types like Information request, Progress update and Clinical summary.
Some of the core document types are listed below. These are all CDA Level 1 documents, whose type is specified by way of a LOINC (Logical Observation Identifiers Names and Codes) [28] and who all have the same structure.

Send	**Receive in return**
eReferral	Notifications – accept/reject, triaging, scheduling Clinical report or discharge summary
Advice request / eConsult	Notifications – accept/reject Clinical report
Progress update	Could be in any direction
Clinical summary	Could be in any direction
Clinical report	Could be in any direction
Discharge summary	A type of clinical report
Care plan	Could be in any direction
Information request	Progress update, clinical summary, care plan
ADT notification	Admission, discharge notifications
General-purpose notification	Examples include accept/reject, triaging, scheduling; could be in any direction
General-purpose document	Could be in any direction

Table 1. Core Interaction types [19]

After the requirement gathering phase, some of the high-priority functional requirements identified were as following, user-stories were created and tracked for each requirement.

- The system shall be able to send information requests, progress updates, and clinical summaries to other CDX-enabled EMR systems.

- The system shall be able to respond to the received information requests, progress updates, and clinical summaries from other CDX-enabled EMR systems.

- The system shall provide a way to view the history of all received and sent CDX documents and how they are related.

- The system shall be able to respond to other CDX documents like e-referrals and advice requests received from other CDX-enabled EMR systems.

- The system shall indicate the detailed delivery status of any sent CDX document.

- The system shall support the ability of the end-users to send CDX documents to multiple primary and secondary recipients.

- The system shall support the ability of the end-users to select specific clinics to route CDX documents to.

- The system shall support the ability of the end-users to attach multiple attachments to the sending CDX document.

- The system shall support the ability of the end-users to send text content and attachments together.

2.2 Quality Attributes

2.2.1 Usability

Some of the key usability requirements were as following.

- The system shall be easy to use for new or infrequent users with no to few training requirements.

- The system shall support the ability of the end-users to perform any task clearly and unambiguously with the minimum number of clicks.

- The system shall have a fluid and flexible navigation to and from panels or displayed pages.

- The system shall make use of visual indicators such as colours to better present safety information to users.

- The system shall provide consistent user interface standards or conventions throughout the system.

2.2.2 Safety

As EMR systems deal with the sensitive patient data, patient safety is now dependent on the EMR system's ability to deal with hazards and failures. The purpose of the following safety requirements is to ensure that system failures do not cause any severe damage to the system that might also affect patient safety. Some of the key safety requirements for the project were as following.

- The system shall be able to record any detectable failure within the system boundary as soon as it happens.

- The system shall provide failure causes and details to the users for mitigation.

- The system shall operate with no to minimal data loss.

- The system should have a monitoring tool to let users know about any potential failure or system warnings so the users can act on them immediately.

2.2.3 Maintainability

Software maintenance is the most expensive phase of development, usually consuming more than half of the development budget. It is essential to plan maintenance into the development lifecycle so we can maintain software efficiently [20]. Some of the key Maintainability requirements for the project were as following.

- Achieve modularity by fragmenting the application into different components so that components can be reused across the application.

- Achieve reusability by making different code library classes generic enough to be used easily in different application modules.

- Achieve modifiability by making it easy to add code to the existing system and easy to upgrade for new features and new technologies.

2.3 Constraints

Some of the key constraints for the project were as following.

- The new messenger system should be implemented inside OSCAR by extending its codebase.

- The new messenger system should make use of the already implemented CDX interoperability infrastructure in OSCAR.

- The new messenger system should make use of OSCAR's inbuilt inbox to hold received clinical documents.

Chapter 3

Design

3.1 System Architecture

Figure 1 below shows a component & connector view of the implemented architecture [26].

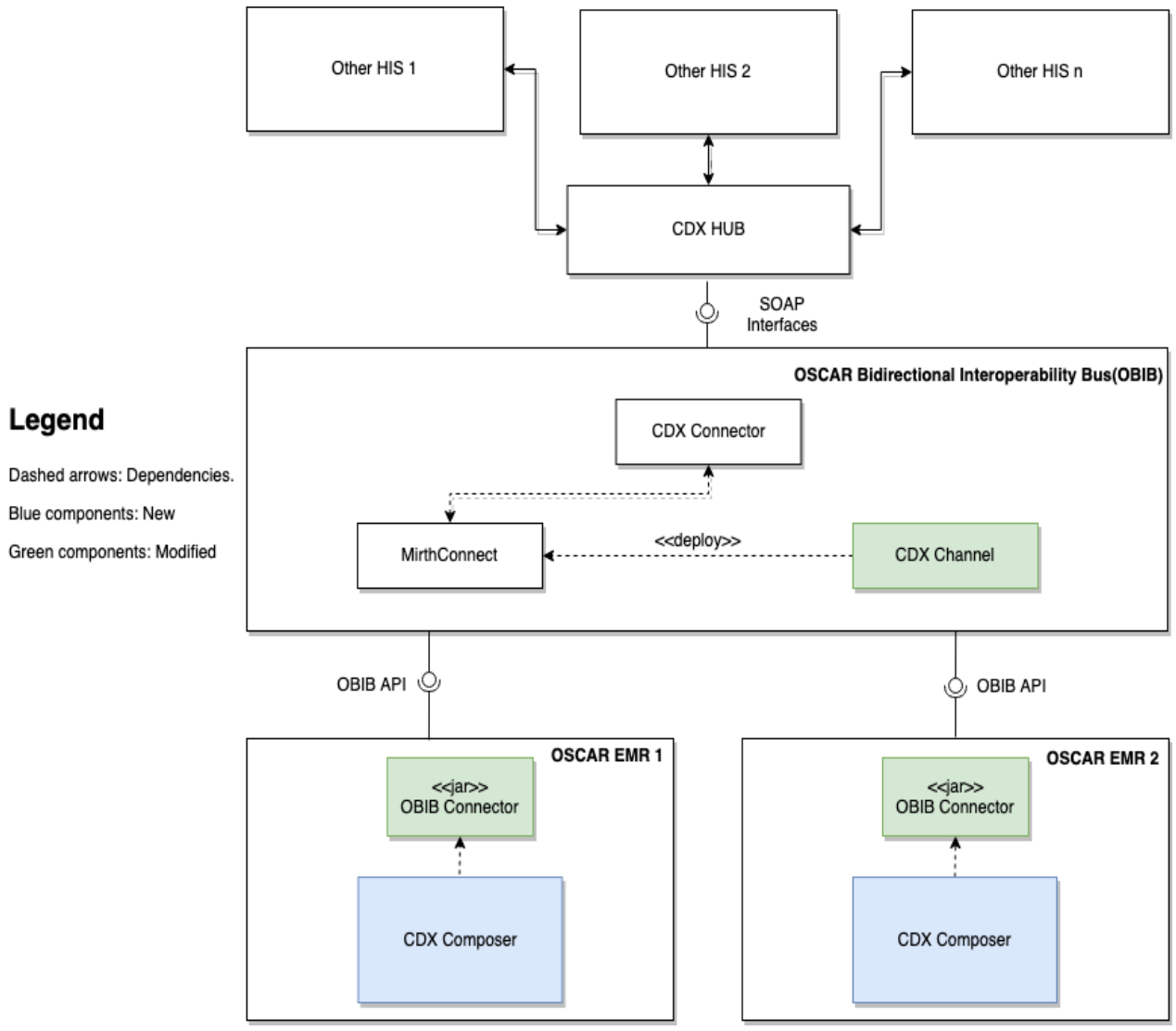

Figure 1. System Architecture

3.1.1 OSCAR EMR

OSCAR is mostly written in Java and Java Server Pages (JSP) served via Apache Tomcat servlet container. OSCAR relies on the apache maven tool for compilation and packaging. Maven is also used for style checking, running unit tests, running integration tests, generating Javadoc, and other reports [7]. The backend storage is managed by MySQL database and Hibernate is used as an interface layer between Java and MySQL. Several frameworks used in the OSCAR web application are Struts, Spring, JPA, Angular and jQuery.

- **CDX Composer**

  The objective was to implement the CDX-based messenger system in the OSCAR to support safe Bi-directional information exchange. We call this CDX-based messenger system "**CDX Composer**" as it is similar to an email tool which let users create and compose a different type of CDX documents, and it makes use of OSCAR inbuilt inbox to receive documents. CDX Composer allows users to create, manage and send clinical documents safely. It makes use of some of the underlying OSCAR functionalities and provides safe bi-directional information exchange between various EMRs. CDX composer connects to the CDX message distribution system via the Oscar Bidirectional Interoperability Bus (OBIB) as shown in figure 1.

- **OBIB Connector**

  OBIB Connector is a Java archive (JAR) made available to the OSCAR. OBIB Connector contains all the functionality required by CDX Composer to interact with the OBIB. The OBIB connection is configured utilizing two artifacts on the OSCAR server: an obibconnector properties file and an obibconnector KeyStore file. The OBIB Connector connects with the OBIB using a secured REST API that is provided by the cross-platform interface engine NextGen Connect [16].

3.1.2 OBIB

OSCAR EMR instances connect to the CDX message distribution system via the *Oscar Bidirectional Interoperability Bus* (**OBIB**). An OBIB can support multiple OSCAR instances. Therefore, an OSP only needs to operate one OBIB. The **OBIB** provides interoperability between the CDX communication system and OSCAR through different sets of services. These services are implemented through Web Services and RESTful clients and NextGen Connect channels. Besides, OBIB is embedded in a Vagrant Virtual Machine (VM) [16].

- **NextGen Connect**

  NextGen Connect is a cross-platform interface engine used in the healthcare industry that supports the management of information using bi-directional sending of many types of messages [21]. NextGen Connect provides standards-based tools to develop, test, and deploy interoperability solutions for healthcare information systems and information exchanges [22]. A NextGen Connect channel is an interface that routes, filters, and transforms messages from one source to one or many destinations.

- **CDX Channel**

  The server side of OBIB is implemented based on the interoperability engine NextGen Connect. NextGen Connect channels receive messages from OSCAR, convert those messages to CDA documents (CDX HL7) and transfer them using Web Services to CDX Hub. Multiple CDX Channels are deployed on NextGen Connect to create and receive CDX documents.

  OBIB services channel receives JSON messages via a REST API, sends these messages to the correct destination, transforms the JSON messages to CDA Documents and sends the CDA documents (CDX HL7) messages to the CDX Web Services using the CDX Connector Java Library. **CDA document parser** channel transforms CDA Documents to OBIB JSON Documents. **CDA Registry Parser** channel converts the CDX XML registry messages (clinics and providers) into OBIB JSON messages. **OSP Support** channel receives and stores messages and forwards them through email. **Document Storage** channel stores the metadata of the documents sent to CDX and the metadata of the CDX responses in the OBIB database [16].

- **CDX Connector**

  NextGen Connect communicates with the CDX infrastructure through secured SOAP interfaces. This connection is implemented in the CDX Connector component. CDX Connector is a Java library that helps OBIB to connect to various CDX Web Services.

3.1.3 CDX Hub

CDX is an interface that supports the sharing of a wide range of clinical documents between various EMR systems. *Clinical Document eXchange (CDX)* communication system has a

centralized distribution service where submitting parties submits content to a centralized location and consuming parties request new content on a scheduled or triggered basis where a consuming/submitting party represents a location that one or more provider's work.

Figure 2. Exchange of CDX Documents between EMRs

Chapter 4

Implementation

Some of the work has already been published in the User Manual for CDX Composer.

4.1 Technology Stack

The CDX Composer makes use of some frameworks and interfaces like Spring, Hibernate and, Struts to store and manipulate health data. It accesses the database through the Java Persistence API (JPA). JPA then translates the information into usable POJOs through Data Access Objects (DAO). Finally, the front-end interface was developed using web technologies like HTML, JavaScript, CSS, Bootstrap, and jQuery and is presented through Java Server Pages (JSP) which is rendered on the practitioner's browser. All these frameworks must work together in order to provide OSCAR EMR users with a smooth experience.

4.2 User Interface

The user interface was developed by taking constant feedbacks from OSCAR users including specialists and by utilizing technologies like AJAX, HTML, JavaScript, CSS, Bootstrap, and jQuery.

After the requirement gathering phase, Wireframes were used in the project to get users and client approval on the layout of key pages and the navigation. Some basic wireframes were created initially and then further improved over time. Wireframing is a quick and effective way to identify usability issues early in the design process and can save considerable time and money in the testing and amends phase later in the project.

Unnecessary clicks that add no value to the user experience were avoided as they only make it more difficult for users to achieve their goals. For an instance, instead of having separate "Add" buttons for the primary recipient and secondary recipient, a single "Add" button was used to implement adding both primary and secondary recipients' information into their respective input boxes. Figure 3 shows the main user interface of CDX Composer.

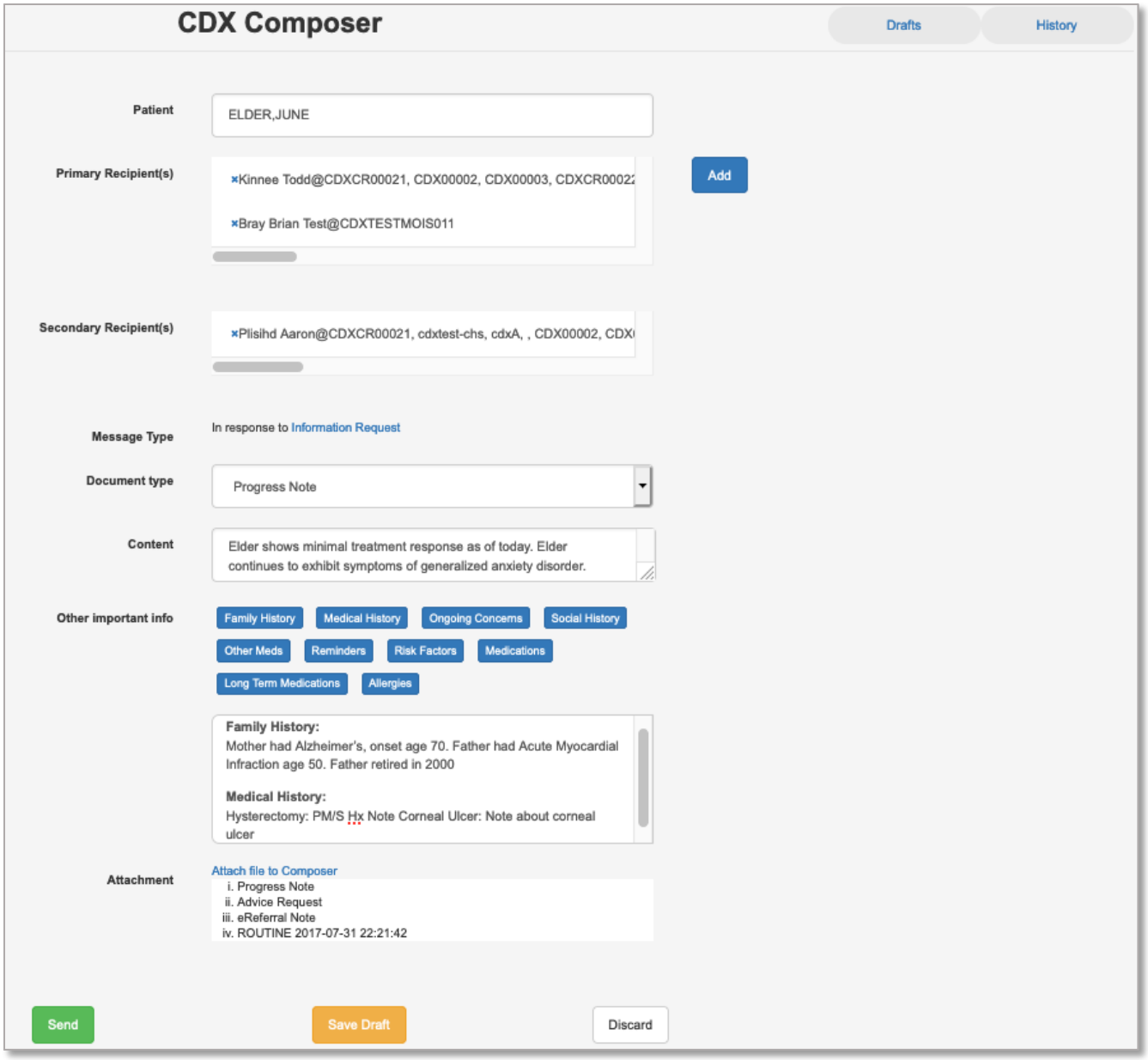

Figure 3. User Interface of CDX Composer

4.5 Controller

The Struts Action class is the Controller as it determines what should happen next in the processing of the request (from the browser). The Action class has an execute method that contains the controller logic. An action class handles the client's request and prepares the response. It also decides where the response should be forwarded. The struts-config.xml contains the routing information that determines which Controller (Action class) the request is

forwarded to. Form data from the JSP page is forwarded to the CDX Composer Action class where data is validated and sent to OBIB using OBIB Connector.

4.6 Features

CDX Composer improves and automates various clinical workflows and provides numerous meaningful features.

- **Different document types**

CDX Composer allows selecting and sending different CDX documents like information requests, progress notes and patient summaries from a single interface.

- **Search and find patient**

CDX Composer allows users to automatically search patients by their first and last name and add patients directly to the document. This feature simplifies the previously implemented workflows in OSCAR of going to the specific patient details page to create or compose documents.

- **Search and find a recipient(s)**

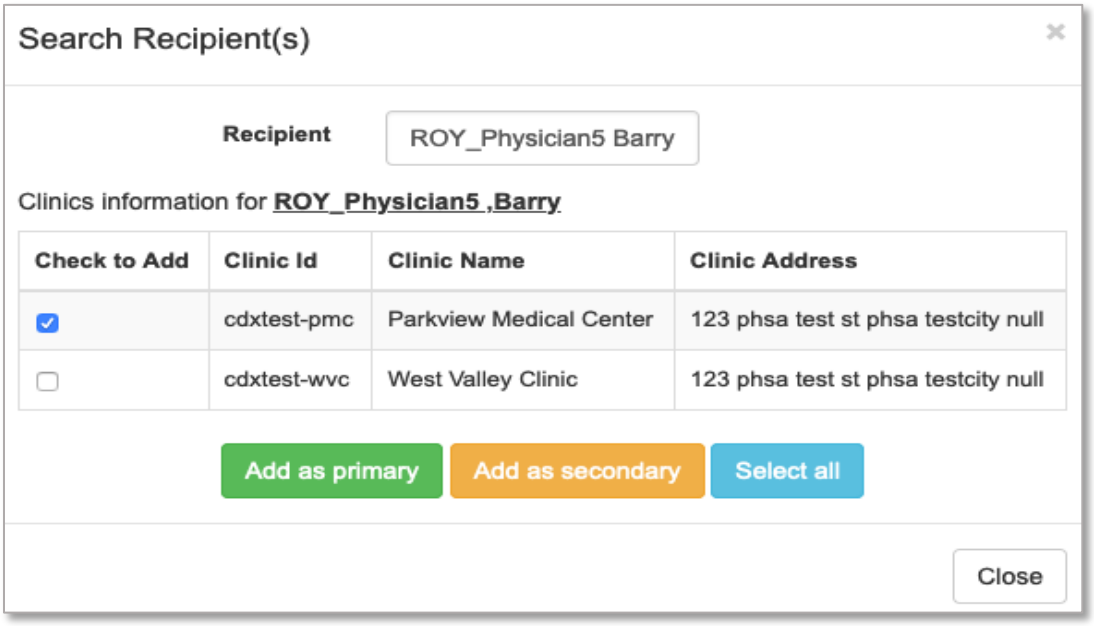

Figure 4. Search and add recipient modal

The CDX infrastructure has a provider registry, which contains address information about all CDX-capable providers and clinics. Even though, OSCAR has traditionally kept its database of specialists. The CDX Composer allows users to query the CDX provider registry and add CDX capable providers directly into the document without using the OSCAR database. By clicking

"Add", Primary Recipient(s) and Secondary Recipient(s) can be directly searched by Name from the CDX provider registry and added to the document as shown in Figure 4. Users can also remove the selected Primary Recipient(s) and Secondary Recipient(s) from the document by clicking the “x” button before the names.

● Selective Message routing

Some providers may work for more than one clinic. In this case, the CDX Composer allows users to select and route the document to specific clinics. As shown in figure 4, users can select all or any specific recipient clinic from the options.

● Auto-populated patient data

CDX Composer provides a way to embed patient-specific data into the document. It provides a set of buttons to automatically include additional patient-specific data like medical history, family history, medications, risk factors, allergies and reminders into the document as content.

● Attachments

CDX Composer allows attachments to be sent along with the document. It provides a way to attach patient-specific documents like lab reports, text documents and CDX documents to the sending CDX document. Also, Attachments and free text (content) can be sent together.

● Document delivery status

As shown in Figure 7, users can view the overall delivery status of the document on the CDX Composer History page. Also, the detailed distribution status of the document (Delivery status for each clinic) can be seen by clicking on the “Show Detailed Distribution Status” button on the detail document view page as shown in figure 5 below.

Distribution Status of the document:

Clinic Id	Clinic Name	Status Code	Status Name	Status Time
CDXTEST0023	George Test Clinic	400.10	UNKNOWN: Status Unknown. Please contact CDX support to investigate.	null
cdx-chaj17	Chaj17 test clinic	200	QUEUED	Wed Dec 16 18:26:00 PST 2020
cdxpostprod-ietcb	Intrahealth E2E Test Clinic B	100	DELIVERED	Wed Dec 16 18:26:00 PST 2020
cdxtest-jtp	Jays testing place	200	QUEUED	Wed Dec 16 18:26:00 PST 2020

Figure 5. Detailed Distribution Status

● **Receive**

CDX Composer utilizes the already implemented inbox feature of OSCAR to receive various CDX documents. Responses received from a specialist (via CDX) will arrive in the receiver's inbox.

● **Reply**

CDX Composer allows the user to respond or reply to the incoming or received documents. Users can open any CDX document from the inbox to see a "Reply" button. After clicking on the reply button, users would be directed to the CDX Composer window with some pre-populated fields. Users can also add more information in the composer and send the document as a response to the incoming document.

● **Drafts**

CDX Composer keeps track of all the CDX document drafts saved through CDX Composer. Users can anytime edit and send the saved document by clicking the "Edit" button in front of the specific saved draft.

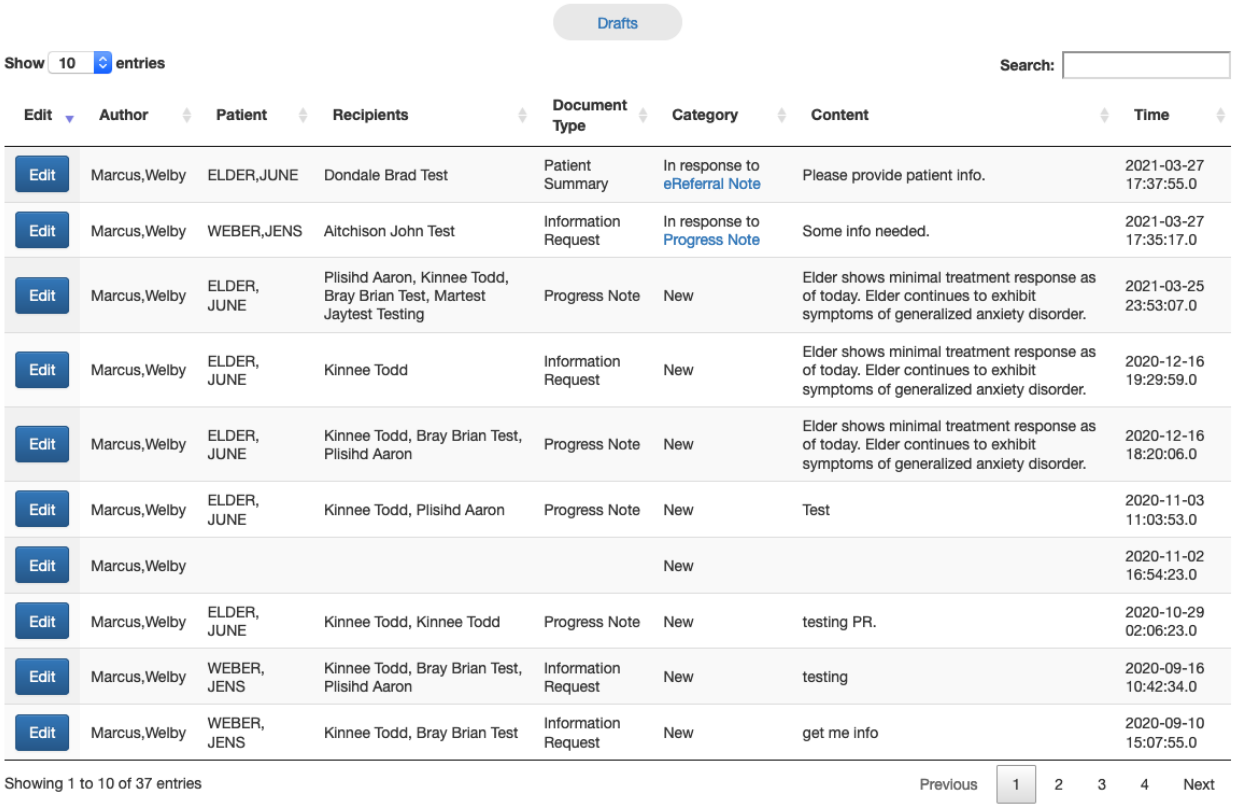

Figure 6. CDX Composer Drafts UI

- **History**

CDX Composer keeps track of all the CDX documents sent through CDX Composer. Users can view the detailed sent document by clicking on the “View” button on the history page as shown in Figure 7. Users can also view the detailed document to which they are responding by clicking the document type link on the CDX composer history page.

History

Show 10 entries

Search:

Document Details	Author	Patient	Recipients	Document Type	Category	Content	Time	Delivery Status
View	Marcus,Welby	ABBOTT, ABIGAIL	Plisihd Aaron	Progress Note	In response to Advice Request	I'll advice for MRI scan. That can provide more information on the case.	2021-03-26 00:28:29.0	DELIVERED
View	Marcus,Welby	ANT, ANNA	Kinnee Todd	Patient Summary	In response to Information Request	Risk Factors involves History of heavy alcohol consumption. Tobacco smoke - Second hand smoke	2021-03-26 00:24:43.0	QUEUED
View	Marcus,Welby	BARBER, ANNE	Plisihd Aaron	Patient Summary	New	Current Medications: Zyban 150mg Take 1 Tablet Po Bid For 12 Weeks Qty:168 Repeats:52	2021-03-26 00:23:05.0	QUEUED
View	Marcus,Welby	AADDBrown, BOB	Aitchison John Test	Progress Note	New	BOB is doing great, recovered well from the COVID-19.	2021-03-26 00:20:46.0	QUEUED
View	Marcus,Welby	WEBER, JENS	Dondale Brad Test	Information Request	New	Can I get the Blood report and signs of any infection for the patient ?	2021-03-26 00:19:26.0	QUEUED
View	Marcus,Welby	APRICOT, APRIL	Aitchison John Test, Chrobot Larry Test	Patient Summary	New	Allergies: Celebrex 100mg Reaction: Celebrex intolerance Peanuts Reaction: Severe Rash all over Body Penicillins Reaction: Mild rash	2021-03-26 00:17:35.0	DELIVERED
View	Marcus,Welby	ELDER,JUNE	Kinnee Todd, Bray Brian Test, Plisihd Aaron	Progress Note	In response to Information Request	Elder shows minimal treatment response as of today. Elder continues to exhibit symptoms of generalized anxiety disorder.	2021-03-26 00:11:12.0	QUEUED
View	Marcus,Welby	ELDER, JUNE	Kinnee Todd, Bray Brian Test	Information Request	New	Mother had Alzheimer's, onset age 70. are there any other similar symptoms?	2021-01-29 13:40:05.0	QUEUED
View	Marcus,Welby	ELDER, JUNE	Kinnee Todd	Information Request	New	Need a recent diagnosis report for Elder june.	2020-12-19 17:37:15.0	QUEUED
View	Marcus,Welby	ELDER, JUNE	Addaprovider test to, Martest Jaytest Testing	Information Request	New	Require weekly updates on elder.	2020-12-16 18:25:11.0	LOST

Showing 1 to 10 of 72 entries

Previous 1 2 3 4 5 … 8 Next

Figure 7. CDX Composer History UI

Chapter 5

Hazard Analysis

Risk and hazard analysis is an important activity for the development and operation of critical software systems such as EMRs systems. To identify safety hazards in systems, a variety of hazard analysis methodologies have been proposed. System-theoretic hazard analysis method STPA (*System- Theoretic Process Analysis*) was used for this purpose. We have specifically implemented and used the extended STPA model for the analysis of an eHealth interoperability conformance profile [23]**.**

STPA (*System- Theoretic Process Analysis*) is based on a model called STAMP (*System-Theoretic Accident Model and Processes*) which treats safety as a dynamic control problem instead of a failure prevention problem. *System Theoretic Process Analysis* (STPA) is a relatively new hazard analysis (HA) technique. In addition to component failures, STPA assumes that the unsafe interactions of system components, none of which may have failed can also lead to accidents. Unlike the conventional hazard analysis methods, STPA can be applied in the early project phases to design safety into the system architecture and design which can eliminate the costly rework if design flaws are identified late in the development [24]. According to some comparisons, STPA found more hazard scenarios as compared to other traditional analysis methods like fault tree analysis (FTA). A system under analysis is modelled as a network of control loops, described as the "*control structure*". The steps in basic STPA are shown in Figure 8.

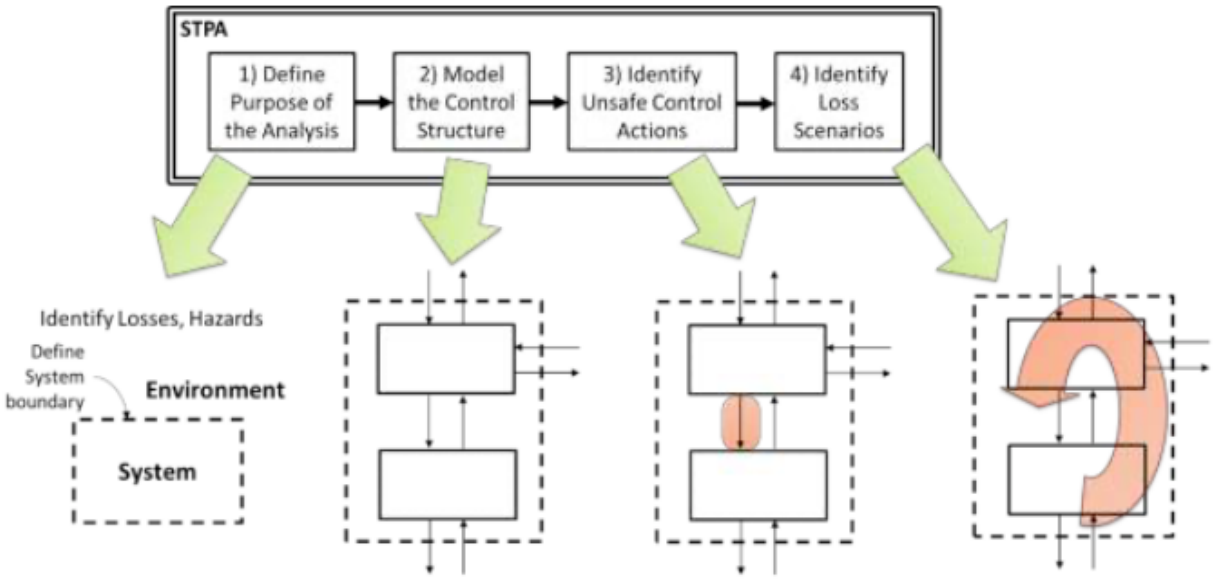

Figure 8. Overview of the basic STPA Method [25]

The first step is to **define the purpose of the analysis**, it deals with identifying *losses* to avoid and a preliminary set of *system-level hazards* and system boundary. Losses may be related to patient harm and anything that has value to the stakeholder. For system boundary, the investigation is limited to the CDX Composer. Table 2 lists the losses and system-level hazards identified during this step.

Losses	System-level Hazards
L1: Patient is harmed as a result of inappropriate intervention by the primary caregiver or the secondary caregiver. L2: Patient is harmed because of missing or delayed intervention by the primary caregiver or the secondary caregiver.	H1: incomplete or incorrect order (may lead to L1, L2) H2: wrong/missing record target (may lead to L1, L2) H3: wrong/missing order recipient (may lead to L2) H4: delayed/missing order (may lead to L2)

Table 2. Scope of the analysis: System-level losses and hazards

The second step is to **model the control structure**, STPA control structures comprise of multiple overlapping and interacting control loops, where each loop contains a *controller* (active component), a *process* (passive component), an *actuator* (used by the controller to influence the process) and a *sensor* (providing the controller with feedback on the status of the process [13].

Figure 9 displays the control structure modelled for the CDX Composer. The decomposed control structure is explained as follows:

- The PCG *creates* documents through CDX Composer.
- The PCG *replies* to a document through CDX Composer after viewing an incoming document in the inbox.
- The PCG checks *history* and *delivery status* of all the sent CDX Documents through the CDX Composer.
- The PCG checks *Draft* of all the unsent saved documents through the CDX Composer.
- The CDX Composer *sends* document using OBIB and the CDX middleware (*Message Routing Controller* – MRC).
- The CDX Composer *gets providers* and *clinics* details capable of receiving CDX messages using the CDX middleware (*Provider and Institution Registry* – PIR).

- The OPC *checks new documents* and *downloads* any document that has been queued for that clinic.
- The SCG *views* a new incoming document through the *Order Processing Controller* – OPC.

Figure 9 shows the control structure for CDX Composer with different actions and feedbacks, exclamation mark (!) and question mark (?) are used for actions and feedback respectively. To manage complexity and make control relationships easy to understand, a solid arrow represents actions and a dashed arrow represents feedbacks.

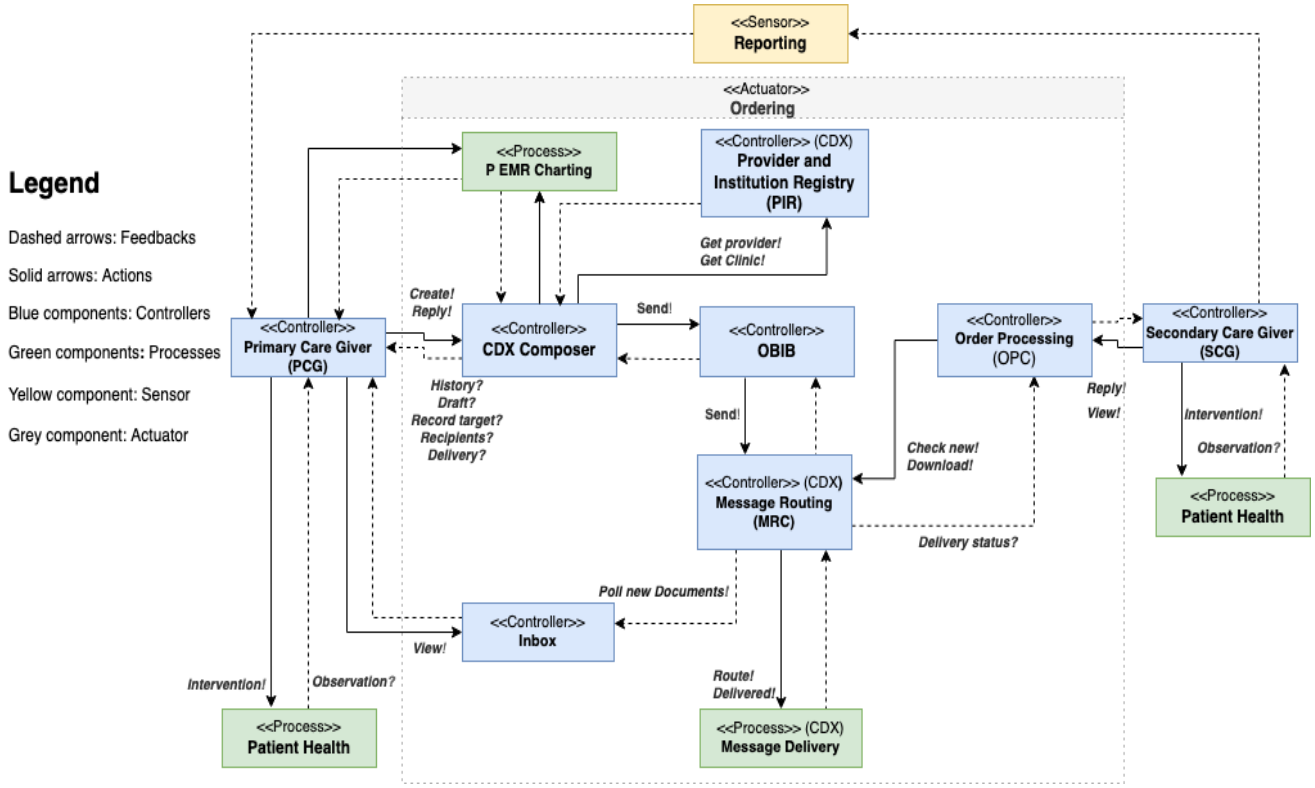

Figure 9. Control Structure with actions and feedback

Once the control structure has been modelled, the next step is to **identify unsafe control actions**. An *unsafe control action* (UCA) is a control action that, in a particular context will lead to a hazard. A UCA contains five parts <Source> <Type> <Control Action> <Context> <Link to Hazards>. For understanding, consider the following UCA.

UCA1: PCG provides create! with mis-identified / ambiguous / missing record target [H2]

- *PCG is a* <Source> which is a controller that can provide the control action.
- *Provides* is a <type> which is the type of unsafe control action (provided, not provided, too early or too late, stopped too soon or applied too long).
- *Create!* is a <Control Action> which is the control action or command itself.

- *With mis-identified / ambiguous / missing record target* is a <Context> which describes any relevant context for this UCA.
- *[H2]* is <Link to Hazards> which is a reference to the related hazard.

UCAs are identified by observing each control action in the control structure with a set of *guide words* like *provided, not provided, too early or too late, stopped too soon or applied too long* to explore circumstances that may lead to hazards.

Once unsafe control actions have been identified, the next step is to ***identify loss scenarios.*** Each loss scenario describes the conditions under which a specific UCA may happen. Two types of loss scenarios are considered, "*why would Unsafe Control Actions occur?*" and "*why would control actions be improperly executed?*". Table 3 lists all the identified UCAs and loss scenarios identified during STPA step 3 and 4 respectively for CDX Composer.

UCA	**Loss Scenarios**	**Controller Constraints**
UCA1: PCG provides *create!* with mis-identified / ambiguous / missing record target [H2]	• PCG picks the wrong patient with a similar name. • PCG has multiple CDX Composers open and creates orders for the wrong name. • PCG enters name / ID incorrectly. • PCG forgets to specify the patient.	CC1: Record target is shown clearly in CDX Composer. CC2: Record target can be searched and added by first and last name. CC3: Record target must be specified.
UCA2: PCG provides create! with incomplete/ incorrect order content [H1]	• PCG forgets to add attachments. • PCG chooses the wrong document type. • PCG forgets to add supporting information like family history, allergies and, medical history to the document. • PCG adds attachments and other supporting	CC4: All Attachments are visible in the CDX Composer. CC5: Buttons to add supporting information such as family history, allergies and medical history must be clearly visible; Populated data is also editable. CC6: All fields are set to default when a patient field is changed.

	information and then changes the patient in the CDX composer.	
UCA3: PCG provides create! with wrong / missing order recipient [H3]	• PCG picks the wrong recipient with a similar name. • PCG has multiple CDX Composers open and creates orders for the wrong recipient name. • PCG enters name / ID incorrectly. • PCG forgets to specify a recipient. • PCG chooses the wrong clinic location. • PCG chooses non-CDX recipients and clinics. • PCG forgets to specify at least one clinic location. • PCG forgets to specify at least one primary recipient.	CC7: All primary and secondary recipients are queried directly from the CDX registry; the Recipient's name and ID are clearly visible in the CDX Composer. CC8: Recipient's clinics name, ID and address are clearly visible; PCG can pick and choose clinics to route the document. CC9: PCG can remove recipients and clinics. CC10: At least one primary recipient is added to the document. CC11: At least one clinic is selected for a recipient.
UCA4: PCG provides reply! with mis-identified / wrong record target [H2]	• PCG has multiple incoming documents opened for different patients and replies to the document with different patient names.	CC1: Record target is shown clearly in CDX Composer. CC12: Record target cannot be removed while replying.

	• PCG changes the record target and enters the name / ID incorrectly.	
UCA5: PCG provides reply! for mis-identified document [H1]	• PCG has multiple incoming documents opened and selects the wrong document to reply.	CC13: The type of document to which the PCG is replying is clearly visible in the CDX composer; the Detailed document to which the PCG is replying can be seen clearly by clicking the link in the CDX Composer.
UCA6: PCG stops providing create! too soon, before providing send! for it [H4]	• PCG creates order but inadvertently exits before sending it (e.g., a browser window is closed too soon). • PCG creates order but wants to send it at some other time. • PCG creates an order, attempts to send it but the CDX infrastructure cannot be contacted or is slow; PCG dismisses the dialog assuming that it will be sent later.	CC14: PCG can save the order draft and send it later. CC15: All unsent orders are logged with the errors and are visible to the PCG.
UCA7: OPC /MRC does not provide delivery status! [H4]	• The document is lost in the abyss. • OBIB is not connected to the OSCAR. • CDX infrastructure is slow or down.	CC16: History is maintained for all the sent documents. CC17: 'LOST' status is clearly visible for the sent documents in the history for which no delivery status is retrieved.

		CC18: Real-time OBIB connection status is clearly visible to the user. CC19: CDX related issues are logged and clearly visible to the user.
UCA8: CDX Composer does not provide get provider! or get clinic! [H3]	• OBIB is not connected to the OSCAR. • Provider and institution registry (PIR) is down. • PCG chooses non-CDX providers and clinics.	CC18: Real-time OBIB connection status is clearly visible to the user. CC20: CDX composer query only CDX-enabled providers directly to the Provider and institution registry (PIR).

Table 3. Unsafe control actions, loss scenarios and controller constraint

Table 3 shows an excerpt of the results of this analysis activity. 8 major UCAs were identified by using the STPA guideword technique. Each UCA is elaborated with one or multiple loss scenarios, describing situations where these UCAs may occur. 20 Controller constraints were identified to prevent the maximum number of loss scenarios identified during step 4. A controller constraint specifies the controller behaviours that need to be satisfied to prevent UCAs. In addition to the System Theoretic Process Analysis, a safety notification system was also implemented in the OSCAR to ensure system safety which is explained in the next chapter.

Chapter 6

Safety Notification System

Some of the work has already been published in the User Manual for the Safety notification system.

6.1 Overview

The safety notification system is part of safety and hazard analysis, to improve the communication and ongoing operations of the system. OSCAR is a legacy system that uses only the log files to keep track of all the unexpected events in the system, logging is undeniably a foundational practice that delivers considerable benefits during the lifespan of the application. However, logging does not provide an effective feature to notify users in real-time and has some significant drawbacks as well, including excessive log data and cryptic log messages. In order to provide users with the real-time system status with brief but comprehensive information, the safety notification system was implemented in OSCAR EMR for CDX interoperability workflows. The safety notification system provides the user with the real-time status of the OSCAR by displaying the colour-coded lights, and detailed information on the unexpected scenarios and errors in the system related to the CDX and OBIB. It also provides users with a feature to ignore notifications when required.

6.2 Design and Implementation

The scope of the safety notification system is limited to the Implemented CDX interoperability workflows in the OSCAR. Figure 10 shows the basic architectural overview of the safety notification system and how the different components interact with each other [25].

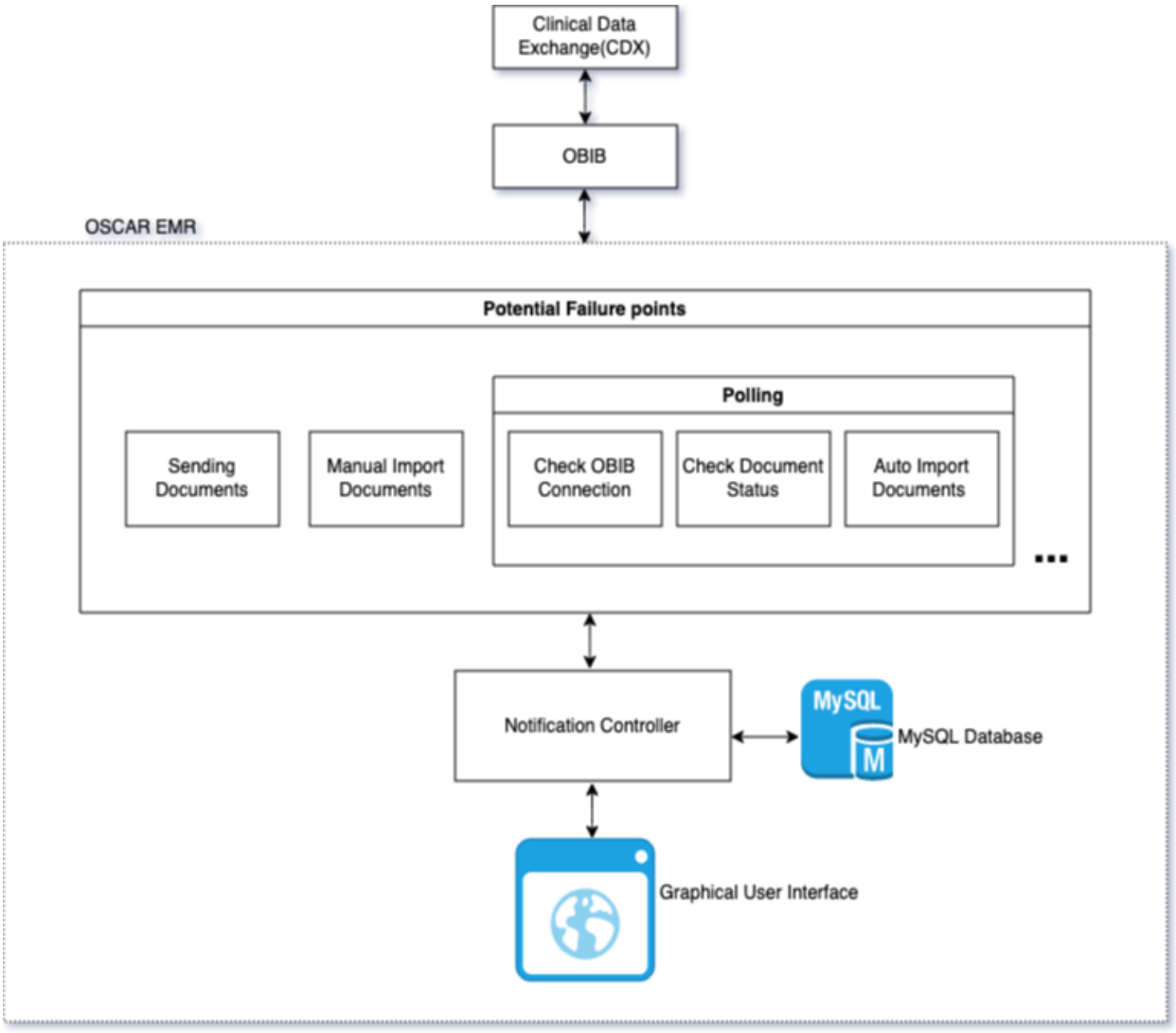

Figure 10. Basic architectural overview of the safety notification system.

6.2.1 Identified Failure Points

To build an effective safety notification system, the first step is to detect vulnerabilities in the system that may lead to failures in certain situations. Hazard analysis for CDX Composer with System Theoretic Process Analysis (STPA) helped in identifying most of the loss scenarios and generating controller constraints. It was needed that the maximum number of failure points are identified in the early development phase. In OSCAR, various potential failure points were detected, some of them are mentioned below.

1. Sending documents

As sending clinical documents to the CDX system is one of the major and often used features of the OSCAR EMR, it is more susceptible to failure. Multiple Java functions were tested and monitored to identify the vulnerable pieces of code. Earlier, there was no mechanism in place to let users know about the status of the sent document, documents were often lost in transit due to several reasons. But now with the safety notification system in place, users will be notified about related system errors or warnings generated during the sending of CDX documents.

2. Manual import of documents

The functionality of manually Importing or receiving documents sent by another EMR system was identified as another potential vulnerability point in the system. Earlier, there was no mechanism in place to let users know about the status of the imported document, documents were often missed when importing to the inbox due to several reasons including unsupported document type. But now with the safety notification system in place, users will be notified of the relevant error or warning information generated during the manual import of documents.

3. Check OBIB connection

This is a case of handling the high-severity failure point. A ‘job‘ was Implemented in Java, the task of this ‘job’ was to periodically check the connectivity of OSCAR EMR with the OSCAR Bidirectional Interoperability Bus (OBIB) in order to provide the real-time status of OBIB connection to the OSCAR users. This is one of the high severity failure points in the system as several other critical workflows are dependent on OBIB to connect to the CDX system. If OBIB is not connected to the OSCAR, then the state of the OSCAR will be set as ‘critical’ and the users will be notified at the earliest.

4. Check document status

The implemented Java ‘job’ queries the CDX system periodically to check the status of all the ‘SENT’ documents that are not yet delivered in order to notify OSCAR users about the real-time status of the documents. There can be a possibility that a critical ‘SENT’ document is lost in the abyss forever, then with the implementation of this job, users will be notified about that document so that the document can be sent again.

5. Auto Import documents

The administration panel in OSCAR provides an option for an admin to set the polling time and a checkbox to disable the auto-import of the received documents. Disabling the auto-import will also disable some of the OSCAR background processes which are important if not critical, in that case, a warning will be generated, and the users will be notified about the impact of disabling auto import. This is a case of handling the low-severity failure point.

6.2.2 Notification Controller

Notification controller acts as a hub for notifications and is implemented using Java, JSP (Java Server Pages), Struts, Spring and Hibernate framework. Notification controller is responsible for tasks like database insertions, deletions, exception handling and, changing the status of the system. The notification controller inserts entries in the database make a change to the status of the system and provide error/warning information to the users. Similarly, the notification controller deletes entries from the database, once the system is recovered from the failure or the persisting problem is fixed and displays the changed status of the system to the users.

6.2.3 Database

OSCAR EMR system uses a MySQL relational database management system. For the safety notification system, a table was created to store and display notifications with all the required fields. The table contains various columns, some of the columns are listed below.

- **Id:** a unique identifier.
- **Type:** type of notification (Error/Warning).
- **Message:** specific cause of the failure.
- **Generated at:** timestamp of the event.
- **Generated during:** a specific failure point.
- **Category:** type of event that caused the failure (Sending, importing)
- **Code:** code specific to the exception.
- **Seen by:** name of the user to dismiss or ignore the notification.
- **Seen at:** time at which notification wad dismissed or ignored.

Concise and specific messages are stored in the ‘message’ column. More detailed messages can still be retrieved from the log files when required.

6.2.4 Notification User interface

The notification user interface is developed using web technologies like HTML, JSP, CSS, JavaScript, and jQuery. Multiple user interfaces were designed and discussed before selecting a suitable one.

1. Notification Types

   Two types of notifications were generated namely, error and warning.

   - **Error** is a type of notification generated by the high-severity failure points of the system which are critical, and any failure or exception can lead to the failure of other critical components or functions of the system. For instance, when OBIB is not connected to the OSCAR.

   - **A warning** is a type of notification generated by the low-severity failure points of the system which are not critical, but any failure or exception can lead to the failure of some non-critical component. For instance, automatic polling of CDX documents is disabled.

2. Notification Lights

   Notification lights are implemented using JavaScript, and CSS. Notification lights are colour-coded and display the current state of the system to the user. Color-coding is listed below.

   - **Green:** The system is working fine without any error/warning.

   - **Yellow:** The system is working fine with some warnings which might require some user attention.

   - **Red:** System is not working fine and requires urgent user attention.

   Notification light changes colour according to the notifications in the database. A green light will be displayed if there are no entries in the database or if all the errors and warnings are dismissed by the user. A yellow light will be displayed if the database contains entries for warning only. In the case of both error and warning entries in the database, a red light will be displayed. Lights are also set to blink for a certain period of time to get the user's attention, once the status of the system is changed.

3. Notification Display Information

Notification information is a JSP page that displays the information about the notifications to the user as shown in figure 11. Users are also provided with functionality to dismiss or ignore any notification with a **'check to ignore'** checkbox. If a user chooses to dismiss any notification, then the selected notifications will be removed from the display and time of dismissal, and the name of the user will be logged in the database to keep track of the dismissed notifications in future.

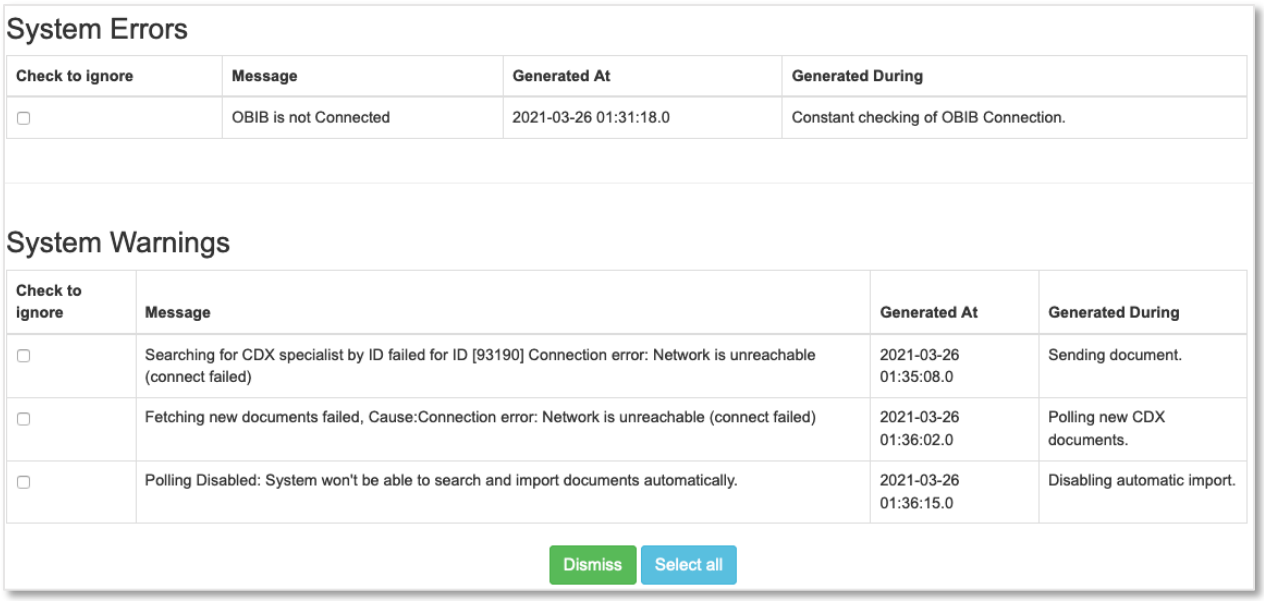

Figure 11. A user interface for notification information

Chapter 7

Evaluation

Referring back to the requirements outlined in chapter 2, we require CDX Composer to be usable, safe, and, maintainable.

- In order to satisfy **usability**, CDX Composer's features were built around users by taking constant feedback from the different clinics and specialists. OSCAR test users have found CDX Composer UI, Safety notification system's colour-coded lights and display UI as compelling and easy to use.

- In order to satisfy **safety**, a Hazard analysis model like System Theoretic Process Analysis was applied in the early development phase and controller constraints were identified to overcome loss scenarios and unsafe control actions. Most of the controller constraints were later implemented in the system to avoid any potential failures. In addition to the System Theoretic Process Analysis, a safety notification system was also implemented to ensure safety by providing OSCAR users with the real-time status of the OSCAR by displaying the colour-coded lights, and the detailed information on the unexpected scenarios and errors in the system related to the CDX and OBIB.

- In order to satisfy **maintainability**, a heavy focus was on the software readability, testability and modular software design. CDX Composer is designed to be easy to understand and use, contains a high degree of code test coverage, and is built up of small, easy to modify code segments. OSCAR EMR is a legacy system; hence it is a challenging task to merge new features into the codebase, therefore, some degree of code refactoring was also performed.

Test cases were written in detail to cover most of the inputs, execution conditions and boundary conditions. A mix of both manual and automatic testing was performed. Some of the successful testing activities are listed below.

- Explicitly turned off OBIB to check if the notification light turns red and then back to green as soon as the OBIB connection is back.

- Disabled automatic polling to check if the notification light turns yellow and back to green after enabling automatic polling.

- Tried importing a bad document to check if the notification light turns from green to yellow with the detailed information.

- Sent clinical document from CDX Composer to other EMR system and checked the delivery status changing from Queued to delivered.

- Sent clinical document from CDX Composer to other EMR system and checked the downloaded document structure, content and attachments.

First, static analysis and testing were performed to detect any visible bugs in the early phase of development. After that, automated test scripts were written to test the CDX Composer and safety notification system. The JUnit testing framework was used for both unit and integration testing.

```
@Test(expected = OBIBException.class)
public void testSubmitDocWithAttachmentAndTextBody() throws Exception {
    try {
        // add recipient (provider), attachments and submit the document
        IDocument response = submitDoc
                .recipient().primary().id("11116").name( firstName: "Todd",  lastName: "Kinnee",  prefix: "Dr.",  suffix: "") IRecipient
                .and().receiverId(clinicIdA) ISubmitDoc
                .attach(AttachmentType.PDF,  reference: "logo.pdf", loadFile( filePath: "/leadlab.pdf"))
                .content("CDX Composer test with attachment and content")
                .submit();

    } catch (OBIBException ex) {
        System.out.println("Message: " + ex.getMessage());
        System.out.println("OBIB Message: " + ex.getObibMessage());
        throw ex;
    }
}
```

Figure 12. Junit test code snippet

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In conclusion, the implementation of CDX Composer in the OSCAR EMR is a part of enhancing the capability of EMR systems to digitally communicate and exchange medical record information to other EMR systems for better patient care and user experience. CDX Composer facilitates safe bi-directional information exchange for some of the core interaction types and provides OSCAR users with a single easy-to-use interface to safely compose different CDX documents with all the necessary information and send it to other CDX-interoperable EMR systems. Hazard analysis model like System Theoretic Process Analysis and implemented safety notification system ensures the safety of CDX interoperability workflows in the OSCAR and helps in the recovery of the system. The CDX Composer creates a more complete health record for a patient, which in turn will help a clinician with better patient plans of care and treatment.

8.2 Future Work

Some of the suggested future work directions are listed below.

- CDX Composer provides a solid platform for bi-directional medical record exchange which can be extended in future to support additional CDA level 1 and CDA level 2 documents.

- Currently, CDX Composer uses OSCAR's old inbuilt inbox to receive and hold incoming CDX documents which can later be implemented in CDX Composer with newer front-end technology and framework like react.

- The safety notification system can also be extended for additional OSCAR workflows other than the CDX interoperability workflows.

Bibliography

1. Casarez, C. (2018, June 1). EHR Interoperability: Why Is It So Difficult? -. *Continuum*. Continuum. Retrieved April 7, 2021, from https://www.carecloud.com/continuum/why-is-ehr-interoperability-so-difficult/
2. Permalink. (n.d.). *Q22. Use of electronic records - Canadian Medical Association*. Retrieved April 7, 2021, from https://surveys.cma.ca/en/permalink/survey31
3. Collier, R. (2015, January 6). National Physician Survey: EMR use at 75%. *CMAJ*. CMAJ. Retrieved April 7, 2021, from https://www.cmaj.ca/content/187/1/E17
4. Yamamoto, L. G., & Khan, A. N. G. A. (2006). Challenges of electronic medical record implementation in the emergency department. *Pediatric Emergency Care, 22*(3), 184-191. https://doi.org/10.1097/01.pec.0000203821.02045.69
5. Blumenthal D, Tavenner M. The "meaningful use" regulation for electronic health records. N Engl J Med. 2010;363(6):501–4. https://doi.org/10.1056/NEJMp1006114.
6. ABOUT OSCAR. (n.d.). *OSCAR Canada Users Society*. Retrieved April 7, 2021, from http://oscarcanada.org/about-oscar/brief-overview/index_html
7. Building the code. (n.d.). *Building the code - OSCAR EMR*. Retrieved April 7, 2021, from https://oscaremr.atlassian.net/wiki/spaces/OS/pages/86092990/Building+the+code
8. Mead C. N. (2006). Data interchange standards in healthcare IT--computable semantic interoperability: now possible but still difficult, do we really need a better mousetrap? *Journal of healthcare information management: JHIM*, *20*(1), 71–78.
9. Zhou, Y., Ancker, J. S., Upadhye, M., McGeorge, N. M., Guarrera, T. K., Hegde, S., Crane, P. W., Fairbanks, R. J., Bisantz, A. M., Kaushal, R., & Lin, L. (2013;2014;). The impact of interoperability of electronic health records on ambulatory physician practices: A discrete-event simulation study. *Informatics in Primary Care, 21*(1), 21-29. https://doi.org/10.14236/jhi.v21i1.36
10. Ho, J. (2017, April 25). BXE2E: a bidirectional transformation approach for medical record exchange. *UVicSpace Home*. Retrieved April 7, 2021, from https://dspace.library.uvic.ca/handle/1828/7988
11. Risk and Safety Management. (n.d.). *Standard Practice for System Safety*. Retrieved April 6, 2021, from https://acqnotes.com/acqnote/tasks/mil-std-882e-system-safety
12. Weber-Jahnke, J., & Mason-Blakley, F. (2011). The safety of electronic medical record (EMR) systems: What does EMR safety mean and how can we engineer safer systems? *SIGHIT Record, 1*(2), 13-22. https://doi.org/10.1145/2047478.2047480
13. Leveson, N., & Books24x7, I. (2011;2012;2019;). *Engineering a safer world: Systems thinking applied to safety*. MIT Press.
14. Health Level Seven International. (n.d.). *HL7 International*. Retrieved April 7, 2021, from http://www.hl7.org/implement/standards/

15. Health Level Seven International. (n.d.). *HL7 Standards Product Brief - CDA® (HL7 Clinical Document Architecture) | HL7 International*. Retrieved April 7, 2021, from http://www.hl7.org/implement/standards/product_brief.cfm?product_id=496
16. Introduction. (n.d.). *Introduction :: OSCAR CDX Interoperability User Manual*. Retrieved April 7, 2021, from https://simbioses.github.io/cdxuserman/003_introduction/
17. (n.d.). *Electronic Delivery of Clinical Data*. Retrieved April 7, 2021, from https://www.interiorhealth.ca/AboutUs/Physicians/Pages/POI.aspx
18. (n.d.). *Pages - Home*. Retrieved April 7, 2021, from https://bccdx.ca/Pages/default.aspx
19. Addendum – Report Codes. (n.d.). *Pages - Documents*. Retrieved April 6, 2021, from https://bccdx.ca/Documents/Addendum%20-%20Report%20Codes%202019-03-22.pdf
20. Software Maintainability: What it Means to Build Maintainable Software. (2019, November 7). *Sealights*. Retrieved April 7, 2021, from https://www.sealights.io/software-quality/software-maintainability-what-it-means-to-build-maintainable-software/
21. NextGen Connect. (2021, January 31). *Wikipedia*. Wikimedia Foundation. Retrieved April 7, 2021, from https://en.wikipedia.org/wiki/NextGen_Connect
22. Brauer, J. (n.d.). Mirth: Standards-Based Open Source Healthcare Interface Engine. *Open Source Business Resource*. Retrieved April 7, 2021, from https://timreview.ca/article/205
23. Weber, J. H., & Costa, O. (2020). Adapting a System-Theoretic Hazard Analysis Method for the Analysis of an eHealth Interoperability Conformance Profile. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, *2020*, 693–702.
24. Leveson, N. G., & Thomas, J. P. (n.d.). STPA Handbook (2018) . Retrieved April 6, 2021, from http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf
25. Tyagi, Aakash. *University of Victoria Engineering & Computer Science Co-Op Work Term Report Term (Fall) Year 2019*.
26. Tyagi, Aakash. *University of Victoria Engineering & Computer Science Co-Op Work Term Report Term (Fall) Year 2020*.
27. Admin, N. M. (n.d.). Emboldening Meaningful Use of EMRs in Canada: Canada Health Infoway. *Canada Health Infoway / Inforoute Santé Canada*. Retrieved April 22, 2021, from https://www.infoway-inforoute.ca/en/what-we-do/blog/digital-health-records/8272-emboldening-meaningful-use-of-emrs-in-canada
28. What LOINC is. (2017, March 03). Retrieved April 23, 2021, from https://loinc.org/get-started/what-loinc-is/