HL7 V2.4 Message Interface Engine Implement

Sang Min Lee[a], Jin Tae Song[a], Il Kon Kim[a], Hune Cho[b], Yun Sik Kwak[b]

[a]*Department of Computer Science, Kyungpook National University, Korea*
[b]*Department of Medical Informatics, Kyungpook National University School of Medicine, Korea*

Abstract

. *In order to use the HL7(Health Level 7)[1] V2.4 message in a medical related application or system, one needs an interface engine for message handling. In this thesis, to make an interface engine, we have done a HL7 V2.4 message modeling and based on this, we have developed an interface engine for message handling.*

Keywords:
Health Level 7; Interface Engine; Message Modeling; Interface Toolkit

Introduction

Computer and Information Communication technology is developed rapidly, so the medical world is activated the hospital computerization, and an individual medical institution's information digitalized work is processed, too. But such a information system is developed individually each medical institution, so interrelated hospitals are difficult to communicate and to integrate system each other[2]. In order to raise the patient treatment service and the medical institution's efficiency, an individual medical institution's patient information is shared.

In this paper, we use HL7, the standard protocol for the data communication between medical information systems, satisfies the service demand level of all kinds of any medical regardless of the type or size of the medical institution, and the latest standard version is 2.4.

HL7 V2.4 message has standardized a greater part of the matters that occur in the medical service, so the contents are massive and various requirements exist. To solve this problem, a HL7 V2.4 interface engine which will create, test and transmit HL7 messages for all medical applications is needed.

In this paper, first we observe HL7 V2.4 message structure. We utilize this structure to make class diagram, and message modeling. HL7 V2.4 message interface engine, provides the simplest function of creating, validating, transmitting message, is Based on message modeling, and we make this interface engine.

HL7 V2.4 Message Structure

Message

A message is the atomic unit of data transferred between systems. It is comprised of a group of segments in a defined sequence. Each message has a message type that defines its purpose. The real-world event that initiates an exchange of messages is called a trigger event. . There is a one-to-many relationship between message types and trigger event codes. The same trigger event code may not be associated with more than one message type; however a message type may be associated with more than one trigger event.

All message types and trigger event codes beginning with the letter "Z" are reserved for locally-defined messages. No such codes will be defined within the HL7 Standard.

Segment

A segment is a logical grouping of data fields. Segments of a message may be required or optional. They may occur only once in a message or they may be allowed to repeat. Each segment is given a name. Each segment is identified by a unique three-character code known as the Segment ID. All segment ID codes beginning with the letter **Z** are reserved for locally-defined messages. No such codes will be defined within the HL7 Standard.

Field

A field is a string of characters. HL7 does not care how systems actually store data within an application. When fields are transmitted, they are sent as character strings. Except where noted, HL7 data fields may take on the null value. Sending the null value, which is transmitted as two double quote marks (""), is different from omitting an optional data field. The difference appears when the contents of a message will be used to update a record in a database rather than create a new one. If no value is sent, (i.e., it is omitted) the old value should remain unchanged. If the null value is sent, the old value should be changed to null. The various chapters of the Standard contain segment attribute tables. These tables list and describe the data fields in the segment and characteristics of their usage

Data Type

For data types which contain multiple components or subcomponents, the examples given in this section do not specify the optionality of the component or subcomponents. This must be specified in the field definitions that follow the formal segment attribute tables.

Except for the TS data type and the maximum or minimum lengths for several other data types (CE, PN, TX, FT), the field length of HL7 attributes is specified in the segment attribute tables, and any specific length of the components

or subcomponents of those attributes must be specified in the field definitions that follow the formal segment attribute tables. In general, HL7 does not specify the lengths of components and/or subcomponents.

In certain data type definitions, square brackets, "[" and "]", are used to specify optional parts of a data type (or of a data type component or subcomponent).

Message Modeling

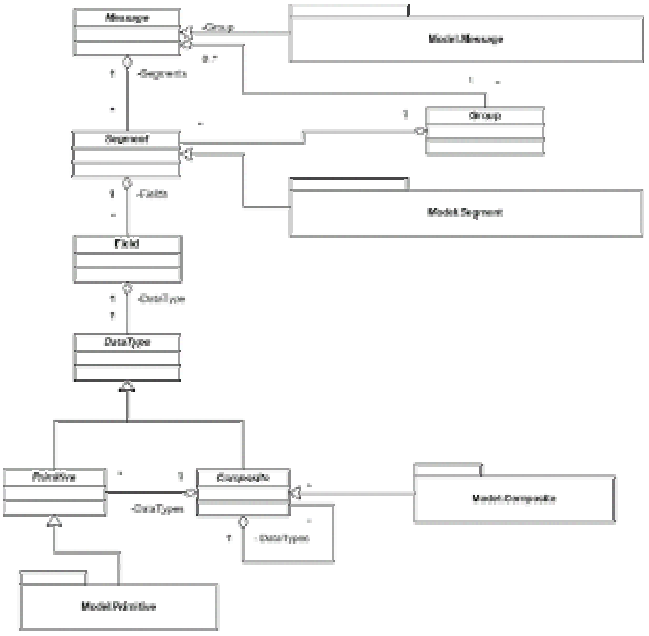

Figure 1 - Message Modeling

HL7 offers a database related to the message structure. This provides all contents related to the message modeling such as Messages, Segments, Fields, Data Types, and Tables. There is a method in which where one can do the message modeling dynamically through in queries. This Method has a lot of advantages for programming. Only a few class structures will be needed to present the modeling method to present the modeling method for the whole message, and time can be reduced. But this method does not suit the HL7 Standard which has various exceptions. For example,, the fifth field of the OBX segment is 'varies'. This form depends on the second field in the OBX segment to decide, its data type. That is, this form is not decided when the message form is created, but which the value of the second field of the OBX segment is decided. There are a few essential fields that must be entered when message types are to be decided. The Field seperator, Encoding character and Message Type fields in the MSH (Message Header) Segment are he most essential details on deciding message forms. Changing the contents of these fields will effect the structure of the whole message, so these fields must be decided when the message is first created. In this case, these fields were modeled in order to handle the various exceptions, enhance the processing speed, and to have a liability.

To process a segment within the message, the required option and repeats option must be handled, and there is also an exceptional case where many segments are bound together to operate as one segment. This case would be called "Group" from now on.

A group has the same role as a segment in a message and the inner part of the group handles many segments such as a message. The segments within the group also acts due to the required option and the repeats option.

A segment is composed of more than one field. The class that is related to the data type is not directly called from the segment, but uses the field as a medium. At this time the name of the data type and the attributes defined by the fields are defined. The field uses the name of the data types to dynamically connect with each data type.

The reason using this method is because of the necessary addition of fields and the need to divide the field and data type. This way, when adding a field, rather than having define a special act dynamically at the segment unit are can use the usual method of making one. The structure of a data type is defined below, but each attribute has a big difference that if a data type has the same attribute as a field, attributes could be unnecessarily mass-produced. Segments continuously add fields using Min_Repeats, Max_Repeats and Position_number .Also, when a operation to get the missing value or the state of a segment is needed, this can be conducted to the field level.

A field and data type is a one-to-one relationship. One field is always connected to one data type. The field supervises various information of the data field and data type integrates various HL7 data types when a field is created, a necessary data type is created and is connected with a data type.

Primitive and composite are derived from data type and the inner part of the composite has an array that includes not only data types which have primitives but also the composite itself. Considering the composite itself may need to the possibility of going into an infinite loop, but due to the HL7 standard it does not repeat more than three times. The happening of all the events due to data input, deletion, and modification is the primitive and it goes directly or via composite to the filed level and announces the outbreak of an event.

Interface Engine Implement

Message Validation

The message validation includes several functions. They are MSH validation, segment validation, field validation, data type validation and table validation. We describe these functions in the following sections.

MSH Validation

First segment of every HL7 message must be MSH. MSH segment has delimeters to parse a HL7 message, value about message structure and value for message transmission. MSH validation has following steps.

- Is a first segment of HL7 message MSH?
- Does MSH has a value about message structure?
- Does MSH has a value for sending / receiving a message?

Segment Validation

When MSH validation is completed, we know message structure. Using this to HL7 database, we get a segments those are members of message. These segments are sequential and have attributes. The segment has two attributes. Compounding these attributes, we get four case those are following.

- Required , Non-Repeat
- Required, Repeat
- Optional , Non-Repeat
- Optional, Repeat

Each segment has only one case among four cases. To validate a segments those are follow a this rule, using a regular expression is very effective. Regular expression is best solution for pattern matching.

For example, ADT_A02 message has a sequence of segments is figure 2.

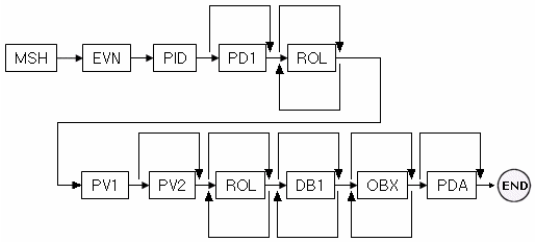

Figure 2 - state diagram for ADT_A02

We can convert a this state diagram into regular expression to validate a segment sequence.

(MSH)(EVN)(PID)(PD1)?(ROL)*(PV1)(PV2)?(ROL)*(DB1)*(OBX)*(PDA)?$

Checking a segment sequence of HL7 message with this regular expression, we can complete a segment validation.

Field Validation

Each segment is a set of fields. The field has a data type, several attributes for data type. Attributes are as follows :

- Length
- Required / Optional
- Repeat / Non-Repeat

According to field, same data types have a different length. Using field delimeter that is got from MSH, we can split a segment. To validate required/optional attribute, we just validate whether field value is null or not. To validate repeat/non-repeat attribute. We just look for a repeat delimeter.(Recommended value for this is '~').

Data Type Validation

Data type has tow kind of data types. These are primitive data type and composite data type. Primitive data type is a basis. Composite data type has a other composite data types or primitive data types as child. Therefore, to validate composite data type, we first validate child data types. So, primitive data type validation is the kernel of data type validateion.

According to property, primitive data types is divided into two. One is string data type, other is numeric data type. To validate these property, using a regular expression that is used to validate sequence of segment is very effective.

For example, SI data type has a positive integer value. The regular expression to validate SI is below :

(+)?[1-9]*[0-9]$

Using this regular expression, we can validate a positive number. We can also validate other numeric data types using this method. But, string data types can have all kind of value, so string data types validation is not necessary.

Table Value Validation

ID and IS data types has a value that is among hl7 defined values or user defined values. These values are stored in database. Difference between ID and IS is where valus is stored in. Values for ID are stored in HL7 defined database and Values for IS are stored in user defined database. ID value must be one of them, but IS value is not. Table number for ID/IS is stored in attributes of field. Querying a this table number to database, we can retrieve a valid value for ID/IS.

Message Creating

HL7 2.4 message being a string type, where the meanings are separated with delimeters. When we create a HL7 message, if we weren't have a common making module, we are faced with many errors. So we define a common making module in a interface engine, we can handle message not a string form but a class form. This way offers convenience and efficiency, because it is based on the field information.

Whole Message Making

HL7 V2.4 message is a tree structure form. A message is in the highest level, there are segment, field, data type the lower part of a message. This level is separated by delimeter in message. When we make whole message, data type is the start position. Each level is going upper the position, it combines the upper node and the lower node. Considering the composite itself may need to the possibility of going into an infinite loop, but due to the HL7 standard it does not repeat more than three times. Due to subcomponents has different delimeter, data type is divided different level.

MSH Making

MSH(Message Header) segment is the most important segment in message. MSH contains message type, message sender, message receiver, separator, version id, time/date of

message, etc. If user change MSH field, whole message structure is changed. So, it has many problem to interpret HL7 message. The user should be carefully handle MSH segment, interface engine offers so small approachable API MSH segment.

Repetition Handling

There are two kinds of the repetition in messages of HL7's version 2.4. One is a segment unit and another is a field unit. Repetition of a group handled as segments is also possible. It is possible to repeat segments and fields next to the original position. However, it is not proper to manage messages and access the right position because the inserted position is not fixed but dynamic.

We suggest that it is connected the segments and fields consecutively under the original message in the standard message architecture. Therefore, we handle the repetition of the message and maintain the standard message architecture.

Varies Value Handling

There is a 'varies' data type in the MFE, OBX and QPD segments. For example in OBX, it is decided a data type of the Varies field when we determine a value of the second field of OBX segment. We evaluate the event when it is changed and decide the data type of the Varies field.

Data Input Handling

We need to check a validation for the data when we insert them. String type is difficult to determine but it is possible to text the date, time, DT, TM and TS. We are able to reduce errors of messages by checking whether the data type connected with tables is a defined value in the tables or not.

Since messages are tree structured, we have to access the sub data types to get the root value. It takes much time so we hold the data in the field to reduce the access. When data types are inserted, we evaluate the event and redefine the value of the field connected to them.

Message Transmission

The important role of HL7 interface engine is transmission of hl7 message between health care applications. That is, hl7 interface engine can transmit a hl7 message from A1 application at A hospital to B1 application at B hospital. If A1 application know IP and port number of B1 application, A1 application transmit hl7 message to B1 application directly. But IP and port number of B1 application can be changed and B1 application is not available. In these case, in order to stably transmit a hl7 message, we propose a following network architecture.

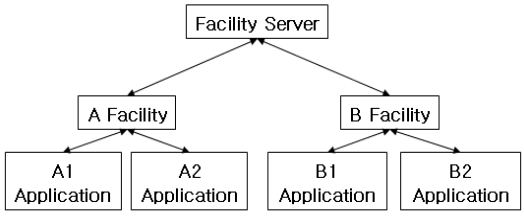

Figure 3 : network architecture

Each node on figure 3 has a following role.

- Application generate HL7 message, send it or receive hl7 message from other application.
- Facility receive message from application and forward it to other application that is located in same facility. Message to other facility is forwarded to facility server. Received message from facility server is forwarded to application by facility.
- Facility server is node for sending and receiving hl7 message between other facility. Facility server forward a hl7 message from sending facility to receiving facility.

Using network architecture like figure 3, A1 application can send a hl7 message to B1 application with name of B1 application. Interaction between A1 application and B1 application is below :

- A hospital (facility) interface engine is registered at facility server.
- B hospital (facility) interface engine is registered at facility server.
- A1 application is registered at A hospital (facility) interface engine.
- B1 application is registered at B hospital (facility) interface engine.
- A1 application sent a message to B1 application at B hospital.
- Message from A1 application is arrived at A hospital interface engine.
- A hospital interface engine parse message and forward it to facility server.
- Facility server receive message from A hospital and forward it to B hospital.
- B hospital receive a message from facility server. B hospital interface engine parse it and get receiving application name.
- B hospital interface engine forward a message to receiving application.
- If receiving facility interface engine is not available, message is stored in message queue at facility server. Similarly, when receiving application is not available, message is stored in facility interface engine.

Advantage from this architecture that is proposed by us is below :

- Sending application can send a message with just receiving application name and receiving facility name.
- When IP and port number of receiving facility and application is changed, there is no need for sending a message.
- When facility or application is not available , we can

stably transmit a message between applications using a message queue.

Conclusion

We observed HL7 V2.4 Message structure and Message Modeling. HL7 V2.4 message interface engine, provides the simplest function of creating, validating, transmitting message, is Based on message modeling, and we make this interface engine. But provided only the genuine interface engine the medical related application developer will have a difficulty understanding and approaching the HL7 message. To avoid this case, offering a toolkit based on the interface engine will enhance the speed of the development. So, We will implement a HL7 V2.4 interface toolkit.

Acknowledgment

This study was supported by a grant of the Korea Health 21 R&D Project, Ministry of Health & Welfare, Republic of Korea(02-PJ1-PG6-HI03-0004).

Reference

[1] HL7 URL : http://www.hl7.org/

[2] Mandle KD, Kohane IS. Healthconnect: Clinical grade patient-physician communication. In Proceedings, AMIA Annual Symposium 1999

[3] Rhapsody Document, Orion, 2002

[4] Eah-Wen Huang, Sheng-Hsiung Hsiao, Der-Ming Liou, “Design and implementation of a web-based HL7 message generation and validation system”, International Journal of Medical Infomatics (2003) 70, 49-58

[5] Andrew Hutchison, Matthias Kaiserswerth, Michael Moser, and Andreas, “Electronic Data Interchange for Health Care”, IEEE Communication Magazine, July 1996