Efficient Processing of Metadata in Java SIG API for HL7

By

Yasir Mehmood

(2007-NUST-MS PhD-CSE (E)-13)

Submitted to the Department of Computer Engineering
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Software Engineering

Advisor
Dr. Muhammad Younus Javed

College of Electrical & Mechanical Engineering
National University of Sciences and Technology
2009

APPROVAL

It is certified that the contents and form of thesis entitled **"Efficient Processing of Metadata in Java SIG API for HL7"** submitted by **Yasir Mehmood**, have been found satisfactory for the requirement of degree.

Advisor: _____

(Dr. Muhammad Younus Javed)

Committee Member: _____

(Dr. Hafiz Farooq Ahmad)

Committee Member: _____

(Dr. Aasia Khanum)

Committee Member: _____

(Dr. Farooque Azam)

Committee Member: _____

(Dr. Ghalib Assadullah Shah)

IN THE NAME OF ALMIGHTY ALLAH

THE MOST BENEFICENT AND THE MOST MERCIFUL

TO MY PARENTS, TECHERS

AND SISTERS

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST CEME or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST CEME or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: Yasir Mehmood

Signature: _____

ACKNOWLEDGEMENTS

First of all I am extremely thankful to Almighty Allah for giving me courage and strength to complete this challenging task and to compete with international research community. I am also grateful to my family, especially my parents who have supported and encouraged me through their prayers that have always been with me.

I am highly thankful to Dr. Muhammad Younus Javed, Dr. Hafiz Farooq Ahmad and members of SEECS HLH Team Leads (Muhammad Afzal & Maqbool Hussain) for their valuable suggestions and continuous guidance throughout my research work.

I am highly grateful to Dr Aasia Khanum and Dr. Farooque Azam for their help and guidance throughout the research work. I am also thankful to all of my teachers who have been guiding me throughout my course work and have contributed to my knowledge. Their knowledge, guidance and training helped me a lot to carry out this research work.

I am also thankful to Ms Sidra Aftab and Mr. Hasan Ali Khattak for their keen interest, guidance and feedback in this research work. I would like to offer my gratitude to all the members of the research group (HLH Team) and my close colleagues who have been encouraging me throughout my research work especially Mr. Muhammad Afzal and Mr. Maqbool Hussain.

Yasir Mehmood

ABSTRACT

Information technology has started focusing on the healthcare enterprises, for providing better medical care. There exist different healthcare enterprise standards that are used for the communication of medical information across health enterprises providing swift and reliable results. HL7 is one of those standards that are used for the exchange of medical information between healthcare systems. The new standard of HL7 named as version 3.0 (v3) is an emerging standard claims at achieving semantic interoperability with its well defined information models like Reference Information Model (RIM), Domain Message Information Model (D-MIM), and Refined Message Information Model (R-MIM). These models are converted to some technology specific format for implementation such as Model Interchange Format (MIF). This format includes metadata information in the form of XML. MIFs are required to be loaded in memory for generation and parsing of messages. Core API developed by Sun with approval of HL7 known as Java SIG API utilizes these files in a non-efficient manner. It loads all associations (no matter these are required or not) present in a particular MIF file. This creates not only the performance issue but also the memory wastage. In this work, an algorithm is proposed to improve the process of message generation and parsing by avoiding unnecessary associations during MIF loading. This technique is based on proxy design pattern. The proposed technique removes the performance bottleneck of the API and makes it space and time efficient.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This chapter introduces the research work that has been taken in this thesis. It includes motivation and definition of the problem. Moreover the objectives and goals are also discussed.

1.1 Motivation

Health industry plays a pivotal role in the progress of nations by adding revenue to a nation's economy. This progress is due to its efficiency that has been indulged in it by the use of Information Technology. Healthcare is becoming even more progressive field these days and in near future will be considering as economy reviving industry. Healthcare stakeholders are ready to invest but reluctant because of no assured interoperability. So interoperability is the main cause of concern in developing healthcare systems. To achieve interoperability, healthcare standards are needed to use in developing healthcare messaging systems.

HL7 is one of several ANSI accredited Standards Developing Organizations (SDOs) operating in the healthcare arena [1]. HL7 has developed several standards in healthcare domain like conceptual standards in the form of HL7 RIM, document standards in form of Clinical Document Architecture (CDA), application standards like HL7 Clinical Context Object Workgroup (CCOW), messaging standards like HL7 version 2.x (v2.x) and HL7 v3. Messaging standards are of high importance because they define how the information can be packaged and communicated from one party to the other [2].

1.2 Background

Metadata includes information about structure of the message being generated, format on which messages are based and constraints applied on different associations, attributes etc. All the messages of HL7 v3 are based on MIF and hierarchical message

definition (HMD). Both of these formats are XML based [3] and supported by Java SIG API but proposed methodology is based on MIF not on HMD. For correct generation and parsing of HL7 v3 messages, proper loading of these metadata files is necessary otherwise there will be no assurance of correct message generation. Metadata loading has lot of problems to be handled for the successful generation of messages. The existing loading strategy requires lot of memory for complete loading of metadata which may include cross references to each other that result into not only performance bottleneck rather generates sometimes memory errors like stack overflow.

Message Communication is also very necessary beside message generation and parsing. Currently Java SIG API does not provide classes for communication of HL7 v3 messages.

1.3 Problem Statement

Problem statement consists of two parts. One part is related to the loading of metadata while generation and parsing of messages which is the main focus of this research work while other is related to the communication of the messages because health information represented in HL7 v3 messages is useless without communication.

1.3.1 Metadata Loading

Metadata loading is an important step in generation and parsing of HL7 V3 messages. Technique currently used by Java SIG API has lot of issues related to performance in terms of memory and time. This not only results in performance degradation but sometimes results in system crashes due to some errors like stack overflow.

1.3.2 Message Communication

Message communication is also an important step but Java SIG API has no provision for that. This work added required classes for the communication of HL7 messages as well.

1.4 Problem Solution

The goal of this research is to provide a mechanism for efficient generation and parsing of HL7 V3 messages.

1.4.1 Efficient Metadata Loading

In this thesis remedies of the above mentioned problems are taken. For handling the issues of metadata loading, an improved technique based on the existing one is proposed. Using this technique metadata is loaded efficiently and problems related to memory and time wastage are removed.

1.4.2 Reliable Message Communication

Java SIG API does not provide any mean for communication of messages. Some of the implementations are available for communication like Minimal Low Layer Protocol (MLLP) implementation for HL7 v2.x by HAPI, MIRTH etc. In this work implementation of MLLP by HAPI is integrated in Java SIG API for adding reliable communication infrastructure.

1.5 Outlines of thesis

The rest of the document is organized as follows.

Chapter 2 provides literature review. In this chapter different concepts related to healthcare industry are discussed, in chapter 3 some details about HL7, its components are discussed, chapter 4 discusses information about Java SIG API. In chapter 5, proposed system architecture is discussed while in chapter 6 research methodologies are discussed. Chapter 7 discusses details about design and implementation, in chapter 8 evaluation and results of the implemented system are discussed, and in chapter 9 thesis work is concluded and some future directions for research are discussed.

CHAPTER 2

LITERATURE REVIEW

Healthcare issues are among the most critical one faced by our society today. There have been repeated efforts over the years to apply all fields of science and technology to improve healthcare. The initiative all along has been to focus on providing healthcare by dealing with day-to-day tasks involved in doing so [4]. Healthcare has number of domains like laboratory, pathology, radiology etc where the involvement of information technology is emerging everyday thereby improving the patient care and thus saving the cost.

The healthcare data needs to be shared among various hospitals, clinics, doctors, physicians etc for variety of purposes like test order, result, patient record etc. The sharing of information among the healthcare systems and functional organization has been categorized four levels according to the conceptual framework. These levels are shown in Figure 1.

Level 1: Non electronic data*:* In 1950s a system was developed in America in which most of the medical transactions are carried out by means of paper. This system is still in use in most of the hospitals around the world. In this system there is no use of IT, instead most of the work is done through telephone and postal system. The basic drawback of this approach is the cost of handling paper and phone calls associated with the test, redundancy of data and the delays in case of paper based ordering and reporting of results [5]. If the transactions are carried out electronically then America could save $11 billion annually [6].

Level 2 Machine Transportable data: In 1970s hospital computing has been primarily about billing, with little attention paid to the needs of the clinician. By the mid- 1980s, programs were developed which were no longer limited to accounts and billing, rather provide access to laboratory results, medication-dispensing information, on-line text of is charge summaries, and more[7]. Only 10% of American hospitals started to use Health information technology in 1999 [5]. In this the transmission of

non standardized information is done via basic IT for example fax, pdf files, documents etc. the main drawback of this approach was that it only transmit the data electronically but it was unable to work or manipulate the information in that data.

Figure 1: Health Information by Time[1]

Level 3 Machine Organizable data: transmission of structured messages containing non-standardized data. At level 3, the system could combine the information from various remote sources but this form of transmission requires the translation of sender's information at receiver's end. The translation often results in misinterpretation due to incompatibility of vocabularies at both ends. Hence the resulting system were more error prone and generate redundant information, they limit the efficiency of Clinical Decision Support System and cause overload for clinicians. Also this level requires large amount of investment in interfaces in order to translate the heterogeneous vocabularies.

Level 4 Machine Interpretable data:

This is the most idealized level as it focuses on the exchange of structured messages containing standardized and coded data. Like Level 3, it can also get data from various remote locations and merge that but all of the systems would exchange the data in

[1] Health Life Horizon Project, "http://hl7.seecs.edu.pk/index.php?id=2"

same format and same vocabulary so there is no need of translating the information at sender and receiver's end and therefore it there is no chance of misinterpretation, ambiguity and incompatibility of vocabularies at the end systems. So the resulting systems are more efficient, accurate and time saving. At level 3 there were so many interfaces required for the translation of multiple vocabularies resulting in great investment. At Level 4, the use of same vocabulary eliminates the need for many interfaces. It only requires one interface to all the external systems. So in case of level 4 almost $31.8 billions cost is saved annually.

Interoperability between the systems becomes more effective due to the elimination of redundant tests and the saving of cost and time [8].

2.1 Healthcare Standards

According to the research and observations in developed countries like America, England etc most of the patients died or injured due to medical errors. The situation is worst in countries like Pakistan. The healthcare community provides solution to this problem by introducing standards in exchanging clinical information. There are different standards that are available for healthcare environment. A brief description of some important standards is given below:

2.1.1 Digital Imaging and Communication in Medicine DICOM

DICOM is a medical standard for storing; printing and transmitting the information related to medical images [9, 10]. DICOM enables the integration of scanners, servers, workstations, printers, and network hardware from multiple manufacturers into a picture archiving and communication system (PACS) [10]. DICOM has been widely used in hospitals and is and now it is also used for small level like dentists' and doctors' offices [9, 10]. The DICOM standard claims the conformance tests and it mentions the classes it support but in real situation DICOM fails to provide conformance issue as it lacks the details of implementation of different features of standard [10]. It also doesn't provide a way to test or check the implementation's conformance to standard.

2.1.2 American Society for Testing and Materials (ASTM)

ASTM is a Standard Guide for Properties of a Universal Healthcare Identifier (UHID). This standard was created by the Association for Information and Image Management and ASTM International. This standard is meant to provide the patient confidentiality and access security based on UHID. The sample UHID is used solely for the healthcare. It is able to link the health care records in both manual and automated environments and it can be easily mapped to currently used healthcare identifiers [6]. However it doesn't identify or addresses the implementation issues and infrastructure requirements.

2.1.3 Standard Generalized Markup Language SGML

The Standard Generalized Markup Language (ISO 8879:1986 SGML) is an ISO Standard Mata language that provides the user a way to define markup languages for documents. SGML only provides an abstract syntax for concrete implementation. ISO SGML shrinks a document to a regular expression in a known grammar so that it can be parsed easily. As such, it makes possible computer processing of information recorded in all the various forms that a narrative can take. SGML can be applied in almost all the areas of medical science and it can contribute more than to just encode the separate text blocks.

2.1.4 Logical Observation Identifiers Names and Codes (LOINC)

Logical Observation Identifiers Names and Codes (LOINC) are a database and universal standard to distinguish the medical laboratory observations. It is available for free and was created due to the demand of electronic database for clinical purpose. It applies universal names and codes for medical terminologies related to Electronic Health Record. With the time the data base do not remain restricted to just medical laboratory code names but rather it is expanded to include nursing diagnosis, nursing interventions, outcomes classification, and patient care data set. The database currently

includes over 41,000 observation terms that can be accessed and understood universally [6].

2.1.5 Study Data Tabulation Model (SDTM)

SDTM (Study Data Tabulation Model) defines a standard structure for human clinical trial (study) data tabulations that are to be submitted as part of a product application to a regulatory authority such as the United States Food and Drug Administration (FDA). The Submission Data Standards team of Clinical Data Interchange Standards Consortium (CDISC) defines SDTM [6].

SDTM is based on observations collected about subjects participated in a clinical study. Each observation can be expressed by a series of variables, corresponding to a row in a dataset or table. Each variable can be classified according to its Role. A Role determines the type of information conveyed by the variable about each distinct observation and how it can be used.

2.1.6 Introduction of HL7

HL7 is an international standard, developed in 1987 for electric exchange of clinical data in healthcare domains. Initially it aimed to provide point –to-point transmission or patient's data related to admission, discharge, orders and test result. But today HL7 provides different formats for information exchange related to all areas of healthcare including billing, care guidelines, doctor and support staff's information and so on.

The "seven" in HL7 represents its position at the 7^{th} layer of Open System Interconnection (OSI) series of protocols from International Organization for Standardization (ISO). HL7 focuses on the information that is carried in the messages despite of the formatting of messages or configuration of end point systems [8].

HL7 is the most widely used standard amongst all the other healthcare standards. The popularity and usage of HL7 is shown in the Figure 2.

Figure 2: Usage of Different standards among healthcare providers[2]

HL7 standard is used for exchanging information between the clinical information systems and between the systems dedicated for specific clinical task such as Laboratory, Radiology etc. The information is exchanged by the use of HL7 messages. The HL7 messages consist of various fields containing the data. The HL7 messages were first introduced in its v 2.x series which was a market success .but later on HL7 version 3 was introduced with new entirely different concept.

2.1.6.1 HL7 v 2

HL7 standard was initially introduced in series of version 2.x such as version 2.1, 2.3, 2.4, 2.5. At that time version 2 was very successful that it was used in more than 93% of US hospitals, that is the main reason that the concepts and terminologies used in version 2 just focuses the US paradigm.

The methodology and the development process of version 2 are not clear and properly planned. Hence its members do not have straight directions to follow and construct the messages. The version 2 is compatible with only limited data types. The prior knowledge of data types, message type and structure is required in version 2. The segments and message definitions are reused so many times and to cater this reuse, most of the data fields are declared optional. The metadata is not available in

[2] "Usage if Different standards among healthcare provides," [Online]. Available: http://hl7.seecs.edu.pk/index.php?id=2

structured form, instead it is extracted from the word processing documents and the messages were created by making changes in these documents.

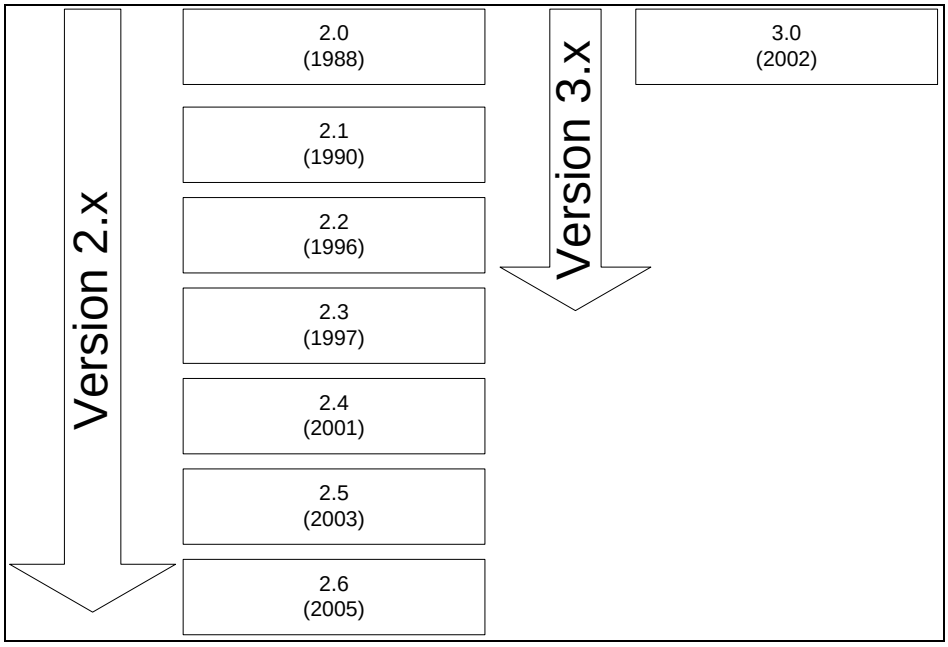

Figure 3: HL7 Version 2.x and 3.x

The version 2 basically provides an easy way to exchange healthcare message between the clinical systems and its transmission instead of focusing on healthcare data.

Hence there was a need to improve this version to address the major issues of healthcare. HL7 version 3 was developed to solve the problems faced by the developers of version 2.

2.1.6.2 HL7 v 3

HL7 v 3 standard was introduced to address the issues that were remained unsolved by version 2. The development of version 3 was the major challenge for HL7 members. After 4 years of continuous efforts and work, a methodology was designed to support all the healthcare workflows through a series of electronic messages. These version 3 messages are based on XML syntax.

Most of the countries around the world were using version 2 so it would be very difficult for these countries to use an entirely different version and spend money. Also

HL7 v3 is not compatible with the older 2.x series. So the major challenge was to provide a converter or translator from version 2 to version 3. Also version 3 is very complex

The Version 3 is not just limited to US hospitals; instead it can be widely used across the world in any healthcare environment. The version 3 messages focuses on the healthcare data instead of providing ways to transmit the data.

In HL7 version 3, the specifications become more detailed, clear and accurate by using the Object Oriented Approach. The Object Oriented is based on Unified Modeling Language UML design principles leading to the creation of more refined, unambiguous and structured messages. No prior knowledge is required for creating the version 3 messages. HL7 version 3 uses XML syntax instead of ASCII coding used in version 2. The difference between version 2 and 3 is shown in Figure 4.

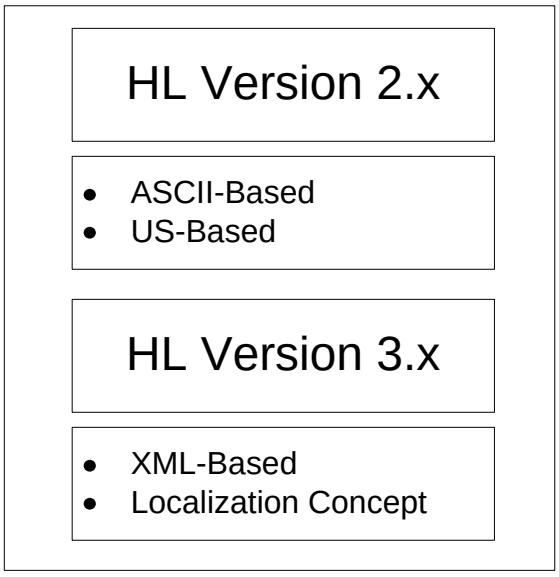

Figure 4: Difference between HL7 version 2 and 3[3]

Reducing the optionality in data field is considered as the major issues by version 3 developers as optionality makes it harder to claim conformance for message. The conformance of HL7 v3 will be testable by introducing the concept of Application Roles. Application Roles are used to define the behavior of messaging between the healthcare systems in an abstract way. The conformance to HL7 v3 is tested by

[3] "HL7 V2&V3 Comparison," [Online]. Available: http://hl7.seecs.edu.pk/index.php?id=2

assuring that the applications supports all the messages, trigger events and data elements linked with one or more application roles.

2.2 Health Life Horizon Project

Health Life Horizon (HLH) project is an ongoing project at School of Electrical Engineering and Computer Science (SEECS). Main aim of HLH is to develop HL7 v3 framework in order to provide better healthcare services to different communities of world. Main focus will be the development of HL7 middleware for the healthcare information systems in use today. Work on this project will be carried out in different phases.[4] There are three main phases of this project i.e.

1. HL7 Core Engine
2. HL7 Database Mapper
3. HL7 Transportation Component

2.2.1 HL7 Core Engine

HL7 Core Engine is related to message management. Main component of this module is HL7 message management tool which comprises of HL7 v2 to HL7 v3 Mapping tool, and HL7 v3 message generator and message parser. For message generation and message parsing HL7 java SIG API is used so Java SIG API is also considered as part of this component. Working of this API is main focus of this thesis. Beside these there are components related to HL7 Ontology core engine and application role deployment tools.

2.2.2 HL7 Database Mapper

HL7 database mapper is also an important component of HLH project. Currently HL7 specification does not provide any guidelines for direct mapping of database contents to the HL7 messages. It is related to the mapping of HL7 message contents to the database contents so that HL7 v3 messages may be generated efficiently in lesser time.

[4] "System Architecture," [Online]. Available: http://hl7.seecs.edu.pk/index.php?id=2

2.2.3 HL7 Transportation Component

Without transportation HL7 messages are of no good use. For providing full advantages of HL7 v3 there is a strong need of communication of messages among different stakeholders as described in Chapter 1. HLH defines three mechanisms for communication of HL7 v3 messages: MLLP, ebXML, and webservices. HLH architecture is shown in Figure 10 (Chapter 4).

CHAPTER 3

HEALTH LEVEL 7 VERSION 3.0

3.1 Introduction

HL7 is a standard organization. It was established in 1987 in order to standardize healthcare information systems [11]. It was accredited by American National Standard Institute (ANSI) in 1994. HL7 is one of several ANSI -accredited Standards Developing Organizations (SDOs) operating in the healthcare arena. Almost all of the SDOs operate in their domain or specific to their domain e.g. some are specific to pharmacy domain, some are specific to medical devices, and some are specific to medical imaging etc. Similarly HL7 also covers some domains. HL7's domain is clinical and administrative data.

3.2 HL7 Basic Modeling Concepts/HL7 Foundation Components

The HL7 Version 3 specifications consist of several documents. Some contain the HL7 version 3 specification and some contain information which is necessary or crucial for the development and transport of the messages of HL7. HL7 foundation document falls in the category which is necessary for the development and transport of messages. Figure 5 shows the pictorial view of the composition of foundation document [12].

Foundation components include Information Model, Vocabulary, Implementation technology specification, data types, common message element types and examples [12].

3.2.1 Information Model

Information model describes the information within specific domain of interest. It shows the required classes of the domain, there properties in terms of their attributes, relationships with other classes in the domain, constraints which applies on those

classes and there states etc. Different types of information models are defined in HL7, these information models are designed in such a way that they can be used to express or represent the information of different contexts.

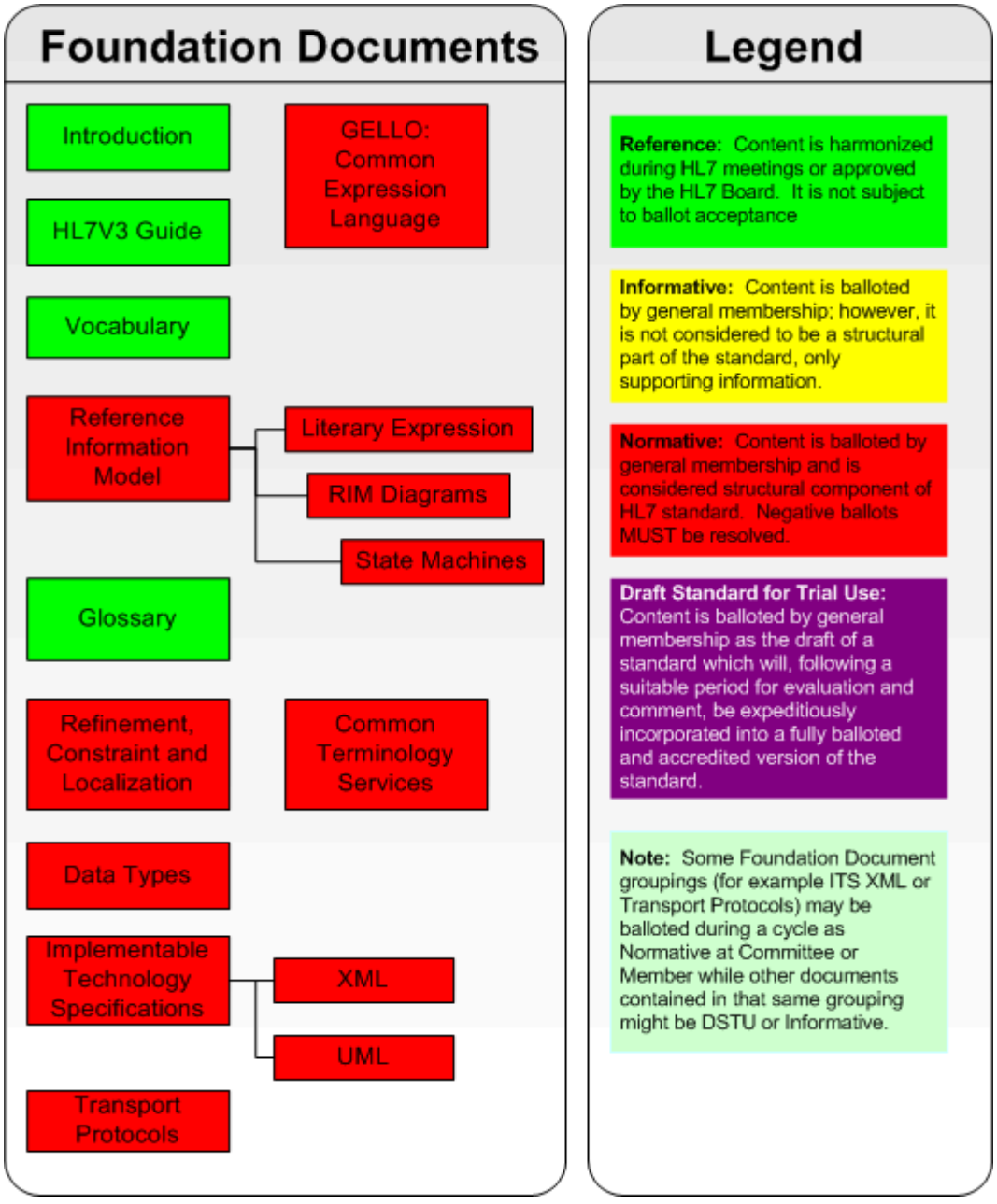

Figure 5: HL7 Foundation Components

An information model consists of different components like classes, their attributes, and relationship between classes, data types for all domain attributes, and state transition models for some of the classes. HL7 information model is based upon UML

and can be presented graphically using the standard UML notations. UML stands for unified modeling language and it is used for object oriented modeling. There are three types of information models named as:

1. Reference Information Model
2. Domain Message Information Model
3. Refined Message Information Model

3.2.1.1 Reference Information Model

Reference Information Model (RIM) of HL7 is a static model of healthcare [13] as it (RIM) is viewed from within the scope of HL7 standards development activities.

HL7 RIM is the ultimate source from which all HL7 Version 3 (HL7 V3) protocol specifications are defined and all HL7 V3 standards draw their contents related to information. The HL7 RIM is one of the significant components of version 3 development process. RIM is the root of all information models because all information models which are used in HL7 are based on it i.e. R-MIM, D-MIM etc.

This model provides the static view of the information needs of the HL7 V3 standards [13]. It is represented in diagrammatic form as shown in Figure above. It is comprised of six back bone classes act, participation, entity, role, actrelationship, rolelink details about these is given below:

Act

Act class represents the actions that are executed and must be documented as healthcare is managed and provided.

Entity

Entity class represents the physical things and beings that are of interest to, and take part in health care.

Role

Role class establishes the roles that entities play as they participate in health care acts.

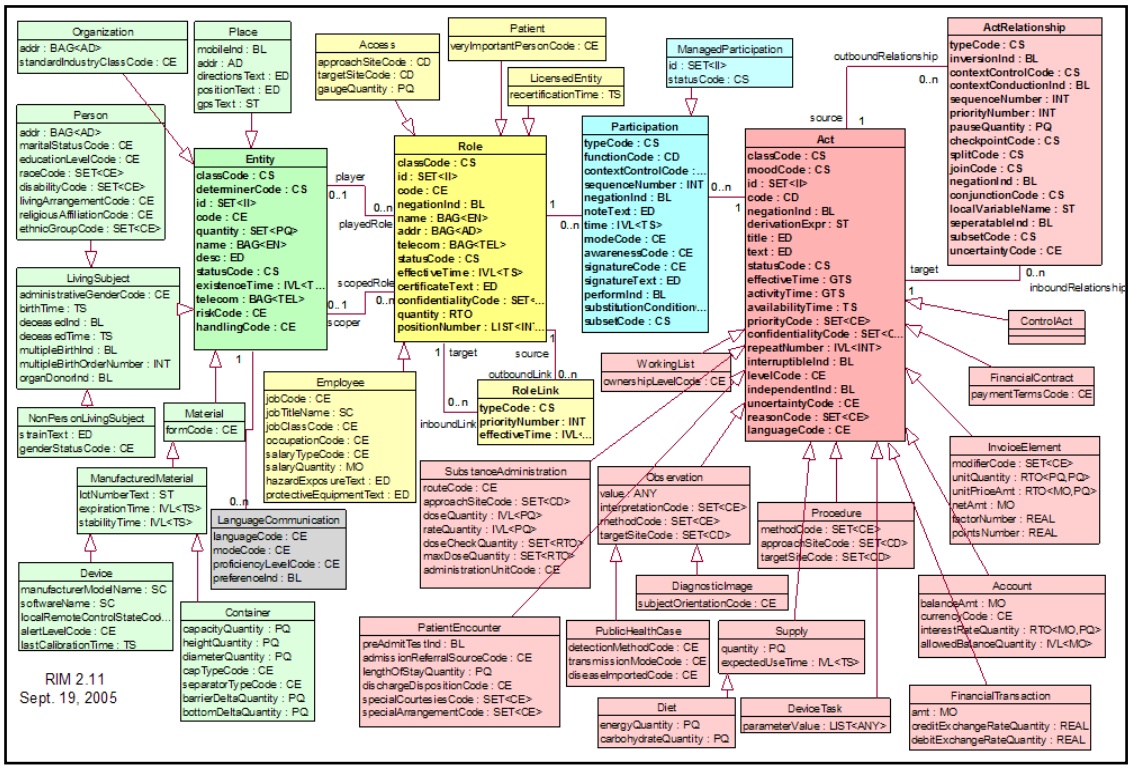

Figure 6: Reference Information Model

Participation

Participation class expresses the context for an act in terms such as who performed it, for which it was done, where it was done, etc. These are used for linking one role class with an act class.

ActRelationship

ActRelationship class represents the binding of one act to another, such as the relationship between an order for an observation and the observation event as it occurs. These are used for linking two act classes.

RoleLink

RoleLink class represents relationships between individual roles. These are used for linking two role classes.

These classes are the building blocks of RIM. All of the RIM structure is totally based on these classes. First three are called back-bone classes of RIM because all of other classes are based on these classes.

3.2.1.2 Domain Message Information Model

RIM demonstrates the abstract picture of the healthcare and is unable to represent the details about a specific domain but it provides necessary information that can be used for representing a particular domain. Domain Message Information Model (D-MIM) derived from RIM is used to capture information about a particular domain [14]. It is a subset of RIM that includes a fully expanded set of class clones, attributes, and relationships that are used for the generation of a message in a particular domain. For example set of class, attributes, and class clones that are used in the generation of messages of laboratory domain can be totally different from those that are used in the generation of messages of patient administration domain.

Like RIM it is represented in the form of a diagram. Its diagrammatic model is also very complex; it has its own conventions and notations that are developed by HL7 in order to represent some semantic meaning of the message. Complete understanding of these conventions and notations is very important in order to understand a D-MIM. These conventions and notations are given below in detail.

Entry Point

Each D-MIM model will have at least one entry point. It is the point from which the message for a particular domain begins [15]. In D-MIM it is represented by a rectangle with a black arrow originates from it and points to some class in D-MIM, that pointed class is called focal class or root class. Figure 7 shows the pictorial form of entry point. Entry point box contains following information

- Name of entry point mentioned in bold fold at the top left corner
- RMIM-Artifact ID mentioned below the name,
- Description of entry point.

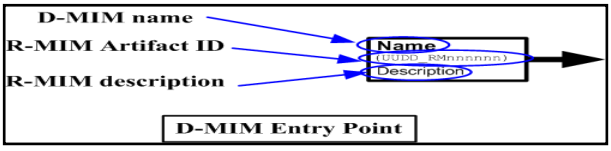

Figure 7: D-MIM Entry point

Classes and Colors

Like RIM, basic building blocks of D-MIM models are also RIM classes. The main difference is that in D-MIM each class can appear multiple times (but that is not the case in RIM). These multiple instances of a class in a model are called clones of that class. Each class has its own significance and color which distinguishes it from others. In D-MIM these classes are represented by different colored boxes [16]. The act related classes appear in boxes of red color, entity related classes appear in green and role related classes appear in yellow boxes while non-core RIM classes (classes which are not based on RIM) appear in dark blue colors as shown in Figure 8.

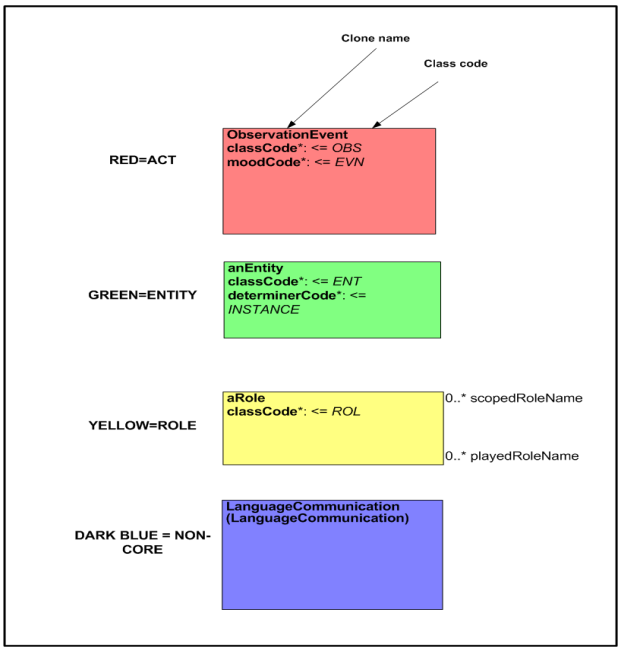

Figure 8: D-MIM Classes & Colors

Relationship Classes

Relationship classes are used for linking back-bone classes (act, entity, and role). These are shown by arrowed boxes in D-MIM and are of three types actrelationship which are used in linking two link classes, participation classes used for linking act with role class and rolelink classes used for link two role classes. These are shown in pictorial form in Figure 9.

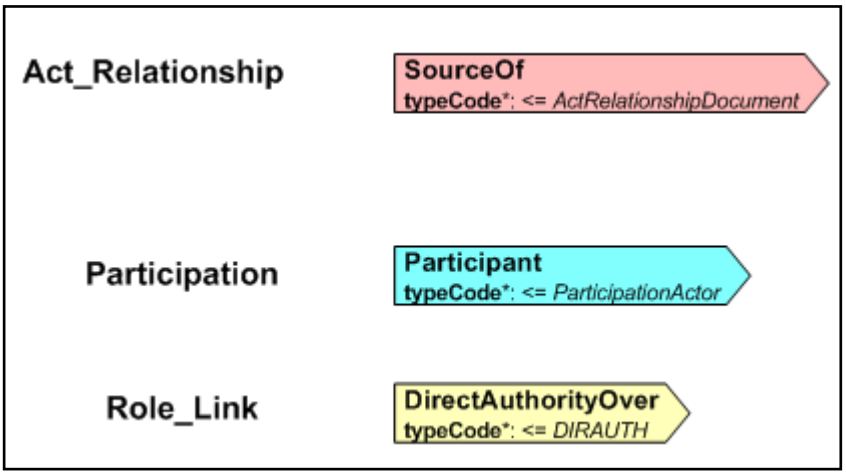

Figure 9: Relationship Types

3.2.1.3 Refined Message Information Model (R-MIM)

R-MIM is a subset of D-MIM [17]. It contains only those classes, attributes and associations required to compose the set of messages defined for a particular area in a domain. In D-MIM different entry points describes information about different possible R-MIMs in them. Same diagrammatic convention is used for expressing R-MIM as that is used for D-MIM. The main visible difference is that R-MIM contains only one entry point as compared to those of D-MIM which contains multiple entry points.

3.2.2 Static Structures

Static structure is same as that in unified modeling language (UML). It consists of classes and relationship between these classes.

3.2.2.1 Class

A class is an abstraction of things or concepts that are subjects of interest in a given application domain. Classes are the people, places, roles, things and events about which information is kept. Classes normally have a name (anonymous classes are exception), properties in the form of attributes, relationship between classes and states [18].

3.2.2.2 Relationships

Relationships define how the classes in a particular information model can be related to each other. We can categorize relationship between classes in two forms, generalization and association [18].

Generalization is the relationship between classes not between objects. It is association or relationship between two classes, i.e. super class and sub class. In generalization all properties of super class are inherited in subclass [18].

Associations are the relationships at the object level. These are used to relation two objects. Objects may be instances of two different classes or of same class [18].

3.2.3 Attributes

Class attributes are the core components of any information model. These are the sources for all of the information content in HL7 message [18]. Most of the attributes in information models are descriptive attributes. Descriptive attributes are used to describe aspects of classes that are important for communication between health care systems. In addition to descriptive attributes there are three more kinds of attributes which are used in information model named as identifier attributes, classifier attributes and state attributes. Identifier attributes can be used to identify an instance of a class. Sometimes only one attribute is enough to represent an object and sometimes there is need of using more than one attributed for identifying object. Value of identifier attribute is unique among all instances of a class. The classifier attributes form the backbone of the RIM (Entity, Role, and Act). These attributes are named "classCode". These attributes can be used to provide a great amount of flexibility and extensibility in the information model. State attributes are used in subject classes, these contains values which regularly changes. These attributed are used to indicate the current state of the class. A subject class must have only one state attribute. The state attribute must be assigned the data type "set of code value" that allows multiple state flags to be specified. State attributes are named status_cd and are associated with vocabulary domains defined by HL7 that correspond to the state machine defined for the subject class [18].

3.2.4 Constraints

Constraints are used to put some restrictions on the attributes. Constraints narrow down the set of possible values that an attribute can take on. Constraints may be specified in the RIM, D-MIM, R-MIM or hierarchical message description (HMD) [18]. In RIM the constraint is relevant for an attribute in all messages containing the attribute, in other two models the constraint is specific to all of the messages derived from that particular model (D-MIM or R-MIM). Constraints specified in a higher level (e.g., the RIM) may be further constrained in a lower level (e.g., D-MIM or HMD). However, the subordinate constraint must conform to the constraint on the higher level [18].

3.3 Vocabulary

Within HL7, a vocabulary domain is the set of all concepts that can be taken as valid values in an instance of a coded field or attribute [18]. Vocabulary domain consists of set of concepts, in different implementations of an interface the same concept can be represented using different coding systems. Thus, each concept in a vocabulary domain has a one-to-many relationship to codes that might be used as representations for the concept in a message instance [18].

3.4 Data Types

Data types are the basic building blocks of attributes. They defined the structural format of the data carried in attribute [18]. Some data types have very little intrinsic semantic content and the semantic context for that data type is carried by its corresponding attribute. However HL7 also defines quite extensive data types such as PNMP, person Name Part, which provides all the structure and semantics to support a person name. Every attribute in the RIM is associated with one and only one data type, and each data type is associated with zero or many attributes [18].

Table 1: Data type Categories from Normative 2006

Data Type Category	Explanation	Example Data Types
Basic Data Types	These describe 31 of the 42 data types that HL7 defines	Text,Codes,Identifiers,Names,Addresses,Quantities
Generic Collections	Data Types where a number of values may be included. They are not complete data types as any collection will have an associated Data Type from one of the other Data Type groups listed here	Sequence, BAG and SET
Generic Type Extensions	Ability to extend existing Data Types through a formal extension language	Not supported in current XML ITS
Timing Specifications	All timing requirements	IVL, Time Interval

3.5 Common Message Element Type

Common message element types (CMET) are a work product produced by a particular committee for expressing a common, useful and reusable concept [18]. They are intended for common use across messages produced by all committees that are way they are proposed to, reviewed by, and maintained by the CMET task force of the MnM committee [18]. A CMET is derived from a single D-MIM, defined by the producing committee. Its content is a direct subset of the class clones and attributes defined in that D-MIM, and does not include content from other D-MIMs.

CHAPTER 4

JAVA SPECIAL INTEREST GROUP API

Java Special Interest Group (Java SIG) for HL7 has designed an API for HL7 messaging named as Java SIG API for HL7. This API is used for generation and parsing of HL7 V3 messages. It plays a key role in HLH project. Architecture of HLH project is as follow:

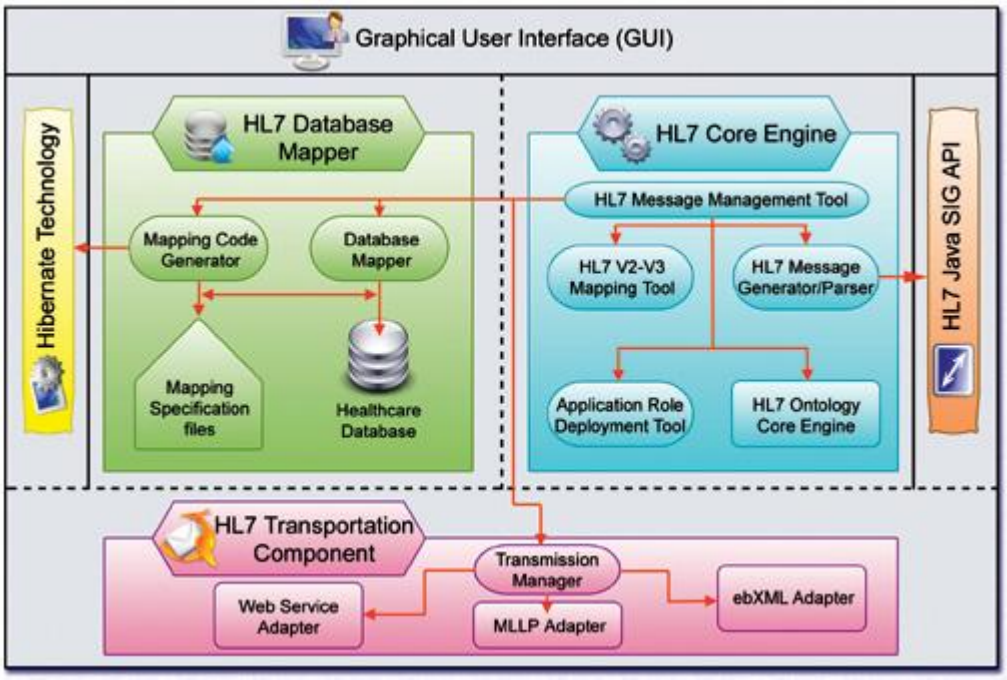

Figure 10: HLH Project Architecture[5]

It is the basic need of HL7 core engine to properly work. Role of HL7 core engine is mostly related to HL7 message management. HL7 message generator, HL7 message parser, HL7 V2 to V3 mapping and other messaging related functions fall under its roles. Here message generator and parsers use HL7 Java SIG API for generation and parsing of messages.

[5] "System Architecture," [Online]. Available: http://hl7.seecs.edu.pk/index.php?id=2

4.1 Java SIG API Component Overview

HL7 Java SIG API can be divided into different components named as:

1. Message Type Loaded
2. Message Content Handler
3. Data types
4. RIM Objects
5. Builder

In pictorial form it is shown in Figure 11:

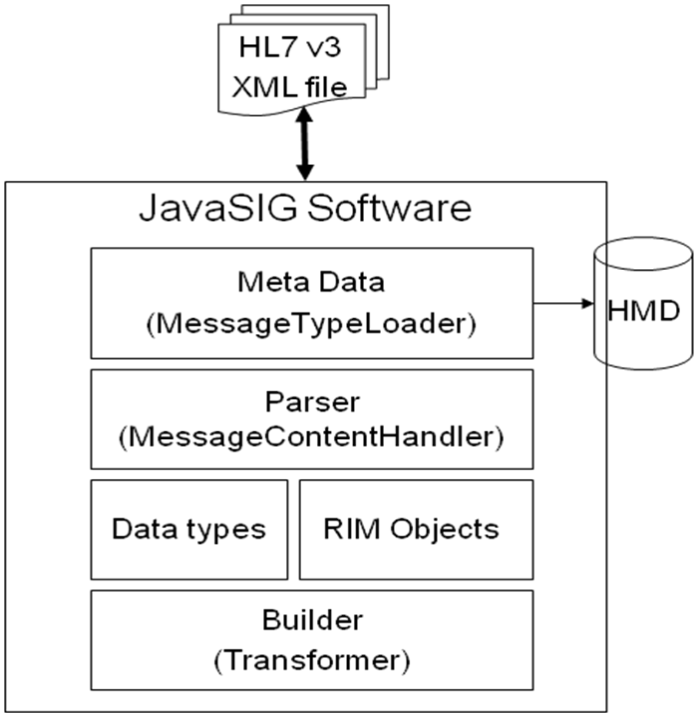

Figure 11: HL7 Java SIG API Components Overview

4.1.1 Message Type Loader

It is also called metadata loader. For generation and parsing of every message it is necessary to load the correct metadata of message types therefore it is called metadata loader. Work presented in this thesis is also related to this component of HL7 Java SIG API. For every message there is a message type or metadata present in the form of metadata interchange format (MIF) files and is distributed with each ballot and

normative edition. These files have the extension "**.mif**". Earlier it was present in the form of hierarchical message description (HMD) files but now in V3 these are provided in the form of XML. For metadata loading HL7 Java SIG API provides interfaces in "**org.hl7.meta**" package and their respective implementation classes are present in "**org.hl7.meta.impl**" package.

4.1.2 RIM Objects

RIM objects are discussed in chapter 3. These objects are important for generation of messages. These classes are based on high level abstraction such as patients, observations, procedures, roles, acts, etc [1]. Interfaces for these objects are found in "**org.hl7.rim**" package and implementations for these interfaces are given in "**org.hl7.rim.impl**" package.

4.1.3 Data types

HL7 V3 has defined its own data types. Each attribute in RIM has an HL7 specific data type. For each data type HL7 Java SIG API has defined some interfaces and classes for their proper usage. These classes contain many specialized methods [1]. Interfaces for datatypes are provide in "**org.hl7.types**" package and implementations of these interfaces are provided in "**org.hl7.types.impl**" package.

4.1.4 Message Parser

Message parser component is used for parsing of already generated messages. It makes use of rim objects, datatypes, and metadata loader component for correct parsing of message. Message parsing classes are also called message content handler classes. These classes are present in "**org.hl7.xml.parser**" package of Java SIG API.

Message parser makes use of SAX parser during parsing of messages. As SAX event fires, data present in the SAX event is used to query the metadata classes for loading metadata. The specific metadata allows the parser to instantiate a node as a RIM object of HL7 data type object [1]. Because of many flavors of objects like RIM classes, data types, CMETs (common message element types) there is a need of

specialized content handlers. The specialized content handlers can be dynamically "switched out" during message parsing.

4.1.5 Message Builder

Message builder or message transformer component is basically used for building or generation of HL7 V3 message using above mentioned components. It is very much similar to message parser i.e. in case of message parser it fully relies on small message content handlers while message builder relies on small builder components in fact content handler in parser are called builder during message building or generation. It makes use of rim objects, data types, and metadata loader.

The message builder uses the “Identity transform” feature of XSLT transforms to print the XML output of message. “The “Identity transform” is a special case where the incoming XML message is sent through the XSLT transform machinery, but no actual transform is done. The newly generated output is exactly the same as the XML input: but the output isn't simply copied. The input generated SAX events and these events were used to create the XML output. Since no transform is performed, the resulting message is the same as the input message” [19]. Class for message builders are mostly found in “**org.hl7.xml.builder**” package of HL7 Java SIG API.

CHAPTER 5

SYSTEM ARCHITECTURE

In this chapter proposed system architecture is discussed. System architecture is the conceptual design of the system. It describes the structure and/or behavior of the system in certain circumstances. The main focus of this chapter is to describe different parts/components of the architecture and design of the system. Initially all of the components of the architecture are mentioned after that their details i.e. their purpose and their working is discussed. Then proposed system architecture is shown in pictorial form.

5.1 Proposed Architecture

Proposed architecture of the system is divided into three sub-architectures named as:

1. Metadata Controller
2. Database Mapper
3. Communication Controller

Metadata controller is responsible for handling metadata related issues. It loads the data after processing of metadata interchange format (MIF) file. Database mapper is used for mapping the message contents with the database contents while communication controller is used for sending and receiving messages. It is shown in Figure 12.

5.1.1 Proposed Architecture of Metadata Controller

Metadata controller is responsible for the metadata handling which is the main work of this report. It is shown in the Figure 13. It is divided into 3 main levels named as:

- User Interface Level
- Messaging Level

MIF (RMIM) Controller Level

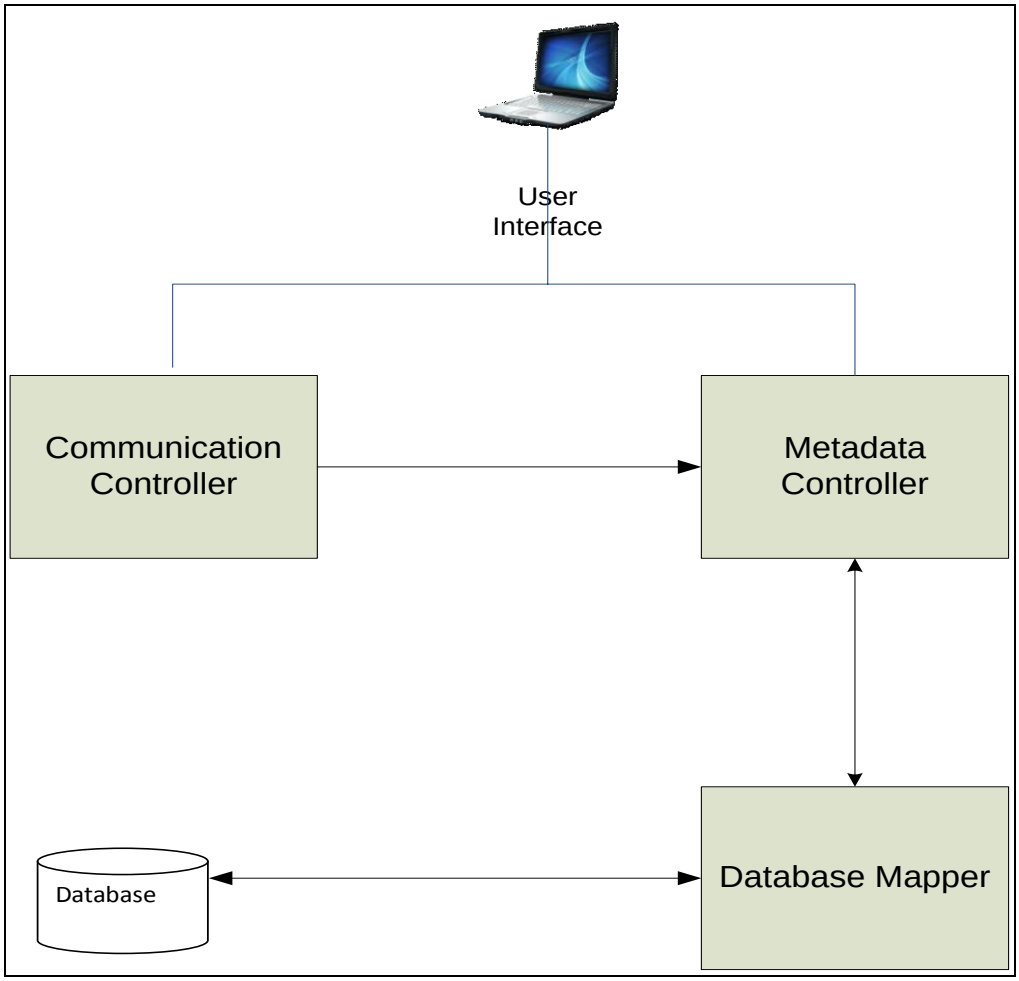

Figure 12: Overall Architecture

User interface level represents different users' querying the system for generation and parsing of messages. When user issues command for generation or parsing of message, the control is transferred to the messaging level. On Messaging level or layer there are two major components named as:

- Message Generator
- Message Parser

From messaging level control is transferred to the MIF (RMIM) controller level for loading and processing of metadata files or MIF files. This level has five components all of them related to metadata handling. These components are named as:

- MIF Handler
- MIF Loader

- MIF Reader
- Association Finder
- MIF Splitter
- MIF Assembler

Details of different important components of the system architecture are discussed below.

5.1.1.1 Message Generator

This component is related to the messaging level. It is used for generation of HL7 V3 messages of a specific format. For correct generation of message it is necessary to load a metadata file called Message Interchange Format (MIF). So for loading of it message generator transfers the control to the MIF loader.

5.1.1.2 Message Parser

This component is used for parsing of HL7 V3 message. Before parsing it validates the message. For both validation and parsing it requires loading of MIF files which is loaded by transferring the control of MIF loader.

5.1.1.3 MIF Handler

It is the first component of MIF controller. There are two main functions of the component first is to load the MIF file by calling MIF loader when the MIF file is loaded MIF loader transfer the control to someone else for processing, second function of MIF Handler is that it takes assembled MIF file from MIF assembler and then it transfers control to the messaging level for generation and parsing of messages.

5.1.1.4 MIF Loader

MIF loader has two main functions. First of all it reads the information about the MIF file from the project properties file named as “cmet-files.properties”. After getting this information it loads the MIF file into memory for processing. After processing it transfers the control to the MIF reader that will read the MIF file and in case some

other MIF files are referenced into that particular MIF file than control is transferred back to the MIF loader for loading of that referenced MIF file.

5.1.1.5 MIF Reader

This component takes loaded MIF file from the MIF loader and starts traversing of it. It has a sub component named as MIF association finder. This component looks for all of the associations present in a MIF file.

5.1.1.6 Association Finder

MIF file is composed of multiple associations. Identification of all of these associations is necessary because some of the associations may be needed in a particular scenario but some of may not be. This component is the sub-component of MIF Reader. It will finds out all of the associations from the MIF file. After identification of all of the associations it will transfer control to the MIF splitter.

5.1.1.7 MIF Splitter

Once all of the associations are identified it is the responsibility of the MIF splitter to split each of the MIF file into different associations so that they can be processed easily. By splitting a MIF file into different associations it is very easy to identify associations that are mandatory, associations that are optional, associations that are required to be loaded in a particular scenario etc. If in an association another MIF file is referenced control is transferred all the way back to MIF loader that loads that association and passes that to the MIF reader and cycle goes on. After identification of associations which are required to be loaded in a particular scenario control is transferred to the MIF assembler so that these can be assembled and loaded in memory.

5.1.1.8 MIF Assembler

MIF Assembler is also an important component of the architecture. It takes the associations from the MIF splitter and finds out the associations which are required in

the scenario. After identification of those associations it assembles them into one MIF file and passes it to the MIF handler which loads them in the memory for usage.

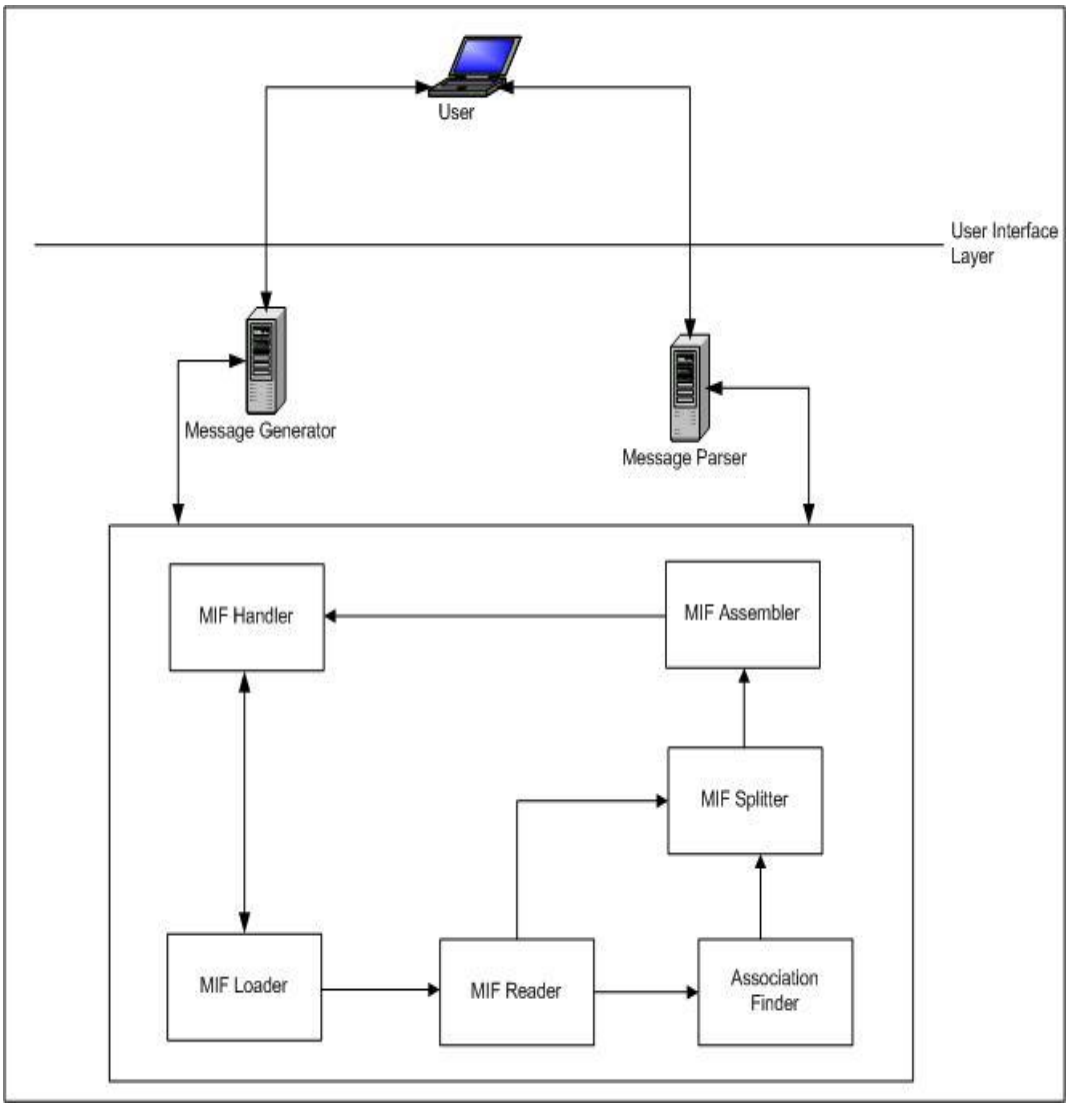

Figure 13: Proposed Architecture Diagram of Database Controller

5.1.2 Proposed Architecture for Communication Controller

Communication controller is developed by another resource working on this project. Currently communication environment is developed using minimal low layer protocol (MLLP) built on top of TCP. Architecture for MLLP is shown in the Figure 14. More details about it can be found from [20].

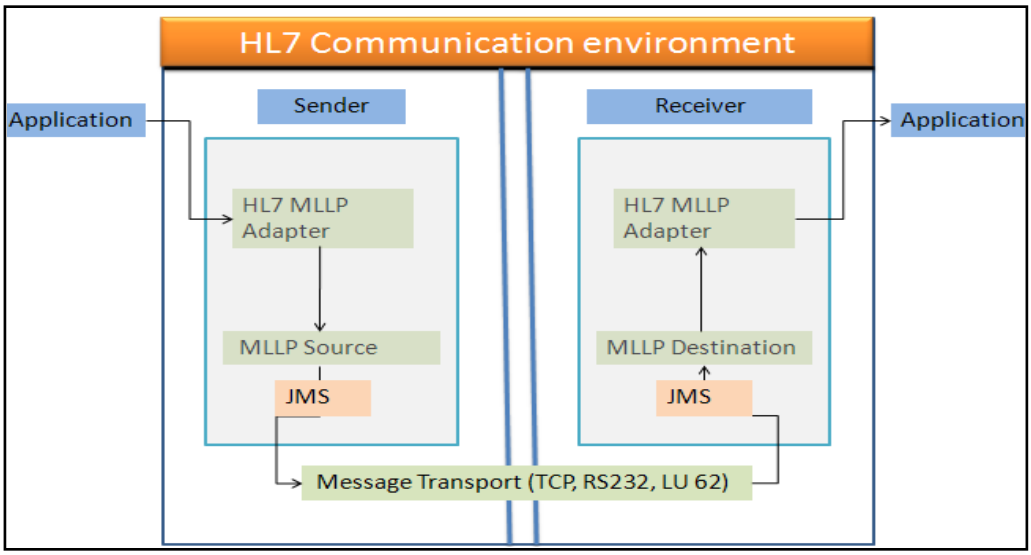

Figure 14: HL7 Communication Environment

The goal of the MLLP Message Transport protocol is to provide an interface between HL7 Applications and the transport protocol that uses minimal overhead. MLLP is based on a minimalist OSI-session layer framing protocol. It is assumed that MLLP will be used only in a network environment. Most of the details of error detection and correction are handled by the lower levels of any reasonable transport protocol (e.g. TCP/IP, SNA) and do not require any supplementation.

The network protocol and the network behavior have to be agreed upon by the communicating parties prior to the exchange of data. MLLP Release 2 covers the absolute minimal requirements in order for it to be a reliable Message Transport protocol [20].

5.1.3 Database Mapper

This component is used for integrating database contents with the message contents. It is used on both sender and receiver side. On sender side it works as Database to HL7 mapper. It takes data contents from data loader and maps them with R-MIM contents from R-MIM controller. After mapping these mapped contents are passed to Message Generator component. On receiver's side it works as HL7 to Database mapper. It takes parsed message contents from Message Parsers and maps them with the RMIM

contents from RMIM controller, after mapping it passes these contents to the database controller for storing into database.

CHAPTER 6

METHODOLOGY

In this chapter research & implementation methodologies are described. Methodology explains the techniques or procedures used by the researcher [21]. This chapter contains material about methodologies which are proposed or implemented in this thesis. This chapter is divided into two parts in first part problem is discussed and in second part methodologies which are used for solution of this problem and motivation behind these methodologies is discussed.

6.1 Problem

As discussed in previous chapter some metadata files are loaded during generation and parsing of HL7 V3 messages. Efficient processing of these metadata files called metadata interchange format (MIF) files is necessary for generation and parsing of messages. Metadata loader component of Java SIG API is responsible for this task. During generation and parsing of message, name of the metadata file is provided to the system. System loads that metadata file and transverse it. Each metadata file contains lot of associations. It is also possibility that in some of the associations may contain reference to some other metadata files which are also loaded with these associations. Now problem with API is that it loads all of the associations present in the metadata file all at once due to which lot of metadata files are loaded sometimes these metadata files contains references to each other in which case some circular queue like situation arises that may result in wastage of memory and ultimately into a deadlock i.e. Metadata file “A” contains reference to metadata file “B”, metadata file “B” contains reference of metadata file “C” and metadata file “C” contains reference back to metadata file “A” in this case all metadata files “A”, “B”, and “C” contains reference of one another and this process will go on until program crashes or some halt condition is applied.

It is possibility that in a certain application scenario there may be no need of loading all of the associations e.g. consider a scenario in which a metadata file is loaded which contains 18 associations. Out of these 18 associations in 4 associations some more metadata files are referenced, these referenced metadata files contains almost same structure i.e. lot of associations and referenced metadata files. If a rough estimate of these associations is taken then there would be more than 70 associations which Java SIG API has load for accurate generation or parsing of message. Out of these associations in one scenario there may be required only 10 to 15 associations rest may not be of any use in a particular scenario for efficient processing there is a requirement of loading only these associations.

Beside message generation and parsing there are also some problems related to message communication. HL7 Java SIG API does not provide any mechanism for message communication. It is also a big challenge to provide communication environment so system not only generate and parse messages but also communicate it with other systems. Some communication specification is described in HL7 Normative edition but at implementation level no work is carried out earlier.

6.2 Methodologies

Problems mentioned in the above section are resolved in the approach describes in this thesis. In proposed methodology a temporary MIF file based on the requirements of the application is created and loaded in the memory. Using this approach only those associations (of the metadata files are loaded) which are required in a certain scenario. Due to this processing time and resources can be reduced by many times. This approach is useful for both message generation as well as message parsing.

To make the message generation and message parsing processes time as well as space efficient; API is optimized to load only those associations which are required in a particular scenario. In each MIF file there are two kinds of associations which are necessary associations as well as optional associations. Necessary or required associations are those which are mandatory to be loaded no matter whether there is any need of them or not, on the other hand optional associations are those which are not

required to be loaded and only few out of them that are required in a particular scenario are loaded.

6.2.1 Loading of Associations for Message Generation

For generation of message initially all of the necessary associations associated with the entry point of the metadata file are identified. It can be done by reading and understanding MIF files. Some time different MIF files are referenced in different associations within the metadata file, and these metadata files have similar structure and sometimes result in the cross referencing. Necessary metadata files can be identified by checking the value of minimum multiplicity attribute of "*targetconnection*" tag which is nested inside the association tag. For necessary association value of minimum multiplicity attribute should be greater than '0'. After identification of necessary associations, system identifies the metadata files referenced in these associations and asks the application developer to load these metadata files. When all of the necessary associations attached with the entry point are identified, then system takes classes referenced in these associations as entry point turn by turn and identify necessary associations further attached with these entry points, and load metadata files if they are referenced in associations and so on.

When finished with necessary or required association, now it's the turn of optional associations. First of all identify all of the optional associations (associations with minimum multiplicity value equal to '0'). After identification of these associations, system asks application developer if she/he is interested in loading these associations. If application developer is interested, the association is loaded (along with MIF file referenced in that association if any); otherwise system removes that association from the metadata file. When done with first level optional associations, then it is the turn of second level associations. System takes the first level associations as entry point and repeats the above mentioned step for both mandatory and optional associations and so on.

After finishing with all of the associations, it is required to load them in the memory as a new temporary metadata file. For this purpose a temporary metadata file is

created, all of the identified associations are written down in this new MIF file. After that all of the MIF files referenced from these identified associations are traversed, similar steps are performed and identified associations from these referenced MIF files replaces the association (which contain reference to other MIF file) of temporary MIF file and this process goes on until all of the associations and MIF files referenced from these associations are handled. At the end the new temporary MIF file is ready. System is given the path of this temporary MIF file and all of the associations of this MIF file are loaded.

6.2.1.1 Algorithm for Generation of Message

1. Identify entry point of the given MIF file.
2. Identify all of the associations associated with the entry point of the given MIF file
3. Identify mandatory associations (out of the associations found in step one) by checking "*minimumMulitiplicity"* attribute whose value is greater than 0 in tag *"mif:targetConnection"* nested in *"mif:association"* tag.
4. Load the associations of step 2 in temporary MIF file in memory.
5. Identify optional associations by checking *"minimumMultiplicity"* attribute whose value is '0'.
    a. Identify those associations which are according to requirements and induct them to the MIF file.
6. Identify the classes referenced in associations identified in step 3 & 5,
7. Take these classes as entry point turn by turn and repeat steps from 2 to 6
8. If another metadata file is referenced in associations identified in step 3 & 5
    a. Then repeat steps from 1 to 7 with referenced metadata file and so on.

6.2.2 Loading of Associations for Message Parsing

For parsing of message similar approach is used. Here idea is to parse both message file as well as MIF file. On the basis of content of the message file associations from the metadata files are loaded in the memory. This is done first by parsing the message

file, and after parsing identify the names of each starting tag and values of those tags are compared with the name attribute of "*targetconnection*" tag in metadata file and in case of choice boxes, it is compared with the "*traversalName*" attribute of the "*participantClassSpecialization*" which is nested in the "*targetconnection*" tag. On the basis of these comparisons, associations and referenced metadata files are loaded in the memory. It results in the easy parsing of message.

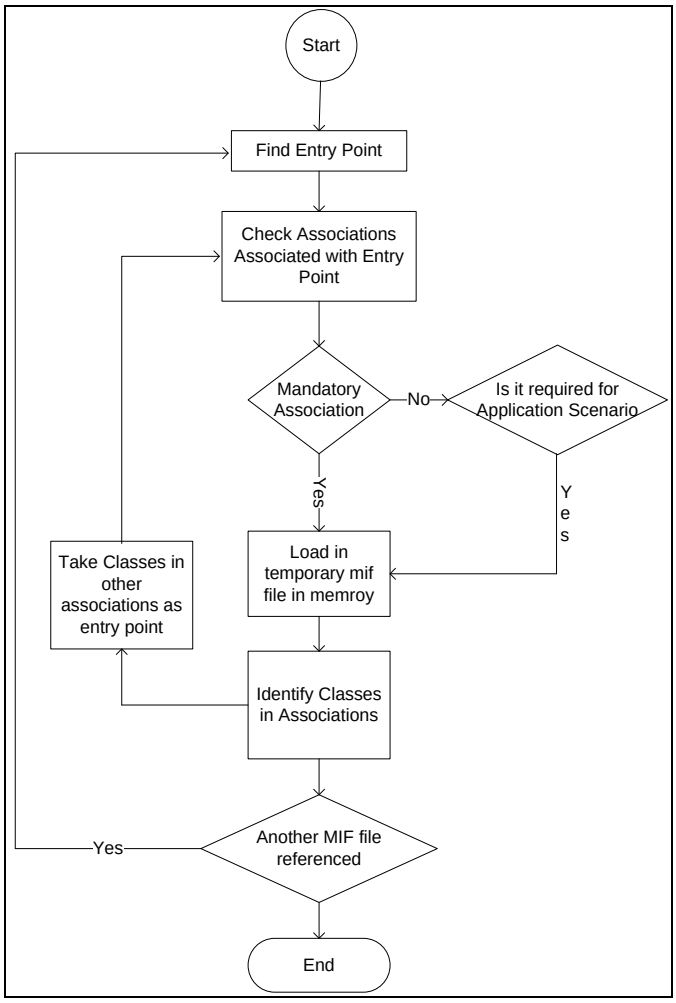

Figure 15: Flow of activities during generation of message

6.2.2.1 Algorithm for Parsing of Message

1. Read value of starting tags of message file
2. Compare these values with the values in metadata file there are two possibilities
    a. If the association is simple then compare value with name attribute of "targetConnection" tag nested inside association tag
    b. If the association is of choice type, then transverse to the participant class specialization tag written as "participantClassSpecialization" and compare the value of tag of message file with value of its name attribute.
3. Load those associations for which values are compared in the memory
4. Leave rest of the associations
5. If other MIF files are referenced, then load them and repeat steps from 1 to 4
6. Parse the message with the new temporary MIF files

These steps can be repeated until all of the associations which are used in the message file are identified and loaded. After loading of these associations it is easy to parse the message.

In worst case scenario, the results of both existing and new techniques for loading of associations in message generation as well as message parsing are same. In the best case scenario, one may be asked to load just one metadata file depending upon the requirements of the message. This technique reduces the time required for the processing of metadata files largely and also improves the space utilization.

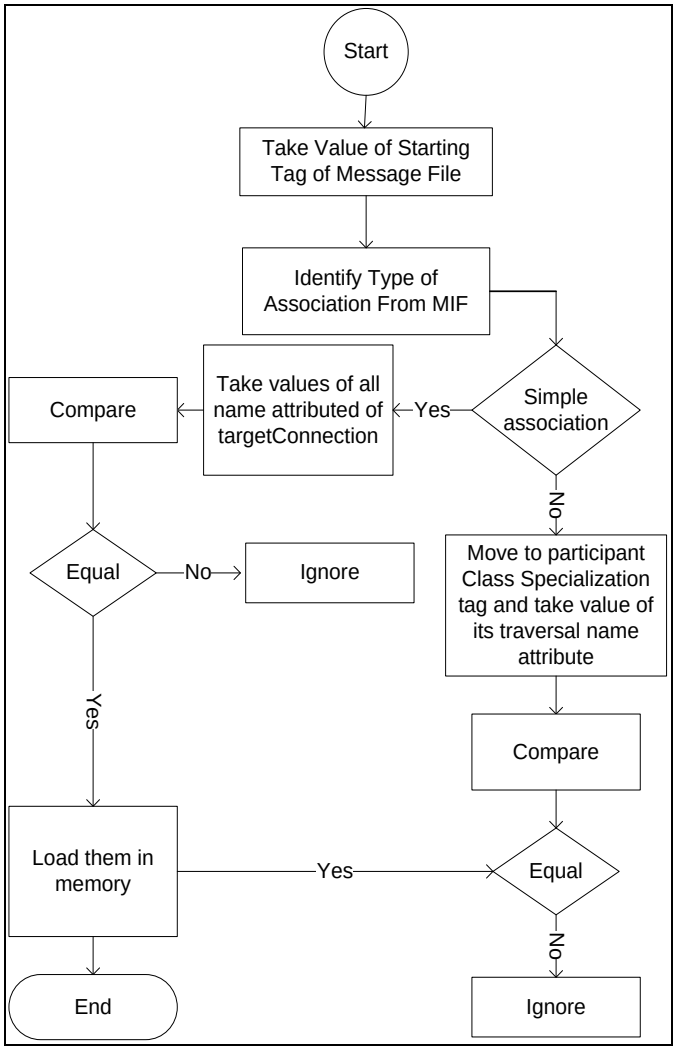

Figure 16: Flow diagram of loading of one association for parsing

6.2.3 Message Communication

HL7 Java SIG API does not provide any packages for message communication neither there is any other messaging API which can be used for communication HL7 V3 messages, although there is one named as HAPI which can be used for communicating HL7 V2.x messages but for V3 messages there are requirements for some modifications. Lot of methods are proposed for transmitting HL7 V3 messages from one point to other like ebXML, webservices, minimum low layer protocol (MLLP) etc. Proposed used in this thesis focuses on the combination of MLLP and

Java message service (JMS). Its architecture is shown in Figure 17 and its details can be found in [20].

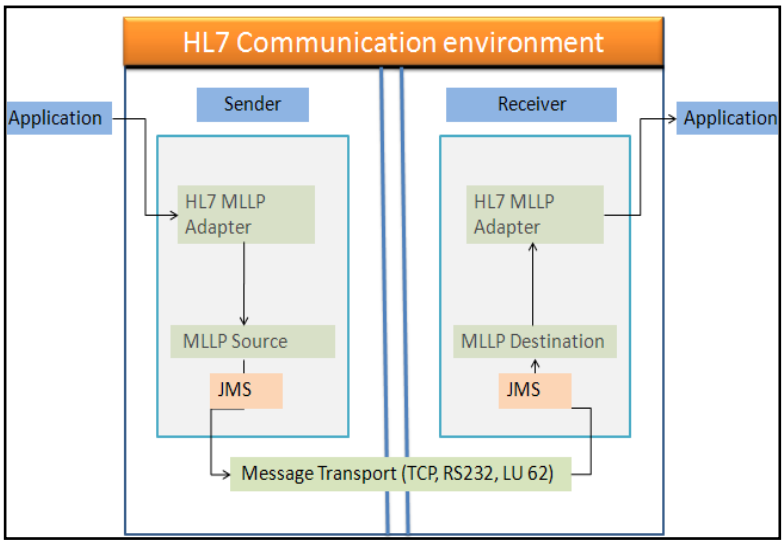

Figure 17: HL7 Communication Environment

CHAPTER 7

DESIGN AND IMPLEMENTATION

This chapter covers the design and implementation details of the application (prototype) in the form of interaction diagrams in design section and class diagrams in the implementation section. This technique is tested in generating and parsing of messages of laboratory domain. For this purpose CITILAB is taken as a case study.

7.1 Analysis to Design & Implementation Details

Figure 18 shows the use case model of the application.

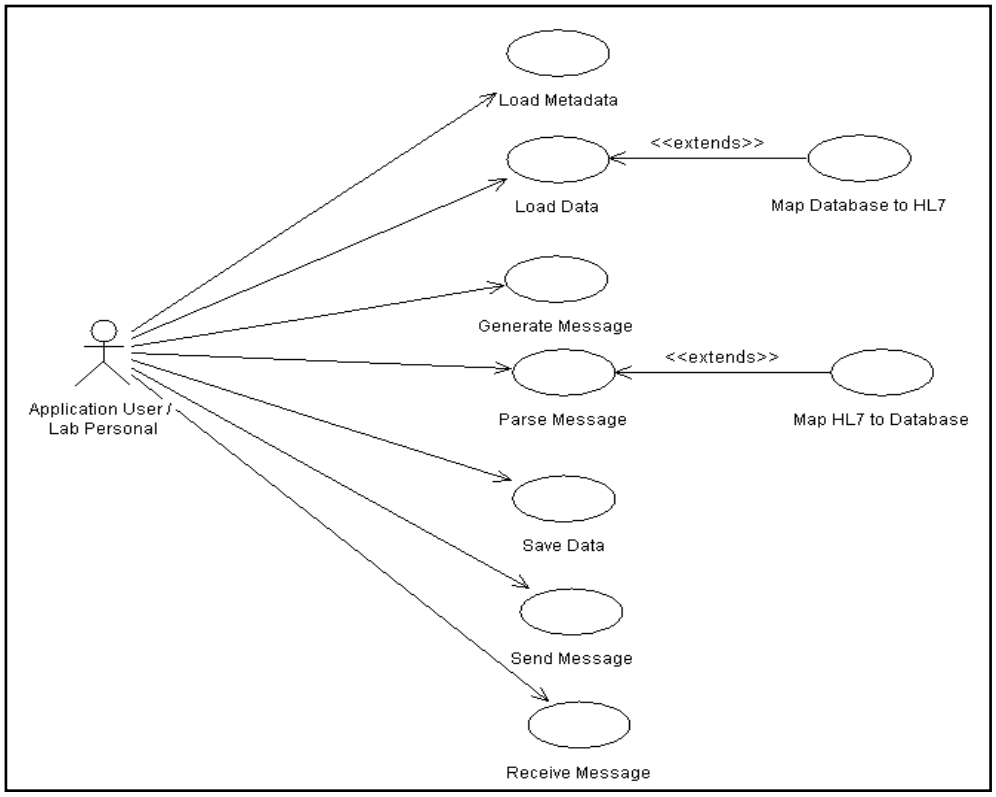

Figure 18: Use case Diagram

In this use case model different use cases for handling of metadata, and other issues involved in the development of the application are discussed. Details of each of these are given below:

7.1.1 Load Metadata

This use case captures the scenario when system wants to load the metadata for generation or parsing of message.

7.1.1.1 Sequence Diagram & Collaboration Diagram

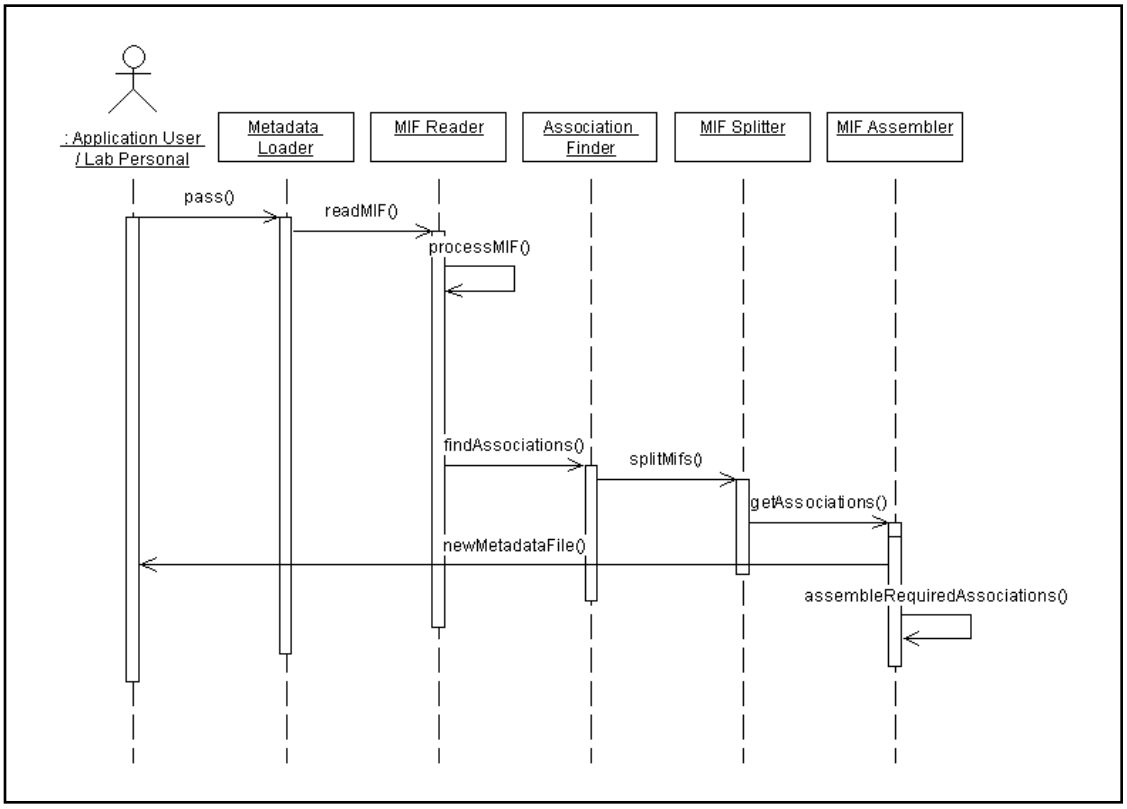

Figure 19: Sequence Diagram of Load Metadata Use Case

When application user or lab personal (as citilab is taken as case study) passes the command for generate message or parse message control is transfers to the metadata loader. Working of metadata loader is not just to load the metadata but it first refines the metadata according to requirements and then loads it. For this purpose it passes the control to the MIF reader which processes the MIF, then control is transferred to the association finder for identification of the associations present in a MIF file after identification of associations control is transferred to the MIF splitter which splits the MIF file in different associations and at the end MIF assembler combines the required

associations into a single MIF and passes the control to the Metadata Loader which loads the metadata file. Its collaboration diagram is shown Figure 20.

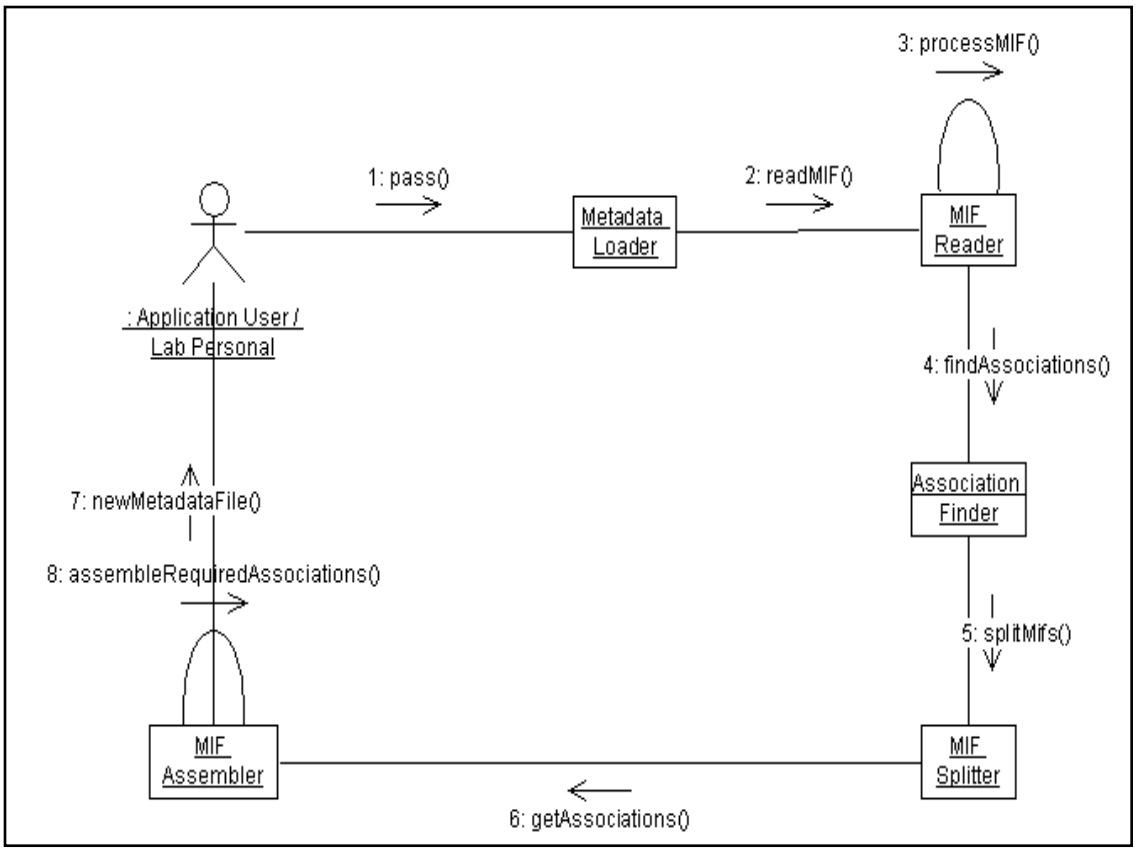

Figure 20: Collaboration Diagram of Load Metadata Use Case

7.1.2 Load Data for Message Use case

Primary actor for this use case is Lab Personal. It captures the scenario for loading of data from the database about different laboratory tests which are being conducted for a patient.

7.1.2.1 Sequence Diagram & Collaboration Diagram

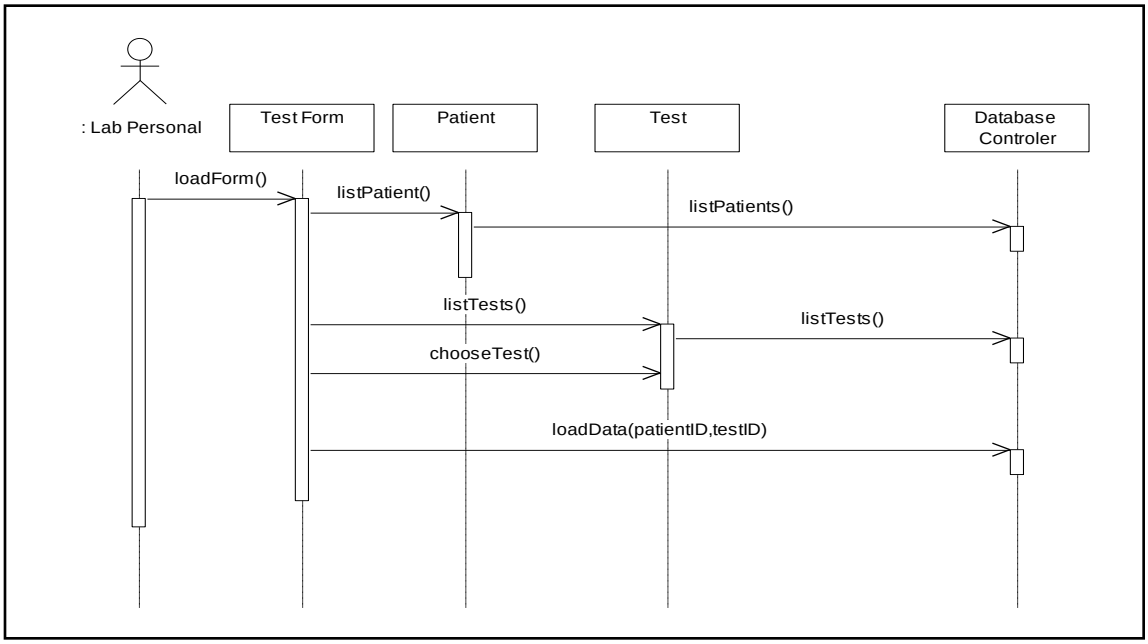

Figure 21: Sequence Diagram of Load Data Use Case

Lab personal loads the form using loadForm() method. After that information about patients and tests is retrieved by using listPatient() and listTests() methods respectively. And data for the selected patient and tests is loaded by using loadData(patientID,testID) methods. Its collaboration diagram is given in Figure 22.

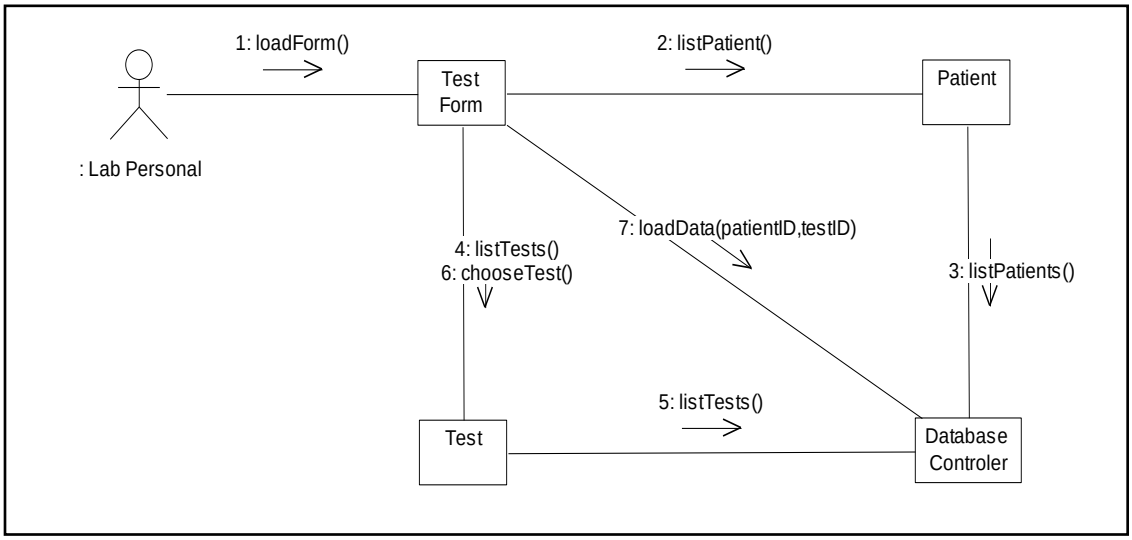

Figure 22: Collaboration Diagram of Load Data Use Case

7.1.2.2 Partial Class Diagram

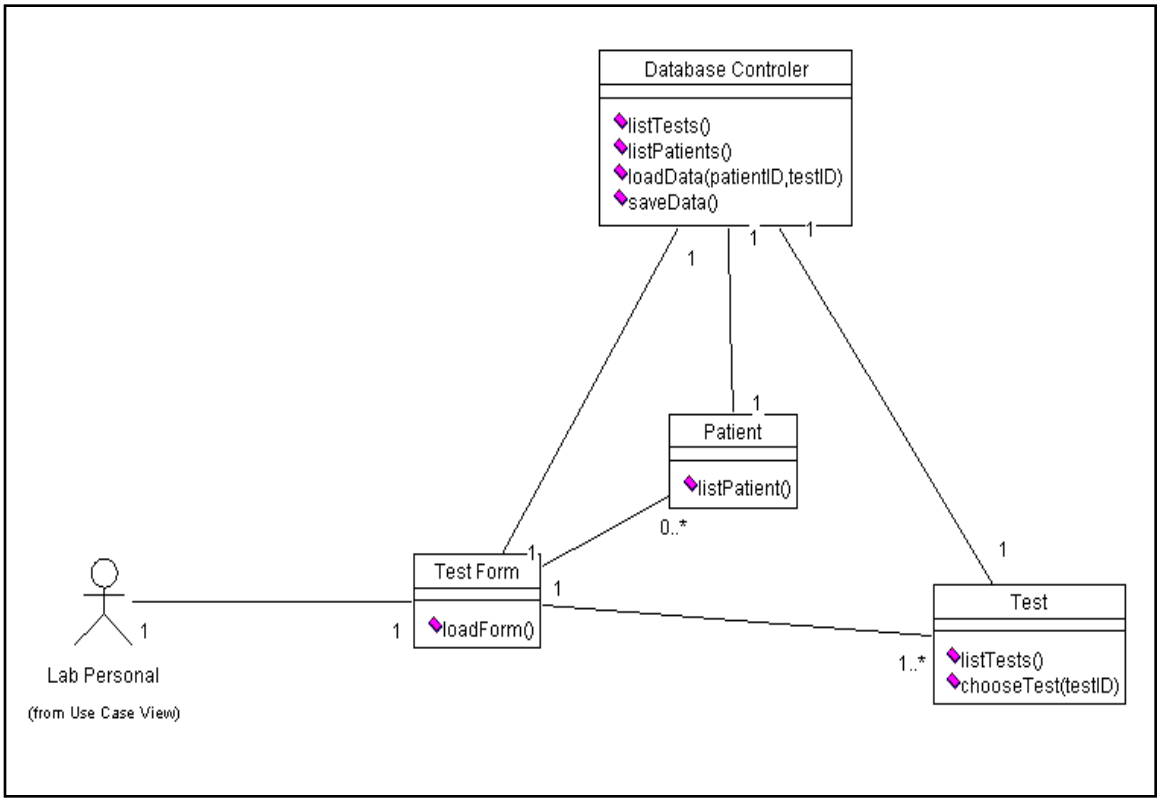

Figure 23: Partial Class Diagram of Load Data

7.1.3 Map Database to HL7

This is a very important use case it captures the scenario where retrieved data about patient and tests is mapped to the RMIM objects. Correct mapping is necessary for the generation of message. Primary actor in this use case is system itself, pre condition for this is that data is successfully loaded and after one to one mapping it will be ready for correct generation of the message.

Data about patient and test is loaded by calling the method loadData(patientID,testID), after that information different elements of the metadata are extracted from the RMIM by calling the extractInfoFromRMIM(msgType). After that data objects are mapped to the RMIM objects using mapDbToHL7 () method.

7.1.3.1 Sequence Diagram and Collaboration Diagram

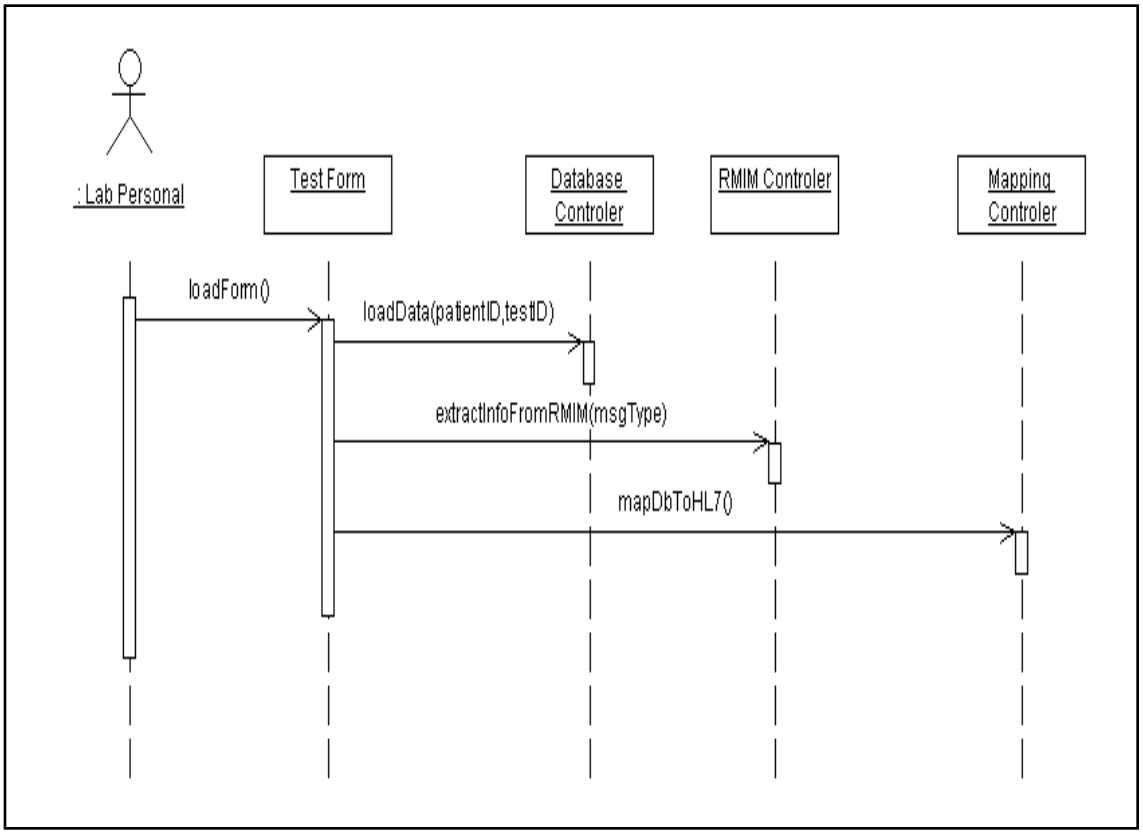

Figure 24: Sequence Diagram of Map Database to HL7

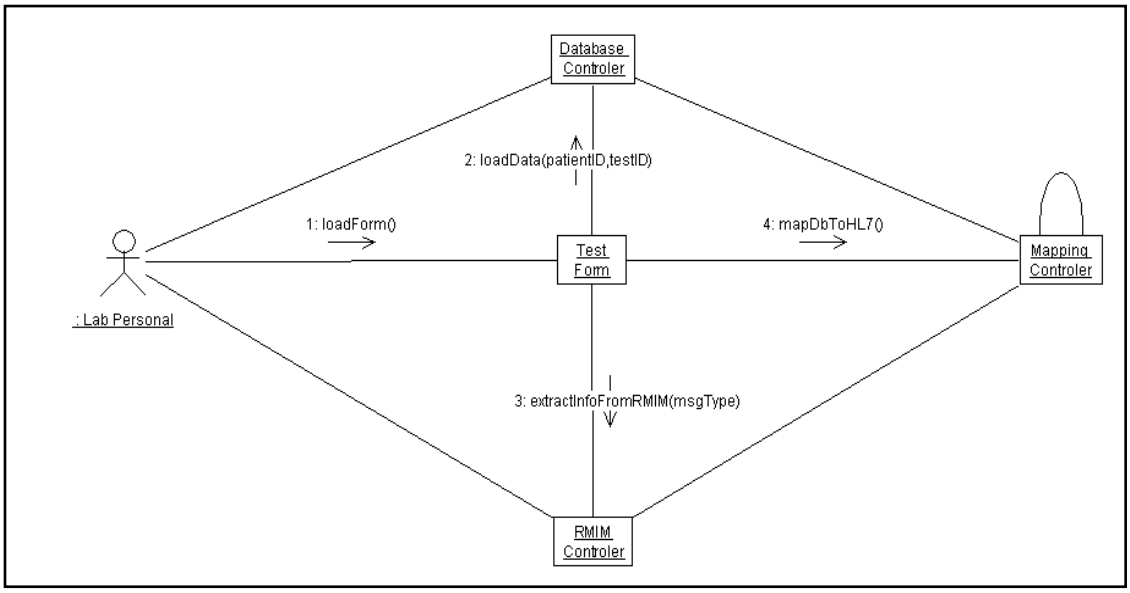

Figure 25: Collaboration Diagram

7.1.3.2 Partial Class Diagram

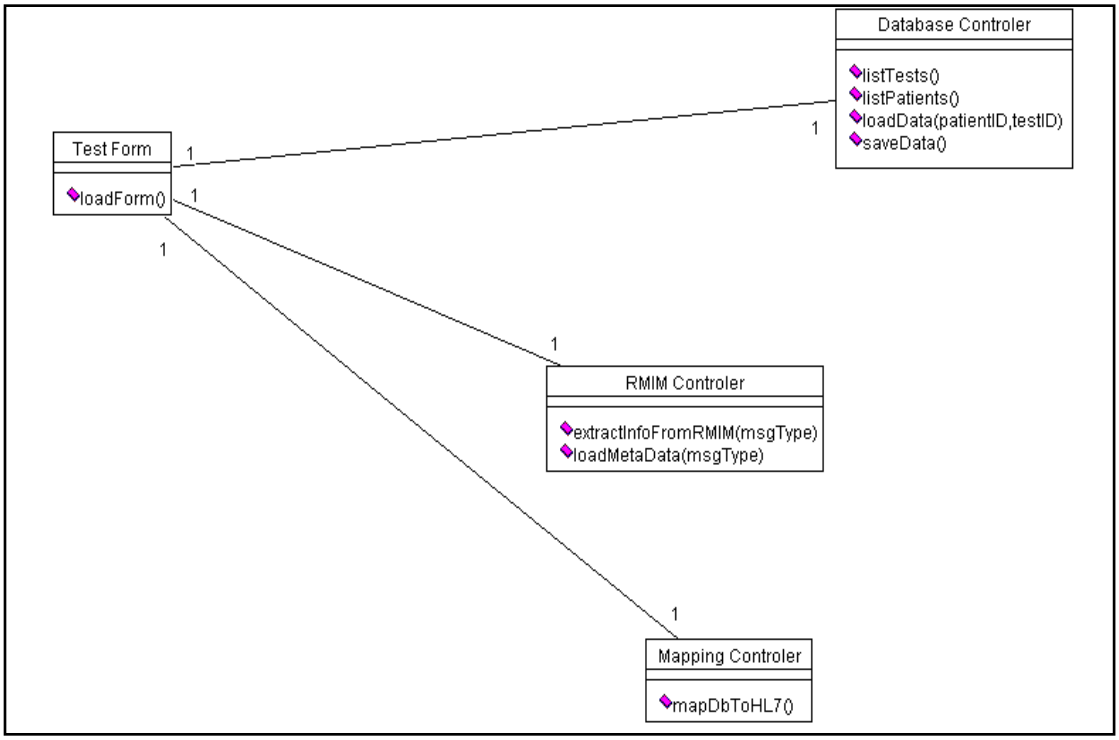

Figure 26: Partial Class Diagram of Map Database to HL7

7.1.4 Generate Message

This use case is responsible for the generation of message. Primary actor for this use case is lab personal and for its correct working it is necessary that data is loaded and mapped to the RMIM elements otherwise it might not work properly. Message will be ready for communication after successful generation.

Lab personal loads patient data from test form by calling loadData(patientID,testID) method. After that metadata of the message type for which message is going to be generated is loaded by calling loadMetadata(msgType). Then there is a mapping controller which maps database objects with the RMIM (metadata) objects by calling mapDbToHL7 (). After that message is generated by calling generateMessage(patientID,testID,msgType).

7.1.4.1 Sequence Diagram and Collaboration Diagram

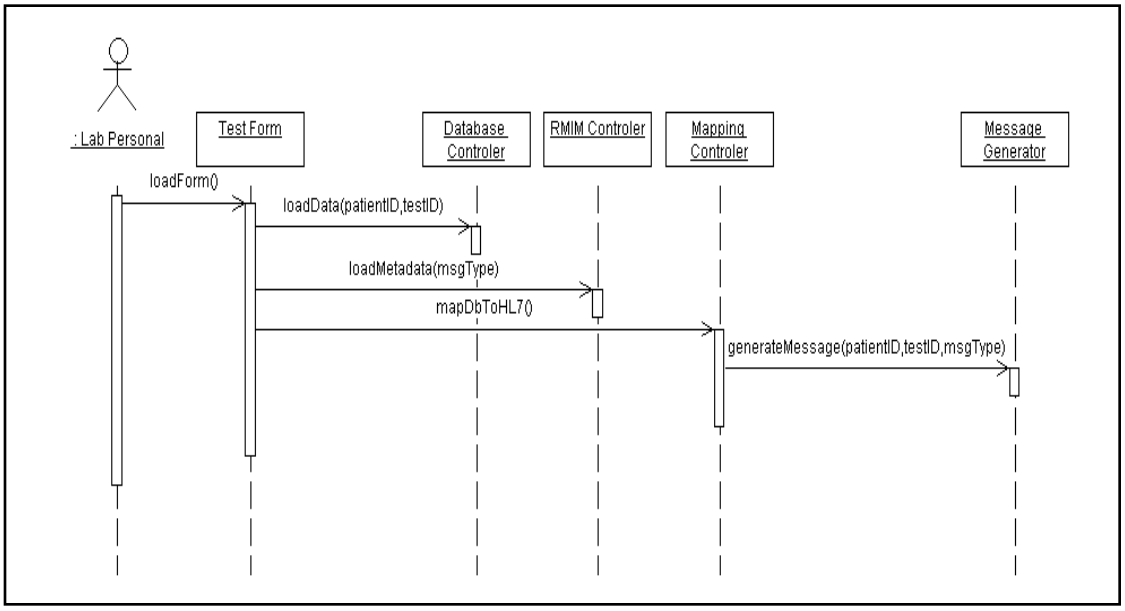

Figure 27: Sequence Diagram of Generate Message Use Case

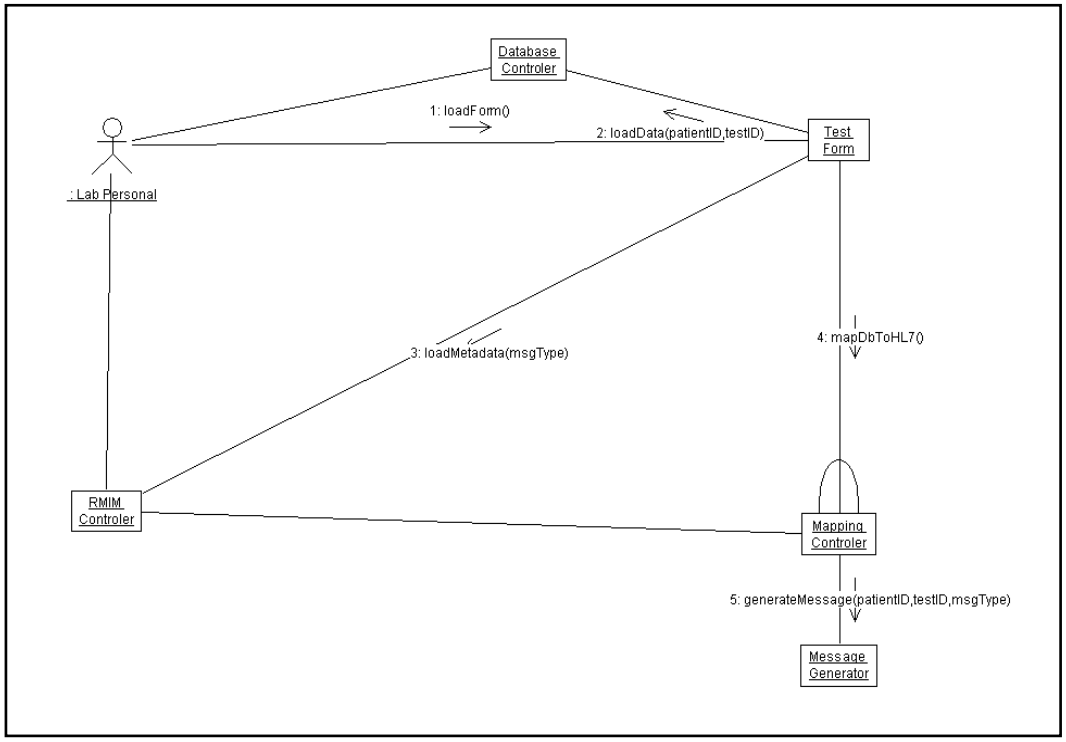

Figure 28: Collaboration Diagram of Generate Message Use Case

7.1.4.2 Partial Class Diagram

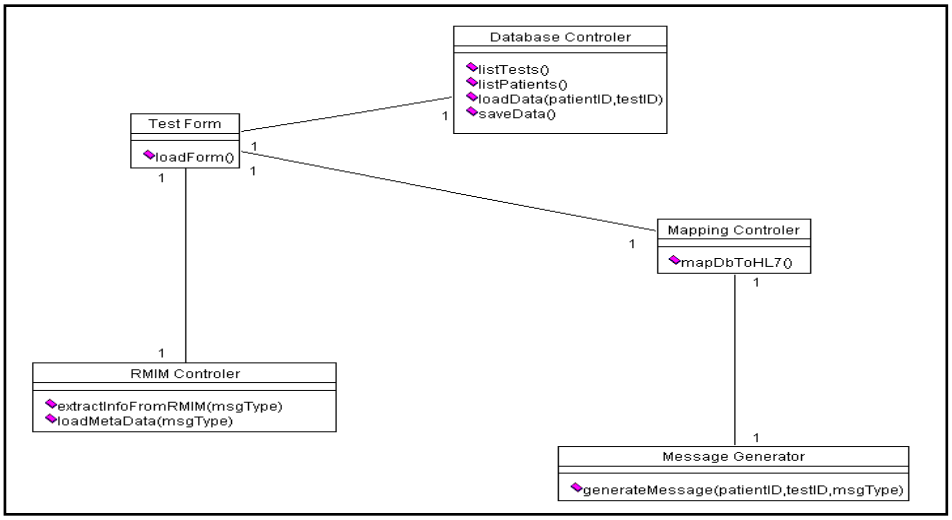

Figure 29: Partial Class Diagram of Generate Message Use Case

7.1.5 Parse Message

Primary actor in this use case is Lab Personal. It is used to parse the message at the receiver's end. Precondition for this use case is that message is successfully sent and validated at receiver's end. After parsing contents of the message would be ready to be mapped to database contents and to be stored in the database.

7.1.5.1 Sequence Diagram and Collaboration Diagram

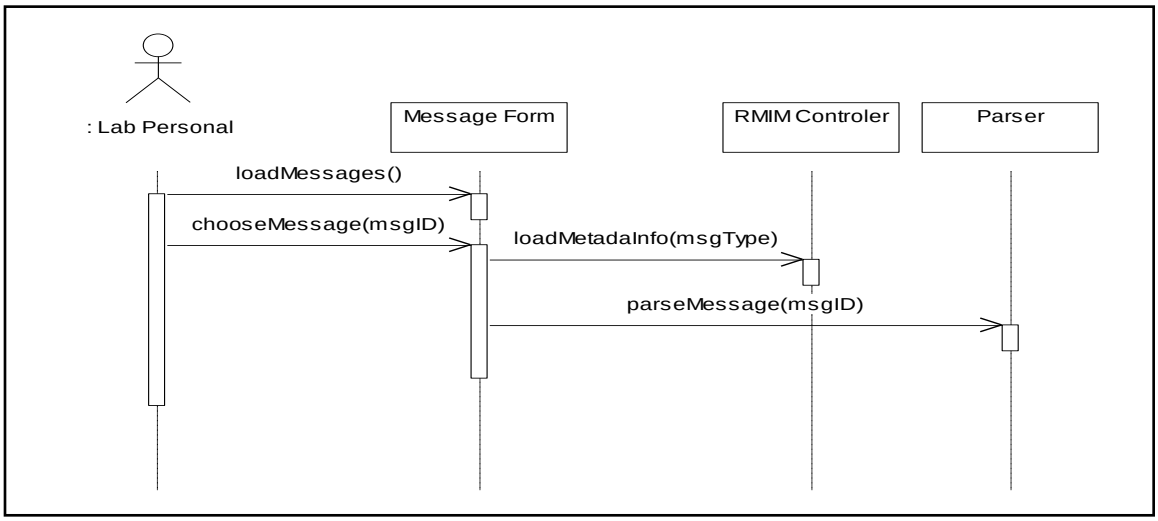

Figure 30: Sequence Diagram of Parse Message

Initially all of the messages are loaded by calling loadMessages() method, after that lab personal chooses a message to which she/he wants to parse by using method chooseMessage(msgID), then metadata information is loaded by calling the loadMetadataInfo(msgType) then parseMessage(msgID) is called in order to parse the message.

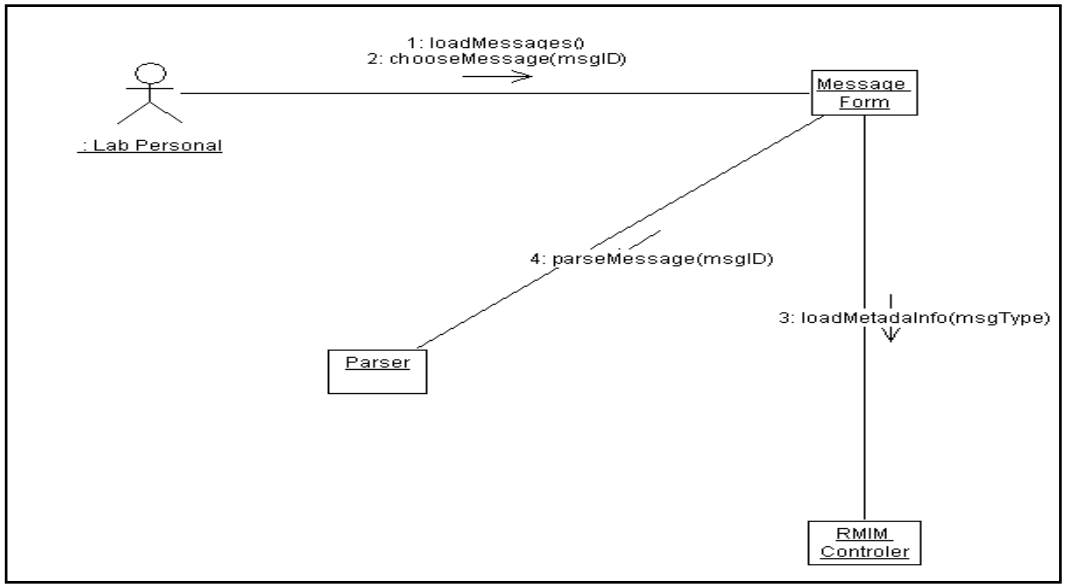

Figure 31: Collaboration Diagram for Parse Message Use Case

7.1.5.2 Partial Class Diagram

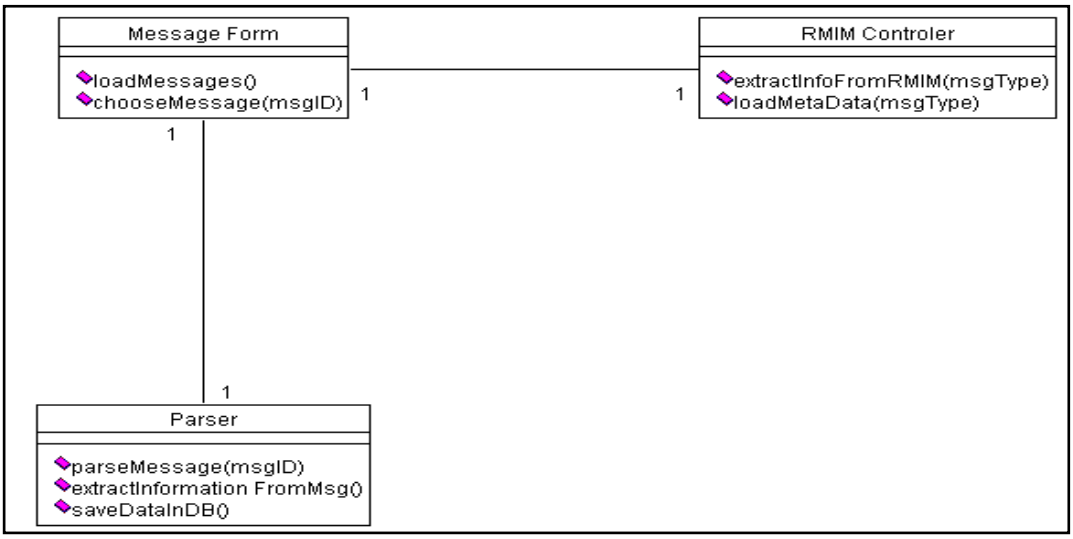

Figure 32: Partial Class Diagram of Parse Message

7.1.6 Map HL7 to Database

This use case is for mapping HL7 objects to the database objects, so that they can be easily stored in the database. Key actor for this use case is system itself. For mapping it is necessary that message is successfully received and parsed at the receiver's end after mapping data contents are ready to be saved in database.

7.1.6.1 Sequence Diagram and Collaboration Diagram

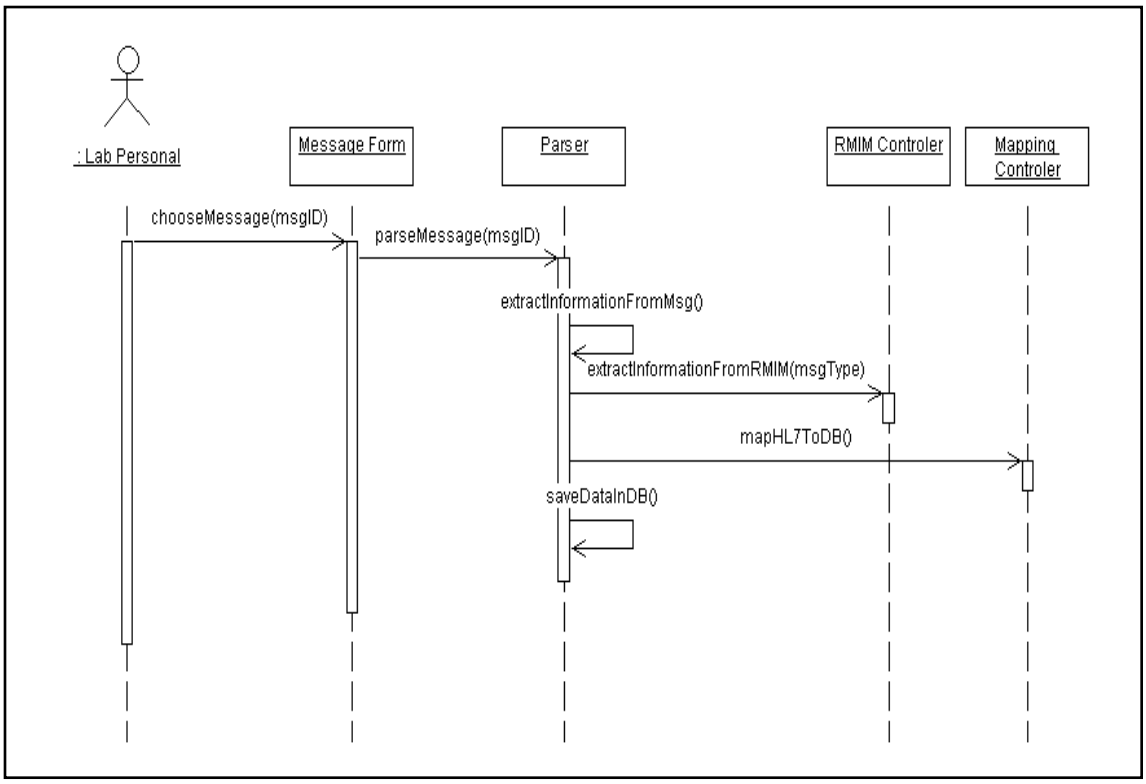

Figure 33: Sequence Diagram of Map HL7 to Database Use Case

Initially a list of messages is shown to the lab personal from there a message can be chosen by calling chooseMessage(msgID), after that message is parsed by calling parseMessage(msgID) method during parsing information is extracted from both message (information which is transferred) and RMIM (information about message type) and then parsed contents are mapped to the database contents. After successful mapping data contents are saved in database by calling saveDataInDB() method.

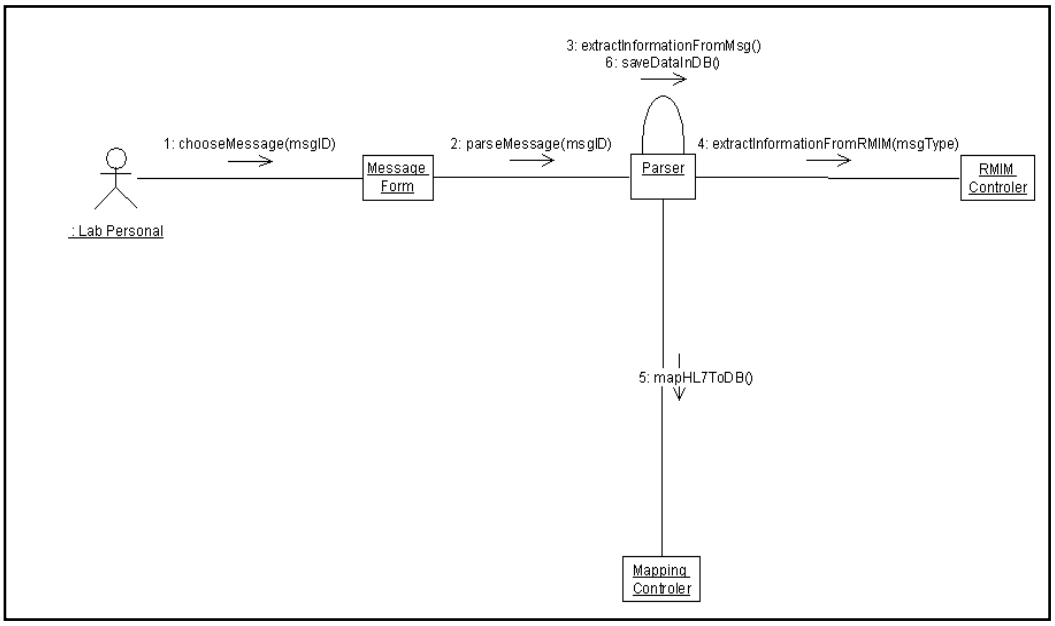

Figure 34: Collaboration Diagram of Map HL7 to Database Use Case

7.1.6.2 Partial Class Diagram of Map HL7 to Database Use Case

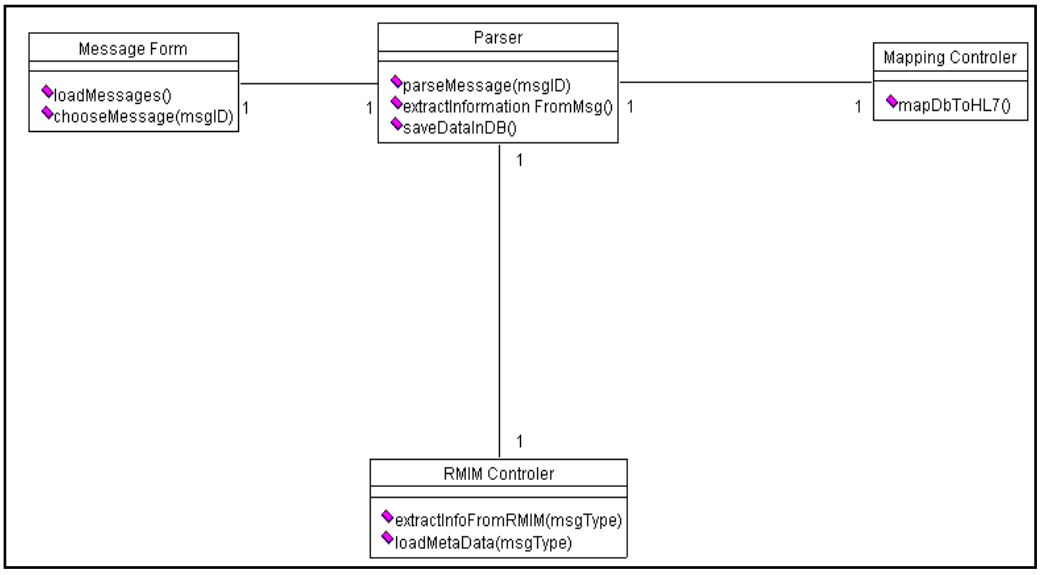

Figure 35: Partial Class Diagram of Map HL7 to Database Use Case

7.1.7 Send Message

This use case defines the scenario of sender application role sending a message. Primary actor for this use case is sender application role.

7.1.7.1 Sequence Diagram & Collaboration Diagram

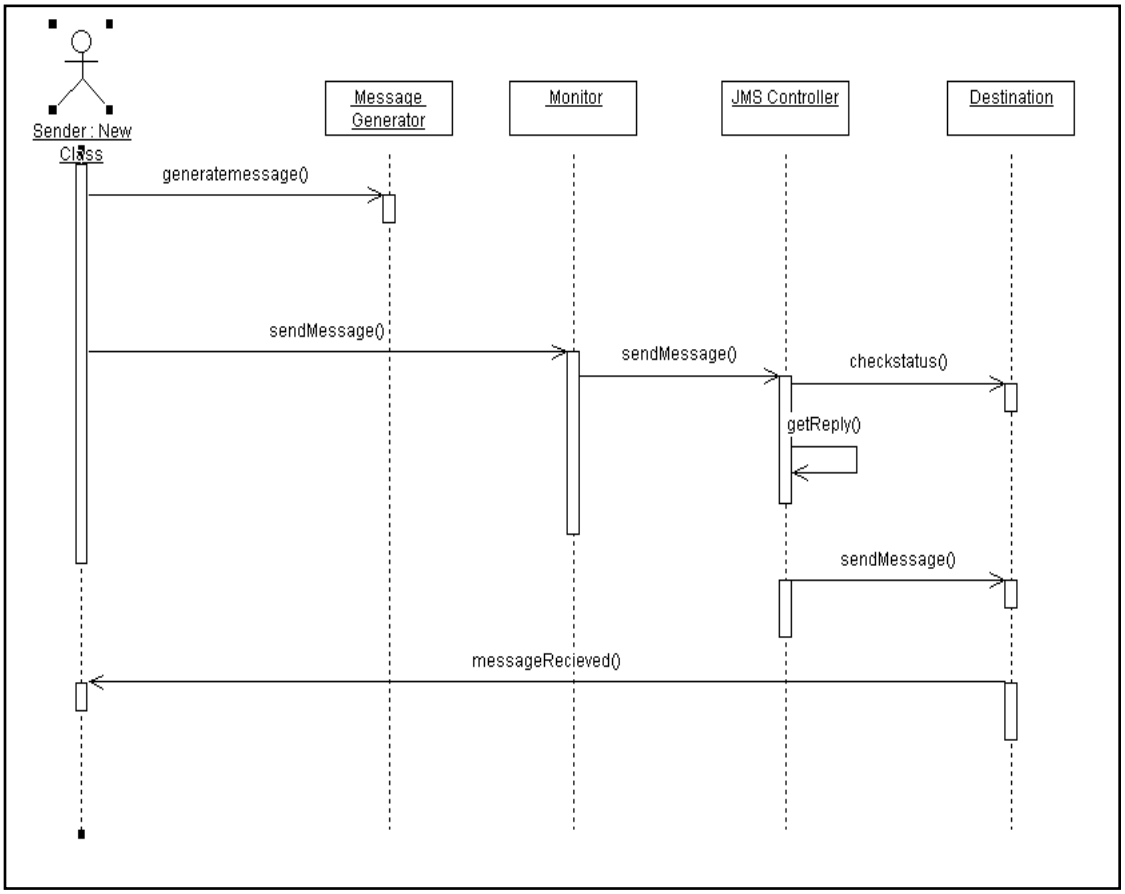

Figure 36: Sequence Diagram of Send Message Use Case

Sender application role receives the message from the message generator. The message is then sent to monitor, which will bypass the 'JMS controller' if the given destination is not busy and send the message to destination. If the destination is not responding, message is handed over to JMS Controller. JMS controller is then going to wait for monitor to tell it when the destination gets ready for message receiving. On trigger from the Monitor, JMS Controller is going to send the message to the desired destination. Message delivery is going to get acknowledged at the end by "messageRecieved()" call.

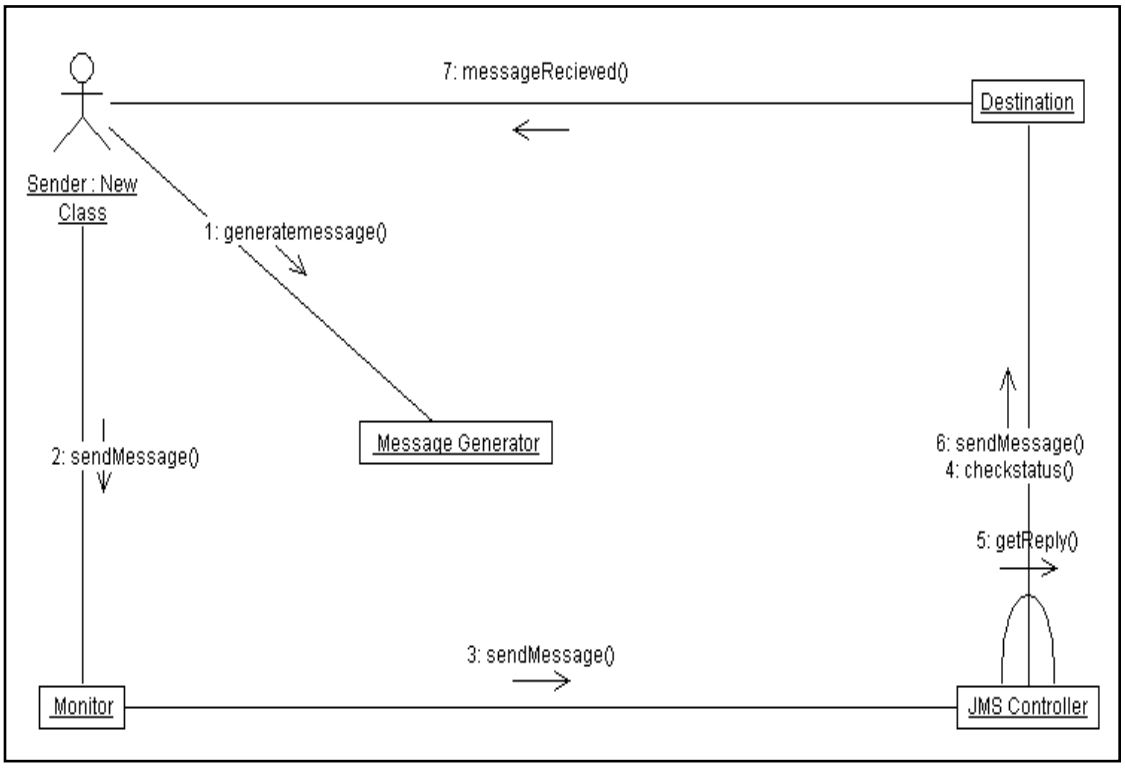

Figure 37: Collaboration Diagram for Send Message Use case

7.1.8 Receive Message

This use case captures the scenario of receiving a message by receiver application role. Primary actor for this use case is receiver application role.

7.1.8.1 Sequence Diagram & Collaboration Diagram

The receiver application role is listening to the communication channel for message retrieval. The server manager, on receiving a message triggers the 'onMessage()' function. 'JMS Controller' will check the status of the 'Receiver' if not ready to take messages, the JMS controller will just en-queue the message. If the Receiver is ready to accept and process messages, JMS is going to pass the message straight to Receiver for further processing.

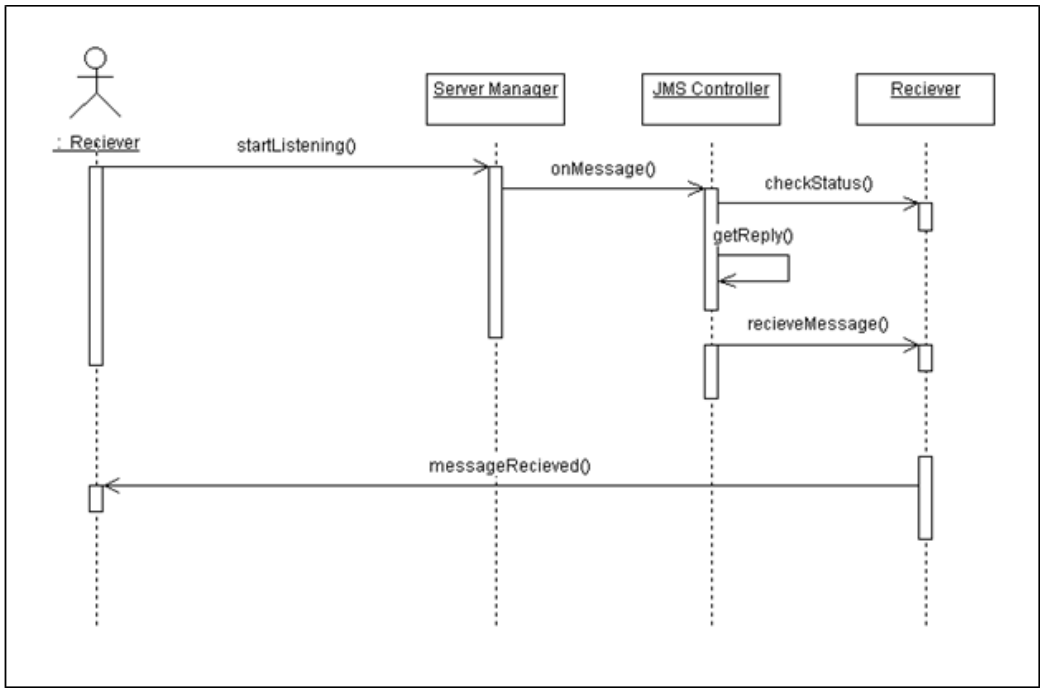

Figure 38: Sequence Diagram of Receive Message Use Case

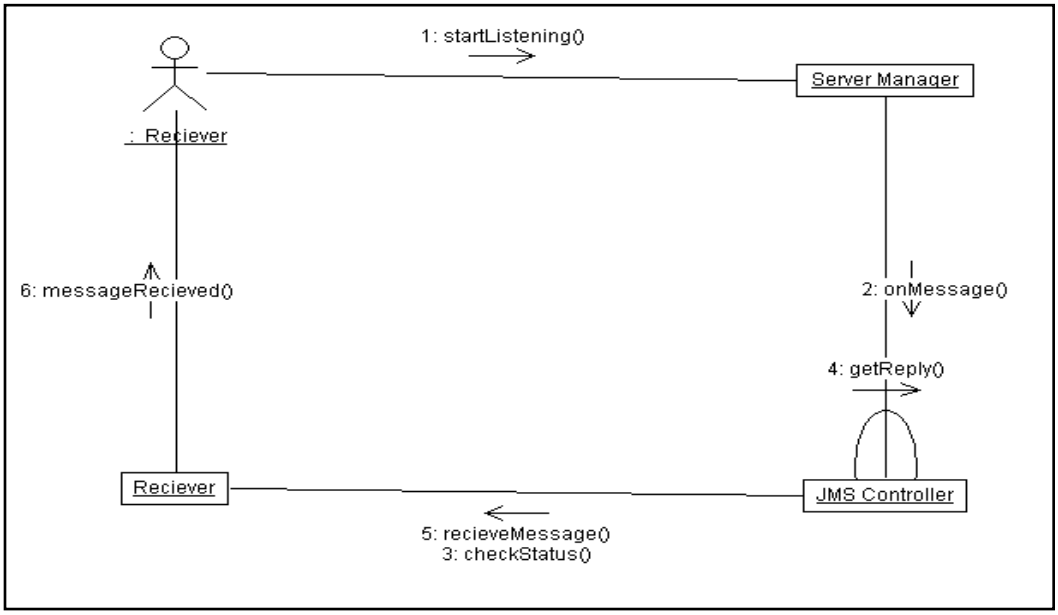

Figure 39: Collaboration diagram for Receive Message Use Case

7.1.8.2 Partial Class Diagram of Send Message & Receive Message

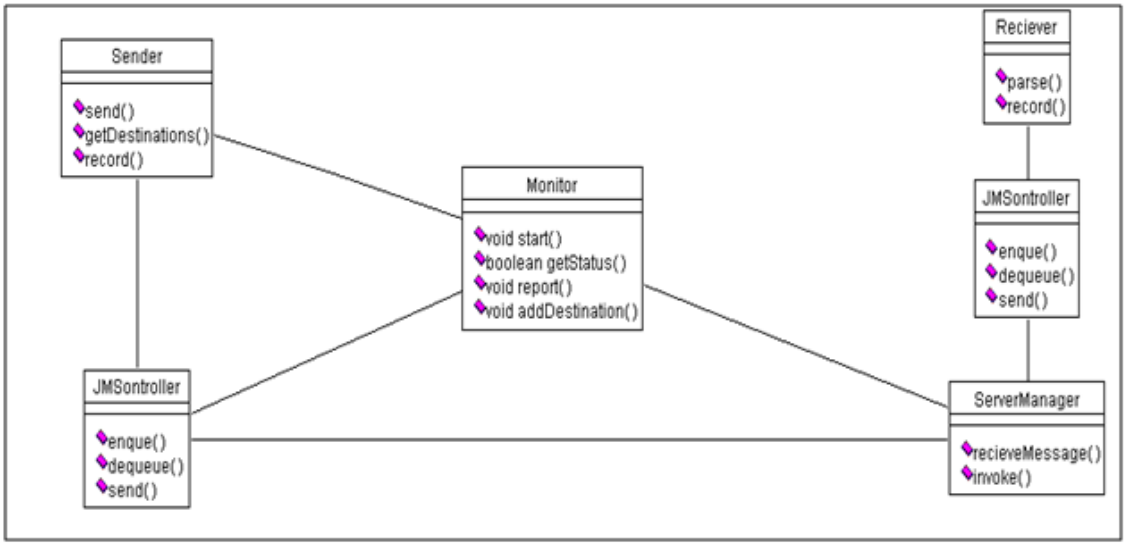

Figure 40: Partial Class Diagram of Send Message & Receive Message Use Case

7.2 Screen Shots

Interfaces are necessary part of any system as these are used by the end users for interaction with the system. Therefore the more interactive, easy and user friendly the interfaces would be, the easier it would be for the end user to communicate use the system. For this purpose lot of interfaces were developed for CITILAB but only few of them are shown below.

7.2.1 Communication Setup

This interface is used by sending side in order to determine status of different branches. It provides information about different branches, their IP address and their status. If the status of a particular branch is “up” than it means that sender can send data to that branch and vice versa. Its interface is shown in Figure 41.

7.2.2 Add Link

This popup is shown when user presses add link button. Here user provides the information about branch and its IP address as a result of it information about that branch is shown in the communication setup. Its interface is shown in Figure 42.

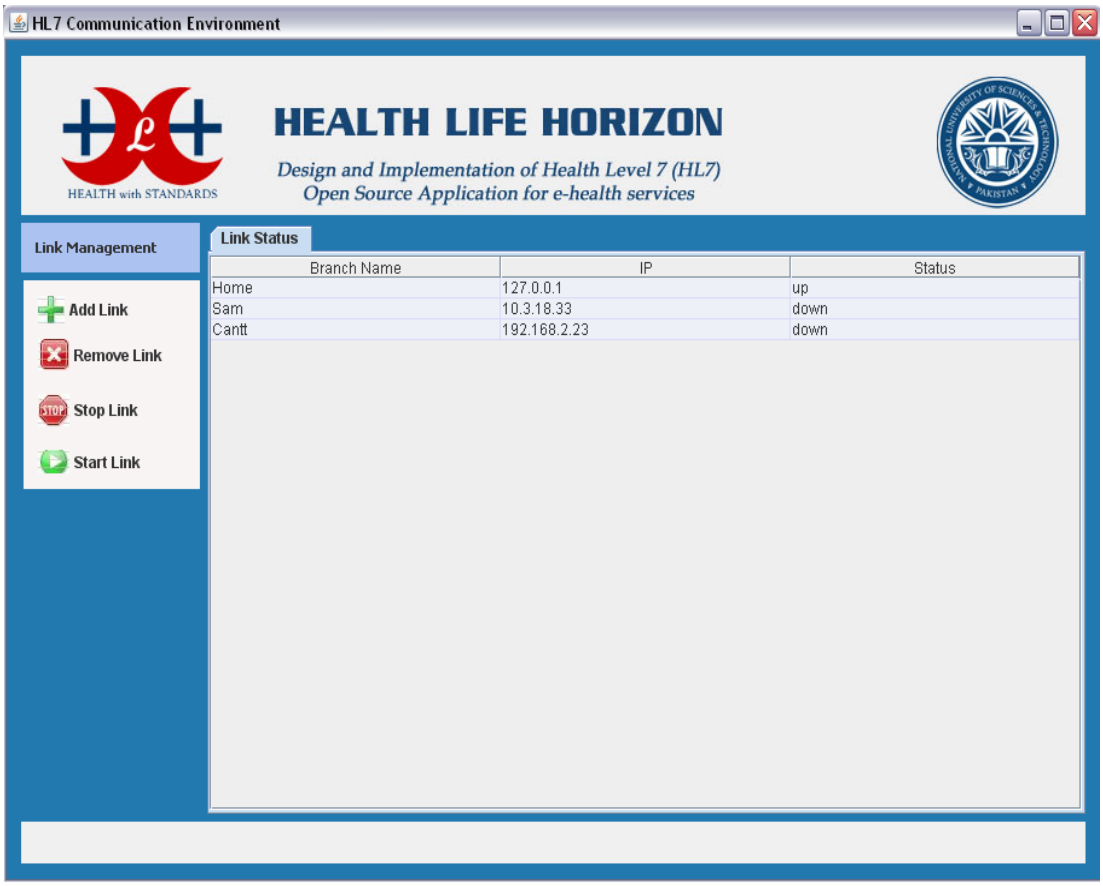

Figure 41: Communication Setup

7.2.3 Test Order Interface

This interface is used by lab personal for getting information about the patients and tests ordered by these patients. Lab personal will has to select a patient and against that patient a list of tests will be displayed. Its interface is shown in Figure 43.

7.2.4 Send Message

This interface shows the message which is generated (by selecting patient and its tests shown in previous interface).When message is generated user will choose a branch from the dropdown list (shown at right corner of interface) and will press "Send Message" button for sending the above message to that particular branch. Its interface is shown in Figure 44.

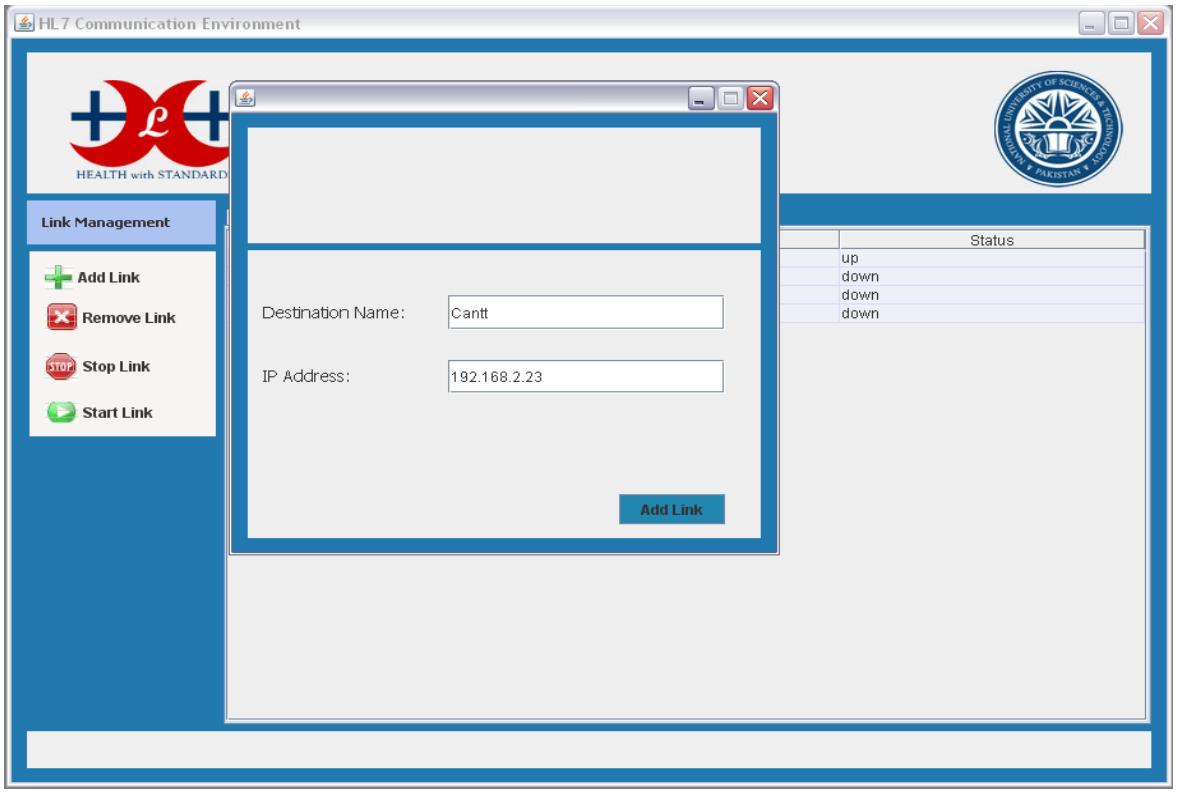

Figure 42: Add Link Interface

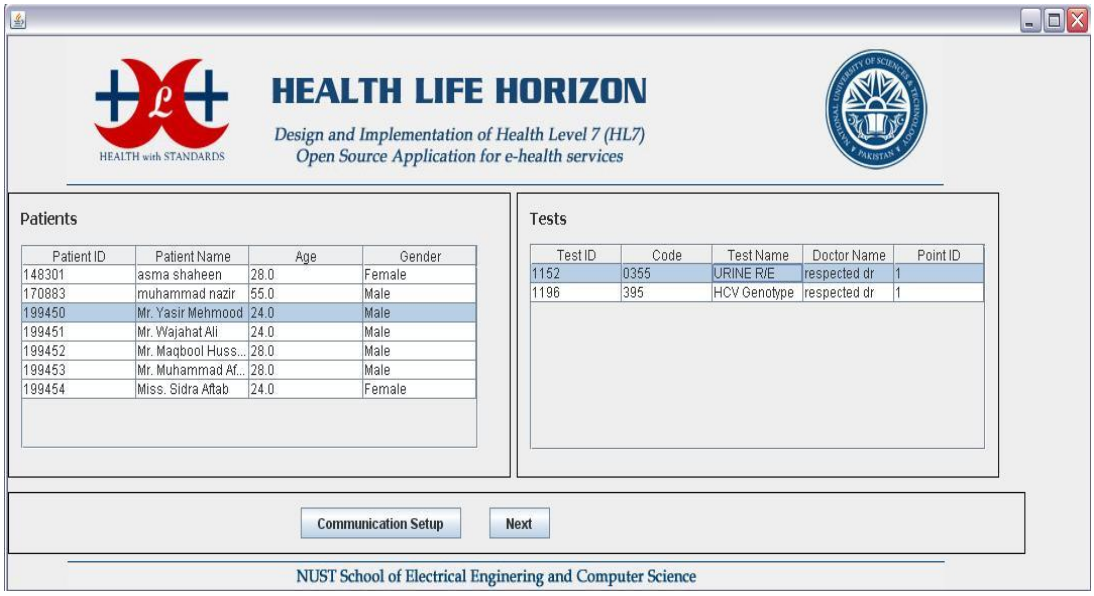

Figure 43: Screen shot for test orders

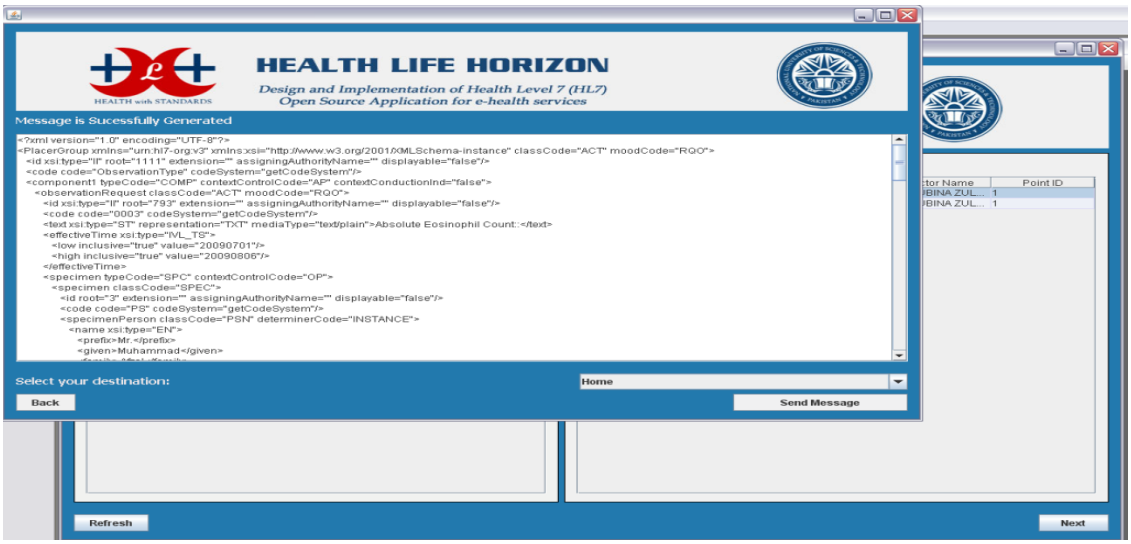

Figure 44: Screen shot for Sending Message

CHAPTER 8

RESULTS AND EVALUATION

In this chapter results of the developed prototype system, discussed in Chapter 5 are evaluated against the existing approach used by HL7 Java SIG API. For this purpose some evaluation criteria is identified, different criteria for message generation and parsing are evaluated, messages are tested on different computers of different specifications.

8.1 System Requirements

System prototype is developed using NetBeans IDE and java development kit 6 so for running this software prototype there is a requirement of Java Runtime Environment 6 and database handling is done using Microsoft SQL Server 2000 which must be installed and database should be conFigured for proper running of this software. In tabular form ideal requirements for this prototype are given in Table 2.

Table 2: Ideal System Requirements

System Processor	2.4 GHz
Hard Disk	40 GB
RAM	1 GB
Operating System	Windows 2000 Server, Windows 2003 Server, Windows XP
Runtime Environment	Java Runtime Environment 6
Database Server	Microsoft SQL Server 2000

8.2 Evaluation Criteria

For evaluation of system different time and space related criteria are proposed and performance of this new technique is measured against the existing technique of HL7 Java SIG API. Details of these criteria are given below.

8.2.1 Time Usage

In this criteria time is measured in generation and parsing of messages for both existing approach as well as the new technique discussed in this thesis. For that purpose two different ways are used in order to access system efficiency using technique proposed in this thesis.

In first method application or prototype is run on a single system of the specifications mentioned in previous section. In this case only time "Test Order" message generation and parsing is measured. In this case parameters of the message are changed i.e. number of tests included in each message are varied and their performance is measured and compared with the results of the existing approach. Results measured in this method are shown in Table 3.

Table 3: Details of results as a result of dynamic message parameters

NO. OF TESTS	MESSAGE GENERATION		MESSAGE PARSING	
	EXISTING APPROACH IN MILLI SECONDS	NEW/PROPOSED APPROACH IN MILLI SECONDS	EXISTING APPROACH IN MILLI SECONDS	NEW/PROPOSED APPROACH IN MILLI SECONDS
1	2259	885	2343	913
2	2277	899	2367	922
3	2290	919	2389	935
4	2312	934	2399	941
5	2330	948	2412	957
6	2345	964	2429	965
7	2367	980	2440	976
8	2382	1004	2456	998
9	2399	1015	2462	1011

Graphical forms of the above table are given in Figure 45.

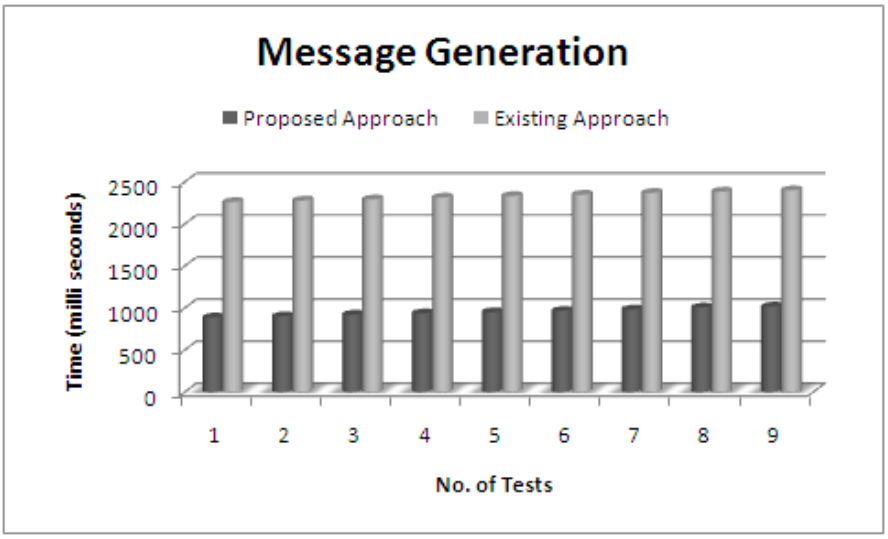

Figure 45: Comparison of approaches message generation

Similarly for parsing of message graph is given in Figure 46.

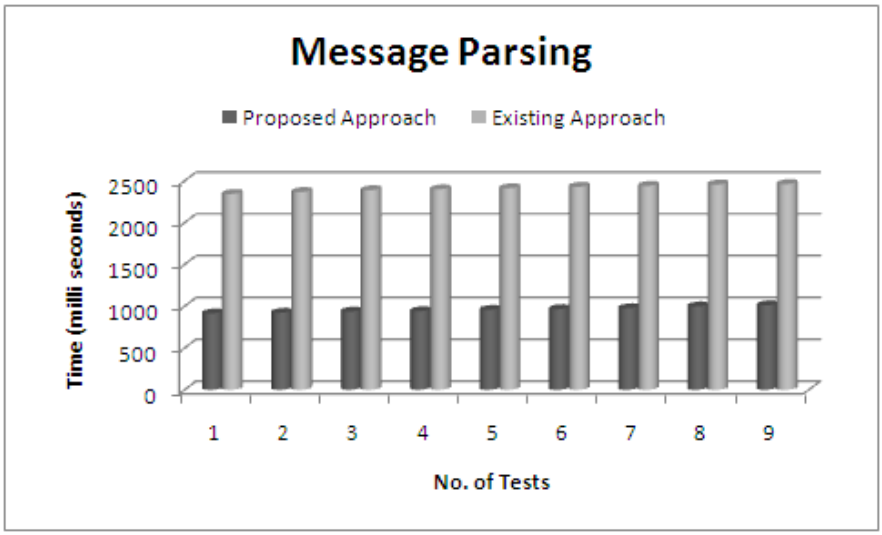

Figure 46: Comparison of approaches in message parsing

In second method system is used in 10 different systems of almost same specifications using new methodology as well as existing technique and different messages are produced. For example if a simple case of "Test Order" message of laboratory domain is taken then this message is generated on 10 different systems and time used for generation of this message is measured for both methodologies. In this case parameters of the message which is generated are static while those of the hardware on which application is running are varied. Details of this measurement are given in table 4.

Table 4: Details of results on different systems

Placer Order		Result Event	
New/Proposed Approach	Existing Approach	New/Proposed Approach	Existing Approach
985	2659	1117	4025
985	3012	1045	4023
985	3025	1245	4025
985	3046	1125	4215
985	3689	1026	4136
985	3656	1045	4289
985	3689	1065	4026
985	4011	1169	4289
985	3698	1123	4726

In this table results gathered by the testing of both “Test Order” and “Test Result” messages of laboratory domain are shown. All of these measurements are taken in milliseconds and these measurements are only for message generation. In graphical form results are shown in Figure 47.

8.2.2 Memory Usage

Memory measurement is taken empirically because it is very hard to measure the performance in terms of memory of the system and it is not in the scope of this thesis. For taking empirical measurements it is assumed that each association which may be used in message generation or parsing is taken as one unit or they are going to consume one unit of memory. On the basis of this principle all of the measurements are taken and their results shows that new technique presented in this thesis is far better than the existing one in term of memory usage.

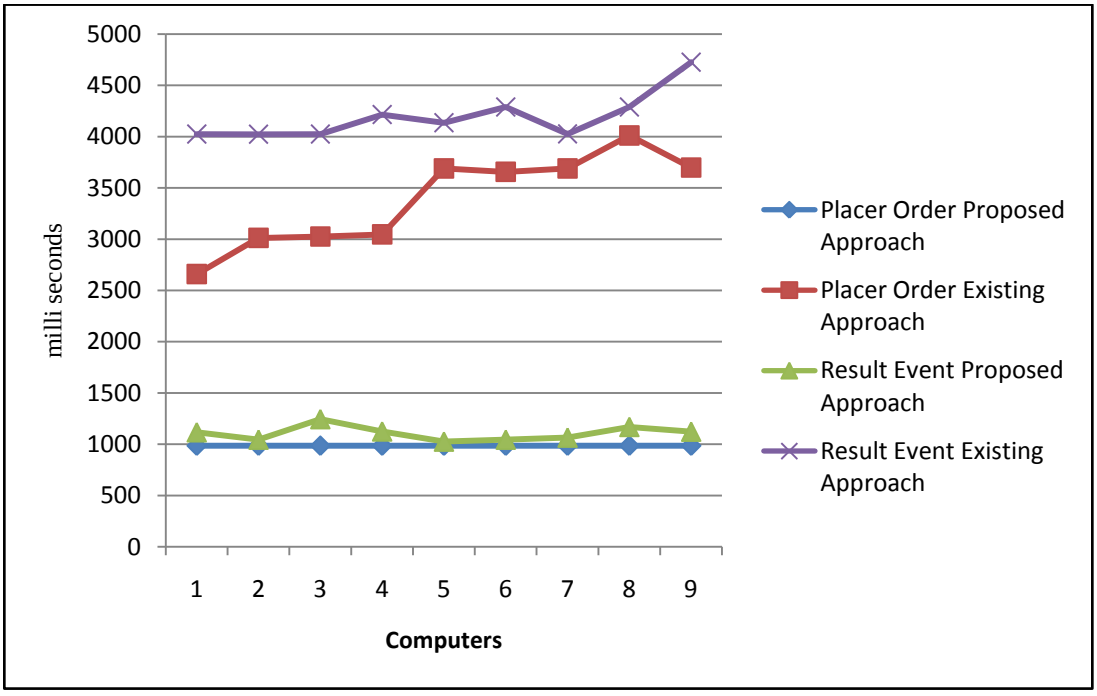

Figure 47: Graphical views of results on different systems

8.3 Message Communication

Currently there Java SIG API does not provide any classes which provide the environment for HL7 V3 messages communication. Work presented in this thesis caters for this thing. Some of the classes of implementation of MLLP are included in this work so that message communication may be achieved. Because without message communication HL7 v3 messages are of no use.

8.4 Significance

In existing methodology, best, average and worst case behaviors are same. The reason is that, in existing methodology all of the associations are loaded no matter whether they are needed or not. This causes memory leakage and wastage of time. In contrast, average case behavior of new (proposed) algorithm (methodology) is far better than the currently used methodology because it loads only required associations due to which lot of memory and time is saved. With this, the problems of cross references and stack overflow errors are solved. Details of it can be seen in section 8.2.

CHAPTER 9

CONCLUSION AND FUTURE WORK

In this chapter research work is concluded and some future directions are described. The chapter is of vital importance because it provides a bird's eye-view of the methodology and gives future directions for new researchers.

9.1 Conclusion

Methodology used by Java SIG API for generation and parsing of messages has some serious performance issues due to use of memory inefficiently. Proposed methodology optimizes the existing methodology. It overcomes the efficiency problems related to time and memory. It improves the metadata loading procedure by loading only required objects rather loading all of the objects in memory. This work is evaluated on different systems of different specifications and in all cases it produced better results than the existing one as shown statistics given in chapter 8. With the usage of this API not only memory leakage or memory wastage is removed but also problem of cross referencing (which resulted in program crashes and stack overflow errors) are handled efficiently. In term of time efficiency it works much faster than the existing one. In worst case it's behavior is same as existing one because in that case it has to load all of the associations in memory while in average and worst case scenarios its performance in terms of memory utilization and time rapidness is lot better than existing one. In short, this technique utilizes memory and time much more efficiently than the existing methodology.

9.2 Future Work

Metadata loading process can be extended by adding machine learning support. E.g. if a message is generated for the laboratory domain, system will keep track of the associations which will be loaded in generation of that message. Record can be kept by adding required associations into database against each message. This process will

continue for some time so that different variation may be stored in the system or record. After that some data mining techniques can be used for retrieving patterns out of the stored messages. It will reduce the burden from application developer to direct system for loading associations.

Another way of bringing efficiency in the process of metadata loading is to put all of the associations in a memory block called page. During generation and parsing of messages each of these associations will be swapped in the memory as per requirements. After utilization of some blocks or pages, these pages can be flushed or cleaned so that the memory may be used by some other processes. By doing so worst case scenario issues will be handled.

REFERENCES

[1] “What is HL7?” [Online]. Available: http://www.hl7.org/

[2] “Health Level 7,” [Online]. Available: http://en.wikipedia.org/wiki/HL7

[3] Meta information loader. [Online]. Available: http://aurora.regenstrief.org /javasig/wiki/HL7 v3Overview

[4] A.Choudhri, L.Kagal, A.Joshi, T.Finin, Y.Yesha. “PatientService: Electronic Patient Record Redaction and Delivery in Pervasive Environments,” *Healthcom 2003*, pp. 41-47, 6-7 June 2003. [Online]. Available: http://ieeexplore.ieee.org/xpl /freeabs_all.jsp?arnumber=1218716. [Accessed August 11, 2009].

[5] J.Walker, E.Pan, D.Johnston, J.Alder-Milstein, D.W.Bates, B.Middleton “The Value of Health Care Information Exchange and Interoperability”. [Online]. Available: http://content.healthaffairs.org/cgi/content/full/hlthaff.w5.10/DC1

[6] J.M.Hook, E.Pan, J.Adler-Milstein, J.Walker “The Value of Healthcare Information Exchange and Interoperability”. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1839305

[7] M.F.Collen “Clinical Research Databases--A Historical Review,” *Journal of medical systems (J Med Syst) 2005,* pp 323-344, November 2005. [Online]. Available: http://www.find-health-articles.com/rec_pub_2132040-clinical-research-databases-historical-review.htm

[8] M. Henderson, F.M. Behlen, PhD, C.Parisot, E.L. Siegel, D.S. Channin, “IHE a Premier: Part 4 .The Role of existing Standards in IHE”vol 21.Dec 2001

[9] http://www.faqs.org/patents/app/20090024152

[10] “Digital Imaging and Communications in Medicine,” [Online]. Available: http://www.ms-technology.com/resource-center/dicom.html

[11] http://en.wikipedia.org/wiki/Health_Level_7

[12] “HL7 Foundation Components,” [Online]. Available: http:// healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide /v3guide.htm#v3fnd

[13] “Reference Information Model,” [Online]. Available: http:// healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide

/v3guide.htm#v3ginfomdl
[14] “Domain Message Information Models,” [Online]. Available: http://healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide/v3guide.htm#v3dmim
[15] “Entry points,” [Online]. Available: http://healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide/v3guide.htm#v3dmimentrypt
[16] “Classes and Colors,” [Online]. Available: http://healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide/v3guide.htm#v3dmimcnc
[17] “Refined Message Information Models,” [Online]. Available: http://healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide/v3guide.htm#v3grmim
[18] “HL7 Messaging Components,” [Online]. Available: http://healthinfo.med.dal.ca/hl7intro/CDA_R2_NormativeWebEdition/help/v3guide/v3guide.htm
[19] “HL7 V3 API Overview,” [Online]. Available: http://aurora.regenstrief.org/javasig/wiki/HL7v3Overview
[20] N. Ilyas, “HL7 communication environment,” BIT Thesis, School of Electrical Engineering & Computer Science, NUST, 2009
[21] “Methodology,” [Online]. Available: http://en.wikipedia.org/wiki/Methodology