M S Ramaiah Institute of Technology
(An Autonomous Institute, Affiliated to VTU)

A **Mid Semester Evaluation Report** on

BUILDING A WEB SERVICE FRAMEWORK (STOW-RS) FOR MEDICAL IMAGING

Submitted By:

NEHA PATHAPATI
(USN: 1MS12CS143)
COMPANY: Philips
BUSINESS UNIT: Health Systems

Under the guidance of:

Professor. Srinidhi. H
Assistant Professor
Dept. of Computer Science and Engineering

in partial fulfillment for the award of the degree of
Bachelor of Engineering in Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

M.S.RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)
BANGALORE-560054
www.msrit.edu, **Jan-May 2016**

ABSTRACT

STOW-RS is a restful service that facilitates third party clients to store DICOM (Digital Imaging and Communication in Medicine) files over the PACS (Picture Archival and Communication System).

DICOM is a standard that defines the format of medical images. PACS provides an infrastructure for the electronic management of images and related information. PACS is employed by hospitals all over the world for long-term storage of medical images and also for facilitating sharing of images between physicians/radiologists. The primary characteristics of PACS are:

- They bring an end to the use of film as the legal mode of archiving medical images.
- They enable simultaneous access to images and related information, by authorized personnel at any time and from any location.
- They provide the ability to diagnose a medical condition from these images with a computer i.e. Computer Aided Diagnostics.
- They improve the scanned image quality, and reduce the need for retakes.

The above mentioned characteristics of PACS has led to its widespread deployment in health organizations.

To further benefit from the functionality of PACS, there should be a mechanism that allows third party clients of a health organization to store DICOM images onto its PACS. Such a feature would prove to be useful in the following situations:

- When a patient gets a medical exam done at a particular hospital, but would like a physician in another hospital to examine his medical reports.
- When medical reports have to transported between health organizations for getting a second opinion on a particular medical condition.
- When a particular health organization does not own the necessary equipment to conduct a medical exam and requires a patient to get the exam done from another hospital.
- When a health organization does not house doctors with sufficient expertise to deal with a specific medical condition/disorder.
- When a patient wants all his medical reports to be consolidated and archived at a hospital he most frequently visits, and so on.

These are the situations when allowing an external organization to post DICOM files onto the PACS of a health organization seems convenient. This is precisely what STOW-RS aims to achieve.

INDEX

1. SYNOPSIS

1.1 A BRIEF OVERVIEW

A subset of the systems responsible for the generation, management, storage, and retrieval of medical and patient related information in a healthcare organization are:

Hospital Information system (HIS)

A Hospital Information System (HIS) is primarily concerned with the administrational needs of a hospital. It is a comprehensive, integrated information system designed to manage all the aspects of a hospital's operation, such as medical, administrative, financial, and legal issues and the corresponding processing of services.

With regard to patient care, it is entrusted with the task of patient ID, admitting/registration, billing, and forms the common source of information of a patient's health history. The HIS is responsible for upholding the confidentiality and the security of patient data at all times and strictly controls its access. It also enhances the ability of health care professionals to coordinate care by providing a patient's health information and visit history at the place and time that it is needed. Hence, it can be regarded as the 'master system' in a hospital.

Radiology Information System (RIS)

The Radiology Information System (RIS) is a subset of the HIS. It is a computerized database used by radiology departments to store, manipulate, and distribute patient radiological data and imagery. The system consists of scheduling of patient exams (An exam is a request for an imaging service, for example, a CT scan, a MRI, and so on.) to specific modalities (defined next), execution and completion of exams, storing and distribution of exam reports, and generation of bills.

Modality

The imaging equipment which provides an imaging service is termed as a modality. It can be a Magnetic Resonance, a Computed Tomography, an Ultrasound, an Angiography, and so on. A modality can also be called a Performing Resource.

Picture Archiving and Communication System (PACS)

PACS provides an infrastructure for the electronic management of images and related information. PACS is employed by hospitals all over the world for long-term storage of medical images and also for facilitating sharing of images between physicians/radiologists. The primary characteristics of PACS are:

- They bring an end to the use of film as the legal mode of archiving medical images.

- They enable simultaneous access to images and related information, by authorized personnel at any time and from any location.
- They provide the ability to diagnose a medical condition from these images with a computer i.e. Computer Aided Diagnostics.
- They promote accelerated access to data, increase productivity, and reduce patient stay.
- They improve the scanned image quality, and reduce the need for retakes.

Review Workstation

At the review workstation, the radiologist dictates a report for the patient based on – a review of the patient's history, the patient's radiological history, and the images of the current exam.

The relation between the functioning of these systems is depicted in Figure.1 below:

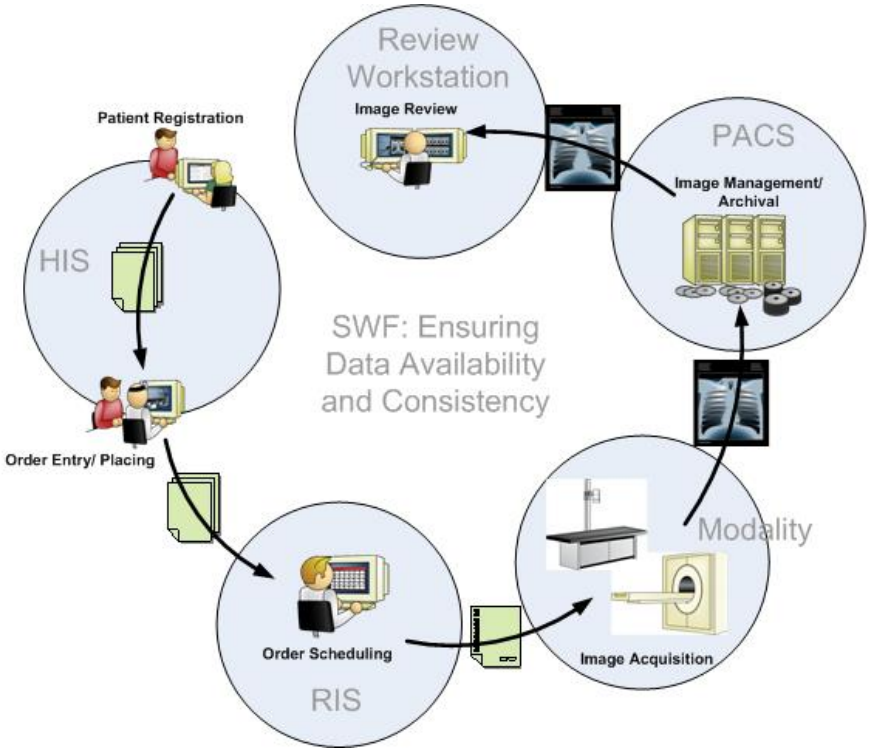


Figure 1: INTERACTION BETWEEN THE SYSTEMS

A patient arrives at a hospital in request for a healthcare service. The patient registers himself at the Hospital Information System (HIS) and is assigned a unique Medical Record Number (MRN). If he/she is a recurring patient to the hospital, his/her MRN is used to identify him. After successful registration, he is examined by a physician who advises him to get an exam done. Consequently, the patient places

an order for an exam/scan at the HIS. The RIS takes charge of scheduling the exam to a specific modality. The images captured by the modality are archived in the PACS. From the PACS, the images/study could be retrieved for viewing, report creation, to study radiological history, and so on, by the radiologist or referring physician.

The interaction between these systems occurs with the assistance of communication protocols and standards. Two of them are:

Digital Imaging and Communications in Medicine (DICOM)

DICOM is a standard for handling, storing, printing, and transmitting information in medical imaging. It includes a file format definition and a network communications protocol. The storage and retrieval of images in PACS occurs in DICOM. It is also the protocol used by a system to interact with the PACS and modality.

Health Level 7 (HL7)

HL7 is the protocol used for the exchange of healthcare information in the application layer of the network stack. HL7 operates predominantly in the administrative domain which includes registration, billing, scheduling, and handling medical records. This is the primary protocol used to communicate with the HIS and RIS.

DICOMweb

It is the web standard for medical imaging. It is primarily a set of RESTful services, enabling web developers to unlock the power of healthcare images using industry-renowned tool-sets. DICOMweb provides web access to underlying imaging systems that speak DICOM through the following services:

- STOW-RS (Store Over the Web - Restful Service)
  This service enables to store a DICOM file in PACS using RESTful Services. For every new DICOM file, this service creates a new resource if it doesn't already exist.
- WADO-RS (Web Access to the DICOM Objects - Restful Service)
  This service enables the retrieval of DICOM files from PACS using RESTful Services.

Both STOW-RS and WADO-RS have multiple action types to store/ retrieve study, series, instance, bulk data, and the meta-data. [A study is set of images associated with an exam. Images in a study can be grouped into a series when - the images obtained are from the same modality and they contain the same series information. Finally, each series is composed of several DICOM instances.]

1.2 PROJECT SCOPE

Consider two healthcare organizations, namely X and Y. A patient arrives at hospital X and is advised by his referring physician to get a CT exam done. However, hospital X does not own the CT modality. Hence, the patient is forced to get his CT scan from another hospital owning CT, say Y. Now, after successful completion of CT exam at Y, the radiologist at Y must be able to post the patient's medical images to his host hospital, that is, X. This is critical considering that the referring physician at X, who had asked for the scan in the first place, must examine the images and advise further treatment. Additionally, the patient's scan images must be assembled with his other medical images/information history for further need.

The goal of our project is to build a STOW – RS service to facilitate third party clients to post DICOM files (all scans are in compliance with the DICOM standard) to the PACS.

A block diagram of the STOW - RS service is shown in Figure.2:

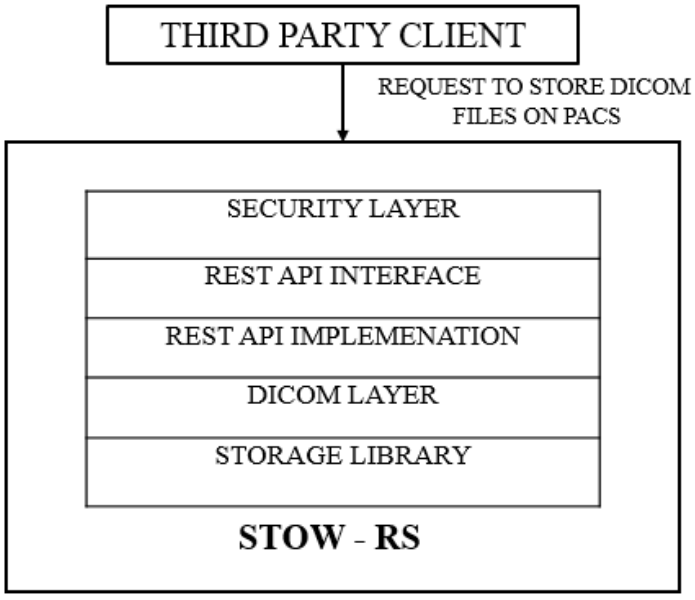


Figure 2: ARCHITECTURE OF STOW - RS

A brief overview of the architecture of STOW is as follows:

- The third – party client is the entity external to the healthcare organization that wishes to store DICOM files from the PACS.
- The Security Layer is responsible for – authenticating, that is, verifying the identity of the external client wanting to store a DICOM file.
- The REST Interface is the REST API used to enable the client to access STOW – RS service.

- The REST Implementation layer refers to the actual implementation of the REST API interface.
- The DICOM Library is used to validate the DICOM information being pushed into PACS by the client.
- The Storage Library is responsible for the physical storing of DICOM files into the PACS.

Thus, a brief overview of STOW-RS and its significance in the medical industry is described.

2. PROJECT PLAN

2.1 PROCESS MODEL

Scaled Agile Framework (SAFe) is the process model which will be followed in the project execution.

The Scaled Agile Framework provides a recipe for adopting the Agile model at an enterprise scale. It helps organizations to scale agile from the team to the program to the portfolio level.

The PORTFOLIO LEVEL comprises of the highest and most strategic layer where programs are aligned to the company's business strategy and investment approach.

At the PROGRAM LEVEL,

- SAFe uses the concept of an Agile Release Train (ART) or Train to deliver the expected value.
- An ART can also be called a Program Increment (PI), and spans for a period of 3 months.
- It is the primary vehicle for value delivery at the program level.
- A Release Planning Meeting (RPM) is conducted on the first three days of each PI.
  It helps to draw up the work to be done in the next 3 months.
- RPM is a form of collaborative, deliberate discovery workshop to determine the initial set of User Stories that will be worked on by the teams. These User Stories are then prioritized into releases.
- Additionally, the tasks undertaken in a RPM include – assigning story points (amount of effort required, can be 1,3,5,7,11,13,15, or 20, indicating 1 hour, 2 hour, etc.) to individual user stories, and determining - the features that will be delivered, the risks involved, the internal dependencies, the external dependencies, the defects, the technical/architectural debts, and the spikes (user stories involving a high degree of uncertainty).
- All the user stories are then wisely distributed into sprints (10 working days – 2 week periods).
- Between 5 and 10 teams work together on a train to deliver business value.
- Every 10 weeks (5 iterations) a train delivers a Potentially Shippable Increment (PSI). A release demo, inspection, and adaptation sessions are conducted.
- After successful delivery of a PSI, planning for the next PSI commences.
- PSIs provide a steady cadence (flow) for the development cycle.
- In SAFe the actual market release is often separated from this cadence. This allows product management to decide when the actual market releases are based on business timing rather than development cycles.

At the TEAM LEVEL,

- SAFE uses a hybrid of Scrum and XP engineering practices.
- All teams operate on the same cadence, i.e. the sprint duration is the same across all teams.
- The teams synchronize their sprint start and end dates for better co-ordination between teams.
- Potentially Shippable Increments (PSI) of a product will only be possible after a number of sprints are completed.
- Define/Build/Test (DBT) teams deliver working, fully tested software every two weeks. There are five to nine members of each team.
- Time is divided into a set of fix length sprints (or iterations) of duration 2 weeks.
- At the start of each sprint, a sprint planning is done. Here, the team members review the user stories assigned to various sprints in the RPM and commit to the user stories they will execute.

2.2 RISK IDENTIFICATION AND RISK MITIGATION STEPS

Risk is an expectation of loss, a potential problem that may or may not occur in the future. It is generally caused due to lack of information, control or time. A possibility of suffering from loss in software development process is called a software risk. Loss can be anything, increase in production cost, development of poor quality software, not being able to complete the project on time. Software risk exists because the future is uncertain and there are many known and unknown things that cannot be incorporated in the project plan.

Risk identification is the process of determining risks that could potentially prevent the program, enterprise, or investment from achieving its objectives. It includes documenting and communicating the concern.

Risk Mitigation, within the context of a project, can be defined as a measure or set of measures taken by a project manager to reduce or eliminate the risks associated with a project. Risks can be of various types such as technical risks, monetary risks and scheduling-based risks. The top 5 risks of our project and their mitigation steps are enlisted in Table. 1 below:

Table 1: RISKS AND THEIR MITIGATION STEPS

Sl. No	RISK	RISK MITIGATION
1.	Difficulties which may arise in the integration	To employ the practice of Continuous Integration. This ensures that the teams are able to produce the required value: the team

	of the multiple STOW layers.	increment. The team should integrate and test its work at least once in a PI. To emphasize on the importance of Continuous Integration, there should be a user story and story points dedicated to integration in the RPM, just like for the other development activities.
2.	Lack of expertise with new technologies leading to decelerated execution of tasks.	Training sessions have to be conducted to increase the proficiency of team members with the new technologies being used like HP Fortify Scan. Also, always available online/offline help must be arranged.
3.	Evolving design of STOW – RS service.	Create user stories to accommodate design improvements/modifications, their implementation, and their testing in all sprints. Brainstorming is also an effective strategy.
4.	Maintaining security.	A foolproof mechanism to authenticate and authorize the third party client posting DICOM instances using the STOW – RS service.
5.	Finally, the possibility of risks not being identified and mitigated at the appropriate levels is itself a massive risk! Also, the tasks of risk mitigations should be explicitly assigned to project team members to assign responsibility.	Code level risks must be resolved at the team level. Risks that may affect the entire Agile Release Train (ART) should be dealt with at the program level. This prevents the same risk from being encountered twice or getting escalated. ROAM board is an efficient tool to mitigate this risk. It must be used during PI planning to identify and analyze risks and issues. To make sure all the risks are covered, the goal of this technique is to Resolve, Own, Accept or Mitigate all risks (ROAM). Hence, potential issues at the Team-level ROAM board don't get transferred onto the Program-level ROAM board. Also, the ROAM tool ensures that all risks are covered i.e. after a ROAM session, responsibility is assigned to solve these issues.

2.3 PROJECT SCHEDULE

A Gantt chart is a type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. A Gantt chart, commonly used

in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

The schedule of the project is summarized in Figure. 3:

Building a web service framework (STOW-RS) for medical imaging. 23 Feb, 2016

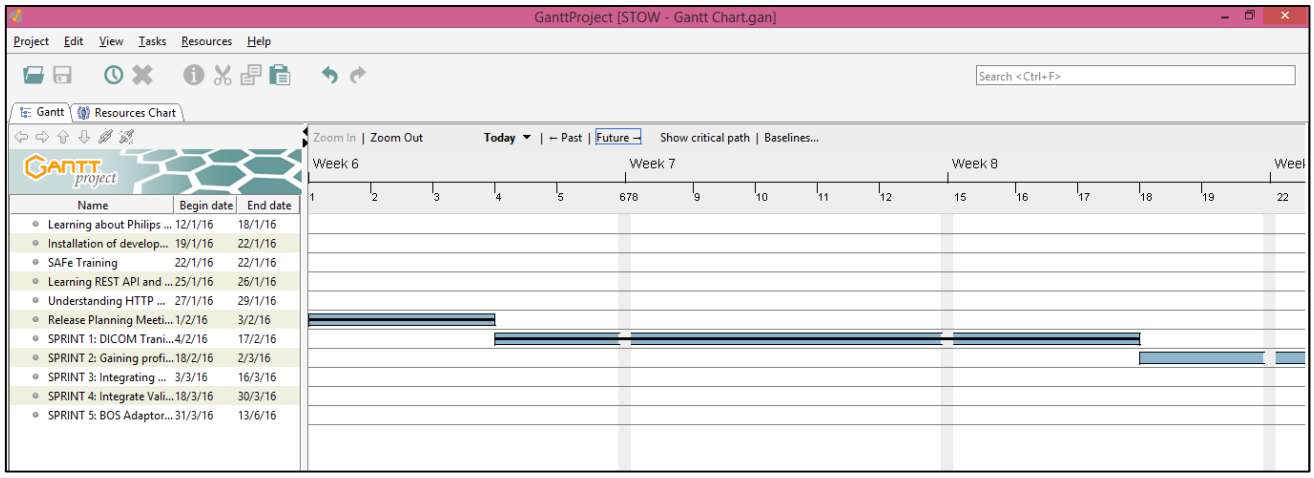
2

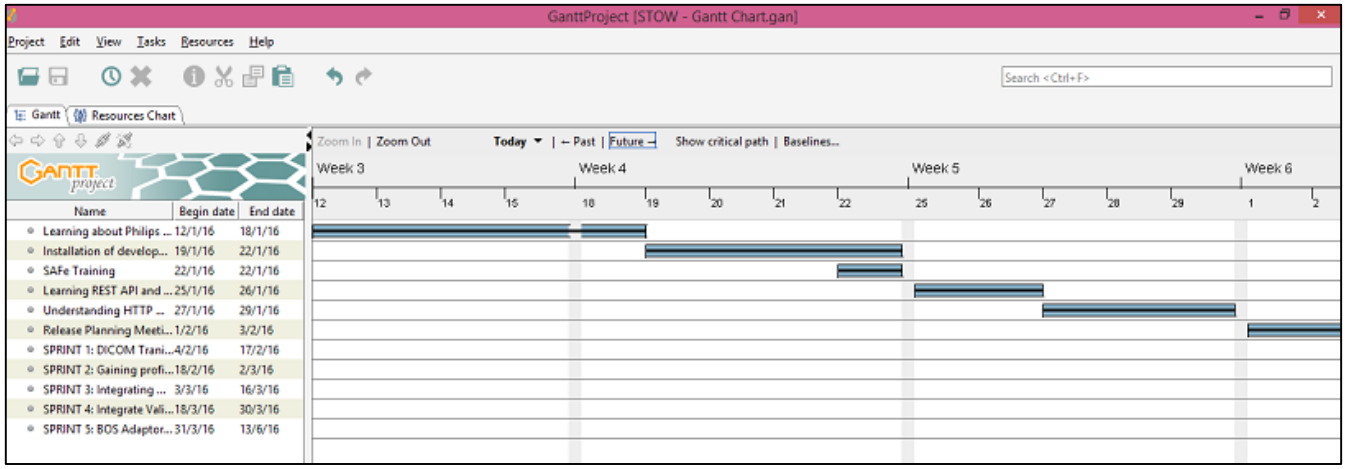
Tasks

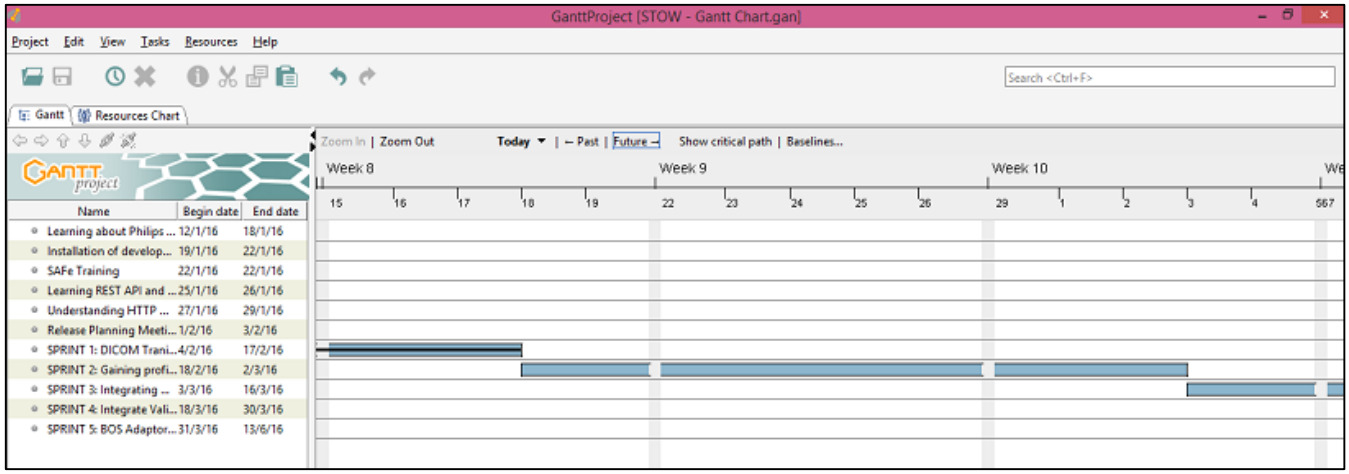
Name	Begin date	End date
Learning about Philips products, DICOM structure, and C#.	12/1/16	18/1/16
Installation of development environment.	19/1/16	22/1/16
SAFe Training	22/1/16	22/1/16
Learning REST API and building a prototype.	25/1/16	26/1/16
Understanding HTTP multipart messages and STOW-RS and WADO-RS specification in the DICOM standard.	27/1/16	29/1/16
Release Planning Meeting for Increment 1.	1/2/16	3/2/16
SPRINT 1: DICOM Tranining, JMeter Training, Learning and building a sample MVC application, Hosting an application over IIS, Understanding STOW-RS and WADO-RS specification.	4/2/16	17/2/16
SPRINT 2: Gaining proficiency with using the MERGE DICOM Toolkit, A C# code for DICOM Validation, A C# code for DICOM validation using MERGE DICOM Toolkit, Creating custom logs, C# code for UID validation.	18/2/16	2/3/16
SPRINT 3: Integrating modules from Sprint 2 to STOW-RS, Code for Stentor Logging, Exploring AE_Titles and their implementation using MERGE DICOM Toolkit in STOW-RS module, Understanding and building a security framework, REST APIs.	3/3/16	16/3/16
SPRINT 4: Integrate Validation, Logging, and Security modules into WADO-RS, Optimize and clean up STOW-RS code, Creation of CDR Extension.	18/3/16	30/3/16
SPRINT 5: BOS Adaptor, Storage Library, Support for conversion of metadata based on incoming request.	31/3/16	13/6/16

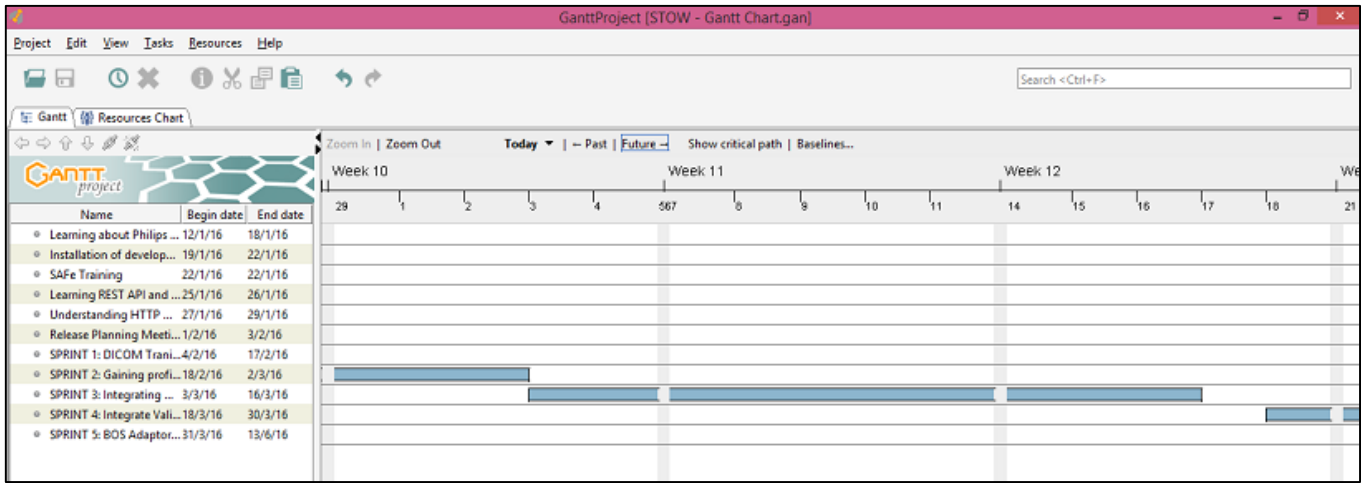
Figure 3: SUMMARY SCHEDULE

The schedule of the project is depicted using the Gantt Chart below (Figure. 4):




GanttProject [STOW - Gantt Chart.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out Today | ← Past | Future → Show critical path | Baselines...
Week 3 Week 4 Week 5 Week 6
Name Begin date End date
Learning about Philips ... 12/1/16 18/1/16
Installation of develop... 19/1/16 22/1/16
SAFe Training 22/1/16 22/1/16
Learning REST API and ... 25/1/16 26/1/16
Understanding HTTP ... 27/1/16 29/1/16
Release Planning Meeti... 1/2/16 3/2/16
SPRINT 1: DICOM Trani... 4/2/16 17/2/16
SPRINT 2: Gaining profi... 18/2/16 2/3/16
SPRINT 3: Integrating ... 3/3/16 16/3/16
SPRINT 4: Integrate Vali... 18/3/16 30/3/16
SPRINT 5: BOS Adaptor... 31/3/16 13/6/16


GanttProject [STOW - Gantt Chart.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out Today | ← Past | Future → Show critical path | Baselines...
Week 8 Week 9 Week 10
Name Begin date End date
Learning about Philips ... 12/1/16 18/1/16
Installation of develop... 19/1/16 22/1/16
SAFe Training 22/1/16 22/1/16
Learning REST API and ... 25/1/16 26/1/16
Understanding HTTP ... 27/1/16 29/1/16
Release Planning Meeti... 1/2/16 3/2/16
SPRINT 1: DICOM Trani... 4/2/16 17/2/16
SPRINT 2: Gaining profi... 18/2/16 2/3/16
SPRINT 3: Integrating ... 3/3/16 16/3/16
SPRINT 4: Integrate Vali... 18/3/16 30/3/16
SPRINT 5: BOS Adaptor... 31/3/16 13/6/16


GanttProject [STOW - Gantt Chart.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out Today | ← Past | Future → Show critical path | Baselines...
Week 10 Week 11 Week 12
Name Begin date End date
Learning about Philips ... 12/1/16 18/1/16
Installation of develop... 19/1/16 22/1/16
SAFe Training 22/1/16 22/1/16
Learning REST API and ... 25/1/16 26/1/16
Understanding HTTP ... 27/1/16 29/1/16
Release Planning Meeti... 1/2/16 3/2/16
SPRINT 1: DICOM Trani... 4/2/16 17/2/16
SPRINT 2: Gaining profi... 18/2/16 2/3/16
SPRINT 3: Integrating ... 3/3/16 16/3/16
SPRINT 4: Integrate Vali... 18/3/16 30/3/16
SPRINT 5: BOS Adaptor... 31/3/16 13/6/16

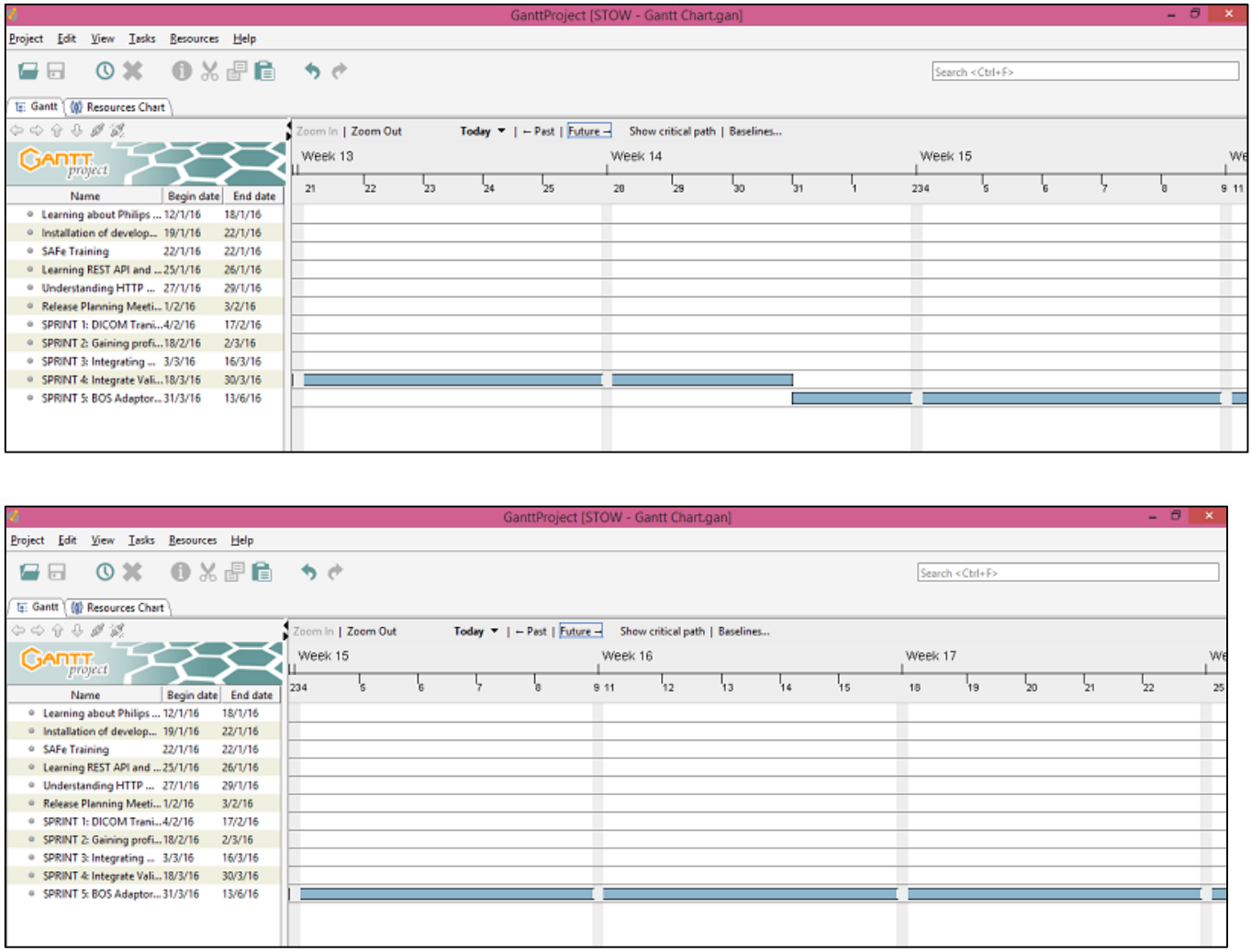


Figure 4: GANTT CHART

The project will reach completion by June 14th. Since the Scaled Agile model is employed, the subsequent tasks to be undertaken will be determined in the RPM of the next PI.

3. SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Software Requirements Specification (SRS) is a requirement specification for a software system. It provides description of the behavior of a system to be developed and may include a set of use cases that describe interactions between the users and the software. In addition to a description of the software functions, the SRS also contain non-functional requirements. Non-functional requirements impose constraints on the design, implementation and quality standards.

The software requirement specification document enlists all necessary requirements for project development.

3.1 PRODUCT OVERVIEW

The goal of our project is to build a STOW – RS (Store Over the Web – Restful Service) service to facilitate third party clients to post DICOM (Digital Imaging and Communication in Medicine) files (all scans are in compliance with the DICOM standard) to the PACS (Picture Archival and Communication System).

A block diagram of the STOW - RS service is shown in Figure. 5:

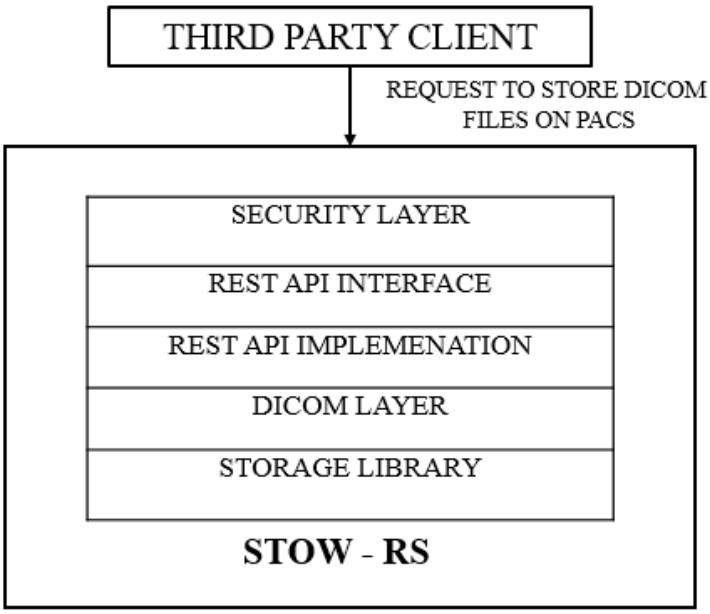


Figure 5: ARCHITECTURE OF STOW – RS

A brief overview of the architecture of STOW is as follows:

- The third – party client is the entity external to the healthcare organization that wishes to store DICOM files from the PACS.

- The Security Layer is responsible for – authenticating, that is, verifying the identity of the external client wanting to store a DICOM file.
- The REST Interface is the REST API used to enable the client to access STOW – RS service.
- The REST Implementation layer refers to the actual implementation of the REST API interface.
- The DICOM Library is used to validate the DICOM information being pushed into PACS by the client.
- The Storage Library is responsible for the physical storing of DICOM files into the PACS.

This project aims to build a web-based service for accessing and presenting DICOM supported as per the DICOM. (Defined by National Electrical Manufacturers Association (NEMA)).

3.2 EXTERNAL INTERFACE REQUIREMENTS

3.2.1 User Interface

The STOW-RS service can be accessed by the third party clients using a simple REST API. A RESTful Web service is a Web service implemented using REST architecture and HTTP.

3.2.2 Hardware Interface

This information is proprietary to Philips. Hence, it cannot be disclosed.

3.2.3 Software Interface

This information is proprietary to Philips. Hence, it cannot be disclosed.

3.2.4 Communication Interface

Communication between the third party client and the STOW-RS service occurs by HTTP messages.

The DICOM file to be stored is encapsulated into a HTTP multipart request message and sent across to the server (STOW – RS service). All request messages are HTTP/1.1 multipart messages. For DICOM instances being sent by the client, the HTTP Request field Content-Type is set as: **Content-Type: application/dicom**. The DICOM Request Message has a multipart body, where: **Content-Type: multipart/related; type=application/dicom; boundary = {Message Boundary}.** The multipart request body contains every instance to be stored. Each instance is in a separate part of the multipart body. Each part in the multipart body represents a DICOM SOP Instance with the following HTTP headers: Content-Type: application/dicom.

The STOW RESTful Service will return a HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module - If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success “response code. If there are instance specific errors, the response code shall be a 409 and the response payload shall contain the Store Instances Response Module, which contains additional information regarding instance errors. In all other conditions, the RESTful Service shall return an "HTTP 202 - Accepted" response code. The response payload may contain a Store Instances Response Module, which specifies additional information regarding instance warnings or failures.

3.3 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs and the behavior associated with it. Functional requirements may be technical details, data manipulation, processing and other specific functionality that define the needs a system should accomplish. Functional requirements specify the particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system. The functional requirements of STOW-RS are as follows:

3.3.1 Accept STOW requests from third party clients

All request messages are HTTP/1.1 multipart messages. HTTP Request field Content-Type is used in the header lines by the client in an HTTP/1.1 transaction to indicate the type of data being sent to the Service. The request message can be DICOM or metadata and bulk data depending on the "Content-Type", and is encapsulated in a multipart request body.

The STOW - RS service must be able to process storage requests with the following HTTP header format:

- Resource
  {SERVICE}/studies[/{StudyInstanceUID}]
  Where,

{SERVICE} is the base URL for the service. This may be a combination of scheme (either HTTP or HTTPS), host, port, and application;

{StudyInstanceUID} (Optional) is the study instance UID for a single study. If not specified, instances can be from multiple studies. If specified, all instances shall be from that study, and instances not matching the StudyInstanceUID shall be rejected.

- Method

  POST

- Headers

  Content-Type - The representation scheme being posted to the RESTful service. The types allowed for this request header are as follows:

  - Multipart/related; type=application/dicom; boundary = {MessageBoundary}
  - Multipart/related; type=application/dicom+xml; boundary = {MessageBoundary}.

- The DICOM Request Message has a multipart body.
  - Content-Type:

    Multipart/related; type=application/dicom; boundary = {MessageBoundary}.

    Specifies that the post is XML metadata and bulk data.
  - The multipart request body contains every instance to be stored. Each instance is in a separate part of the multipart body.
  - Each part in the multipart body represents a DICOM SOP Instance with the following HTTP headers:
  - Content-Type: application/dicom.
- The XML Metadata and Bulk Data Request Message has a multipart body.
  - Content-Type:

    Multipart/related; type=application/dicom+xml; boundary = {MessageBoundary}
  - The multipart request body contains all the metadata and bulk data to be stored.
  - If the number of bulk data parts does not correspond to the number of unique BulkDataURIs in the metadata then the entire message is invalid and will generate an error status line.
  - Each body part is either DICOM XML metadata or a bulk data item from a SOP Instance sent as part of the Store operation.

The first part of the multipart message must be XML metadata and contains the following HTTP headers:

- Content-Type: application/dicom+xml; transfer-syntax = {TransferSyntaxUID}

Subsequent items will contain the following HTTP headers (order is not guaranteed):

- Content-Type: application/dicom+xml; transfer-syntax={TransferSyntaxUID}
  An uncompressed bulk data element encoded in Little Endian binary format with the following headers:
- Content-Type: application/octet-stream
  Content-Location: {BulkDataURI}
  A compressed pixel data object from a SOP Instance in the Study with the following headers:
- Content-Type: {MediaType}
  Content-Location: {BulkDataURI}
  Metadata and its associated bulk data shall always be sent in the same POST request.

3.3.2 Validation of STOW request and DICOM file header

Firstly, the format of Study Instance ID in the STOW – RS request URL should be validated. Secondly, the Study Instance ID of the STOW- RS request URL should be checked for consistency with the Study Instance ID field of the DICOM file header. Thirdly, the header of the DICOM file, consisting of the 128 bytes of preamble, 4 bytes of prefix, and the data set (comprising of data elements) should be checked.

3.3.3 Ensuring security

Mechanism for storing a DICOM file will be through the HTTPS protocol, using DICOM UIDs for Study Instance level. For enabling the HTTPS protocol, HTTP BASIC Authorization over SSL and SSL Client Certificates must be used. Authentication and authorization of the client attempting to store DICOM files on the STOW – RS service must be implemented.

3.3.4 Storing the files

The STOW – RS service creates new resources for the given DICOM instances on the Server or appends them to an existing resource on the Server. One or more DICOM instances associated

with one or more study instance unique identifiers (SUID) are stored in the appropriate file store location.

3.3.5 Issuing a HTTP Response Message

The RESTful Service shall return an HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module.

If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success "response code.

If the status for all instances included in the POST request is Failure, the RESTful Service shall return an appropriate failure status line with a response code. If there are instance specific errors, the response code shall be a 409 and the response payload shall contain the Store Instances Response Module, which contains additional information regarding instance errors.

In all other conditions, the RESTful Service shall return an "HTTP 202 - Accepted" response code. The response payload may contain Store Instances Response Module, which specifies additional information regarding instance warnings or failures. The standard HTTP response codes are listed in Table .2:

Service Status	HTTP/1.1 Status Codes	STOW-RS Description
Failure	400 - Bad Request	This indicates that the STOW-RS Service was unable to store any instances due to bad syntax.
	401 - Unauthorized	This indicates that the STOW-RS Service refused to create or append any instances because the client is not authorized.
	403 - Forbidden	This indicates that the STOW-RS Service understood the request, but is refusing to fulfill it (e.g., an authorized user with insufficient privileges).
	409 - Conflict	This indicates that the STOW-RS Service request was formed correctly but the service was unable to store any instances due to a conflict in the request (e.g., unsupported SOP Class or StudyInstanceUID mismatch). This may also be used to indicate that a STOW-RS Service was unable to store any instances for a mixture of reasons. Additional information regarding the instance errors can be found in the XML response message body.
	415 - Unsupported Media Type	This indicates that the STOW-RS Service does not support the Content-Type specified in the storage request (e.g., the service does not support JSON metadata).
	503 - Busy	This indicates that the STOW-RS Service was unable to store any instances because it was out of resources.
Warning	202 - Accepted	This indicates that the STOW-RS Service stored some of the instances but warnings or failures exist for others. Additional information regarding this error can be found in the XML response message body.
Success	200 - OK	This indicates that the STOW-RS Service successfully stored all the instances.

Table 2: HTTP/1.1 STANDARD RESPONSE CODES

3.4 SOFTWARE SYSTEM ATTRIBUTES

The software system attributes of STOW-RS are:

3.4.1 Reliability

The STOW – RS service must create new resources or append to existing resources the incoming DICOM instances. Storing of the same instance should not occur multiple times. The DICOM files must be accepted only from an authorized and an authenticated client. The request URL and the DICOM file header must by validated. The adequate HTTP response message must be returned to the client after completion of the STOW operation. All operations must be logged for further reference. The service must meet all of the functional requirements without any unexpected behavior. It should be tested and debugged completely. All exceptions should be well handled.

3.4.2 Availability

The STOW – RS service should always be up and running. An authorized client must be able to post DICOM files using STOW – RS service at any time instance.

3.4.3 Security

Storing of DICOM files will be through HTTPS protocol, using DICOM UIDs for Study Instance level. The client must be authorized and authenticated before he is allowed to post a DICOM file using the STOW – RS service. It should also not be possible for the DICOM files of third party clients stored to be accessed by unauthorized personnel's within our organization. Security is achieved through the use of SSL client and server certificates.

3.4.4 Portability

The STOW-RS service should be able to function upon any CPU configuration.

3.4.5 Maintainability

The service should be written clearly, concisely, and optimally. The code must be well documented with explanatory comment lines. Modularity of code has to be ensured for easy and accelerated version upgrades and changes.

3.4.6 Performance

The STOW – RS service must run on Windows Server for best performance. The entire sequence of operations – receiving of DICOM files from clients, validation, security check, storing files, logging, and sending back an appropriate HTTP response message should be completed within a few milliseconds.

3.5 SYSTEM REQUIREMENTS

At the client side, the system requirements are: Workstation clam machines, 8GB RAM, 200 GB hard disk. At the server side, 32GB of RAM and 17 GB of hard disk is needed. The software requirements include .NET Framework 4.5, Internet Information Server (IIS), licenses for DICOM toolkit. For storage, NFS, SAN, and/or NAS are required.

3.6 PERFORMANCE REQUIREMENT

Performance benchmarking is yet to be done**.** However, the service is required to function within very few milliseconds. JMeter is the tool that will be used to monitor the load performance of STOW-RS.

3.7 DATABASE REQUIREMENT

Not applicable. We will not be using a database.

3.8 DESIGN CONSTRAINTS

Some of the dll's used are in 32-bit, and hence they have to be converted to 64-bit.

3.9 OTHER REQUIREMENTS

Non-functional requirements are requirements that specifies the criteria which can be used to judge the operation of a system rather than specific behavior of the system. The non-functional requirements required for the project are as listed below.

3.9.1 Logging

All errors, warnings, exceptions, and informational messages have to be logged in a custom location for further reference.

3.9.2 Auditing

A security-relevant chronological record that provides documentary evidence of the sequence of activities that have occurred needs to be maintained. This record, called the audit trail can be used to for reexamination of events occurred and for security purposes.

4. LITERATURE SURVEY

4.1 INTRODUCTION

STOW – RS (Store Over the Web – Restful Service) is a service to facilitate third party clients to post DICOM (Digital Imaging and Communication in Medicine) files (all scans are in compliance with the DICOM standard) to the PACS (Picture Archival and Communication System).

Most medical organizations, be it hospitals, diagnostic centers, etc. cannot own modalities (imaging equipment) for all the innumerable types of medical examinations. Also, in order to undergo the medical exam recommended by his general physician, it is inconvenient for a patient to again consult a physician in different medical organization that operates the modality required for his medical exam. The patient should be able to take a quick exam at the foreign hospital and have his reports available to his general physician without him performing this task manually. This way he saves valuable time, avoids having to pay consultation fees twice fold, and can be assured of getting diagnosed from his physician who is well versed with his medical history. Also, all his medical exams reports can be consolidated and stored at a single location – his host hospital, for a quick reference in the future. Thus, there is a need for a hassle free mechanism for a patient to undergo medical examinations (such as CT, MRI, etc.) at medical organizations other than the one hosting his referring physician and have his reports available to his physician. This is what STOW-RS aims to achieve.

The ultimate and only reference for STOW-RS is the Digital Imaging and Communication in Medicine (DICOM) Standard. This standard was established by the National Electrical Manufacturers Association (NEMA) which recognized the emerging need for a standard method for transferring images and associated information between devices manufactured by various vendors. The DICOM Standard is an evolving standard and it is maintained in accordance with the Procedures of the DICOM Standards Committee. Proposals for enhancements are forthcoming from the DICOM Committee member organizations based on input from users of the Standard. These proposals are considered for inclusion in future editions of the Standard. A requirement in updating the Standard is to maintain effective compatibility with previous editions.

The DICOM standard specifies a web-based service i.e. STOW-RS for storing DICOM persistent objects (e.g., images, medical imaging reports). This service is intended for distribution of results and images to healthcare professionals. It provides a simple mechanism for storing a DICOM persistent object, through HTTP/HTTPS protocol, using DICOM UIDs (Unique Identifiers). Data may be retrieved either in a presentation-ready form as specified by the requester (e.g., JPEG or GIF) or in a native DICOM format

using a web-based service called WADO-RS. This standard relates only to DICOM persistent objects (not to other DICOM objects or to non-DICOM objects). Access control beyond the security mechanisms generally available to web applications is outside the scope of this standard.

The 2016 release of the DICOM standard is being referenced for STOW-RS. The DICOM standard is a multipart document. Specifically, the part 3.18 of the standard provides implementation details for STOW-RS. The releases of the DICOM standard are open source.

The definitions of some important terms are a below:

- Digital Imaging and Communications in Medicine (DICOM)
  DICOM is a standard for handling, storing, printing, and transmitting information in medical imaging. It includes a file format definition and a network communications protocol. The storage and retrieval of images in PACS occurs in DICOM. It is also the protocol used by a system to interact with the PACS and modality.
- A study is set of images associated with an exam. Images in a study can be grouped into a series when - the images obtained are from the same modality and they contain the same series information. Finally, each series is composed of several DICOM instances.
- Unique Identifiers are the identifiers used to identify study, series, and images.
- STOW-RS (Store Over the Web - Restful Service)
- This service enables to store a DICOM file in PACS using RESTful Services. For every new DICOM file, this service creates a new resource if it doesn't already exist.

4.2 DEVELOPMENT OF STOW-RS

Consider two healthcare organizations, namely X and Y. A patient arrives at hospital X and is advised by his referring physician to get a CT exam done. However, hospital X does not own the CT modality. Hence, the patient is forced to get his CT scan from another hospital owning CT, say Y.

Generally, this is what happens in such a scenario - Now, after the patient gets the exam conducted at Y, he now has to physically collect the test report. Then, he has to carry it back to his host hospital to be treated by his physician - This is critical considering that the referring physician at X, who had asked for the scan in the first place, must examine the images and advise further treatment. Sometimes, the patient is forced to get his reports checked by a physician in Y, thereby doubling his medical fees. Additionally, there is the hazard of all his medical reports not being systematically assembled at X.

In contrast, with the development of STOW-RS, after successful completion of CT exam at Y, the radiologist at Y must be able to post the patient's medical images to his host hospital, that is, X, and he can get diagnosed by his physician who his aware of his medical history. Additionally, the patient's scan images must be assembled with his other medical images/information history for further need.
Hence, the goal of our project is to build a STOW – RS service to facilitate third party clients to post DICOM files (all scans are in compliance with the DICOM standard) to the PACS.
The only reference for STOW-RS is the DICOM NEMA standard.

The DICOM standard defines STOW-RS as a restful web service that creates new resources for the input SOP Instances on the Server or appends them to the existing resources on the Server.

All request messages are defined to be HTTP/1.1 multipart messages.

The organization of SOP Instances into message parts is said to be dependent on whether the SOP Instances are structured as binary instances, or metadata and bulk data.

The HTTP Request field Content-Type is used in the header lines by the client in an HTTP/1.1 transaction to indicate the type of data being sent to the STOW-RS. All lines are RFC822 or RFC7230 format headers. All HTTP header fields whose use is not defined by STOW-RS are assumed to have the meaning defined by the HTTP standard.

The premise gives a high level picture of the type of service STOW-RS has to be, the type of request messages for the service, and how the store objects have to be organized into HTTP request messages sent by the client.

The DICOM standard includes details of the following:

The URL format to be used by the client to access the STOW-RS service.

- The header format of the Http Request Message – the method (POST), and the content types for storing binary instances, XML metadata and bulk data, and DICOM JSON metadata and bulk data.
- The format of the DICOM request message body for DICOM instances, JSON, and XML objects including the content-type, and the organization of objects into HTTP request messages.
- The response (i.e. the response status line and the response body) of the RESTful Service as a HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances.

- A clear tabulation of HTTP response codes to be issued by STOW-RS.
- Sample HTTP response status lines and response message bodies.

However, the accurate implementation methodology for STOW-RS has not been explicitly stated in the DICOM standard. Specifically, some of the critical aspects that are vaguely described in the DICOM standard:

- The security of STOW-RS.
- A concise mechanism for validation of incoming files from third party clients.
- The mechanism to validate and dynamically update the record of trusted AE_Titles (AE_Titles are used to identity the modality sending the files to STOW-RS).
- Precise implementation details of storing DICOM, Dicom + xml, and JSON objects.

4.3 CONCLUSION

The DICOM standard is the only reference for STOW-RS. Hence, it has been studied and followed rigorously during the STOW-RS implementation. The aspects not clearly described in the standard are implemented as per the team's best expertise. Hence, it could be improved in the areas mentioned earlier thereby to form a more definitive guide.

4.4 REFERENCE

Digital Imaging and Communication (DICOM) standard 2015 published by National Electrical Manufacturers Association.

5. DESIGN

5.1 INTRODUCTION

The goal of our project is to build a STOW – RS (Store Over the Web – Restful Service) service to facilitate third party clients to post DICOM (Digital Imaging and Communication in Medicine) files (all scans are in compliance with the DICOM standard) to the PACS (Picture Archival and Communication System).

This Design document begins with the description of the modules of STOW-RS, followed by its architectural design, and finally the data flow, the sequence, and the class diagrams to illustrate STOW-RS.

NUMBER OF MODULES:

There are 5 modules in STOW-RS.

DESCRIPTION OF MODULES:

1). **Accept STOW requests from third party clients**

The client requests are received through the REST API. All request messages are HTTP/1.1 multipart messages. HTTP Request field Content-Type is used in the header lines by the client in an HTTP/1.1 transaction to indicate the type of data being sent to the Service. The request message can be DICOM or metadata and bulk data depending on the "Content-Type", and is encapsulated in a multipart request body.

2). **Validation of STOW request and DICOM file header**

Firstly, the format of Study Instance ID in the STOW – RS request URL should be validated. Secondly, the Study Instance ID of the STOW- RS request URL should be checked for consistency with the Study Instance ID field of the DICOM file header. Thirdly, the header of the DICOM file, consisting of the 128 bytes of preamble, 4 bytes of prefix, and the data set (comprising of data elements) should be checked.

3). **Ensuring security**

Mechanism for storing a DICOM file will be through the HTTPS protocol, using DICOM UIDs for Study Instance level. For enabling the HTTPS protocol, HTTP BASIC Authorization over SSL and SSL Client Certificates must be used. Authentication and authorization of the client attempting to store DICOM files on the STOW – RS service must be implemented.

4). Storing the files

The STOW – RS service creates new resources for the given DICOM instances on the Server or appends them to an existing resource on the Server. One or more DICOM instances associated with one or more study instance unique identifiers (SUID) are stored in the appropriate file store location. The Storage Library is responsible for this action.

5). Issuing a HTTP Response Message

The RESTful Service shall return an HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module.

If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success "response code.

If the status for all instances included in the POST request is Failure, the RESTful Service shall return an appropriate failure status line with a response code. If there are instance specific errors, the response code shall be a 409 and the response payload shall contain the Store Instances Response Module, which contains additional information regarding instance errors.

In all other conditions, the RESTful Service shall return an "HTTP 202 - Accepted" response code. The response payload may contain Store Instances Response Module, which specifies additional information regarding instance warnings or failures.

ALGORITHM DESIGN:

1. Initialization of DICOM library.
2. Initialization of logging mechanism.
3. Host STOW-RS service on port 443 to listen for incoming STOW-RS requests.
4. A STOW-RS request arrives.
5. Validate format of Study Instance ID from STOW-RS request URL. If format is invalid, abort STOW-RS service.
6. Retrieve Study Instance ID, Series Instance ID, and SOP Instance ID from DICOM file header (inputted by client to store).
7. Store the DICOM file at a predefined location in the hierarchy of the form: StudyUID/SeriesID/SopUID.
8. Construct appropriate HTTP Response Message (status code + status message) and return to client.

5.2 ARCHITECTURAL DESIGN

A block diagram of STOW-RS service is shown in Figure.6:

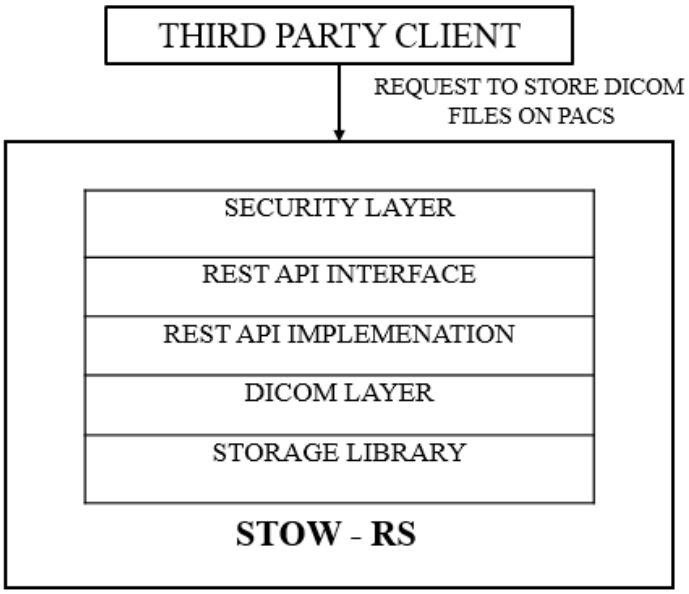


Figure 6: ARCHITECTURE OF STOW-RS

A brief overview of the architecture of STOW is as follows:

1). The third – party client is the entity external to the healthcare organization that wishes to store DICOM files from the PACS.

2). The Security Layer is responsible for – authenticating, that is, verifying the identity of the external client wanting to store a DICOM file.

3). The REST Interface is the REST API used to enable the client to access STOW – RS service.

4). The REST Implementation layer refers to the actual implementation of the REST API interface.

5). The DICOM Library is used to validate the DICOM information being pushed into PACS by the client.

6). The Storage Library is responsible for the physical storing of DICOM files into the PACS.

The STOW-RS can be decomposed into the following modules:

1). Accept STOW requests from third party clients

The client requests are received through the REST API. All request messages are HTTP/1.1 multipart messages. HTTP Request field Content-Type is used in the header lines by the client in an HTTP/1.1 transaction to indicate the type of data being sent to the Service. The request

message can be DICOM or metadata and bulk data depending on the "Content-Type", and is encapsulated in a multipart request body.

2). Validation of STOW request and DICOM file header

Firstly, the format of Study Instance ID in the STOW – RS request URL should be validated. Secondly, the Study Instance ID of the STOW- RS request URL should be checked for consistency with the Study Instance ID field of the DICOM file header. Thirdly, the header of the DICOM file, consisting of the 128 bytes of preamble, 4 bytes of prefix, and the data set (comprising of data elements) should be checked.

3). Ensuring security

Mechanism for storing a DICOM file will be through the HTTPS protocol, using DICOM UIDs for Study Instance level. For enabling the HTTPS protocol, HTTP BASIC Authorization over SSL and SSL Client Certificates must be used. Authentication and authorization of the client attempting to store DICOM files on the STOW – RS service must be implemented.

4). Storing the files

The STOW – RS service creates new resources for the given DICOM instances on the Server or appends them to an existing resource on the Server. One or more DICOM instances associated with one or more study instance unique identifiers (SUID) are stored in the appropriate file store location. The Storage Library is responsible for this action.

5). Issuing a HTTP Response Message

The RESTful Service shall return an HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module.

If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success “response code.

If the status for all instances included in the POST request is Failure, the RESTful Service shall return an appropriate failure status line with a response code. If there are instance specific errors, the response code shall be a 409 and the response payload shall contain the Store Instances Response Module, which contains additional information regarding instance errors. In all other conditions, the RESTful Service shall return an "HTTP 202 - Accepted" response code. The response payload may contain Store Instances Response Module, which specifies additional information regarding instance warnings or failures.

The STOW-RS service is structured in such a way that each of the modules have to be executed in order and successfully to store the DICOM files of the third party clients onto the PACS. The failure of any one module will cause the service to fail for that particular client request. Hence, there is a complete dependency between the modules.

5.3 GRAPHICAL USER INTERFACE

There is no typical GUI developed, as STOW-RS forms the server side of the web service.

However, during development of STOW-RS, the client accessing the STOW-RS service is simulated/mocked using the Fiddler Tool. Fiddler is an HTTP debugging proxy server application. Fiddler captures HTTP and HTTPS traffic and logs it for the user to review. Any browser or web application (and most mobile devices) can be configured to route its traffic through Fiddler.

The client's request to store a DICOM file on the STOW server is triggered using the Fiddler Tool. The HTTP response returned by the server to the client, which contains the HTTP status code and the HTTP status message depicting the status of completion of the STOW-RS request is also viewed using Fiddler.

A snapshot of the Fiddler tool on the client side is shown in Figure. 7:

There are 6 fields and 1 control for the client in this tool, and they are:

1). HTTP Method

Here, it is set to POST as STOW-RS allows third party clients to store DICOM files on the STOW server.

2). Request URL

It refers to the URL of the server where the STOW-RS service is hosted.

An example of the request URL would be:

https://130.147.168.55:443/stow/studies/1.2.3.20163114312.10000

Here,

- https is used for secure http communication (configured using client and server SSL certificates, and by binding the thumbprint of server certificate to the default HTTPS port of 443).
- "13.147.168.55" refers to the IP address of the server over which the STOW-RS service is hosted.
- "stow" refers to the name of the controller of STOW-RS.
- "1.2.3.20163114312.10000" refers to the Study Instance UID.

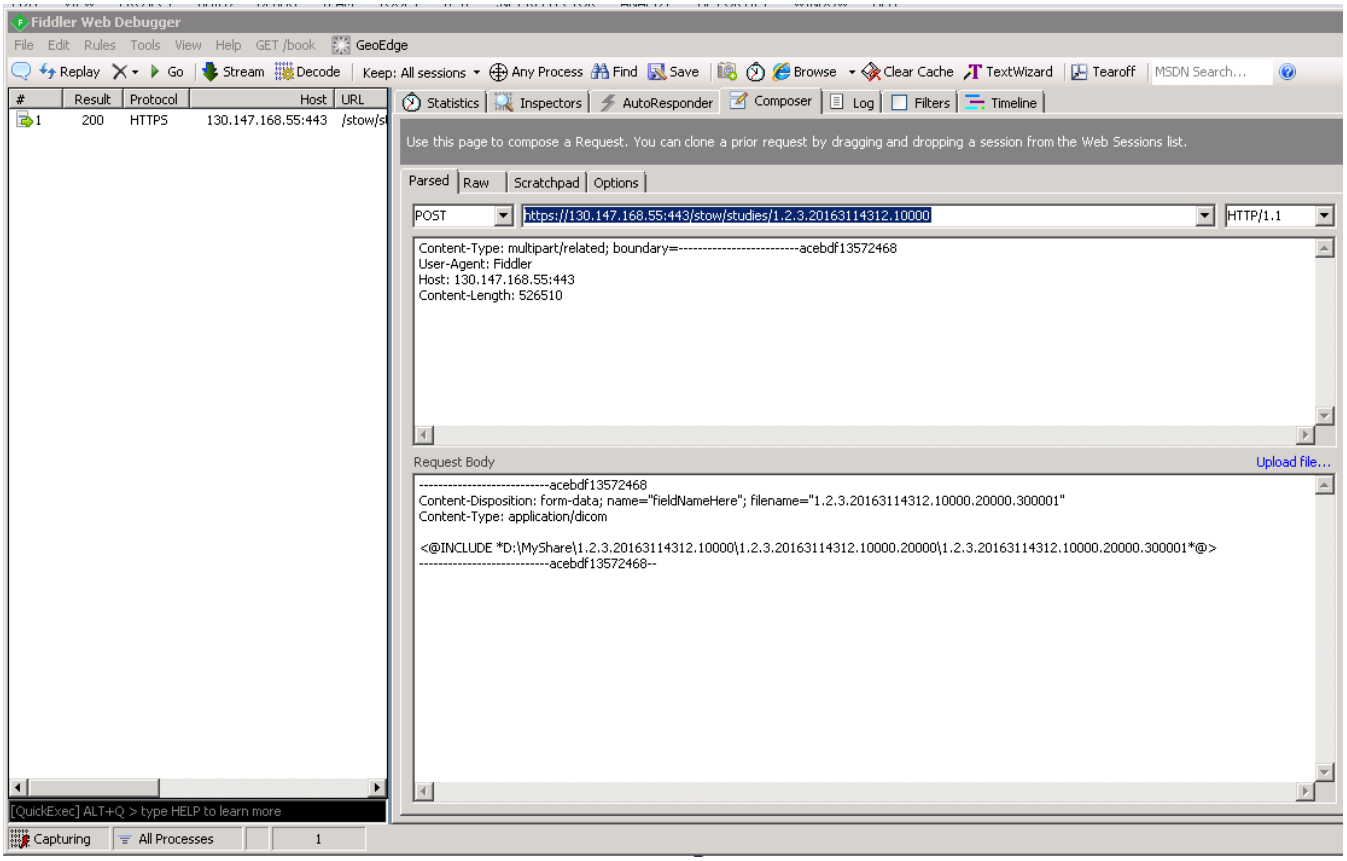


Figure 7: A SNAPSHOT OF FIDDLER TOOL

3). HTTP Version

It refers to the version of HTTP protocol and set to 1.1, that is, HTTP/1.1.

4). Upload File

This control allows the client to browse and upload the file he wants to post using STOW-RS.

5). HTTP Request Header

It contains the following four fields:

- Content type – which specifies the type of content the client is sending to server. It has to be set to multipart/related.
- Boundary – to distinguish between the multiple parts of HTTP request message. Each part contains a DICOM instance.
- Host – the IP address and the port number of the STOW-RS server.
- Content Length – This field indicates the number of bytes of data the client is uploading using STOW-RS.

6). HTTP Request Body

The contents of the HTTP Request are enclosed inside a pair of boundaries (defined in HTTP Request Header). The various fields of the HTTP Request Body are:

- The Content-Disposition response-header field is used as a means for the server to suggest a default filename if the client requests that the content is saved to a file.
- The content type is set to application/dicom to signal that the client is sending DICOM instances to the server. It can be application/dicom+xml or application/json depending on the data being sent.

7). Execute

This allows the client to send the STOW-RS request to the server.

5.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. The data flow diagram for STOW-RS is depicted in Figure. 8:

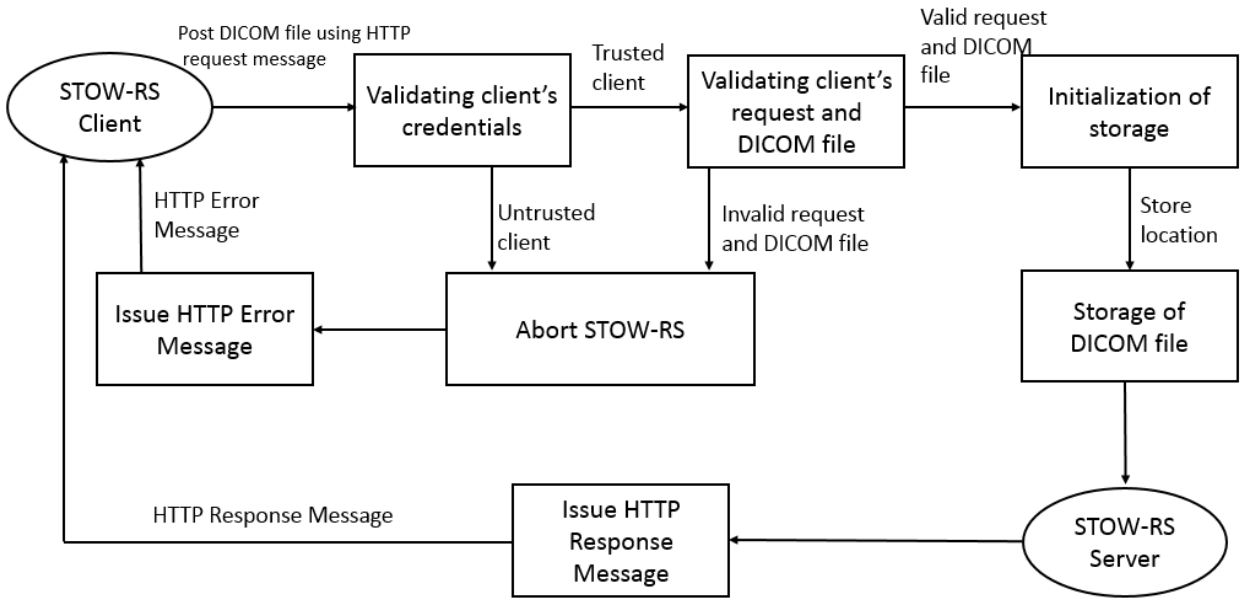


Figure 8: DATA FLOW DIAGRAM OF STOW-RS

5.5 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the

specification of simple runtime scenarios in a graphical manner. The sequence diagram of STOW-RS is shown in Figure. 9:

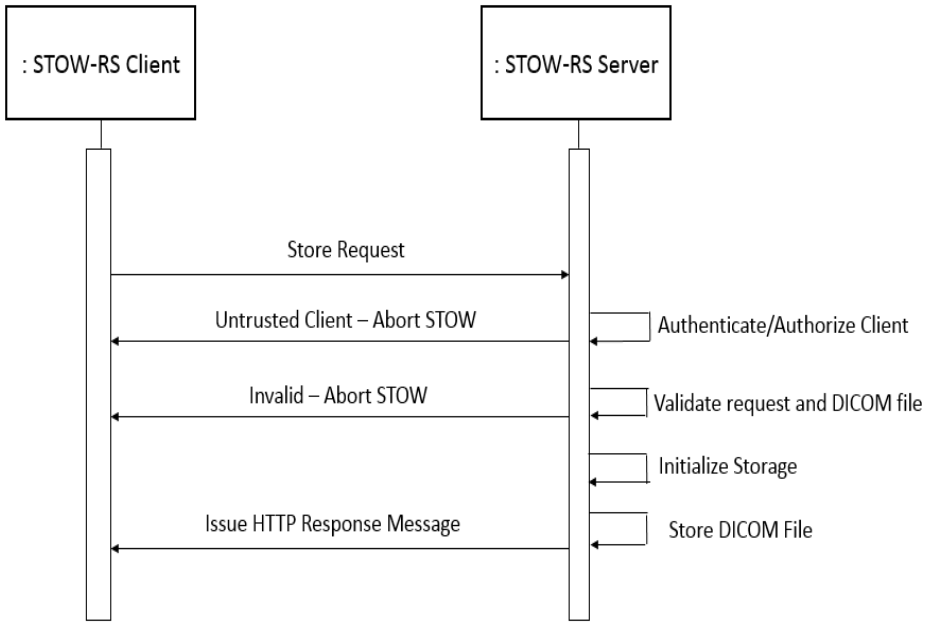


Figure 9: SEQUENCE DIAGRAM OF STOW-RS

5.6 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram for STOW-RS is shown in Figure. 10:

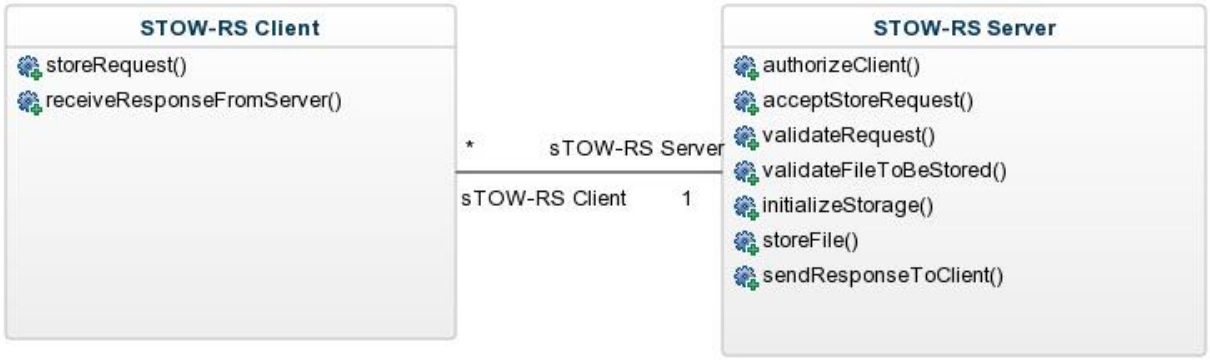


Figure 10: CLASS DIAGRAM OF STOW-RS

5.7 REFERENCE

Digital Imaging and Communication (DICOM) standard 2015 published by National Electrical Manufacturers Association.

6. REFERENCES

[1] *Digital Imaging and Communication (DICOM) standard*, National Electrical Manufacturers Association, 2015.

[2] Philips, [Available]: http://www.philips.co.in/healthcare/resources/support-documentation/ihe-pacs-systemsandwebviewing.