Smart Interface Template and Conceptual Framework for Auto Interoperability in Healthcare Delivery System

Sanjib PANDEY[1,2,*], Abdul KAZI[2], Dahya MAHUL[2,3], John BABATUNDE[4]

[1] Oxleas NHS Foundation Trust, Memorial Hospital, Shooters Hill, London, SE18 3RG UK
[2] Chelsea and Westminster NHS Foundation Trust, Unit G3, Harbour Yard, Chelsea Harbour, London, SW10 0XD, UK
[3] The Doctors Laboratory, The Halo Building, 1 Mabledon Place, London,WC1H 9AX, UK
[4] Principal Consultant Systems and Infrastructure Ltd, Freelance (Currently with IBM), 27 Old Gloucester Street, London, WC1N 3AX, UK
E-mail (*): s.pandey@nhs.net

* Author to whom correspondence should be addressed; Tel.: 44 7861218562

Received: June 21, 2020 /Accepted: September 17, 2020 / Published online: September 30, 2020

Abstract
This paper describes two components of the health care delivery system, the Smart Interface Template (SIT) and the Auto Data Mapping (interoperability) Process. The SIT, a reusable interface development method, has been designed and implemented successfully in the Chelsea and Westminster NHS Foundation Trust, London, UK. It has been used to develop various new Health Level Seven (HL7) message interfaces such as Patient Admission, Discharge and Transfer Messages (ADT), Order Messages (ORM), Result Messages (ORU), and Patient Visit (PV1 & PV2) between the diverse health care delivery systems. The Auto Data Mapping (ADM) on the other hand, is a conceptual framework for automating data mapping processes between the healthcare applications. This framework has the potential of increasing the efficiency of data by reducing human involvement in data mapping, particularly in heterogeneous healthcare delivery applications. The Auto Data Mapping (ADM) framework is an evidence-based tool which employs a computer-aided knowledge-based technique to generate Health Level 7 (HL7) message mapping rules.

Keywords: Health Level 7; Integration Engine; Knowledge-Based; Rule-Based; Intelligent Agent; Machine Learning

Introduction

There are several clinical information systems, with each having its own pre-defined clinical settings, such as disease name, patient-related information, and laboratory test and result data sets. The diverse vendors develop many of these systems, hence maintaining interoperability between the systems is difficult. In addition, the lack of proper standardization of healthcare data sets makes the interoperability process very complicated and challenging.

Interoperability in relation to clinical systems is "… *the ability of different information technology systems and software application to communicate, to exchange data accurately, effectively and consistently and to use the information that has been exchanged*" [1]. In general, some of the challenges experienced in the existing technologies employed in integration engines today can be summarized as follows:

1) Lack of shared IT infrastructure in hospitals, coupled with a vast number of IT solution vendors supplying varying IT solutions and equipment. This creates a situation where several

IT solutions do not interoperate with each other: Examples are where one system codes Blood Pressure as a "Hypertension" while another codes the same parameter as "High Blood Pressure"; or where one system codes Heart Attack as a "Heart Attack" while the other codes it as a "Myocardial Infraction".

2) Lack of standardization in electronic clinical documents such as inpatient/outpatient discharge summary, Emergency Department (ED) discharge summary, clinical letters, clinical notes etc.
3) Lack of standardization in other clinical setting, some of them are given below:
    - NHS Number : MRN, Patient ID, Clinical ID (required unique single point patient access ID)
    - Medication – lack of standard electronic medical message,
    - Data Inconsistency in GP code, Practice Code, Consultant Code, Diagnostic Coding,
    - Patient demographic information and doctors or consultant information (e.g., gender: 1, 0 or Male, Female, etc.)
4) Data migration challenges stemming from the diverse formats and data sizes.

We discuss the SIT and summaries our work in addressing interoperability challenges in clinical information systems. As part of the SIT design, we consider the following essential elements necessary to develop a robust, reliable and error-free health care delivery application:

- Systems must be capable of working with solutions from diverse vendors and health care service, providing organizations and should be capable of utilization common health information standards.
- It is vital to develop common interoperability standards for the health sector and it is mandatory that stakeholders support and subscribe to interoperable structures.
- It is essential to establish standard common technical data for attributes such as patient's name, gender, data of birth, race, ethnicity, religion etc.; and for other parameters such as vital signs, standard care plan fields, laboratory tests and results, medication, allergies, discharge summary letter, appointment letter etc.

The details of the SIT method, SIT architecture, workflow and experimental results are explored and analyzed in Material and Method Section of this paper.

The second section of this paper is the ADM framework. The ADM is a conceptual framework for automated data mapping between the healthcare applications. This framework has the potential to increase the efficiency of work and reduce the human involvement in data mapping (interoperability) for heterogeneous healthcare delivery applications. The auto mapping utilizes intelligent agent in "getting the right information from the right knowledge-base to the right destination in the right format at the right time" [2]. We propose an evidence-based approach to auto map process and generate the HL7 message mapping rules. The central component of the ADM framework works in such a way that if a rule is not found in the rule-based system, a supervised machine learning based process will automatically be applied which then generates the required rule(s). In healthcare systems, application data should be mapped between the source and destination systems correctly and accurately. So, the development of automapping algorithm is a very challenging tasks as healthcare system are interrelated with heterogeneous application. Thus, the proposed conceptual framework will provide automated mapping capable of efficiently facilitating data exchange between different healthcare delivery applications. The details of the ADM framework are explored and analyzed in the section of "conceptual framework for auto mapping" of this paper.

Material and Method

Background Information: Integration Engine

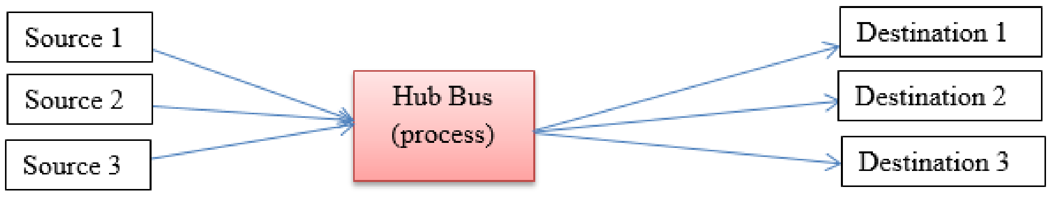
The integration engine is the backbone of a typical health care delivery system, without which it is almost impossible to exchange patient-related clinical data between the heterogonous healthcare delivery systems. The integration engine is essentially a software that receives a message from a source, processes it (according to instructions/user setup), and transforms it to a destination using various protocols/ports [3]. The prime function of an integration engine is to listen and pick up an inbound message (a message that comes from the hospital main Electronic Patient Information System, for instance, CERNER) and generate an outbound message which is then passed to an appropriate destination. The destination could be a different application (health care delivery system). There are various integration engines which are developed by different vendors and are being used in healthcare delivery systems today, some of the common ones are: InterSystem - Ensemble, Orion Health, Corepoint Health, Submit Healthcare, Siemens, OpenLink, ColverLeaf, Mirth, Iguana and Summit [4]. Oracle also has the "*Oracle health science integration engine*", details about this engine can be found in [5].

Further information about Healthcare Integration Engines can be found in KLAS Research [6]. The Scala, Open eHealth platform, Mirth, NextGen, King College IE, etc. are some examples of open-source Integration Engine. The eHealth platform (Java/Groovy/Camel/HAPI and code available on GitHub) [7]. The Mirth is a Java-based open source integration engine, the experiences of Mirth and more detail about it found on [8]. Al-Sakran [9] described a conceptual design to implement integration strategies with suitable technologies to tackle interoperability problems.

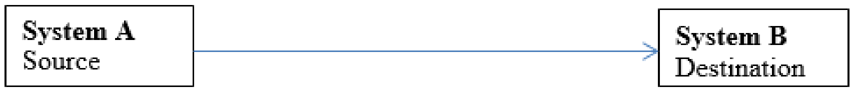
Integration Engine: Implementation

The integration engine can be implemented in different ways. Below are the conventional approaches for achieving integration engines:

a. Centralize environment

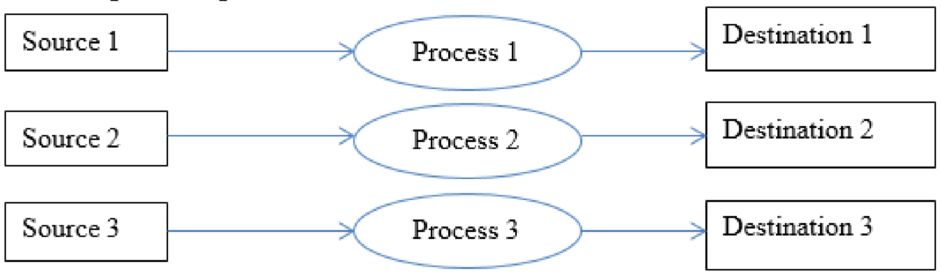


b. Point-to-point environment



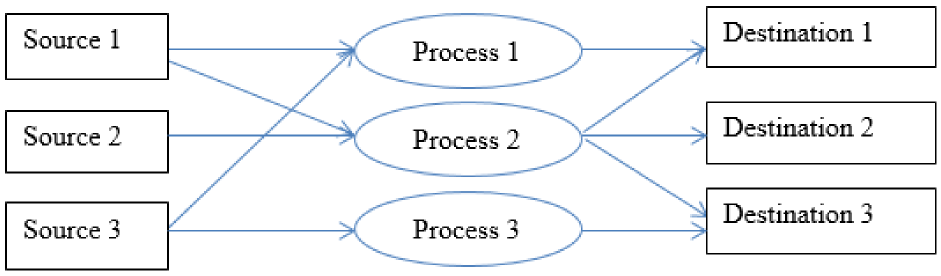
A point-to-point interface is a direct connection between two systems and is quite popular if data sets are well known between systems. In a point-to-point environment, both systems must communicate with the same type of data standard/format and protocol and it is important that both data formats and protocols are unlikely to change. It can reduce the dependency function to change the message format in many medical software to send or receive messages.

c. Independent process



Each source and destination have connected with an individual process.

d. Mixed Approach



Smart Interface Template

We have designed and developed a reusable Smart Interface Template (SIT), which is suitable and applicable for use in any kind of new interface (e.g., admission, discharge, transfer patient, lab order, lab report, patient discharge document, patient appointment, etc.) development for transferring HL7 messages between various medical applications. The SIT has four classes, namely, Environment Setup Class, Common Class, Log Services and Web Services, each of which has specific functionalities and responsibilities which support the building of SIT. The SIT was designed and implemented in Ensemble/HealthShare Integration Engine using Cache Object Script [10]. The SIT architecture and components are discussed in the next sub-sections.

Smart Interface Template Architecture

The SIT takes the message from inbound (source), processes the message and sends it to outbound (destination). The SIT communicates with inbound and outbound systems and based on the instructions, it further processes or drops the messages. The SIT architecture is illustrated in Figure 1.

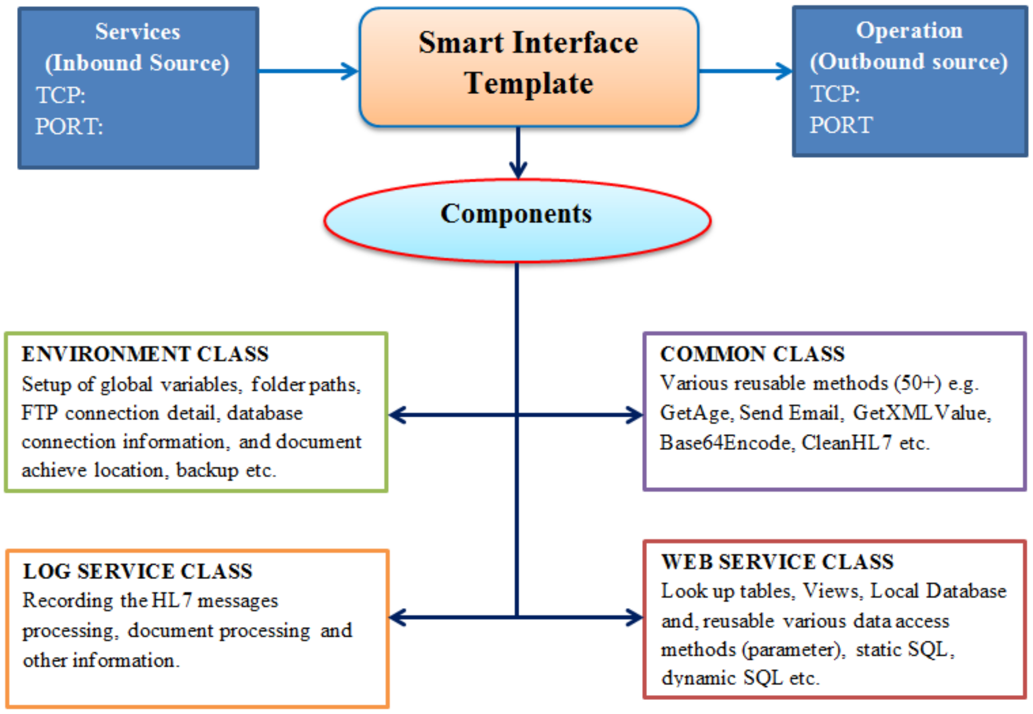

Services (Inbound Source)
TCP:
PORT:
Smart Interface Template
Operation (Outbound source)
TCP:
PORT
Components
ENVIRONMENT CLASS
Setup of global variables, folder paths, FTP connection detail, database connection information, and document achieve location, backup etc.
COMMON CLASS
Various reusable methods (50+) e.g. GetAge, Send Email, GetXMLValue, Base64Encode, CleanHL7 etc.
LOG SERVICE CLASS
Recording the HL7 messages processing, document processing and other information.
WEB SERVICE CLASS
Look up tables, Views, Local Database and, reusable various data access methods (parameter), static SQL, dynamic SQL etc.

Figure 1. Smart Interface Template architecture (TCP: Transmission Control Protocol, FTP: File Transfer Protocol, HL7: Health Level 7, SQL: Structured Query Language)

The classes or components are described below:

1) Common Class. In this class, where various methods have been written; all these methods are reusable and can be used in different ways. The SIT can call these methods according to the requirements. Example of some methods are:

- GetAge (provide the age)
- FTP download (download the files from a remote server)
- upLoadFile (upload the files in the remote server)
- readGetValueXML (get the given clinical data from XML file)
- FileCopyOrMove (move or copy file(s) from one location to another location)
- Base64Encoding and Base64Decoding (PDF file encoding and decoding, send information into HL7 message OBX segment; read encoding HL7 message and decoding it)
- SendEmail (send email)
- Alert Email (send the alert email)
- Statistic information (Realtime inpatient discharge, appointments information, ED discharge summary etc.)
- Etc.

2) Environment Class. In the Environment class; all required global variables have been setup, for instance, FTP connection details (IP address, username, password), document location, file path, database setup, setup email information, file process monitoring path, all the downstream system information (operation, process name), etc.

3) Log Service Class. This class is responsible for recording the HL7 message processing information into the log table. The recording components are ID, MRN number, NHS Number, Message ID, Message Type, Message, Components (source and destination information), Date and time. This class also responsible to log the information of all inpatient, outpatient document processing (e.g., discharge summary, ED discharge summary, appointment, report, etc.).

4) Web Service Class. This method is responsible for getting the information from local database (lookup tables, views). Web service class has consisted the database connection information, data warehouse information, SQL Server, DB Visualizer connection, create and manage the local database, method to extract the data from Excel sheet, CSV, PSV file, export the data into Excel etc. There is various reusable parameter-based data access method written such as:

- Get_GP_Detail_Inforamtion(*GPCode*),
- Get_Consultant_Information(*consultantCode, LookupField, displayValue*), GetPatient(*ID*),
- GetConsultantSpecialist(*hospitalNumber, appointmentDate, displayFiledList*)
- etc.

These all retrieve the data according to requirements. Furthermore, various static and dynamic SQL statements is written to lookup the data from lookup table or views, for instance: Obtain Patient Information, GP Information, GP Practice Information, GP Address, Patient Appointment, Reschedule Appointment, etc.

Smart Interface Template Components

The SIT has various sub-methods that are responsible and capable for performing specific tasks. The Request, Response, Message Check, Message Type, Message Call and other message segments are sub-methods of SIT. The order of sub methods, relationship and how they communicate with other classes is shown in Figure 2.

Workflow of Smart Interface Template

The SIT needs to be connected by Inbound (source) and Output (destination) listeners; this allows the SIT to read source message and check the message, to determine whether the message is valid or invalid. The invalid message will be dropped, while a valid message will be processed. The steps in message processing are illustrated in the flowchart in Figure 3.

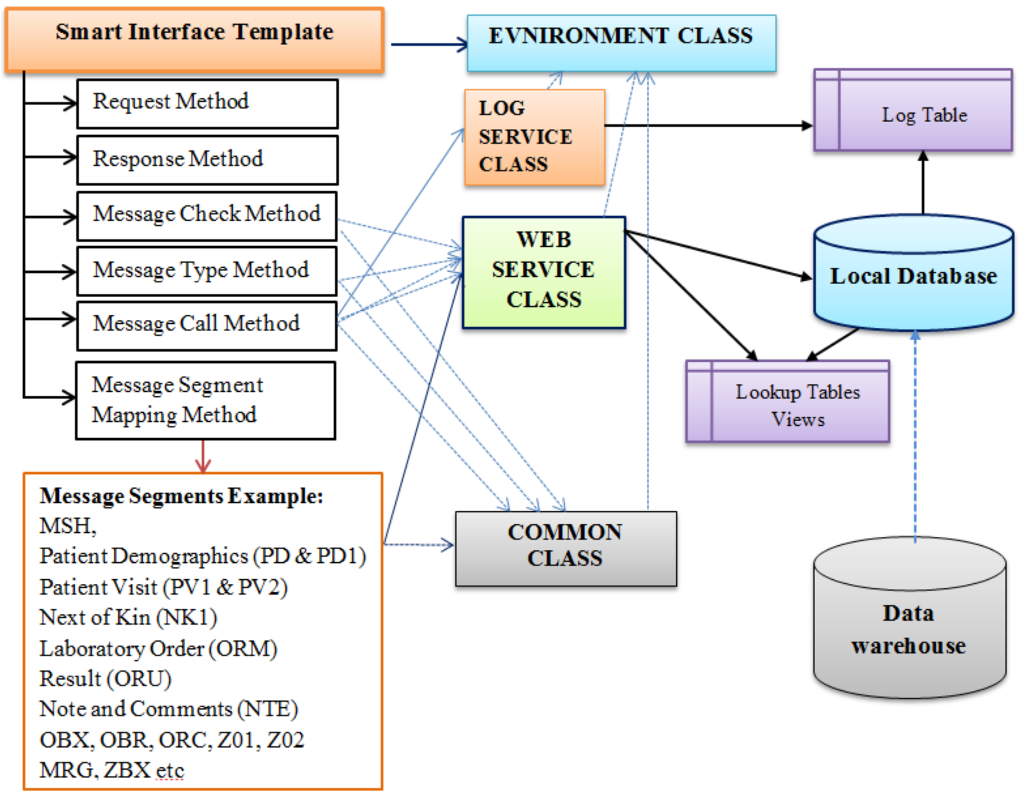


Figure 2. Smart Interface Template components and communication with other classes

This approach is flexible and will ensure interoperability in healthcare applications. It will not only be fast and efficient but will also be easy to use and will be capable of solving most interoperability problems in healthcare applications.

Experiment and Result

The SIT was developed in Ensemble/HealthShare integration engine using Intersystem Cache Object Script Language. Cache Object Script Language is an inbuilt coding platform of Ensemble/HealthShare integration engine. Currently, we have implemented more than 20 interfaces at Chelsea and Westminster NHS Foundation Trust, that has been developed based on SIT architecture. The screenshot of SIT and its Sub Methods or Components is shown in Figure 4.

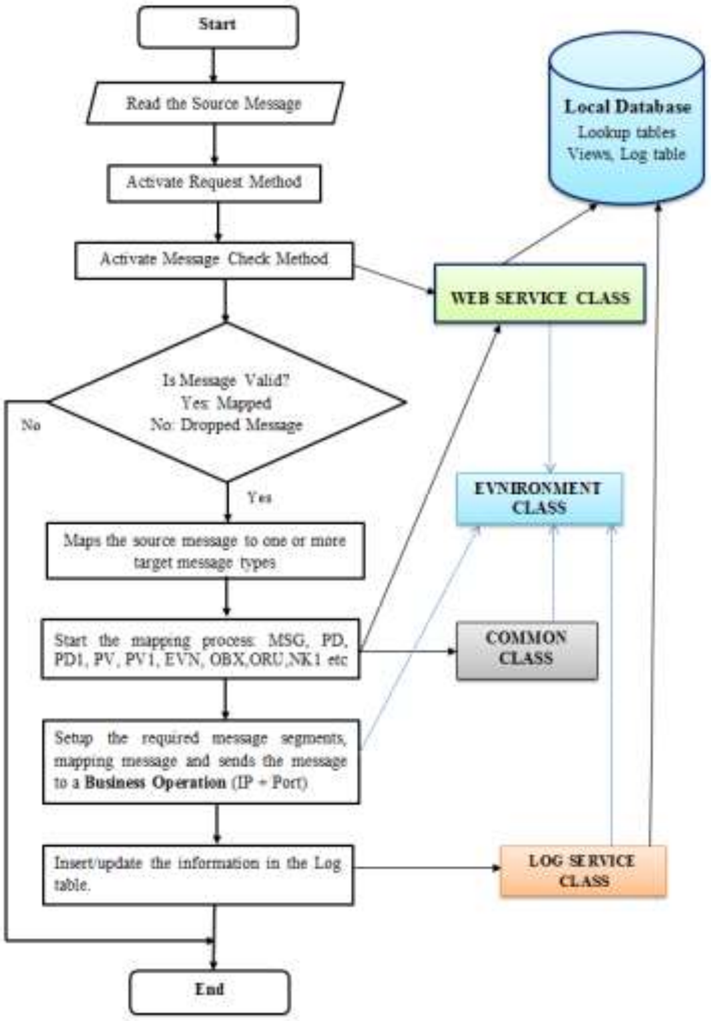
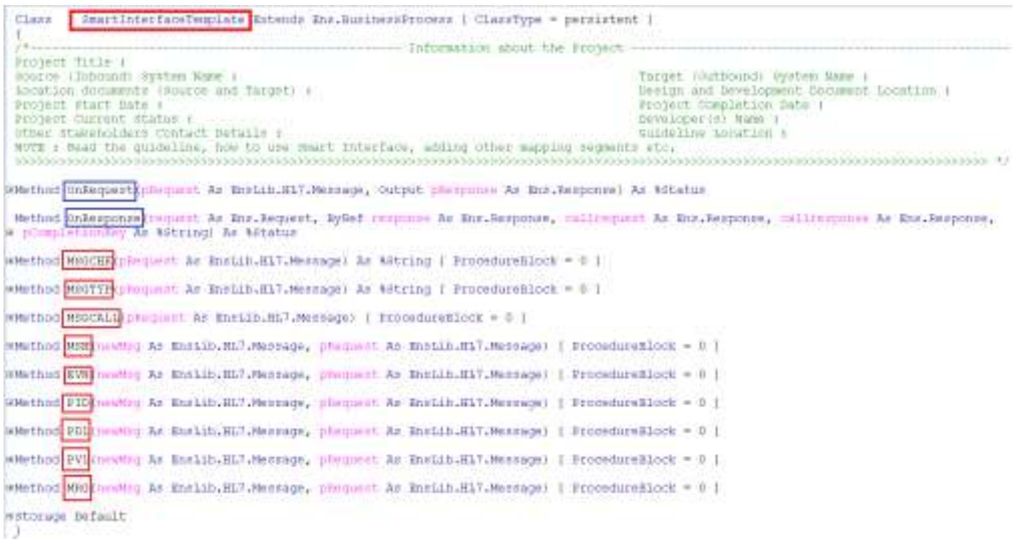

Start
Read the Source Message
Activate Request Method
Activate Message Check Method
Is Message Valid?
Yes: Mapped
No: Dropped Message
No
Yes
Maps the source message to one or more target message types
Start the mapping process: MSG, PD, PD1, PV, PV1, EVN, OBX,ORU,NK1 etc
Setup the required message segments, mapping message and sends the message to a Business Operation (IP + Port)
Insert/update the information in the Log table.
End
Local Database
Lookup tables
Views, Log table
WEB SERVICE CLASS
EVNIRONMENT CLASS
COMMON CLASS
LOG SERVICE CLASS

Figure 3. The workflow of Smart Interface Template



```
Class  SmartInterfaceTemplate Extends Ens.BusinessProcess [ ClassType = persistent ]
{
/*----------------------------------------- Information about the Project -----------------------------------------
Project Title :
Source (Inbound) System Name :                                         Target (Outbound) System Name :
Location documents (Source and Target) :                               Design and Development Document Location :
Project Start Date :                                                   Project Completion Date :
Project Current Status :                                               Developer(s) Name :
Other Stakeholders Contact Details :                                   Guideline Location :
NOTE : Read the guideline, how to use Smart Interface, adding other mapping segments etc.
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> */

±Method OnRequest(pRequest As EnsLib.HL7.Message, Output pResponse As Ens.Response) As %Status

 Method OnResponse(request As Ens.Request, ByRef response As Ens.Response, callrequest As Ens.Response, callresponse As Ens.Response,
± pCompletionKey As %String) As %Status

±Method MSGCHK(pRequest As EnsLib.HL7.Message) As %String [ ProcedureBlock = 0 ]

±Method MSGTYP(pRequest As EnsLib.HL7.Message) As %String [ ProcedureBlock = 0 ]

±Method MSGCALL(pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Method MSH(newMsg As EnsLib.HL7.Message, pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Method EVN(newMsg As EnsLib.HL7.Message, pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Method PID(newMsg As EnsLib.HL7.Message, pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Method PD1(newMsg As EnsLib.HL7.Message, pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Method PV1(newMsg As EnsLib.HL7.Message, pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Method MRG(newMsg As EnsLib.HL7.Message, pRequest As EnsLib.HL7.Message) [ ProcedureBlock = 0 ]

±Storage Default
}
```

Figure 4. Screenshot of Smart Interface Template (MSH: Message Header, EVN: Event Type, PID & PID1: Patient Identification, PV & PV1: Patient Visit, MRG: Merge Patient Information)

The main features of the methods or components of SIT template are as follow:

1) Method OnRequest (*request, response*): this method has the request and response arguments and it is called in the first instance and this method will call and perform the logic, rules, tasks that define in the other three methods (message check, message type and message call) sequence one after another.

2) Method onResponse (*request, response, callrequest, callresponse, pCompletionKey*): The method has the following signatures:

- Request: *request object sent to SIT process.*
- Response: *response object returned by this SIT process.*
- Callrequest: *incoming response object.*
- pCompletionKey: *the key value associated with the incoming response.* [Intersystem documentation]

Method MessageCheck (*request*): This method is responsible for dropping invalid messages or carrying out further checks on valid messages. To carry out further checks, the segments of the message, other validity will be checked by analyzed both documentations (inbound and outbound system) and other predefined business rules. Below is a demonstration of the process of checking source message type and destination message type for a particular message segment:

```
READ: source message
CHECK: If source message's header segment has the following message type?
    ADT^A28 (e.g. Patient creation), ADT^A31 (e.g. Demographic update)
    ADT^A01 (e.g. Admit patient, ADT^A02 (e.g. Transfer patient)
IF YES THEN
    ACCEPT message and do the further process
NO
    REJECT and DROP the message
END
```

3) Method MSGTYP (*request*): This is the message type method. Once the source message types have been accepted, then this method will activate and allows to map the source message into a specific target message. Message can be mapped into one or more target message types.

Read message: get source message and mapped to target message type (these methods will be setup according to the mapping documentation of both system (source system and destination system)

For Example:

```
IF source message type "ADT^A28" THEN target message type "ADT^A28"
IF source message type "ADT^A18" THEN target message type "ADT^A40"
IF source message type "ADT^A01" THEN target message type "ADT^A01"
IF source message type "SIU^S12" THEN target message type "ADT^A05"
```

4) Method MSGCALL (*request*): This is the message call method. This method will setup the message type and required segments of a particular message type. So, again, both documentation and requirements will be observed to determine the segments of the message type. For instance:

ADTA28 = MSH, EVN, PID, PD1, PV1

Note: ADTA28 is the patient creation message type; the required segments for this message are MSH (message header), PID/PD1 (patient information), PV1 (patient visit information), EVN (event type code), etc.

The following method will start to mappings the process based on the message call. In this example message call has ADTA28 message type and has five segments (MSH, EVN, PID, PD1, PV1); so, it is required to create each method for them to mapping the message.

5) Method MSH (new message): all the segments value/information of MSH (MSH3, MSH4, MSH5, MSH6, MSH7 etc.)

SET MSH3 VALUE = "required information", "MSH: 3"

6) Method EVN (new message): a segment of EVN

7) Method PD (new message): all the PD segment values; e.g.

```
SET PID3 = "hospital number", "PID:3"
SET PID5 = "patient last name, first name, middle name, title etc.", "PID:5")
```

SET PID7 = "patient dob", "PID:7"

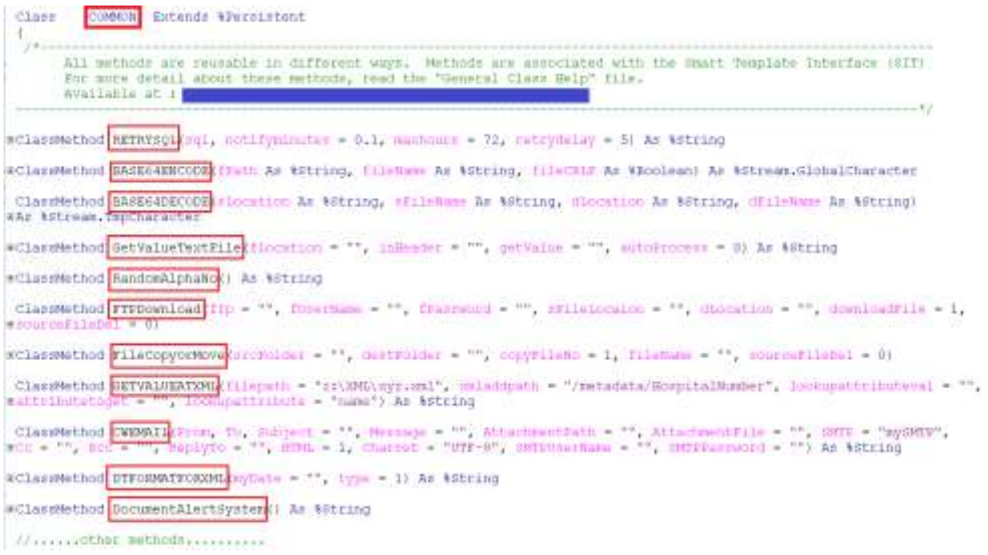
8) Method PD1 (*new message*): patient additional information e.g.

SET PD13 = "GP Practice Code", "PD1:3"
SET PD14 = "GP National Code", "PD1:4"

9) Method PV1 (*new message*): patient visit information e.g.

SET value PV18 = "Refereeing Doctor", "PV1:8"
SET value PV17 = "Attending Doctor", "PV1:7"
SET Value PV144 = "Patient Admit Date", "PV1:44", etc.

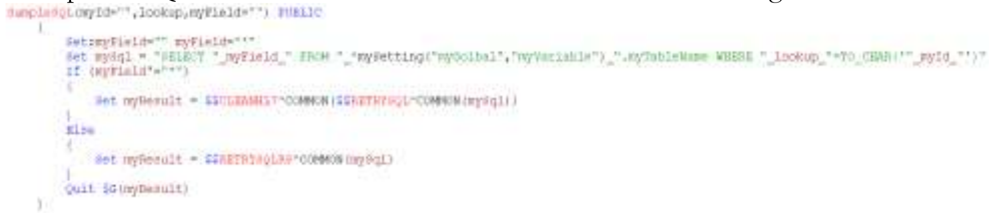
The required methods are OnRequest, onResponse, MSGCHK, MSGTYP, MSGCALL, MSH others are message's segments methods; users can add or remove segments method according to the requirements. Most of the methods are associated with other classes. The sample of Common Class file is shown in Figure 5.



```
Class   COMMON Extends %Persistent
{
 /*-----------------------------------------------------------------------------------------
      All methods are reusable in different ways.  Methods are associated with the Smart Template Interface (SIT)
      For more detail about these methods, read the "General Class Help" file.
      Available at : [redacted]
 -------------------------------------------------------------------------------------------*/

ClassMethod RETRYSQL(sql, notifyminutes = 0.1, maxhours = 72, retrydelay = 5) As %String

ClassMethod BASE64ENCODE(Path As %String, fileName As %String, fileCRLF As %Boolean) As %Stream.GlobalCharacter

 ClassMethod BASE64DECODE(sLocation As %String, sFileName As %String, dLocation As %String, dFileName As %String)
As %Stream.TmpCharacter

ClassMethod GetValueTextFile(fLocation = "", inHeader = "", getValue = "", autoProcess = 0) As %String

ClassMethod RandomAlphaNo() As %String

 ClassMethod FTPDownload(ftp = "", fUserName = "", fPassword = "", sFileLocation = "", dLocation = "", downloadFile = 1,
sourceFileDel = 0)

ClassMethod FileCopyorMove(srcFolder = "", destFolder = "", copyFileNo = 1, fileName = "", sourceFileDel = 0)

 ClassMethod GETVALUEATXML(filepath = "z:\XML\xyz.xml", xmladdpath = "/metadata/HospitalNumber", lookupattributeval = "",
attributetoget = "", lookupattribute = "name") As %String

 ClassMethod CWEMAIL(From, To, Subject = "", Message = "", AttachmentPath = "", AttachmentFile = "", SMTP = "mySMTP",
Cc = "", Bcc = "", ReplyTo = "", HTML = 1, Charset = "UTF-8", SMTPUserName = "", SMTPPassword = "") As %String

ClassMethod DTFORMATTOXML(myDate = "", type = 1) As %String

ClassMethod DocumentAlertSystem() As %String

 //......other methods..........
```

Figure 5. Screenshot of Common Class

The sample of SQL statements in the Web Service class is shown in Figure 6.



```
sampleSQL(myId="",lookup,myField="") PUBLIC
    {
        Set:myField="" myField="*"
        Set mySql = "SELECT "_myField_" FROM "_^mySetting("myGolbal","myVariable")_".myTableName WHERE "_lookup_"=TO_CHAR('"_myId_"')"
        If (myField="*")
        {
            Set myResult = $$ULEANMIT^COMMON($$RETRYSQL^COMMON(mySql))
        }
        Else
        {
            Set myResult = $$RETRYSQLRP^COMMON(mySql)
        }
        Quit $G(myResult)
    }
```

Figure 6. SQL statement in Web Service Class

The main functionalities of Service, Process, and Operation of Integration Engine are as follow:

- **Services**: (inbound source – define by port) accept the message that comes from source and it will define by port.
- **Process**: (inbound and outbound message mapping) process (code) to map the message according to requirement.
- **Operation**: outbound source (destination) define IP address

Conceptual Framework for Auto Mapping Message

The manual mapping process typically requires significant human labor and could be time-consuming with a huge chance of human errors being introduced to process or the data itself as there will be no automated filtering, alerting, reminder methods are available. In view of this problem, this paper sets out to introduce a method that makes the mapping process automated by using new automation techniques and functionalities. To achieve this, the integration engine is required to understand the structure and format of the message, message route, message destination, etc. Various infrastructures can be applied in health care, but still, due to the lack of standards in Integration Engine design, it is often challenging to achieve mapping and fast transmission of data between diverse systems. Furthermore, it is still difficult to cope with various heterogeneous health care delivery system, for instance, products (such as architecture, interface, medical device integration), services (such as implementation, maintenance), and systems/applications (such as EPR, Radiology, Pathology, digital audio/video, patient-related data, clinical data, etc.) through one unique healthcare integration engine. As a consequence, it is predicted that the healthcare integration market is worth an estimated total of about $2.75 billion by 2018 [11].

After studying various online papers, journal and experiences of working at Integration department, it has been identified that the integration engine at least most have the following features to provide better services:

1. Capable of transferring the message/data without error within minutes.
2. The robust interface (mapping and processing data and logic efficiently)
3. Advanced migrating facilities: Securely migrating the data into other health care delivery system, if required (e.g., Epic, Cerner)
4. Powerful Data Exchange Mechanism (securely and effectively exchange the data between the heterogeneous healthcare delivery system)
5. Intelligent Workflows Mechanism: alert (e.g., alter the right people if something goes wrong), reminder, data validating, auto data mapping and processing methods, etc.)
6. Hybrid Technique: Integration engine able to communicate with various source of data e.g. JSON (JavaScript Object Notation), XML, Cloud based Technology, FHIR (Fast Health Interoperability Resources, some RDBMS (Oracle, SQL server, MySQL, etc.) big data analytic, Excel, etc.
7. Excel Compatible Capacity (operating systems, network platform, communication channel, etc.)

Data mapping is a fundamental step in the design data interface. Data mapping is defined as the process of creating various data element mappings between two or more different data models. Data mapping is a very critical part of the data interface because the format of data is not the same in the medical software. The same data can be present in different sets; therefore, it is very difficult to one-to-one mapping. Required other several supporting tools to mapping the data, for example, some source system send the GP Code only in the HL7 message and destination system expecting or required GP code and detail information of GP, so to meet this requirement need to additional support to obtain the GP detail information might be database lookup table. It is also necessary to understand the context in which the data is used so as to have insight into the structure of data and the format of each data element.

Auto Mapping Process: Hypothesis

This subsection discusses the automapping development process. The proposed conceptual framework is based on the following hypothesis:

Our proposed conceptual framework has the following hypothesis.

1) This framework is not intended to develop a new integration engine.
2) This framework is based on the exiting Ensemble/HealthShare Integration Engine
3) This framework is more concerned about how to develop and deployment the intelligent automated healthcare data mapping process between the heterogeneous healthcare delivery system.
4) This framework emphasis on how to exchange the message automatically from one main hospital EPR system to diverse downstream healthcare devilry system and *vice versa*.

5) This framework is an appropriate guideline to develop and deployment the intelligent mapping process based on InterSystem HealthShare Integration Engine using the available resources (e.g., Cache' Object Script, PowerShell, MySQL Server, DB Visualizer, C#, Java, Excel, etc.)

In this sub-section we are focusing on how to deploy a robust intelligent automate mapping process in a heterogeneous healthcare delivery system. However, still, human input and involvement are very vital in the mapping process because every EPR and other healthcare delivery systems do not have a unique data pattern, format and layout. In general, this process is very time-consuming, and a large chunk of time is spent in contacting suppliers to understand their documents and processes. It is envisaged that the framework discussed in this paper will provide an efficient automated mapping process, which in turn will reduce if not completely remove the need for human interactions.

Conceptual Framework for Auto Mapping

The conceptual framework for the auto-mapping process diagram is shown in Figure 7.

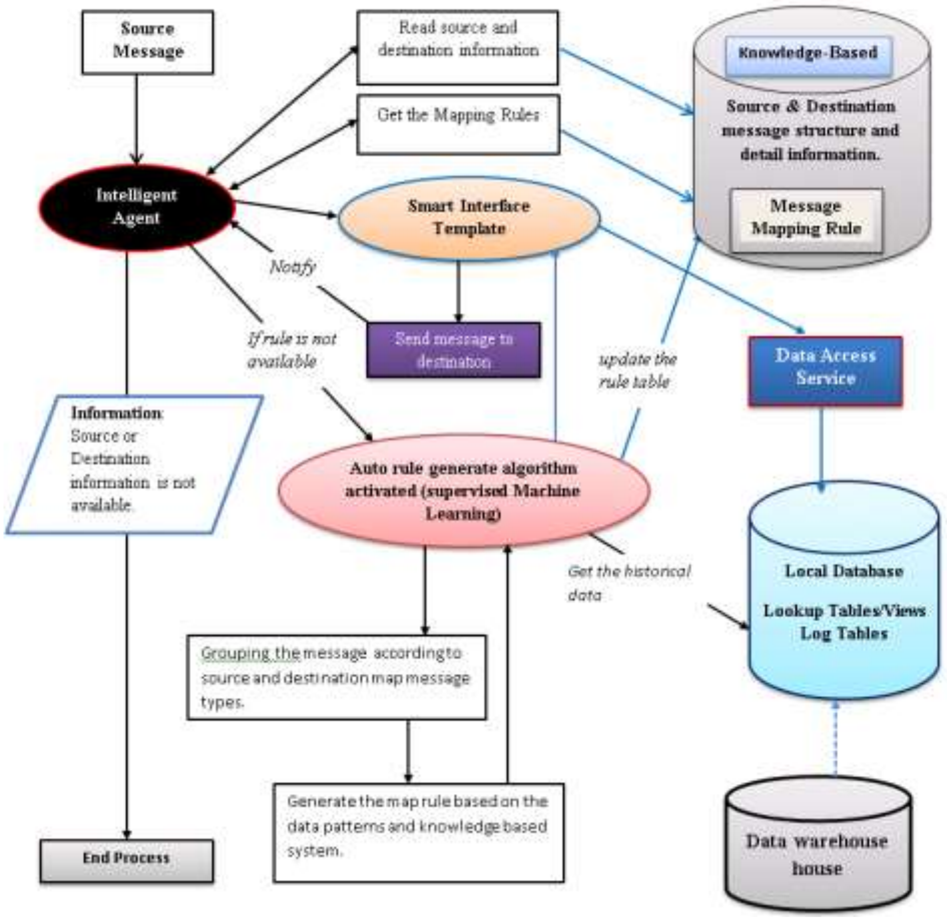

Source Message
Read source and destination information
Get the Mapping Rules
Knowledge-Based
Source & Destination message structure and detail information.
Message Mapping Rule
Intelligent Agent
Smart Interface Template
Notify
If rule is not available
Send message to destination
update the rule table
Data Access Service
Information: Source or Destination information is not available.
Auto rule generate algorithm activated (supervised Machine Learning)
Get the historical data
Local Database
Lookup Tables/Views
Log Tables
Grouping the message according to source and destination map message types.
Generate the map rule based on the data patterns and knowledge based system.
End Process
Data warehouse house

Figure 7. The conceptual framework for data auto-mapping

Knowledge-Based System

The knowledge-based system will capture the required information on the health care delivery system. That means it will consist of all the inbound and outbound system's information basically data sets, types, and architecture. An intelligent agent has the subcomponent knowledge reasoning that is responsible for representing the knowledge.

Rule-Based System

A production rule has been applied to get and set the HL7 message based on the knowledge-based.

Integration Engine

The independent (inbound listeners → Intelligent Agent → outbound listeners) configuration has been setup, which means each inbound source (healthcare information system where HL7 message come from) connected to Intelligent Agent (based on our conceptual framework) connect to Process (code which read the HL7 message and maps with outbound (destination system), and send to the operation (outbound system).

Intelligent Agent

The intelligent agent is an automated process which gets the message from HL7 and performs the following tasks:

1) Check the type of HL7 message (e.g., ADT, ORU, and ORM).
2) Filter the message: process only a specific type of message that comes from inbound listeners.
3) Read the inbound source dataset information
4) Read the outbound source dataset information
5) If both inbound and outbound dataset is similar then, it is easy, just straightway map the message.
6) If there are no similarities, then the rule-based system will support the map of the data.
7) If inbound has the dataset and outbound does not have to appropriate dataset (filed) to feed the information, then we proposed to apply a supervised machine learning technique, which feeds the appropriate dataset (information) into the outbound source and vice versa. The detail more about the technique is as follow:

```
IF Inbound filed is not available on Outbound
     Activate intelligent agent
Else
     Read rule-based system and mapped the data
End if

Intelligent Agent for Missing Field
While Not EOF()
{
   OPEN the historical log tables
   READ the HL7 message (only given message type e.g. ORU, ADT^A01,
ADT^A28 etc.)
   STORE consist records into working memory "in appropriate label"
   READ individual message from working memory and get the value of
     specific segment (the value is missing in the current message)
   Check continuously and find the maximum number of using value
   Obtain the maximum using value into missing field
   Update the rule table
Exit
}
    Read the history HL7 message which retrieve from database and consist
    into working from memory "in appropriate label"
     Read the message and find the inbound filed and track the data
without bound filed
     Check continuously and find the maximum number of using value
     Pass the value for missing field
   }

Supervised Machine Learning Approach
```

Machine learning techniques are being applying in clinical information system, the most useable supervised machine learning algorithms are linear regression, logistic regression, neural network, decision tree, support vector machine, random forest, K-Nearest Neighbors [12]. A Supervised Machine learning technique will have the following functionalities:

Categorization Process: The Categorization process will check the type of message in inbound and check the type of message in outbound. Then, prepare the instruction to prepare the HL7 message into different categorizations and instruct the Record Prepare Process to produce required lists of records. The Categorization Process Algorithm will be created, which separates the message into a different label based on the knowledge-based and provide the instruction to Record Prepare Process to produce the set of datasets. In this case, the label will be defined using the decision tree.

Record Preparing Process: This process is very important that is responsible for searching and collecting and putting together the historical HL7 message based on the instruction of the Categorization process. The Record Prepare Algorithm will be developed to achieve this task. This algorithm will search the HL7 message from the historical file and document log file and store all the output (HL7 message) into working memory.

Searching Process: This process observed all the input data structure from inbound and mapped data into an outbound data structure. If any input data structure is different from the outbound data structure and separates these records by giving an appropriate labeling name and produce the list.

Conclusion

The SIT has been designed and implemented successfully and has been used to develop a new HL7 message interface in a Chelsea and Westminster NHS Foundation Trust. Currently, more than 18 interfaces have been implemented using the SIT and the use of SIT has been proven for ease of use and reliability in the development of new HL7 interfaces in Ensemble/HealthShare environment. The conceptual framework for auto data mapping presented in this paper has not been implemented. The implementation of this conceptual framework will certainly make a significant impact on the health care delivery system.

List of abbreviations

HL7: Health Level Seven
MHS: Message Header
ADT: Patient Admission, Discharge and Transfer Message
PD & PD1: Patient Demography Information
NK1: Next of Kin
OBR: Observation Request
OBX: Observation Result (e.g., Radiology,Pathology)
PV1 & PV2: Patient Visit Information
ORM: Order Message
ORU: Result Message
ORC: Common Order
MRG: Merge Patient
NTE: Notes and Comments
ZO1 & Z02 Flexible Standard
SIT: Smart Interface Template
ADM: Audio Data Mapping
TCP: Transmission Control Protocol
FTP: File Transfer Protocol
SQL: Structured Query Language
KBS: Knowledge-Based System
RBS: Rule-Based System

Acknowledgement

The supports and contributions (regarding to setup an integration engine and coding) that the authors received from Tovio Hartikainen and Alice Shrestha were tremendous and we would like to thank their efforts. We are also thankful for all work done by other integration team members.

The authors wish to express their deep gratitude and sincere thanks to the Tovio Hartikainen and Alice Shrestha for their assistance in setting up the Integration Engine. Their valuable advice towards this research work is highly appreciated. The supports that the authors also received from Usman Bahadur and Alain Nkouatcha were excellent, and we would like to thank their efforts. Author John BABATUNDE particularly contributed to the proofreading and made some contributions in the conceptual framework for the auto data mapping process.

Funding

The authors received no specific funding for this work.

Competing Interests

The authors have declared that no competing interests exist.

References

1. National Alliance for Health Information Technology. "What Is Interoperability?" [Internet] 2005 [cited 2018 December 11]. Available from: www.nahit.org
2. HL7 International [Internet] [cited 2020 June 6] Available from: https://www.hl7.org
3. Khoumbati K, Themistocleous M, Irani Z. Integration Technology Adoption in Healthcare Organisations: A Case for Enterprise Application Integration. Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 2005, pp. 149a-149a, doi: 10.1109/HICSS.2005.331
4. Enterprise Integration Engine for Large Scale Interoperability. [Internet] [cited 2020 June 6] Available from: https://www.slideshare.net/orionhealth/enterprise-integration-engine-for-large-scale-interoperability
5. Healthcare Industry Digital Transformation. [Internet] [cited 2020 June 6] Available from: https://www.oracle.com/industries/healthcare/
6. KLAS Research is a healthcare, found [Internet] [cited 2019 April 11]. Available from: https://klasresearch.com/market-segment/integration-engines/19
7. IPF Open eHealth Integration Platform [Internet] 2018 [cited 2019 April 11]. Available from: http://oehf.github.io/ipf/
8. Bortis G. Experiences with Mirth: an open source health care integration engine. Proceedings of the 30th International Conference on Software Engineering 2008, pp. 649-652. Doi:10.1145/1368088.1368179

9. Al-Sakran HO. Framework Architecture for Improving Healthcare Information Systems Using Agent Technology. International Journal of Managing Information Technology 2015;7(1):17-31. Doi: 10.5121/ijmit.2015.7102
10. InterSystems Documentation Cache & Ensemble 2018.1 [Internet] [cited 2019 April 11] Available from: https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls
11. Integration Engines Tackle Healthcare's Lack of Standards [Internet] 2014 [cited 2018 April 11]. Available from: https://www.informationweek.com/healthcare/clinical-information-systems/integration-engines-tackle-healthcares-lack-of-standards/d/d-id/1269322
12. Pandey SR, Ma J, Lai CH, Regmi PR. A supervised machine learning approach to generate the auto rule for clinical decision support system. Trends Med 2020;20:1-9. Doi:10.15761/TiM.1000232