# HL7 Communication Environment (HLCE)

**by**

**Nadeem Ilyas**

(2005-NUST-BIT-130)


Project Report in partial fulfillment of

the requirements for the award of

**Bachelor of Science degree in Information Technology (BIT)**

In

**School of Electrical Engineering & Computer Science (SEECS)**

**National University of Sciences and Technology (NUST)**

**Islamabad, Pakistan**

**(2009)**

# CERTIFICATE

It is certified that the contents and form of thesis entitled "**HL7 Communication Environment"** submitted by Nadeem Ilyas have been found satisfactory for the requirement of the degree.

**Advisor: _____**

**Asst. Professor Dr. Raihan Ur Rasool**

**Co-Advisor:_____**

**Associate Professor Dr. Hafiz Farooq Ahmed**

# DEDICATION

**IN THE NAME OF ALMIGHTY ALLAH**
**THE MOST BENEFICENT AND THE MOST MERCIFUL**

**TO MY DEAREST PARENTS**

# ACKNOWLEDGEMENTS

First and for all, I am extremely thankful to Allah the Almighty, for completion of this project. I am also thankful to my family members and especially my father who supported all of my expenses and my mother who motivated and prayed throughout the course of the project.

I am thankful to my project supervisor Dr Raihan Ur Rasool for his support, and encouragement during the course of this project. I would like to thank my Co-Advisor Dr. Farooq Ahmed for providing immense guidance, always keeping me on my toes, and pushing me to work harder.

Also, I would like to pay special thanks to my team leads Mr. Muhammad Afzal and Mr. Maqbool Hussain. Without their persistent help and technical guidance, the completion of this project would have been impossible for me.

# List of figures

# List of Tables

# Abstract

Interoperability among heterogeneous healthcare applications is difficult due to not following standard message communication specification to exchange clinical and administrative data. Health Level Seven (HL7) is the leading standard playing active role to make healthcare applications interoperable.

Focus of this project is to develop communication environment for HL7 v3 message communication. The project is entitled as "HL7 Communication Environment (HLCE)". HL7 V3 implementation requires development of various components namely message manipulation (parsing/generation), database mapping component, and transportation component. Objective of HLCE is interaction with transportation component of HL7. It provides a reliable and robust communication environment for the transportation of HL7 V3 messages. HL7 specifications provides three ways of communication e.g. MLLP, Web Services and ebXML. HLCE is using web services and Minimal Lower Layer Protocol (MLLP) as transportation mechanisms for HL7 messages. To bring in the feature of robustness and reliability, Java Messaging Service (JMS) is used in the system architecture.

JMS is a Sun Microsystems' specification for supporting loosely coupled communication among applications. It enables a synchronous communication among applications in a reliable manner. Message is guaranteed to be delivered at most once by the JMS specifications.

HLCE incorporates the advantages of web services and HL7 standards. Web service brings the notion of platform and language independence for the sake of implementation while HL7 V3 brings solution for global healthcare interoperability.

*Chapter 1*

# INTRODUCTION

This chapter covers the introduction to the problem domain and motivation behind the project. Moreover this chapter also covers the description of arrangement of the documentation.

## 1.1 Introduction

Information technology is revolutionizing every walk of life. IT has the potential to improve the quality, safety and effectiveness in healthcare industry. However, adoption of IT in this industry is not frequent. Low acceptance of IT in healthcare industry is due to the complexity of IT investment, which is something that goes beyond the acquisition of technology or equipment. It tends to alter the work processes and culture. Making sure that physicians, nurses and other staff use it is indeed a big challenge.

"An EU project has demonstrated that information technology can provide enormous benefits if the technology is properly implemented" [1]. The "e-Health Impact" project developed a methodology for assessing the economic impact of e-health solutions and then evaluated the economic benefits of introducing new technology in healthcare. Electronically enhanced healthcare has long been promoted as reducing costs, improving quality and efficiency and treating more patients with the same resources. However, no reliable data had been available to support this claim.

The 'e-Health Impact' project, which finished in May 2006, conclusively demonstrated that there is over a 2:1 ratio between economic benefits and costs [1]. In other words, the benefits gained from implementing e-health systems are more than two times greater than the additional cost of implementing them.

## 1.2 Interoperability and Healthcare

Interoperability and health information exchange are best understood as business concepts not specifically as technical concepts. The technical feat of how banks cobbled together the ATM network or point-of-service credit cards may have been interesting at some point in history, but the lasting conversion of these developments is the portability of finance and credit all over the world and its forward movement into every setting where commerce occurs.

Similarly, it is interesting to visualize the technical complexity of healthcare standards, security, architecture, and other technical advances that have made healthcare next on the list of industries that can become interoperable and consumer-centric. There will be a huge impact of that interoperability on the structure and functioning of the healthcare.

### 1.2.1 Realizing the benefits:

Although the benefits of health care information exchange and interoperability (HIEI) are large, they may be difficult to realize.

First, interoperability benefits are highly detached across many stakeholders. Some could lose from disruption of long-standing industry practices, particularly vendors who rely on custom integration of their products for revenue and who use the lack of interoperability as a customer maintenance strategy [2]. Second, the negative network externalities and first-mover disadvantage that penalize early adopters make it difficult to synchronize the behavior of the market so that interoperability can gain a foothold. Just like the fax machine, the last to install an interoperable EMR benefits from everyone else's prior investment, and the first to install bears most of the cost. Third, interoperability first movers have faced many barriers and challenges that have resulted in partial success, slow progress, and outright failure. "Interoperability may be beneficial, but it is certainly not easy" [2].

## 1.3 Problem Definition

Problem domain of the project is highlighted in following paragraphs.

### 1.3.1 Lack of Interoperability

Lack of interoperability is one of the biggest challenges for the health care industry today. Interoperability helps in reduction of medical errors by enabling the sharing of clinical and administrative data among healthcare centers. In USA alone the medical errors cause almost 0.2 million deaths and more than 1 million fatal injuries. Lack of interoperability is mainly due to lack of a standard communication specification.

### 1.3.2 Lack of Reliability

HL7 communication specifications provide us three ways for transportation of HL7 V3 messages using

- Minimal Lower Layer Protocol (MLLP),
- Web Services and
- Eb-XML.

The implementation of MLLP and web-services for HL7 V3 for message transportation is unreliable as the receiver should always be in listening mode to receive message. There can be a scenario when communication channel is not working or receiver crashes, sender will have to wait until the receiver starts listening again or the communication channel gets restored. Messages can get lost as well in such scenarios.

## 1.4 Proposed Solution

HL7 V3 is one of the well known standards to make healthcare applications interoperable. Implementation of HL7 V3 is divided into various components. Main

components are Transmission component, Parser component, Generator component, Database mapping component and Communication Environment.

Main focus of my project is Communication Environment. To bring the notion of reliability Java Message Services (JMS) is embedded into the solutions architecture. This enables synchronous as well as asynchronous communication among healthcare applications in a reliable manner.

JMS is Sun Microsystems' standard API model for message queuing systems. It provides different levels of message delivery assurance and it does not matter if it is used with wireless or wired communication protocol. Message persistence can be achieved using any RDBMS or ODBMS with JMS.

## 1.5 Outline of Thesis

The arrangement of the rest of document is as following.

Chapter 2 covers the literature survey that I conducted; it covers some description of some open source healthcare applications available and some reference documents available. Chapter 3 covers the analysis part. Requirements gathering phase is described in this chapter its main output is SRS document. Chapter 4 covers the design phase of the project, some important use cases are explained there. Chapter 5 covers the description of the implementation part. Chapter 6 covers the conclusion part. It includes the experience gained through the project.

*Chapter 2*

# LITERATURE REVIEW

This chapter focuses on the healthcare reference systems and documents that I thoroughly reviewed for development of HL7 communication environment. There are different open source and proprietary software available related to healthcare industry. Some of famous open source healthcare applications are:

- Mirth Server
- Glass Fish ESB
- BOTS (open source EDI translator)
- Open EMED
- I2b2

For embedding the notion of reliability and robustness in the system, there was need of an open-source message queue. Following implementations were available:

- JBOSS Messing (open source JMS implementation)
- Open Message Queue(open source JMS implementation)
- JORAM (open source JMS implementation)

HL7 specifications provide us with the latest developments in HL7 standard. HL7 specifications are of two types, ballot and normative. Normative is the form of HL7 specifications which is published after the HL7 board meeting approval that cannot be changed, while 'ballots' are recommendations by the participants and which is subject to change when advanced and useful idea about a domain is confronted.

JAVA Caps is another system that is used for integration and reliability purpose. It is a huge system build upon open Enterprise Service Bus (ESB)

technology, which is open source. But the product itself is not open source as it is sold to enterprises. Its documentation and API docs were quiet related to project.

Parser and Generator of HL7 v3 messages are implemented in JAVA SIG API, the API used for integration with the Parser/Generator component.

## 2.1 Introduction to HL7 V3

Health Level 7 (HL7) is a non-for-profit organization, established in 1987, provides international healthcare standards. They provide messaging standards, services and now EHR (Electronic Health Records), which enables different healthcare applications to exchange administrative and clinical data. HL7 developed two messaging standards; V2 and V3. HL7 V2, instead of its wide acceptance in USA, lacks interoperability which is mainly due to absence of standard information model and optionality.

There are some differences related to design and architecture of HL7 V2 and V3. Version 2 of HL7 is an EDI-based approach for exchanging healthcare information and has enjoyed widespread acceptance due to its easy implementation.

Version 3 of HL7 is a modeled and structured approach that uses XML for communicating healthcare information. HL7 Version 2 enables adopters to establish and agree upon EDI message formats for healthcare applications. HL7 Version 3 provides with the possibility of generating messages through a proper efficient procedure and structured document content that is used by healthcare applications in the healthcare enterprise.

The main advantage of HL7 Version 3 is to abolish unnecessary negotiation between healthcare partners that Version 2 frequently requires. Version 3's use of a formal object model and the rigorous application of UML to produce specific document types that provide a way to introduce greater interoperability among HL7-compliant systems than Version 2 implementations can assure.

## 2.2 Java SIG API

The Java SIG team approached their design with the following principles.

- Influence the fundamental RIM without reifying artifacts of the RIM (e.g., RMIMs or DMIMs) for assurance of more robust and stable design. When new message types are approved, they do not need new object classes.
- Minimize any interpretation of what a RIM object represents. This assures the widest applicability of the objects' use in any domain of healthcare.
- Exploit the wealth of effort represented by HL7's repository of hierarchical message descriptions (HMDs) in a systematic way.
- Provide a generic solution for developers about HL7 domain and not one that targets any specific subset of interest areas, such as laboratory, patient record, or pharmacy.

In designing the API for HL7 Version 3 according to the principles outlined above, the Java SIG had two key goals in mind:

- Provision of capabilities that are only specific to HL7 that are not currently present in Java language or other APIs. This reduces the confusion of the developer and defines the scope of HL7 V3 API.
- Implementations of HL7 Version 3 capabilities that access the object-oriented capabilities of the HL7 Version 3 RIM so that developers can concentrate on the higher-level tasks their healthcare applications are designed to perform.

## 2.3 Mirth server

Mirth name is derived from a Swiss word meaning knife, is open source software and is specifically designed for healthcare applications that want their system to be made interoperable with HL7 V3 compliant systems. Figure 1 shows the administration window provided for the MIRTH server.

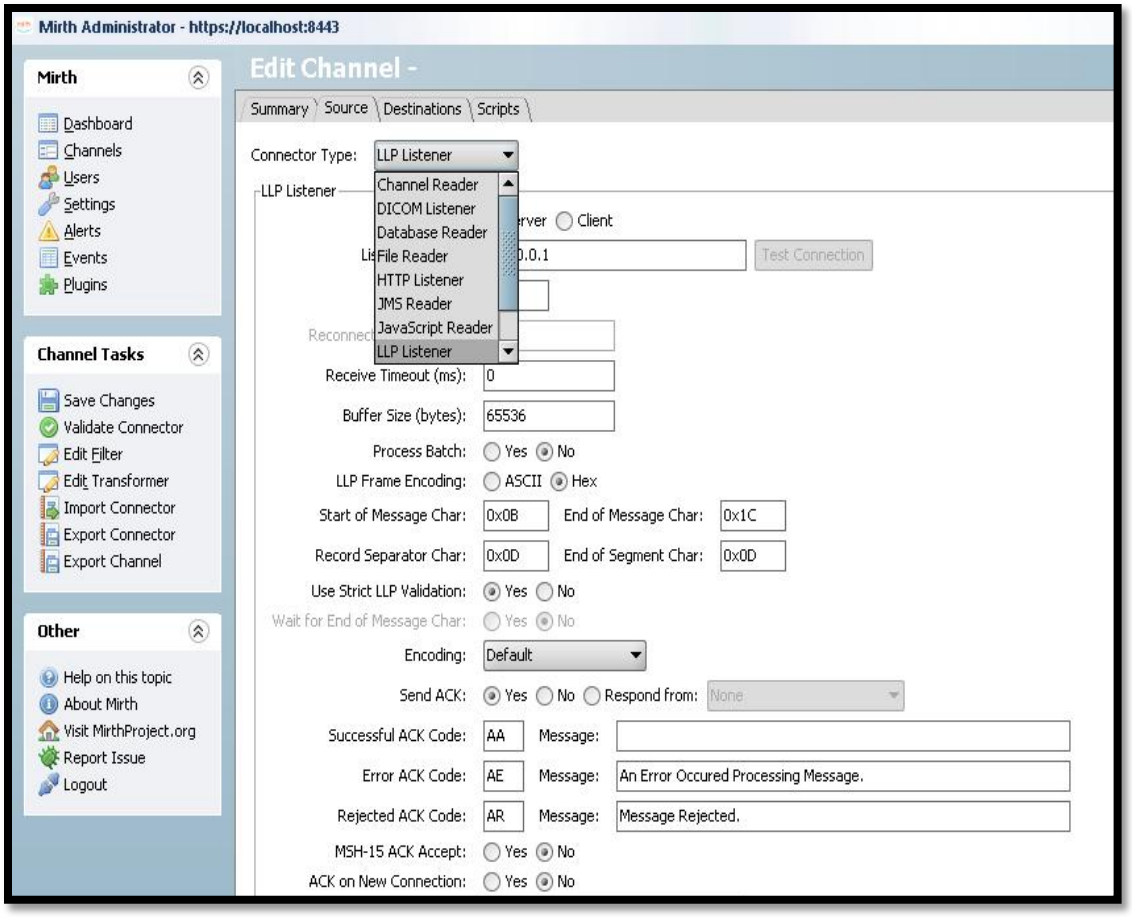


Figure 1: Screenshot of the Mirth Server Configuration

Mirth's current stable version of the system available for download is 1.8.0. It is available for windows as well as Linux platform. It was quiet related to the project for me to study this server as this is developed in JAVA.

It supports one-one or one-many message routing. It supports major protocols used for transportation of medical information. Supported protocols include DICOM,

Database reader, File reader, HTTP, JMS, Java script reader, LLP reader, Web Services and TCP. Database reader includes mysql driver, MS ACCESS, Oracle 10g support and MS SQL Server.

It also provided custom scripting options to modify the incoming or outgoing information according to users' needs. Main dashboard shows the status of messages, whether they are delivered, queued, dumped etc. It also shows the channel status being down, busy, waiting etc. Being open source gives the software developers to build upon the source or contribute to the system. The main deficiency of the Mirth server is its documentation.

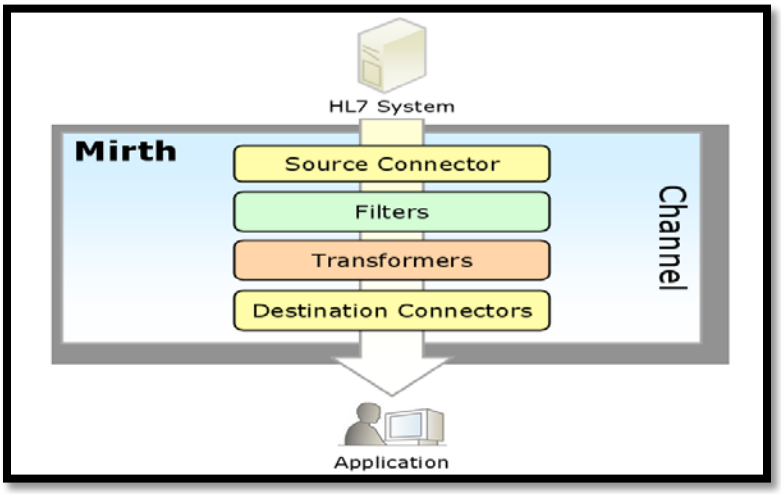


Figure 2: Mirth Architecture [3]

Main architecture of Mirth server is shown in figure 2. It acts as a channel for adding interoperability between HL7 compliant and HL7 non compliant systems.

Though Mirth server is a very good scalable, open source system but it lacks:

- Proper validation procedure for HL7 V3 messages.
- End-to-End transportation mechanism

## 2.4 Java CAPS

Sun JAVA Composite Application Platform suite (CAPS) is a software suite from Sun Microsystems that is standard based and extensible. It is build upon open source technologies and maintained by Open Enterprise Service Bus (ESB) community.

Java CAPS is an excellent candidate to build a messaging system as it works as a backbone for the enterprise communication. It is a many-to-many middleware messaging system, suitable for implementing both event-driven and service-oriented architectures.

As a backbone messaging infrastructure, Java CAPS supports a number of Java Message Service (JMS) implementations from both Sun Microsystems and third parties. This is one of the reasons for this software suite to be reasonable for many to many messaging middleware.

In JAVA CAPS both point-to-point and publish/subscribe messaging are supported and JMS API Kit is available to interface external applications directly to the JMS infrastructure.

Message transformation logic can be implemented in Java (Java Collaborations), Extensible Style Language (XSLT Collaborations), and Business Process Execution Language (BPEL). Message routing can be implemented in Java and BPEL which gives some choice of implementation. It provides about 80 adapters for various communication standards. For health care industry it has support for:

- processing HL7 messages (V2, V3)
- for HL7 protocol connect
- for HIPAA

This software suite is build upon open source technologies but the Java CAPS API docs are available at [4].

It can be extremely helpful in getting some idea about the workflows and method calls in the JAVA CAPS suite. This will ultimately be helpful in deciding the architecture of our system.

## 2.5 Java Message Services

JAVA Message Services (JMS) is a specification by Sun Microsystems that describes a common way for Java programs to create, send, receive and read distributed enterprise messages [5].

JMS is mainly used in software environments where loosely coupled communication is required. This increases the efficiency of the environment by allowing the communication among modules without the communication dependency on other modules.

JMS solves the issue of scalability which mainly resulted in heavy load of communication on the server side. If the server is too busy to handle the requests from the client, message can be queued and responded systematically.

JMS enables asynchronous communication among the applications to promote loosely coupled environment. JMS promises once and only once guaranteed delivery of messages in point to point implementation model.

### 2.5.1 JMS administration

There are two types of objects in JMS programming model connection factories and destinations. Administrator adds and manipulates these objects using the administrative console. The JMS Client looks up for the objects in the Java Naming

Directory Interface (JNDI) namespace. Clients bind to those objects to use them. Figure 3 shows the administration architecture for JMS.

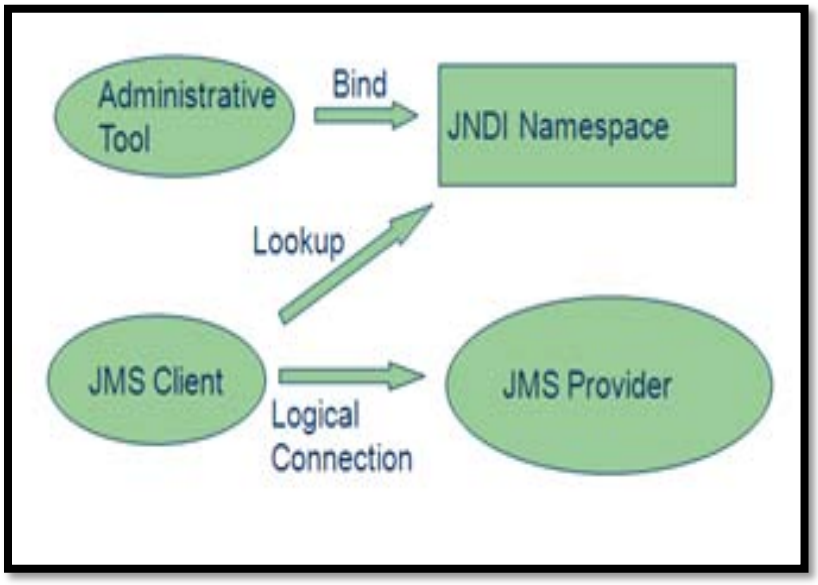


Figure 3: JMS Administration [6]

### 2.5.2 JMS API Programming Model

In JMS API programming model connection factory creates a connection object on behalf of the client and passes it on. Using connection object the session object is created. A connection can have multiple sessions. Each of the session has a consumer and a message producer. Message delivery is guaranteed in point to point model.

Factory design pattern is used when there is complexity involved in the implementation of certain objects and clients wants some interface to shield the complexity and provide the instance of the object on client's behalf. Figure 4 shows programming model for JMS.

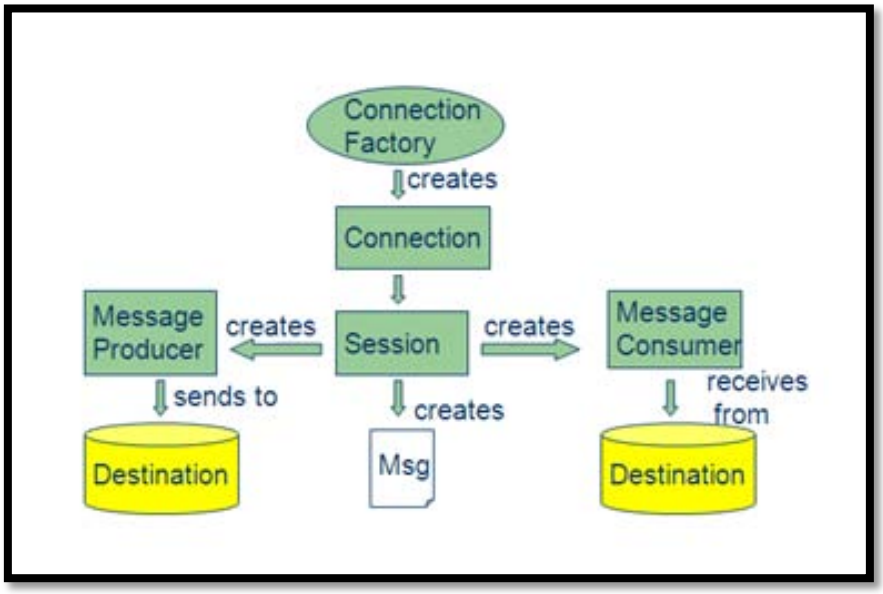


Figure 4: JMS API Programming model [7]

## 2.6 HL7 Specifications

Transmission Specifications address the following aspects about the communications environment that is considered common to all HL7 Version 3 messaging implementations.

HL7 V3 message is not a simple message but a composite message. It is composed of:

- Transmission Wrapper
- Control-act Wrapper and
- Message payload

Transmission Wrapper is compulsory for every message. It tells who was the generator of message, what was the time of creation. Control-act wrapper contains the instructions to handle the message, it is not compulsory for all messages. Message payload contains the main information that the sender wants to transfer. HL7 specifications provide specifications for the composite HL7 version 3 messages.

HL7 specifications provide a protocol for reliable message delivery. Sequence number protocol is provided for synchronization of communication among components.

HL7 specifications provide generic "Communication Roles" that support the modes of HL7 messaging and message control events that describe a framework for generic HL7 messaging.

*CHAPTER 3*

# ANALYSIS

Main motivation behind HL7 Communication Environment is to provide a reliable transportation mechanism for exchanging clinical and administrative data using HL7 V3 specifications. Main features of the end product are reliability and robustness.

In analysis phase the main tasks are to identify the problem and requirements gathering. Primary output of this phase is an SRS document, which contains functional and non functional requirements. Use cases are also an important part of SRS document.

In this phase solution is not emphasized but the problem definition is the main target. As the better requirements gathering will result in a good solution.

In Object Oriented Analysis and Design (OOA/D) approach focus is on identifying and defining objects in problem domain. Iterative model is best suited for small and medium software projects.

## 3.1 Types of requirements

There are five types of requirements:

- Functional
- Supportability
- Usability
- Reliability and
- Performance

**a) Functional:**

It focuses on core functionality of software system for example security, capabilities etc.

**b) Non functional:**

All the remaining categories of requirements are categorized under non-Functional requirements. Non-functional requirements aid the functional requirements to get accomplished. Examples are adaptability, maintainability, internationalization etc.

## 3.2 Use Case Model for HL7 communication environment

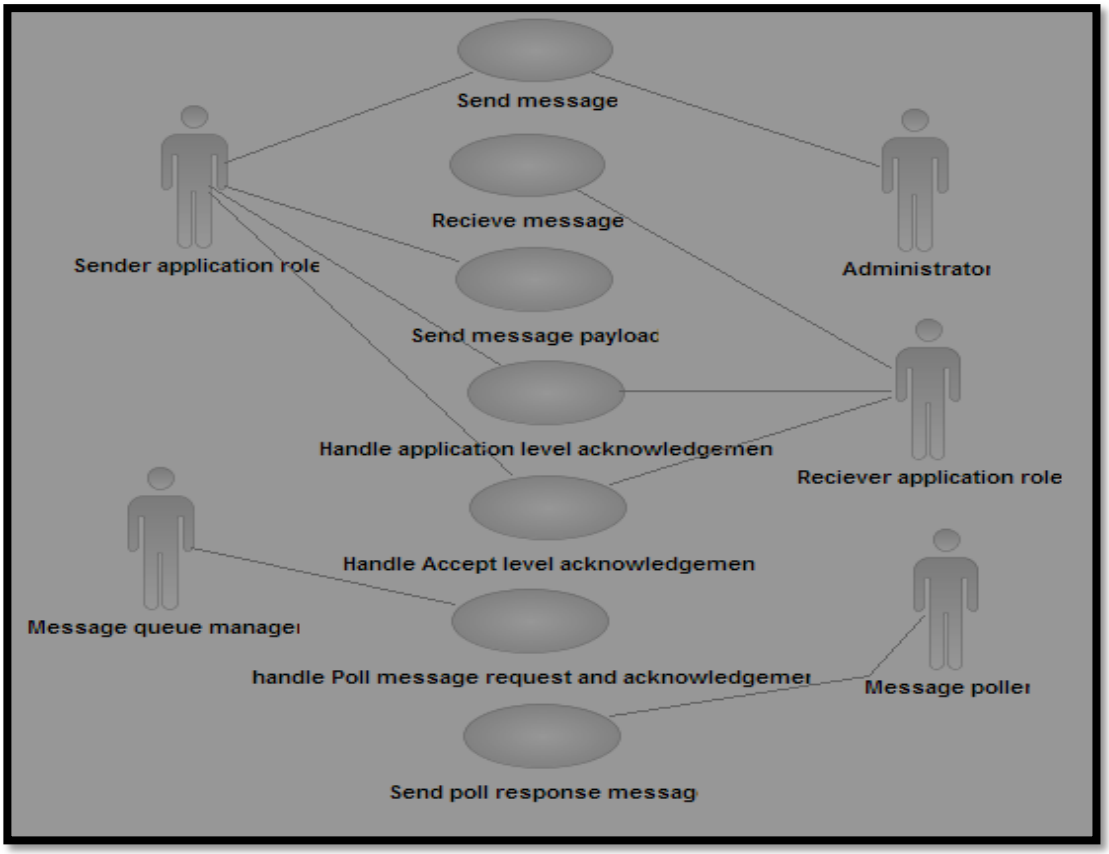


Figure 5: use case model I

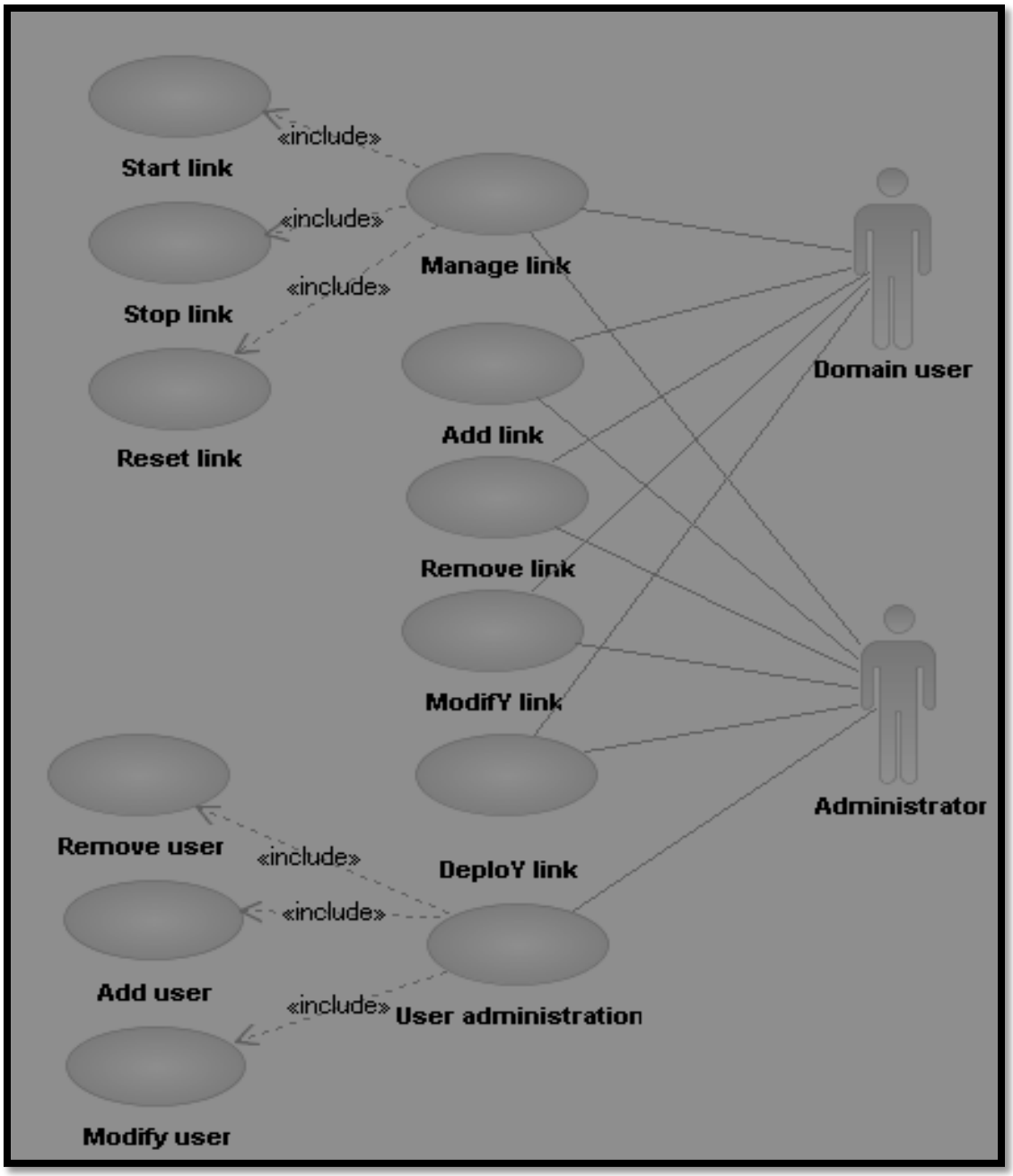


Figure 6: Use case model II

## 3.3 Detailed Description of Use Cases

Following tables contain the description of the use cases for HL7 Communication Environment.

Table 1: Send Message Use Case Description

| Use Case ID: UC_CELab01<br>**Use Case Name: Send Message** | |
|---|---|
| **Actors** | Application sender role, Administrator |
| **Purpose** | To transmit the HL7 message to intended receiver (using web service basic profile). |
| **Cross Reference** | |
| **Features** | For first time it will support web service basic profile for transmission, but it should be extendable to other transmission ways like MLLP and ebXML. |
| | |
| **Course of Events**<br>**1.** The Administrator open window of generated messages.<br>**2.** The system lists all the generated message with information message type, intended receiver, creation time and message description.<br>**3.** The Administrator selects the required message and the configured link for intended receiver for transmission (using web service) and selects the 'send message' command.<br>**a.** For messages that travel too frequently for any given link, the administrator intervention will be minimized by automating this process.<br>**4.** The system sends the message, shift the message to send message list.<br>**5.** The system displays the successful delivery of message. | |
| **Alternate Course of Event**<br>**5.a)** When acknowledgment is needed ( By specifying in message )<br>**i.** System listens continuously for a configured time of span to receive acknowledgment.<br>**ii.** When receiver sends acknowledgment within specified time span.<br>**iii.** The sender checks and validates the acknowledgment and perform step 5 in main scenario.<br>**iv.** When receiver fails to send the acknowledgment in specified time span.<br>**v.** The sender will try to resend the message. (The resending will be on the basis of configured number of times). | |

Table 2: Receive Message Use Case Description

| Use Case ID: UC_CELab02<br>**Use Case Name: Receive Message** | |
|---|---|
| **Actors** | Application Receiver Role |
| **Purpose** | To ensure the successful HL7 message receiving using various transmission specification. |
| **Cross Reference** | |
| **Features** | Message will be received on SOAP based link which will be compliant to HL7 web service basic profile. |
| **Course of Events**<br>**1.** Application receiver role receives message from network.<br>**2.** Receiver validates the message, save it to message queue and checks for any acknowledgements required. | |
| **Alternate Course of Event**<br>**2a)** For acknowledgements required:<br>**a.** Accept level acknowledgement, in which an acknowledgement will be sent to sender after initial message validation<br>**b.** Application level acknowledgement which can be deferred or immediate.<br>**c.** In case of immediate mode, receiver application validates received message thoroughly and send application level acknowledgement to sender application.<br>**d.** If deferred application acknowledgement is required, receiving application will respond with accept level acknowledgement after initial message validation, and application level acknowledgement later. | |

Table 3: Handle Accept level acknowledgement use case description

| Use Case ID: UC_CELab03<br>Use Case Name: Handle Accept level acknowledgement | |
|---|---|
| **Actors** | Sender application role, Receiver application role |
| **Purpose** | To handle accept level acknowledgement sent by Sender application role to receiving application role. |
| **Cross Reference** | Send message, receive message |
| **Features** | Messages are sent using web services |
| | |

| **Course of Events**<br><br>**1.** Sender Application role sends message with accept level acknowledgement request<br>**2.** Receiver Application role responds with accept level acknowledgement after initial validation. |
|---|
| **Alternate Course of Event** |

Table 4: Handle Application Level Acknowledgement use case description

| Use Case ID: UC_CELab04<br>Use Case Name: Handle Application level acknowledgement | |
|---|---|
| **Actors** | Sender application role, Receiver application role |
| **Purpose** | To handle application level acknowledgement from the sender application role by the receiver application role. |
| **Cross Reference** | |
| **Features** | |
| | |
| **Course of Events**<br>**1.** Application sender role sends the message payload with application level acknowledgement request and starts listening for acknowledgement.<br>**2.** Receiver responds accordingly on receiving the message. | |
| **Alternate Course of Event**<br>**2a) If** acknowledgements are required:<br>1. If the mode of application level acknowledgment is immediate, receiver application immediately responds with application level acknowledgement.<br>2. If the application level acknowledgement requested is deferred, then receiver first responds with accept level acknowledgement after initial validation.<br>3. At second stage a deferred application level acknowledgement is sent<br>4. The sender application role responds with an accept application level acknowledgement | |

Table 5: Handle poll message use case description

| Use Case ID: UC_CELab05<br>Use Case Name: Handle poll message | |
|---|---|
| **Actors** | Message poll manager, Message queue manager |
| **Purpose** | To handle the message poller request for un-queuing a message and acknowledgement for message. |
| **Cross Reference** | Send poll response message |
| **Features** | |
| | |
| **Course of Events**<br>**1.** Message poller will generate a request for un-queue a message from message queue manager.<br>**2.** Message queue manager will respond with required message.<br>**3.** If there is some error in the request, message queue will respond with error message.<br>**4.** The acknowledgement for the message will be sent by message poller manager.<br>**5.** If message poller needs more messages from message queue manager, acknowledgement for the message will include request for next message poll.<br>**6.** Above step will continue until all the messages are obtained by message poller. | |
| **Alternate Course of Event**<br>**2.a)**If queue manager does not respond:<br>**1.** If the message queue manager does not respond for a fixed time interval, request will be generated again.<br>**2.** Number of message regeneration is predefined.<br>**4.a)**If no acknowledgement received<br>a) If message queue manage does not get the acknowledgement for a time interval it will retransmit the message. | |

Table 6: Modify Link use case description

| Use Case ID: UC_CELab08<br>Use Case Name: Modify Link | |
|---|---|
| **Actors** | Administrator, Domain user |
| **Purpose** | Modify the configuration of the link. It involves behavioral configuration, destination configuration and source configuration of link. |
| **Cross Reference** | Source management, Destination management |

| | |
|---|---|
| **Features** | The system will provide the functionality of modifying the links easily, validation will be done by user or system after configuration change |

**Course of Events**

1. Administrator/user browses through the available Link(s) to select the desired Link.
2. Administrator/user selects the desired link(s).
3. System displays the available actions for selected Link.
4. Administrator selects appropriate configuration options as per requirements, configurations available are
    1. Source,
    2. Destination(s),
    3. Behavioral (behavior in different scenarios) and
5. System displays the current configurations for the selected link
6. Administrator/user alters them as required and select save setting option before closing the window
7. System validates the changes made by the Administrator/user 'on selecting 'save settings' option.
8. Changes are saved to the configuration file by the system.

**Alternate Course of Event**

**7.a) Validation of configuration changes**

1. If the changes are validated successfully, system saves the settings.
2. If changes made are not valid, an error message is displayed with information about validation error(s).
3. Administrator/user will correct these errors and save changes again
4. On validation, systems save changes to configuration file of the link.

**Post Condition**

Changes made by the administrator/user take effect on redeployment of Link.

Table 7: Deploy Link use case Description

| Use Case ID: UC_CELab09<br>Use Case Name: Deploy Link | |
|---|---|
| **Actors** | Administrator, Domain user |

| | |
|---|---|
| **Purpose** | The need of this use case is to resolve the availability and other issues occur during link starting. |
| **Cross Reference** | Add Link, Modify Link, Import Link settings |
| **Features** | |
| **Pre-Condition** | Administrator/user should follow the steps to add a link |
| **Course of Events**<br>**1.** Administrator/user browse through available 'added link(s)' and select the required link<br>**2.** System displays the available actions for the link(s) selected<br>**3.** Administrator/user selects 'deploy' to deploy link(s)<br>**4.** System deploys link(s) selected by the administrator/user and add those link(s) to 'deployed links' list | |
| **Alternate Course of Event** | |
| **Post-Condition**<br>Deployed link(s) are available for start | |

Table 8: Export Link Settings use case description

| Use Case ID: UC_CELab10<br>Use Case Name: Export Link settings | |
|---|---|
| **Actors** | Administrator, Domain user |
| **Purpose** | To export the link settings to deploy the link with same settings on any other machine. It will automate link creation and save time of adding the link. |
| **Cross Reference** | |
| **Features** | |
| | |
| **Course of Events**<br>**1.** User/Administrator browses through the available links and select desired link(s) from the list<br>**2.** system displays the available actions available for these link(s)<br>**3.** Administrator/user selects 'export settings' from options available to export its settings.<br>**4.** System prompts for the location for saving this 'export file'. Export file contains the configuration of link, used to add a new link on another machine with same settings.<br>**5.** Administrator/user will select the desired location for saving the 'export file'. | |

| |
|---|
| **6.** System creates an ‘export file’ for selected link on desired location. |
| **Alternate Course of Event** |
| **Special Requirements**<br>One export file will be created per link to keep the settings simple. |

Table 9: Import Link Settings use case description

| Use Case ID: UC_CELab11<br>Use Case Name: Import Link Settings | |
|---|---|
| **Actors** | Administrator, Domain user |
| **Purpose** | To deploy new link(s) with some saved settings. This will reduce hassle if one is to start a link with similar settings on another machine. |
| **Cross Reference** | |
| **Features** | |
| | |
| **Course of Events**<br>**1.** Administrator/user selects ‘import link settings’ option from link menu<br>**2.** System prompts for the location of ‘export file’. Export file contains the configuration of link, used to add a new link on another machine with same configurations<br>**3.** Administrator/user provides the location of ‘export file’.<br>**4.** System imports the ‘export file’ and displays the configuration of the Imported link.<br>**5.** Administrator/user selects ‘save link’ option to add this link to ‘added links’ list.<br>**6.** System validates the configurations of link.<br>**7.** Link is added to ‘added channels’ list after validation. | |
| **Alternate Course of Event**<br>**6a)If the link configuration is not valid**<br>**1.** Message is displayed with validation error information.<br>**2.** Administrator/user corrects these errors and save changes.<br>**3.** System adds this channel to ‘added channels’ list. | |
| **Post-Condition**<br>Link is available for deployment | |

Table 10: User Management use case description

| Use Case ID: UC_CELab12<br>Use Case Name: User Management | |
|---|---|
| **Actors** | Administrator |
| **Purpose** | User management, it involves addition of a new user account, removal of an account or modification of a user account. |
| **Cross Reference** | |
| **Features** | |
| | |
| **Course of Events**<br>**1.** Administrator authenticates for user management<br>**2.** System displays available users and their information in user tab of option pane<br>**3.** Administrator selects desired user for appropriate action<br>**4.** System displays available actions for user(s) selected. The available actions for administrator to choose from are:<br>**1.** Add new user, which includes adding a new user to list and define user rights and authentication information<br>**2.** Remove selected user(s), which includes removal of user account information of selected user(s).On selecting remove user, administrator will have an option to save the user settings in some folder for future use<br>**3.** Modify user, which includes redefining user rights, changing user authentication information e.g. user name, password<br>**5.** After finished with user management, administrator will commit the changes he made<br>**6.** System will store these changes in user database. | |
| **Alternate Course of Event** | |

## 3.4 Functional Requirements

### 3.4.1 Message Queuing

Managing the queue of messages at receiving end for the purpose to avoid the lost of messages and to use it for further processing. It also helps to handle the simultaneous arrival

of messages from different senders. More importantly, it is useful for asynchronous communication.

### 3.4.2 Message Transmission

One of the most important functionality of this system is to successfully transfer the message from sender to receiver side. The mechanism used for this process is web service. The sender of the system deploys the web service on particular web server while the receiver then utilizes for its purpose.

## 3.5 Nonfunctional Requirements

### 3.5.1 Reliability

Robustness and reliability are two most important features that will be ensured in our proposed system. Distributed architecture will be implemented to overcome the single point of failure.

### 3.5.2 Performance

For messages that travel too frequently, process of transmission will be automated to enhance the process performance and minimize errors.

### 3.5.3 Safety Requirements

Safety requirements are important for any system. Our proposed system will be able to handle all the safety considerations. Fail safe property will be provided in our solution such that in case damage or possible loss. So, it can recover itself from that state.

### 3.5.4 Security Requirements

Proper user identity authentication mechanism will be provided.

### 3.5.5 Software Quality Attributes

#### 3.5.5.1 Manageability

System should be easily manageable with available options which are compulsory.

#### 3.5.5.2 Maintainability

The designed system must be maintainable for the new components made by open source community or independent developers.

#### 3.5.5.3 User Friendly

The system GUI must be user friendly, easy to understand and handle. The working personnel must feel comforts after training.

#### 3.5.5.4 Easy Integrable

Our system works as a middleware solution for laboratory information systems or hospital information systems have already been developed. So, it requires developing with proper design strategies such that it can easily be integrable with existing system.

#### 3.5.5.5 Accuracy

It is a responsibility of the system to provide the accurate information according to the requirement.

#### 3.5.5.6 Portability

The developed software will be deployed on Microsoft Windows platform.

#### 3.5.5.7 Adaptability

The designed should be able to adopt all the changes made by open source community or independent developers.

## 3.6 Software and Hardware Requirements

- Java Net Beans 6.1 or higher version
- Rational Rose
- GLASSFISH/Tomcat Server
- Windows Operating System

# *CHAPTER 4*

# SYSTEM DESIGN

After analysis and requirement gathering software development enters into design phase. Purpose of the designing phase is to provide a framework or foundation to build the software. It is a universal truth that "good design will always result in a good software design". The design phase of the software development focuses on implementation constraints put up by the user, through requirements gathering phase and devise a design for the system.

## 4.1 Object oriented Design

In object oriented designing (OOD) approach everything is considered as an object. OOD represents interactions among objects having specific attributes. Object design comes in after the requirements gathering and identifying domain model. Assigning responsibilities to objects is the main task here.

### 4.1.1 Responsibility

Responsibility is the actual role that an identified object is going to play in the completion of the system. It is an obligation of the object in terms of behavior. There are two types of responsibilities in a software system:

i. Active (Initiating objects, doing some calculation etc.)
ii. Passive (Object encapsulation etc)

### 4.1.2 Methods

In a software system the responsibilities associated with identified objects are practically implemented using methods or object-methods collaboration.

Object oriented design uses several model diagrams like sequence and class diagrams to provide the demonstration on relationships among objects. There are several object oriented design patterns build by experts to facilitate designers with their experience.

## 4.2 Design Patterns

Design patterns are partial solution to a common problem, such as separating an interface from number of possible implementations or saving resources of the system by molding a piece of code in a specific way. A design pattern does not provide the complete solution but the necessary steps for implementation. It is composed a small number of classes that using delegation or inheritance, provide a modifiable and robust solution.

Some design patterns used in the system are given below.

### 4.2.1 Factory design pattern

| | |
|---|---|
| **Name** | Factory design pattern |
| **Problem description** | Shielding client from different platforms that provide different solutions for the same set of concepts |
| **Solution** | The client need a product, but instead of creating it directly using the new operator it asks for a new product to the factory providing some information about the type of object it needs. “The factory instantiates a new concrete product and then return to the client the created product casted as abstract product” [8]. The client uses the products only as abstract products without being aware about their concrete implementation. Figure 7 shows the Factory design pattern implementation in project. |

| | |
|---|---|
| **Result** | Client is unaware of the concrete product classes<br>Substituting families at runtime is possible<br>Adding new products is a bit tricky as realization for each factory must be created |
| **Uses in the system** | Connection factory is used to get a new connection in JMS implementation |

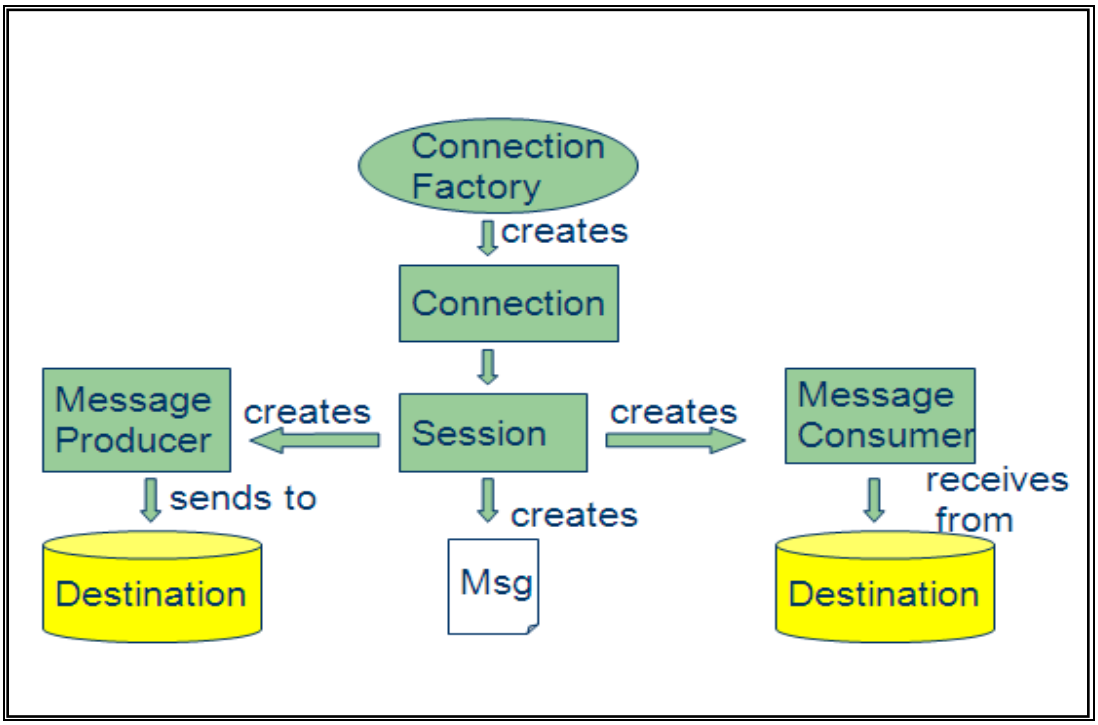


Figure 7: Factory design pattern in JMS [6]

### 4.2.2 Bridge design pattern

| | |
|---|---|
| **Name** | Bridge design pattern |
| **Problem description** | Separation of an interface from implementation so that implementation can be substituted. |

| | |
|---|---|
| **Solution** | Implement an abstraction, this implementation should not depend upon any concrete implementers of the interface. Also extending the interface should not have any effect on the implementers. |
| **Result** | Client is shielded from abstract and concrete implementations<br><br>Independence of implementations and interfaces |
| **Uses in the system** | With interface of receiving a message via JMS, database implementation can also be achieved. |

### 4.2.3 Singleton Design pattern

| | |
|---|---|
| **Name** | Singleton design pattern |
| **Problem description** | It is important for the system to have only one instance of the class at a given time. |
| **Solution** | Create a class with private constructor and a static method returning the instance of the class. There should be a check on the instances of the class, should not be more than one. In this way a class will have single instance at a given time for a system. The same instance is accessible elsewhere in the system. |
| **Result** | Single instance of the class is instantiated<br><br>Gives access to all the resources using same class |
| **Uses in the system** | There are many components that require singleton pattern in the system; most of them are window managers. Listener to the messages also is implemented singleton. |

### 4.2.4 GRASP Pattern

General Responsibility Assignment Software Patterns (GRASP) is suite of pattern that actually describes the fundamental principles of how responsibilities are assigned to objects. Two of the important elements from the suite are:

- High Cohesion
- Low Coupling

#### 4.2.4.1 High Cohesion

Cohesion is a measure of responsibility strength assigned to a class or element. If the strength of responsibility is highly related to the domain of definition, cohesion is considered to be high. On the other hand a class with highly unrelated and tremendous work is considered to be having low cohesion.

High cohesion means too many function calls, as all the classes are divided as per responsibility. On the other hand if cohesion is too low it is difficult to differentiate the responsibility of the class. We should always go for a moderate cohesion in which a class might have responsibilities from two logically related functional areas.

#### 4.2.4.2 Low Coupling

Coupling refers to the measure of interconnection or dependency among modules of the software. According to Constantine's law coupling should be low for good software.

Types of coupling are:

Data coupling:

In this type of low coupling output from one module is input to another module

Control coupling:

It is a moderate level coupling where control flag or control variable is passed between subordinate and super-ordinate to control the sequencing of other module. It is better to divide tasks of modules into further primitive operations.

External coupling:

High level of coupling is acquired when modules of the program are tied to an external environment e.g. coupling of modules with external devices, file formats or some protocols. It is essential but we should try to minimize the number of modules to interact with external environment.

Common coupling:

It occurs due to global data items. For example three modules of a program a, b and c are to work with a global variable x. firstly ‘a’ accesses the module initialize it, then ‘b’ updates this module according to its requirement. Suppose an error occurs and ‘b’ updates it incorrectly, when ‘c’ would read this variable, on processing it would probably cause program to abort. Diagnosing problems in this case are difficult as main cause of failure seems to be ‘c’ rather than ‘b’.

Content coupling:

It is the highest degree of coupling. It occurs when one module makes use of control information or data that lies in the boundary of another module. This type of coupling should be avoided.

Compiler coupling:

It is a variant of external coupling in which source code is tied to specific characteristics of the compiler.

Operating-system coupling:

It is a type of external coupling. It occurs when software modules are tied with operating system. It can bring drastic changes in operation of software when changes in OS occur.

## 4.3 Interaction Diagrams

Interaction diagrams are used to model the behavior of use cases; by describing the way groups of objects interact to complete the task. The two kinds of interaction diagrams are:

- **Sequence** and
- **Collaboration** diagrams.

### 4.3.1 Sequence Diagrams

Interaction diagrams are used when we want to model the behavior of several objects in a use case. They demonstrate how the objects collaborate for the behavior. Interaction diagrams do not give a in depth representation of the behavior.

### 4.3.2 Collaboration Diagrams

Sequence diagrams, collaboration diagrams, or both diagrams can be used to demonstrate the interaction of objects in a use case. Sequence diagrams generally show the sequence of events that occur. Collaboration diagrams demonstrate how objects are statically connected.

## 4.4 HLCE Interaction Diagrams

Interaction diagrams of some important use cases are given below,

### 4.4.1 Send Message

This is a very important use case for HLCE, this use case captures the scenario of sending a message using LLP channel or Web Services channel. This use case also captures the use of JMS controller in the system, which is there for the sake of reliability and robustness.

#### 4.4.1.1 Sequence diagram

Sender application role, after entering the necessary information generates the HL7 V3 message. Message generator returns the HL7 V3 message. The send message window contains different branches or destinations in the combo box; user or sender selects the desired branch and press the send message button. Figure 8 shows the sequence diagram for the use case.

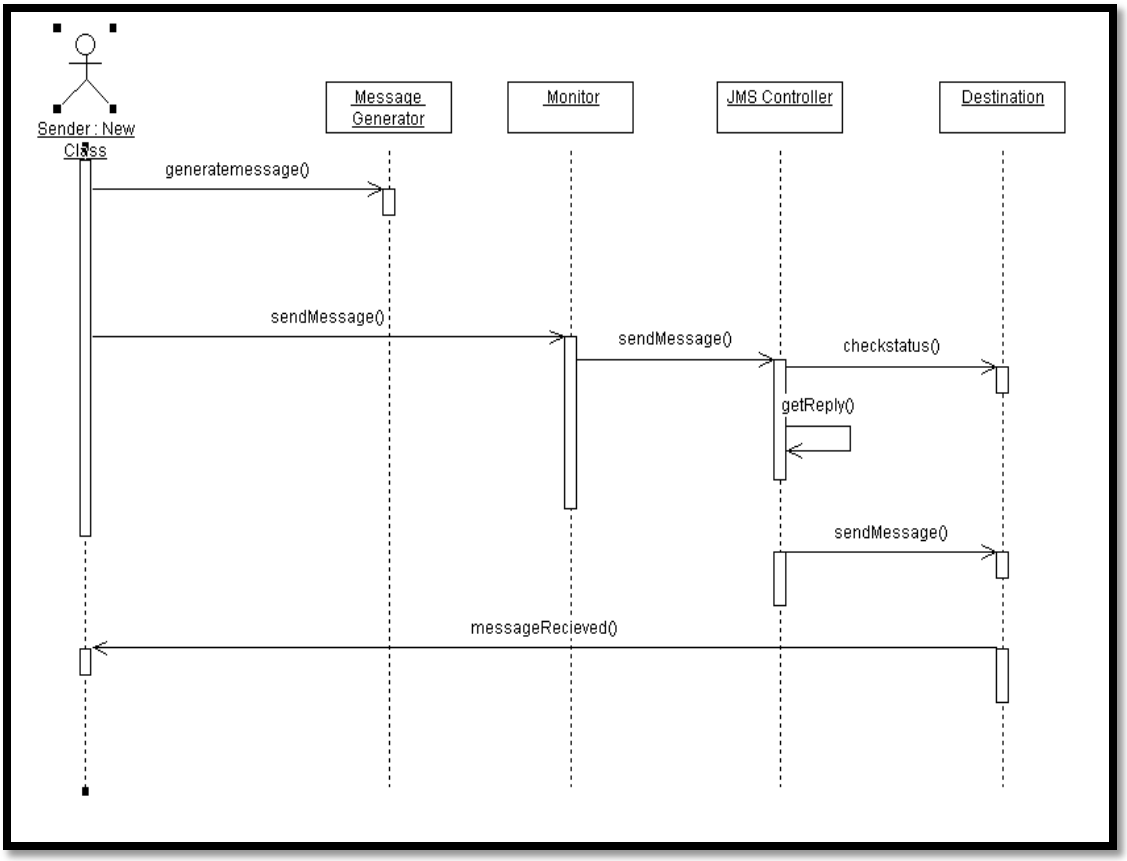


Figure 8: Send Message Sequence Diagram

The monitor will return the status of the destination whether it is listening or not. If destination is listening the monitor will by pas the JMS controller. In the other case if the destination is not listening the JMS controller is handed over with the message, which will keep it in the queue until the monitor signals it to send the specific message.

Monitor will continuously monitor the status of the destination. On successful delivery of the message the acknowledgement is sent to the sender and message is added to send message list.

#### 4.4.1.2 Collaboration Diagram

Given, figure 9, is the collaboration diagram for send message use case.

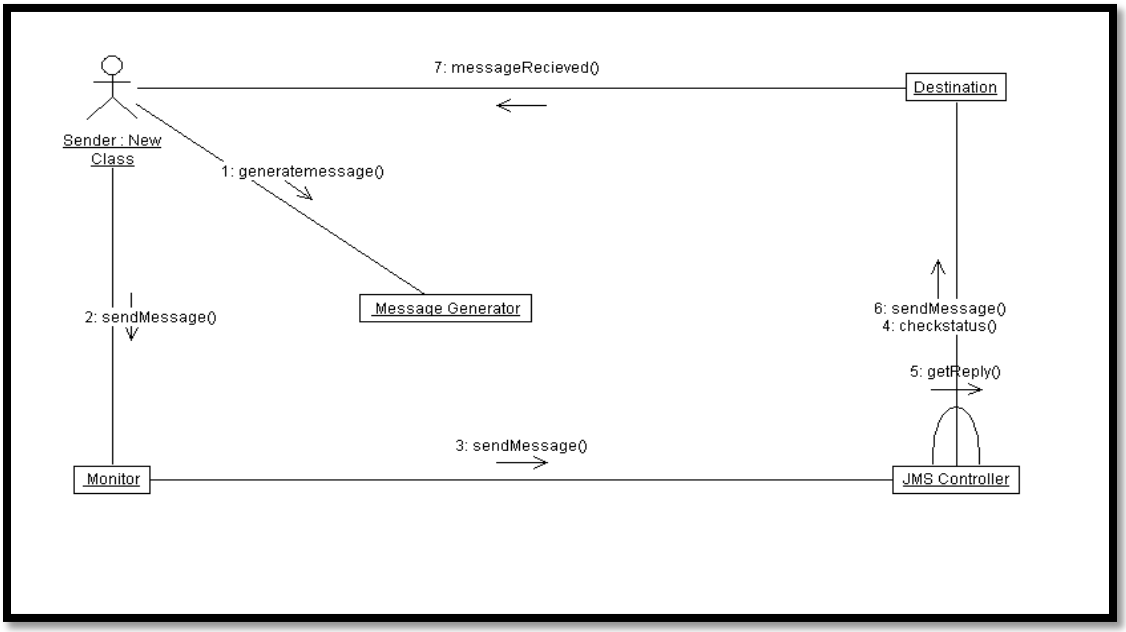


Figure 9: Send Message Collaboration Diagram

### 4.4.2 Receive Message

Receive message is another very important use case of the Communication environment. It captures the scenario of receiving a message over LLP or Web Services channel at the receiving end.

#### 4.4.2.1 Sequence diagram

Receiver application role starts listening to the channel by startlistening() method. Server manager, which is a threaded class, will initiate the JMS controller on receiving a message. JMS controller will check the status of the Receiver continuously, once the receiver is free, the message will be sent to receiver. On receiving the message the Receiver will acknowledge the message. Figure 10 shows the sequence diagram for “Receive Message” use case.

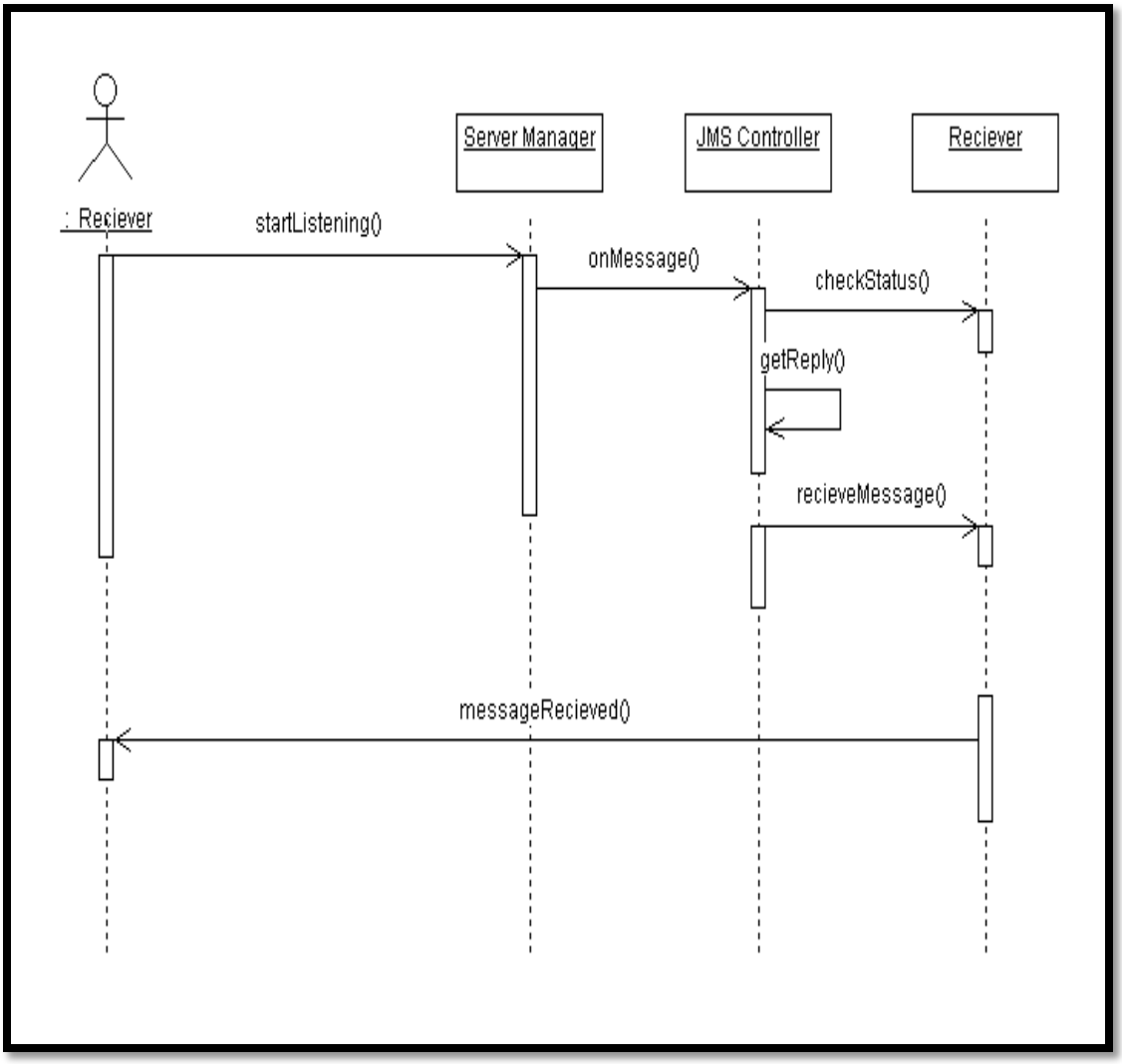


Figure 10: Receive Message Sequence Diagram

### 4.4.2.2 Collaboration Diagram

Figure 11 shows the collaboration diagram for the receive message use case.

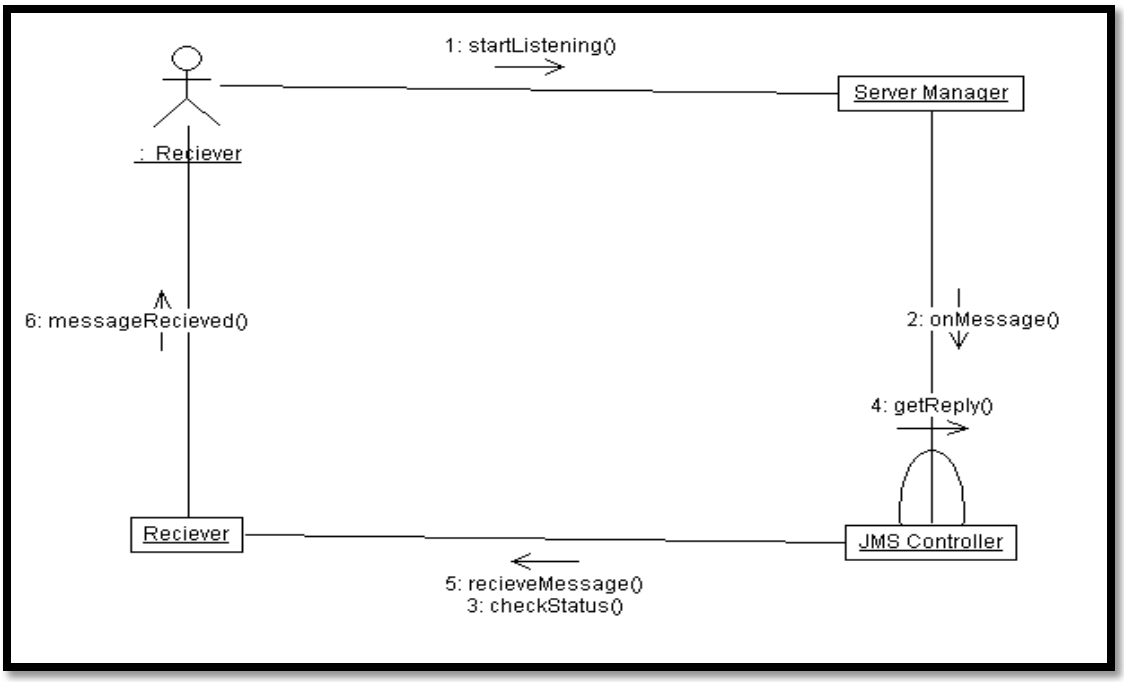


Figure 11: Receive Message Collaboration Diagram

## 4.5 Partial class diagram

Partial diagram of the solution is given below along with cardinalities. Sender and Monitor components are on the sender side while Server manager and Receiver are on the receiver side of the project. JMS Controller is a shared resource among the components on both the sides. Figure 12 shows the mentioned partial class diagrams.

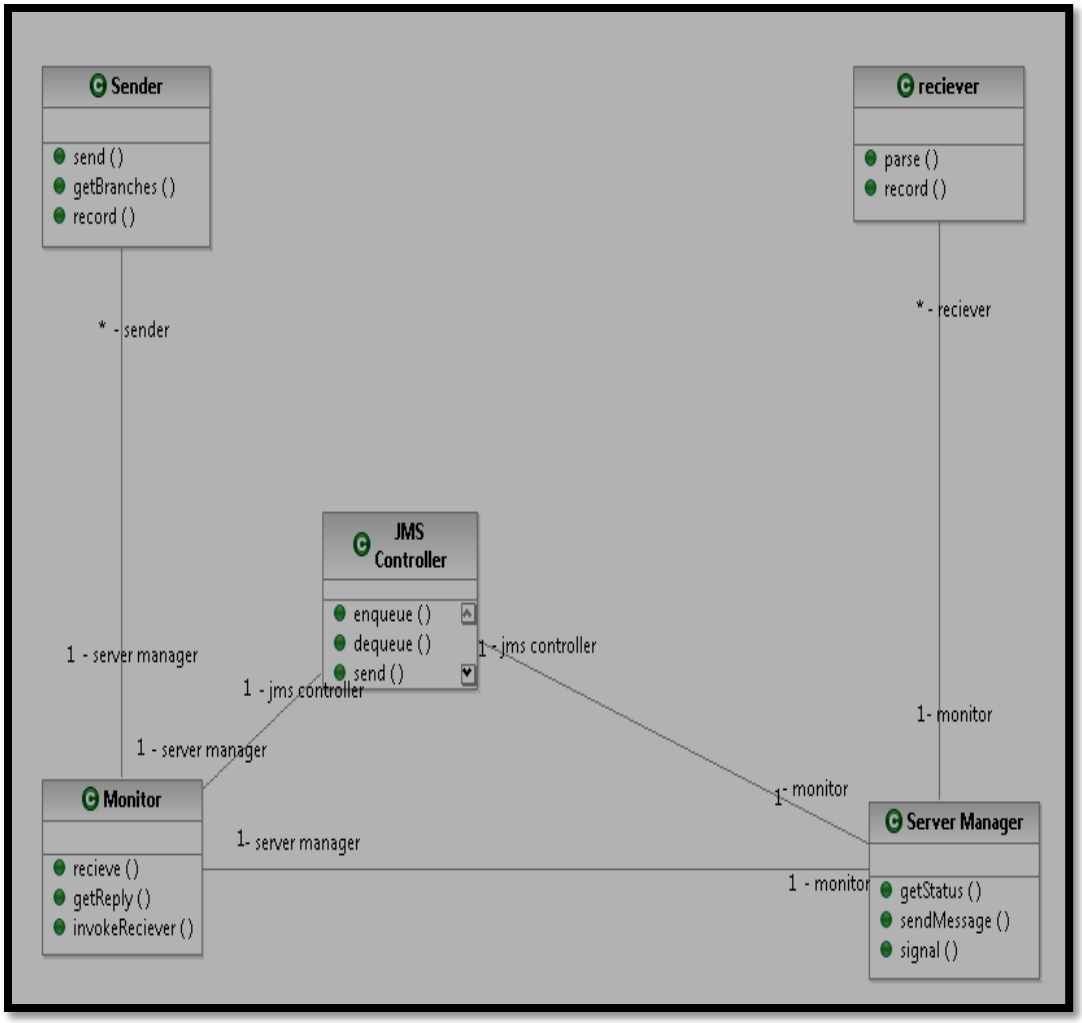


Figure 12: Partial class diagram

## 4.6 Complete class diagrams

### 4.6.1 Message generation

Complete class diagram showing the message generation process. Java SIG API is available implementation for HL7 RIM. Message generator takes the patient ID, test ID and message type as argument and generates the required HL7 message. Parser is responsible for parsing the information. Figure 13 shows the class diagram for HL7 V3 message generation component.

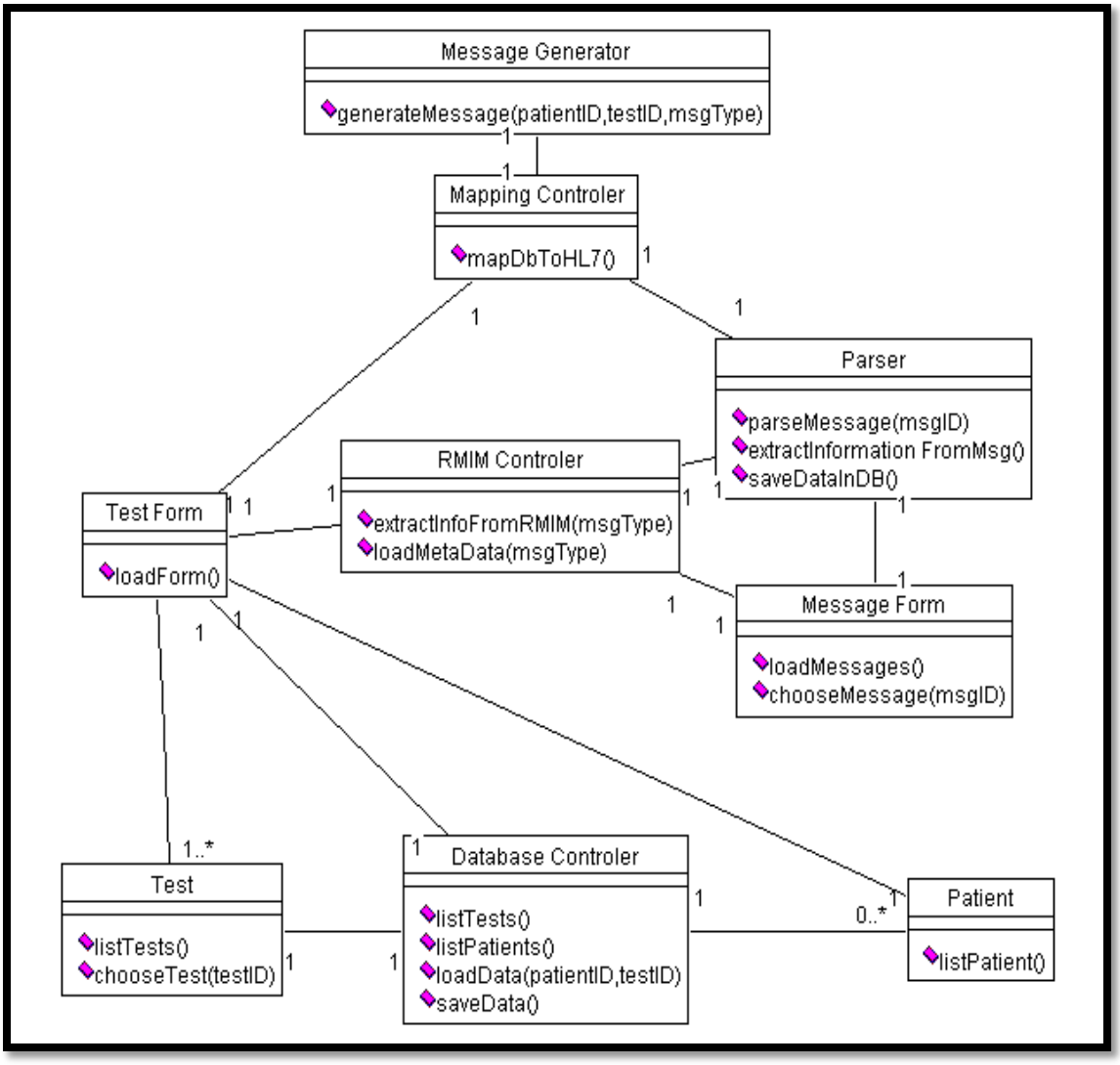


Figure 13: HL7 Message generation process

### 4.6.2 Message communication

Message communication part of the project consists of the sender and receiver module. On receiver side we have Server manager and Receiver while on sender side we have Sender, Monitor and Destination classes. Figure 14 shows the class diagram for the message communication component.

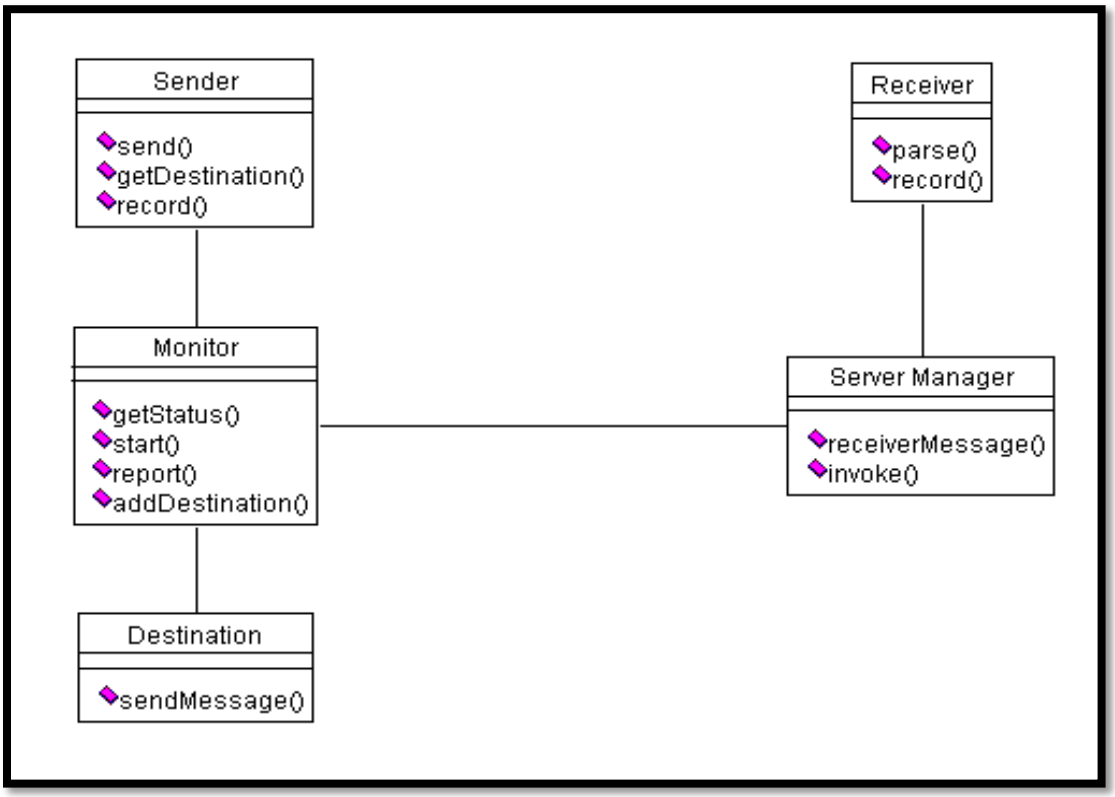


Figure 14: Message Communication

*Chapter 5*

# IMPLEMENTATION

Implementation phase needs some coding techniques from the developers. That is the reason, many implementations are possible for the same design, of which some are more efficient and some are less.

The efficiency of the system depends upon not only designing but also on the logic and programming language. Experience programmers develop efficient logic for given problem while inexperienced ones may follow poor logic. A well-written algorithm and logic can reduce the testing and maintenance effort. Second thing, which effect on system efficiency, is programming language.

## 5.1 HLCE Implementation

HL7 Communication environment is implemented using two protocols, first is Minimal Lower Layer Protocol (MLLP) and second is Web Services over HTTP. First of all there is a need of understanding of both of these protocols.

### 5.1.1 MLLP

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 minimalistic 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 [9].

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. Figure 15 shows the communication using MLLP.

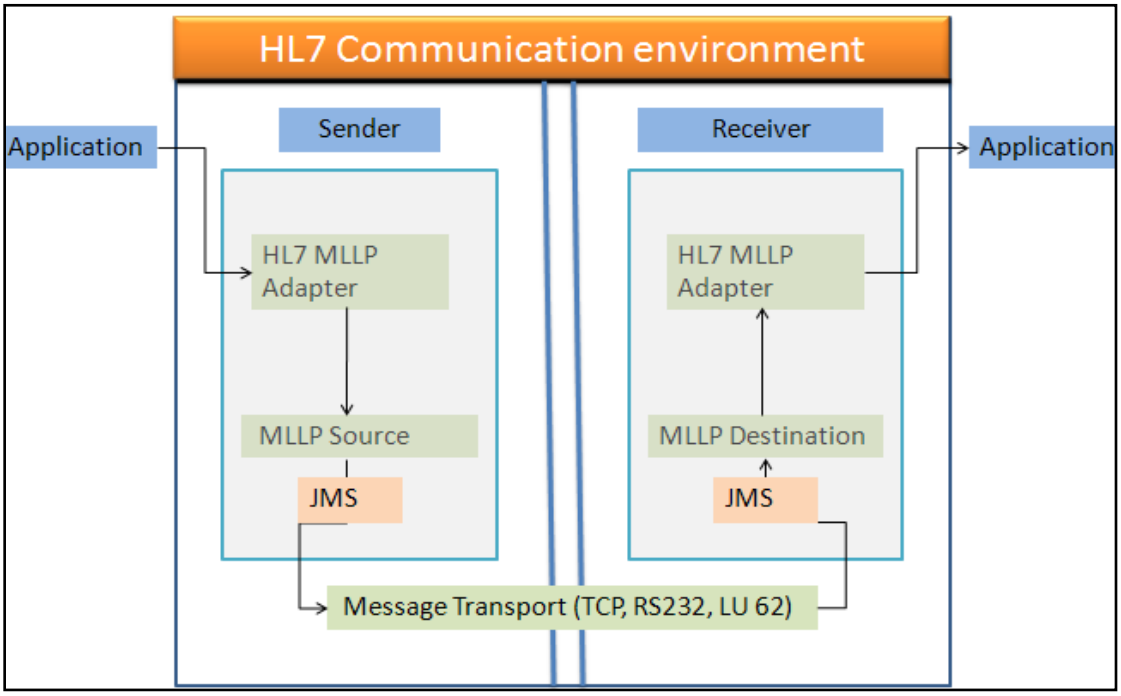


Figure 15: MLLP architecture

### 5.1.2 Web Services

Web services are revolutionizing the Internet by enabling applications to speak a common language: XML. Under the Web services paradigm, a single application can tap into the services of millions of applications scattered throughout the Internet. The potential of this is enormous. Web services allow cooperation, communication, and integration on a global scale [10].

A first-generation Web-services-enabled application contains or directly interfaces with a client that communicates with Web services, as depicted in Figure. This architecture enables the application to find and communicate with remote

systems, but does not implement data reliability, scalability, and re-usage of Web service client logic.

Main advantages of web services are platform independence and choice of language, which makes it acceptable all around the globe. Figure 16 shows the web-services architecture.

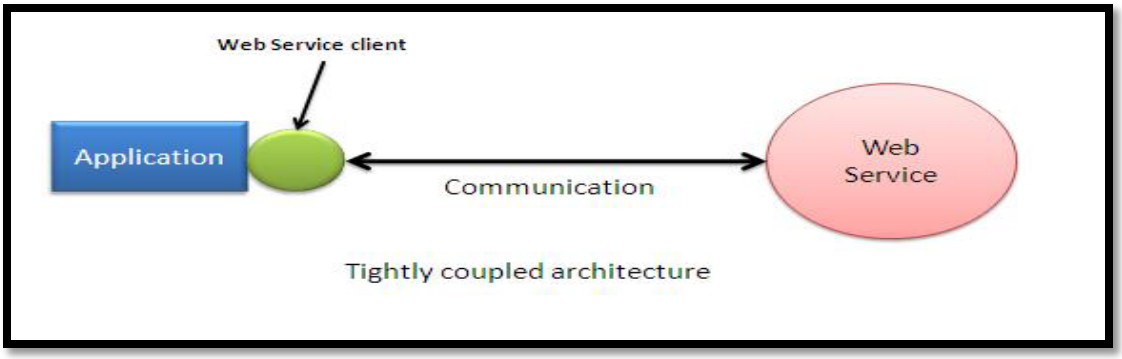


Figure 16: Web Services architecture

#### 5.1.2.1 Web Services with JMS

Web-services enable the implementation of remote functions and messaging but it does not provide a strong and robust communications infrastructure for handling information. An enterprise-class application that communicates with Web-services must make sure that the data can be handled appropriately.

Web-services must be combined with additional technology for robust enterprise messaging. One very strong candidate is the Java Message Service (JMS). JMS provides a reliable, scalable, and loosely coupled architecture for messaging. The combination of Web-services with JMS creates an architecture that can communicate across the Internet, reliably handle data, and integrate with backend systems.

The addition of JMS creates a second generation for architecting Web-services systems, as shown in Figure. JMS decouples the application from the task of Web-services messaging. Applications communicate directly or through an adapter to the JMS server. Figure 17 shows the conceptual model after introduction of JMS in WS architecture.

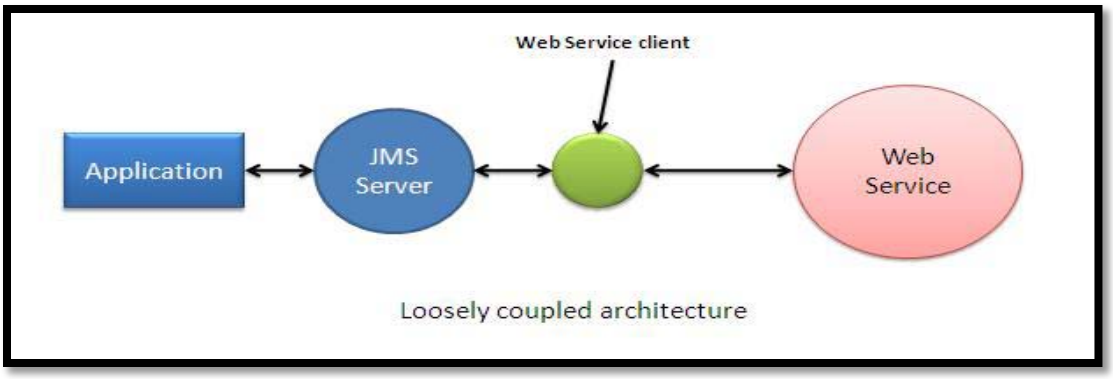


Figure 17: Web Services with JMS

In this new architecture, hybrid JMS and Web-services clients handle the bulk of the messaging duties. Information is passed through the JMS server, which natively handles issues like failover, load balancing, and guaranteed message delivery.

By separating the Web-services client from the application, following advantages are achieved:

- Several applications can readily reuse a single Web-services client.
- It also makes it a simpler process to upgrade the Web service as inevitable software changes occur.
- An application that becomes busy will have its Web-services data by design queued in the JMS server until it is able to process the messages.

In a tightly coupled architecture, the application's Web-services piece would have to wait until the application is ready to begin processing data.

## 5.2 Conceptual Architecture

Keeping in view all the above discussion, the conceptual architecture of the solution is as shown in the diagram below. The generator will generate the message and hand it over to the JMS server, which will pass it over the HTTP to the web service using web service client.

On the receiving end the web service will invoke the parser.

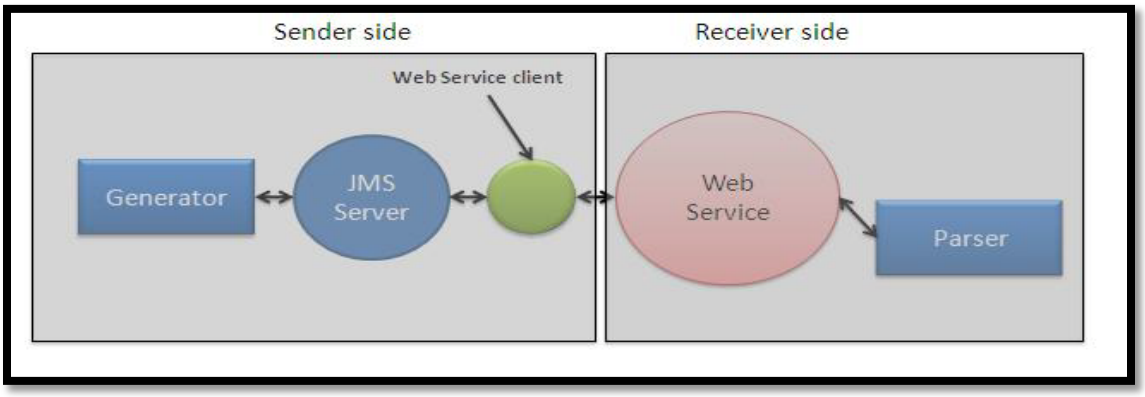


Figure 18: Conceptual Model

Above architecture, as shown in Figure 18, is robust with the inclusion of JMS server inside the architecture. With every destination listed in the communication environment, a separate message queue will be used and if the destination is listening the JMS server will be by-passed.

## 5.3 Architecture using MLLP

The solution architecture using MLLP is given below, the external application will use the MLLP adapter to convert the information in required HL7 V3 format.

MLLP adapter includes Parser and generator component, Parser parses the information coming from source while the generator generates the HL7 V3 message from concerned information.

Messaging infrastructure includes sender and receiver component of HLCE. Responsibility of the sender is to send message over the TCP, and division of message in chunks. While receiver just gather the chunks and recompile the formation to get the original message.

JMS is there in the infrastructure for the sake of reliability, ensures message delivery to the destination. Figure 19 shows the communication using MLLP protocol.

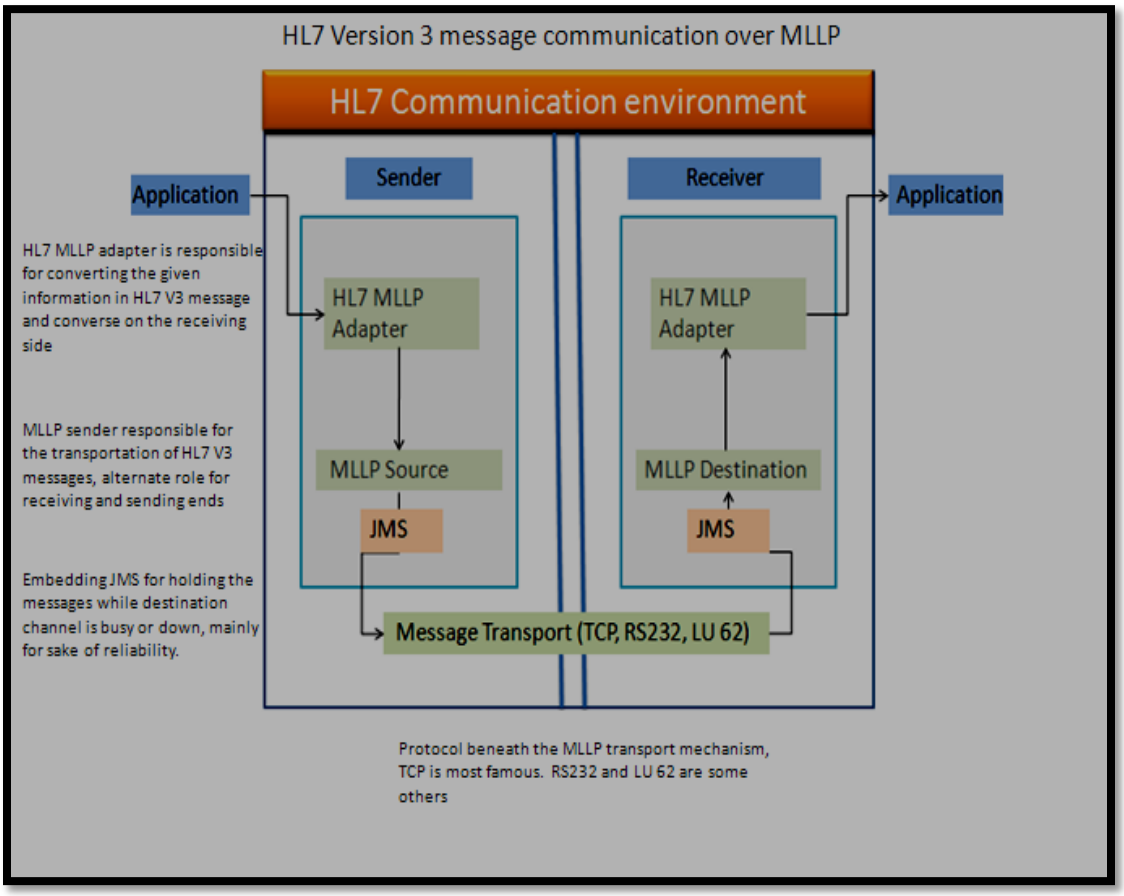


Figure 19: Architecture using MLLP

## 5.4 Architecture using Web Services

External application uses HL7 Web Service adapter to convert information to HL7 V3 message. Adapter is composed of parser and generator. On sending side the

generator component is used to generate a HL7 V3 message and on the receiving side of the environment parser component is used to parse the information back to original.

Web service Source is used to convert the message to a soap message and send it over HTTP. While on the receiving side the receiver component of the application receives the SOAP message and converts it into the HL7 V3 message.

JMS is embedded for the sake of reliability. It will make sure the message delivery to the destination. Figure 20 shows the communication using WS protocol.

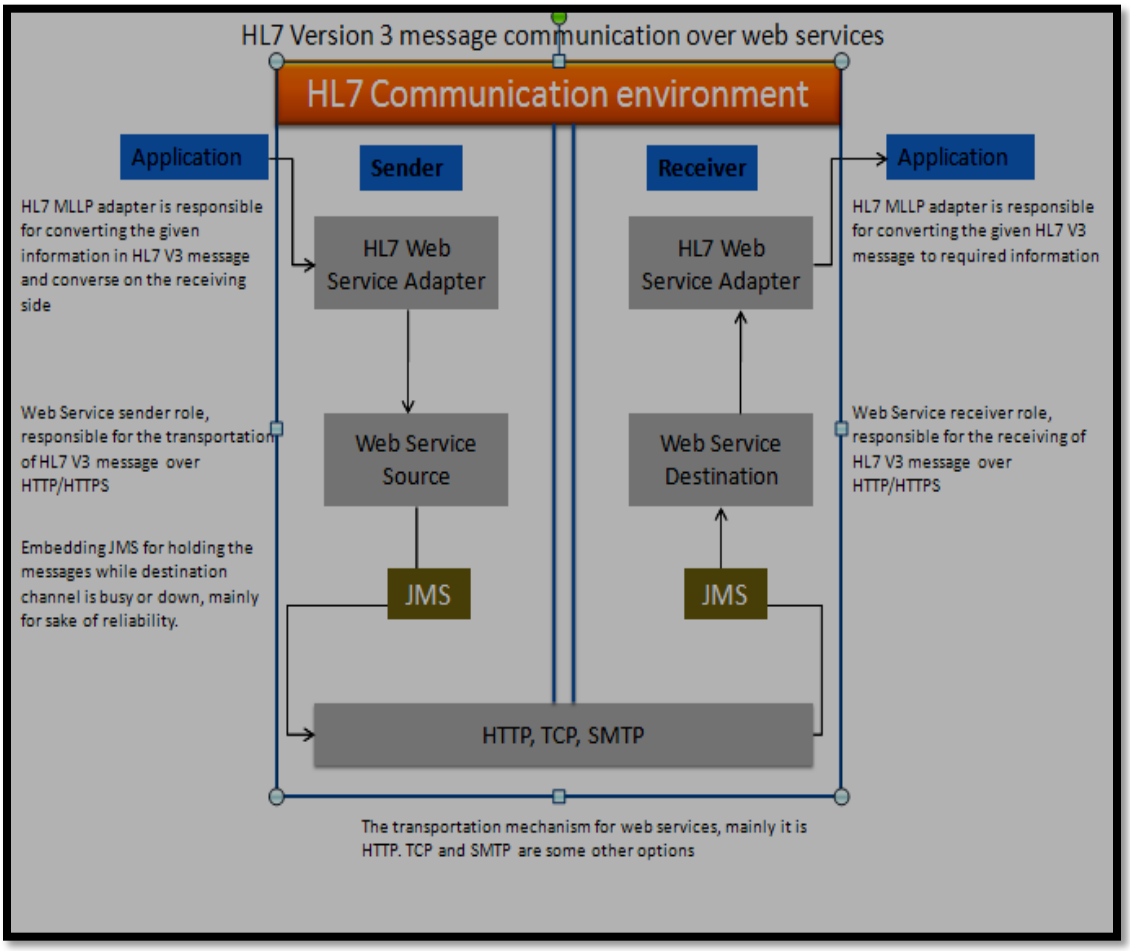


Figure 20: Architecture using Web Services

## 5.5 Mockups of the Implementation

Following are some of the mockups of the solution.

### 5.5.1 Communication Environment Window

The communication environment window displays the status of the links that are added to the communication environment. Again singleton pattern is used here as well to save resources of the system and make this window accessible to all of the system. Figure 21 shows the main window of the communication Environment.

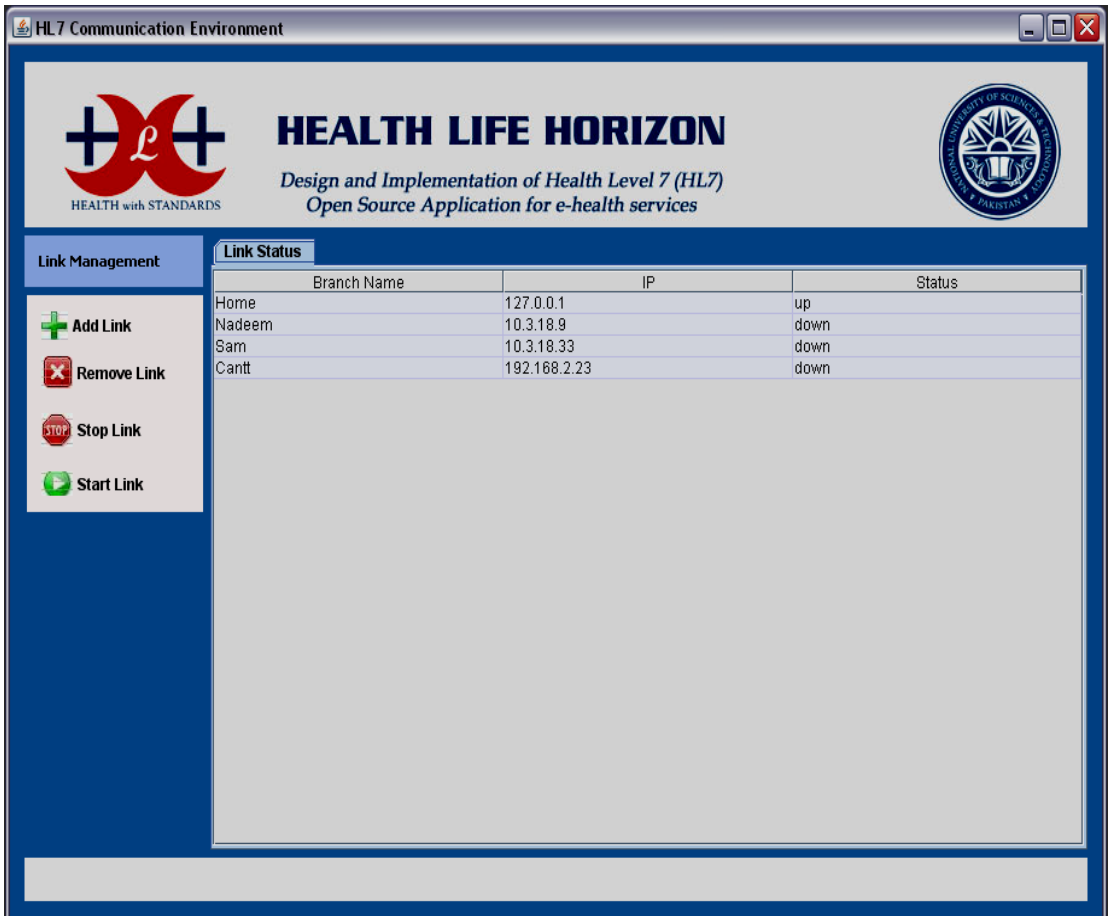


Figure 21: Communication Environment window

The Branch or destination is displayed as UP or listening if the listener component is running on the system, else if the link is broken or the destination is not listening for the messages the link is shown as down.

### 5.5.2 Placer order window

Placer order window displays the data of patients to choose from after selection of desired patient and tests the next button is pressed, which will generate the HL7 V3 message for the patient and tests.

Multiple selections in the table are enabled to facilitate the sender by sending multiple tests on one occasion.

Singleton pattern is used here to reduce the complexity of handling and saving system resources by allowing one instance of the window per application. Figure 22 shows the placer order window for selection of test orders.

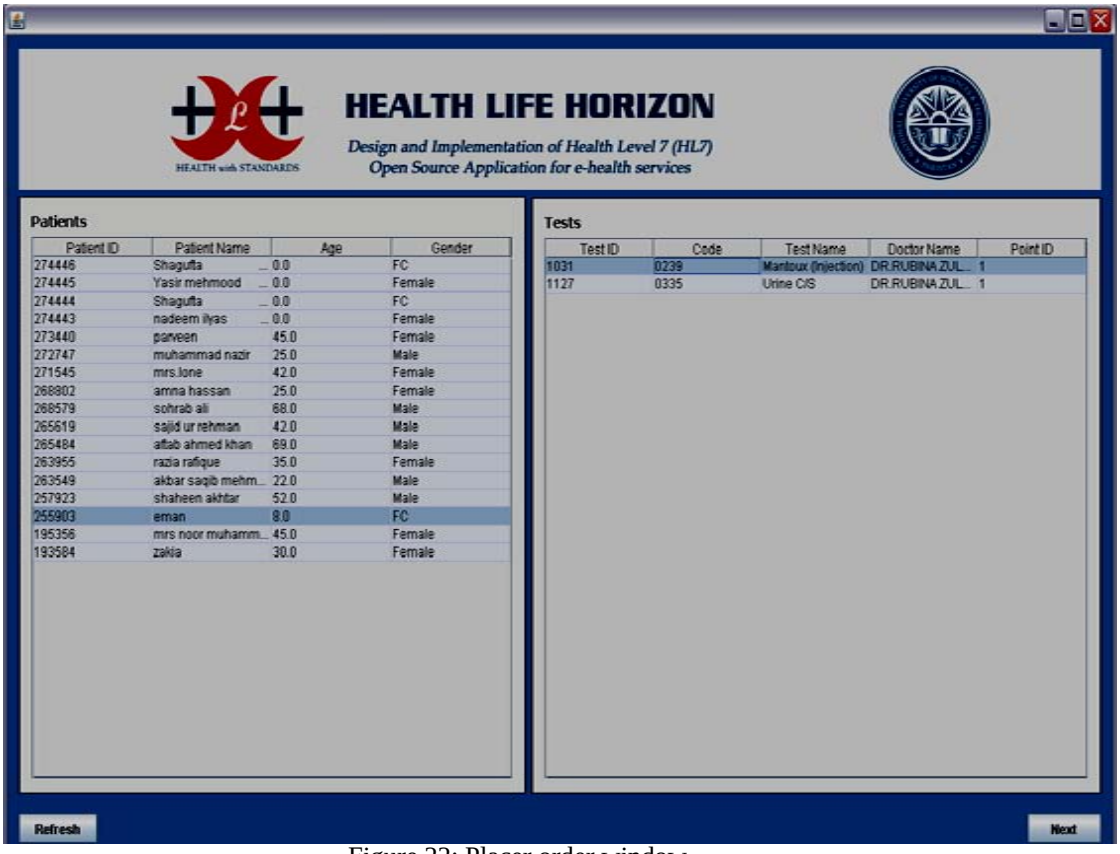

Figure 22: Placer order window

### 5.5.3 Send Message Window

Send message window will appear once generate message is clicked. It displays the HL7 V3 generated message. The user selects the desired destination and clicks the send message button. The message is sent to the destination using MLLP.

Singleton pattern is used here to allow one instance per application. This will save resources and reduce complexity of the application to handle on the runtime. Sample HL7 V3 message is given in the appendix. Figure 23 shows the HL7 V3 message generated from the test order selected by the user.

Figure 23: Send message window

*Chapter* 6

# CONCLUSION AND FUTURE WORK

In a third world country like Pakistan importance of health care system interoperability even increases. Interoperability brings in efficiency and cost reduction in healthcare. Main advantage of interoperability is reduction of medical errors by enabling exchange of clinical experiences among doctors and medical staff.

The aim of the HLH project is to achieve interoperability of healthcare systems using HL7 V 3.0. Under the umbrella of HLH implementation of various modules of HL7 V3 is being carried out. HLCE is one of them. HLCE has still some modifications to do to make it acceptable worldwide.

## 6.1 Future work

Under the umbrella of HLH ebXML will be implemented and the whole project HLH will be released open source.

**a. Open source Release:**

HL7 V3 implementation has four basic components namely parser, generator, database mapping component and communication environment. Deployment of the project at CITI LABS Pakistan was the first step towards building end to end communication environment. After completion of RIM based database and little modification in the code of communication environment, the whole product will be released open source.

**b. Implementation of ebXML:**

HL7 V3 specifications provide three transportation mechanisms for the transfer of messages namely:

- ebXML
- MLLP

- Web-services

After the completion of web services and MLLP implementation, ebXML will also be implemented for giving the users with fair bit of choice.

# REFERENCES


[1] "The Economic Benefits of Information Technology in Healthcare", MTB Europe portal [Online] available: http://www.mtbeurope.info/content/ft611001.htm, Accessed: 27th July, 2009

[2] David J. Brailer, "Interoperability: The key to the Future of Health Care Systems" [Online] available: http://content.healthaffairs.org/cgi/content/full/hlthaff.w5.19/DC1, Accessed: 25th July, 2009

[3] "Mirth architecture" [Online] available: http://www.mirthcorp.com, Accessed: 22nd April, 2009

[4] "Java CAPS an enterprise solution for the communication infrastructure", API docs [Online] available: http://developers.sun.com/docs/javacaps/api/javadocs/index.jsp, Accessed: 23rd March, 2009

[5] JAVA Messaging Service article [Online] available: http://www.sun.com/software/ products/ message_queue/support.xml, Accessed: 20th April, 2009

[6] "JMS administration model" [online] available: http://java.sun.com/products/jms/tutorial /1_3_1-fcs/doc/images/Fig2.1.gif, Accessed: 23rd May, 2009

[7] "JMS API Programming model" figure available: http://www.pair.com/betasoft/images/ JMSProgrammingModel.png, Accessed: 19th August, 2009

[8] "Factory Design Pattern" from OODesign portal [Online] available: http://www.oodesign .com/factory-pattern.html, Accessed: 25th May, 2009

[9] MLLP Article from Wikipedia [Online] available: www.wikipedia.org/wiki/mllp, Accessed: 25th July, 2009

[10] "Enhancing Web Services Infrastructures with JMS" – O Reilly Media [Online] available: http://onjava.com/pub/a/onjava/2002/06/19/jms.html, Accessed: 20th May, 2009

# APPENDIX

## Sample HL7 V3 Message

Below is a sample HL7 V3 message for patient named Nadeem ordered “1 ½ Hr PP” test. This is a test order request message from one lab to another lab. Similar messages can be sent from collection point to laboratory or from hospital to laboratory.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlacerGroup xmlns="urn:hl7-org:v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" classCode="ACT"
moodCode="RQO">
  <id xsi:type="II" root="1111" extension="" assigningAuthorityName="" displayable="false"/>
  <code xsi:type="CS" code="ObservationType"/>
  <statusCode nullFlavor="NI"/>
  <component1 typeCode="COMP" contextControlCode="AP" contextConductionInd="false">
    <observationRequest classCode="ACT" moodCode="RQO">
      <id xsi:type="II" root="1179" extension="" assigningAuthorityName="" displayable="false"/>
      <code code="380" codeSystem="getCodeSystem"/>
      <text xsi:type="ST" representation="TXT" mediaType="text/plain">1 1/2  hrs PP:Serum Glucose 1 ?  hrs PP:</text>
      <effectiveTime xsi:type="IVL_TS">
        <low inclusive="true" value="20091231"/>
        <high inclusive="true" value="20100101"/>
      </effectiveTime>
      <specimen typeCode="SPC" contextControlCode="OP">
        <specimen classCode="SPEC">
          <id root="1" extension="" assigningAuthorityName="" displayable="false"/>
          <code code="BL" codeSystem="getCodeSystem"/>
          <specimenPerson classCode="PSN" determinerCode="INSTANCE">
            <name xsi:type="EN">
              <prefix>Mr.</prefix>
              <given>Muhammad</given>
              <family>Afzal</family>
              <suffix>I</suffix>
            </name>
            <asContent classCode="ASSIGNED">
              <container classCode="CONT" determinerCode="INSTANCE">
                <desc xsi:type="ST" representation="TXT" mediaType="text/plain"/>
              </container>
            </asContent>
          </specimenPerson>
        </specimen>
      </specimen>
      <recordTarget typeCode="RCT" contextControlCode="OP">
        <patient classCode="PAT">
          <id xsi:type="II" root="274449" extension="1" assigningAuthorityName="" displayable="false"/>
          <statusCode code="RoleStatus"/>
          <veryImportantPersonCode code="Patient Person" displayName="Patient" codeSystem="getCodeSystem"/>
          <patientPerson classCode="PSN" determinerCode="INSTANCE">
            <id xsi:type="II" root="1112" extension="24.0" assigningAuthorityName="" displayable="false"/>
```

```
            <name xsi:type="EN">
              <family>nadeem</family>
            </name>
            <administrativeGenderCode code="Male" codeSystem="10173"/>
            <addr xsi:type="AD">^tel#03365647894</addr>
          </patientPerson>
        </patient>
      </recordTarget>
      <author typeCode="AUT" contextControlCode="OP">
        <assignedEntity classCode="ASSIGNED">
          <id xsi:type="II" root="96" extension="" assigningAuthorityName="" displayable="false"/>
          <addr xsi:type="AD">^tel#</addr>
          <assignedPerson classCode="PSN" determinerCode="INSTANCE">
            <name xsi:type="EN">
              <family>PTV</family>
            </name>
          </assignedPerson>
        </assignedEntity>
      </author>
      <dataEnterer typeCode="ENT" contextControlCode="OP">
        <assignedEntity classCode="ASSIGNED">
          <id xsi:type="II" root="1188" extension="" assigningAuthorityName="" displayable="false"/>
          <addr xsi:type="AD">^tel#</addr>
          <assignedPerson classCode="PSN" determinerCode="INSTANCE">
            <name xsi:type="EN">
              <family>M N Baig</family>
            </name>
          </assignedPerson>
        </assignedEntity>
      </dataEnterer>
      <verifier typeCode="VRF" contextControlCode="OP">
        <sequenceNumber value="1"/>
        <noteText representation="TXT" mediaType="text/plain">noteText</noteText>
        <time xsi:type="IVL_TS">
          <low inclusive="false" value="20080606"/>
          <high inclusive="true" value="20090606"/>
        </time>
        <modeCode xsi:type="CS" code="ParticipationMood"/>
        <signatureCode code="ParticipationSignature"/>
        <assignedEntity classCode="ASSIGNED">
          <id xsi:type="II" root="3112" extension="" assigningAuthorityName="" displayable="false"/>
          <assignedPerson classCode="PSN" determinerCode="INSTANCE">
            <name xsi:type="EN">
              <family>Verifier Name</family>
            </name>
          </assignedPerson>
        </assignedEntity>
      </verifier>
      <dataEntryLocation typeCode="ELOC" contextControlCode="OP">
        <locatedEntity classCode="LOCE">
          <id xsi:type="II" root="14" extension="" assigningAuthorityName="" displayable="false"/>
          <location classCode="PLC" determinerCode="INSTANCE">
```

```
            <name xsi:type="EN">
              <given>Routine Test</given>
            </name>
          </location>
        </locatedEntity>
      </dataEntryLocation>
      <subjectOf2 typeCode="SUBJ" contextControlCode="AN" contextConductionInd="false">
        <sequenceNumber value="1"/>
        <seperatableInd value="false"/>
        <annotation classCode="ACT" moodCode="EVN">
          <code xsi:type="CE" code="ActCode" codeSystem="getCodeSystem"/>
          <text representation="TXT" mediaType="text/plain"/>
          <author typeCode="AUT" contextControlCode="ON">
            <assignedEntity classCode="ASSIGNED">
              <id xsi:type="II" root="3115" extension="" assigningAuthorityName="" displayable="false"/>
              <assignedPerson classCode="PSN" determinerCode="INSTANCE">
                <name xsi:type="EN">
                  <prefix>Miss</prefix>
                  <given>Sidra</given>
                  <family>Aftab</family>
                  <suffix>I</suffix>
                </name>
              </assignedPerson>
            </assignedEntity>
          </author>
        </annotation>
      </subjectOf2>
    </observationRequest>
  </component1>
</PlacerGroup>
```