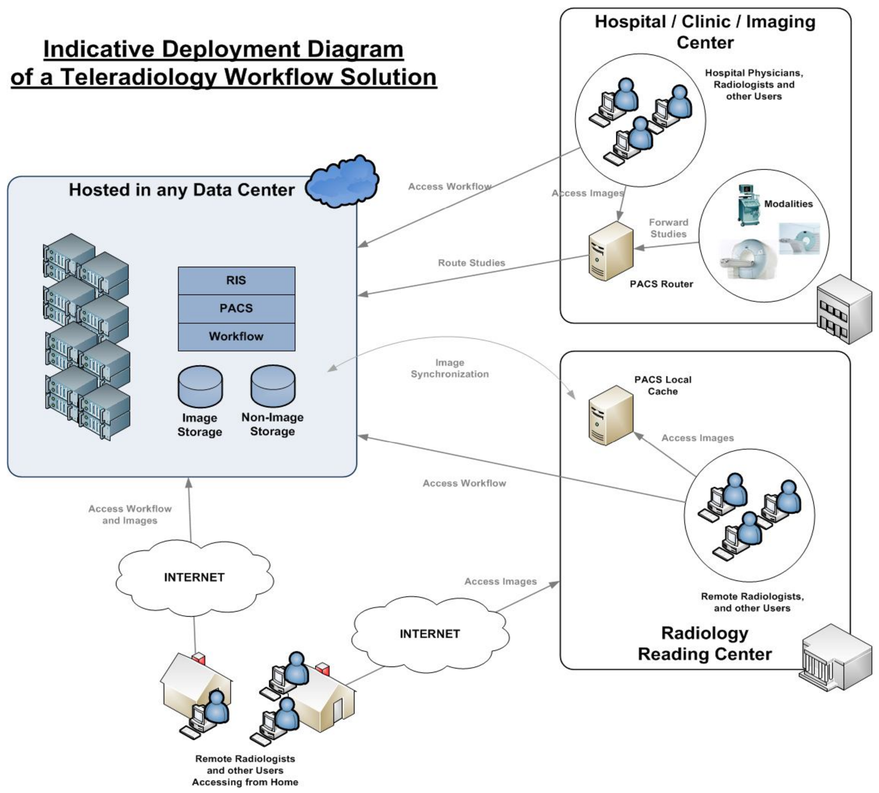
Challenges in Implementing Application Hosting for Teleradiology Platform

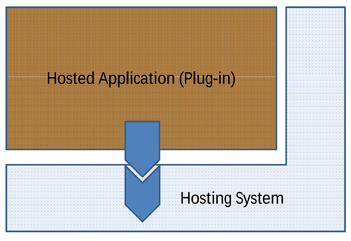
V S Venumadhav Vedula, Raghavendra Rao and Ricky Bedi

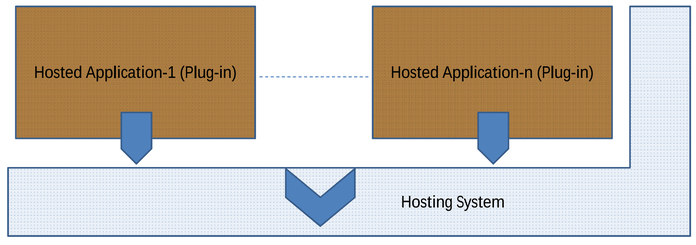
Introduction: Teleradiology providers need to diagnose and report images of various modalities and sub-specialties in a fast and efficient manner. A single viewer application does not suffice for advanced diagnosis of images from different modalities. Suite of clinical applications from multiple vendors needs to be integrated on a single system in order to avoid radiologists reading cases on disparate systems. Teleradiology medical imaging platform (TMIP) based on plug-in architecture is implemented, such that different clinical review and analysis applications can be easily plugged in to the platform. Application Hosting API specified in DICOM Standard (WG-23) is considered as basis for the design of TMIP. Hosting system owned by TMIP addresses teleradiology workflow drivers such as quick turnaround time, image download speeds, portability, deployment, UI Harmonization, licensing, collaboration, security and shared services (CD Burning, DICOM Print, etc.).



Above diagram depicts data flow in a teleradiology workflow. Image and Non-Image data accessed from client system is made sure to be available in quick and efficient manner, ensuring diagnostic quality of the images is maintained.

DICOM Application Hosting: Standard specifies an interface between two software applications Hosting System (1) that provides data to the Hosted Application (2) which analyzes the data eventually returning the results of the analysis to the hosting system. Multiple hosted applications can be hosted by the same Hosting System as long as hosted applications adhere to the hosting system's API.

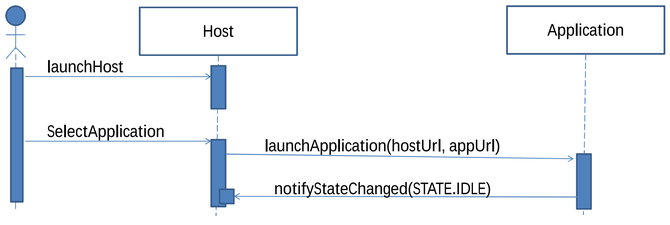




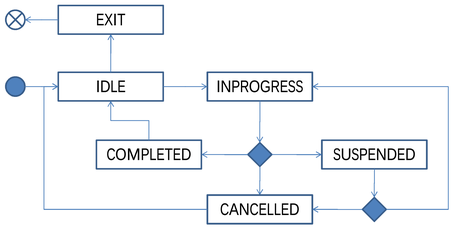
API Design Considerations:

- ✓*Language Independence*
- ✓*Platform Independence*
- ✓*Extensible (backward compatibility)*
- ✓*IP Protected*
- ✓*Security*
- ✓*Leveraging Existing Technology*
- ✓*Simultaneous Launching*
- ✓*Distributed Execution*

Application Launch Sequence:



Hosted Application State Diagram:

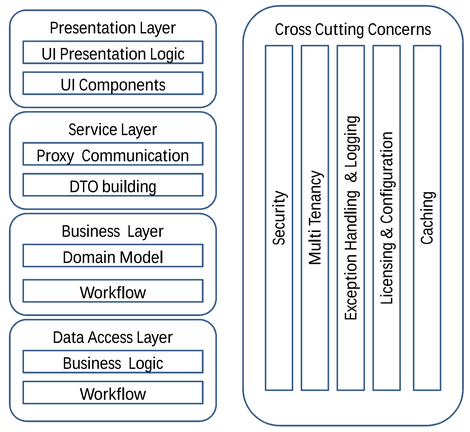


Interface Specifications:

- Base Interfaces: 1) Application 2) Host 3) DataExchange
- Application:
  - getState() : State
  - setState(newState: State) : bool
  - bringToFront(screenArea: Rectangle) : bool
- Host:
  - generateUID() : UID
  - getAvailableScreen(screen: Rectangle) : Rectangle
  - getOutputLocation(protocols: string[]) : String
  - notifyStateChanged(state: State) : void
  - notifyStatus(status: Status) : void
- DataExchange: Data being exchanged can be passed as either the files or may be described as models.

Teleradiology Medical Imaging Platform: Unlike standalone workstations, in Teleradiology , viewing applications are controlled by RIS. This is essential as the multi-tenant applications are to provide adequate levels of data security. Hosting API is extended to support services for RIS.

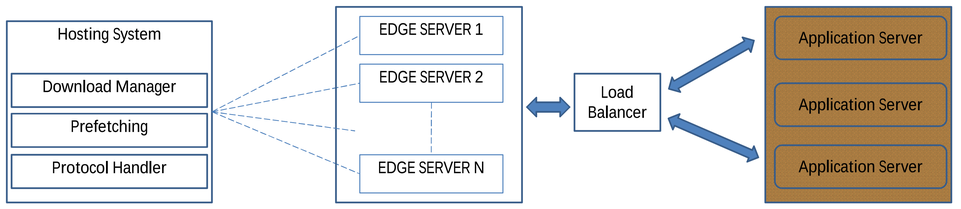
Teleradiology SaaS Application Architecture:



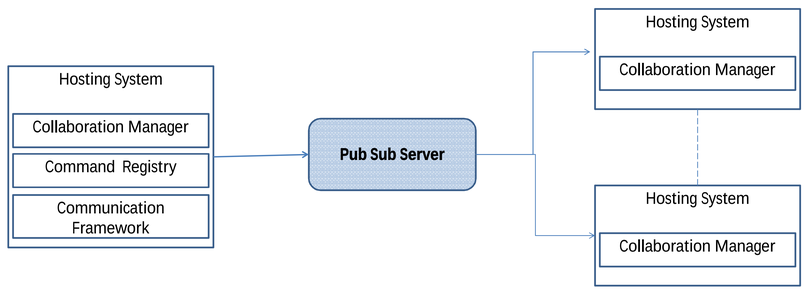
Hosting API Extensions (Instance management):

- LaunchStudy (string studyUID, string sessionToken, string hostName, string AETitle, int securedPort, URL reportServiceURL, string procedureID);
- ActivateStudy (string studyUID, string procedureID);
- CloseStudy (string studyUID);
- Exit();

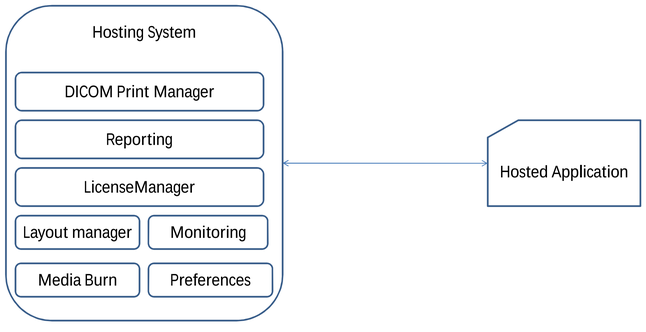
Data Retrieval Management: Hosting System owns the data retrieval manager and intelligently downloads data from proprietary EDGE Servers. Thus data communication protocols are abstracted from Hosted Applications. Output results from hosted application are stored in folder specified by the Hosting system and storing in the server falls under host's responsibility.



Collaboration: Image sharing is another key requirement in teleradiology. It enables online review of the exams between multiple radiologists. Hosting System owns the collaboration framework, such that distributed execution of the application commands in remote machine is easily achievable. Reverse role negotiation is also supported by the collaboration framework.



Shared Services: There are multiple other common functionalities that Hosting System should be owning, to name a few DICOM Print, Reporting, Media Burning, Licensing, State Management and Monitoring.



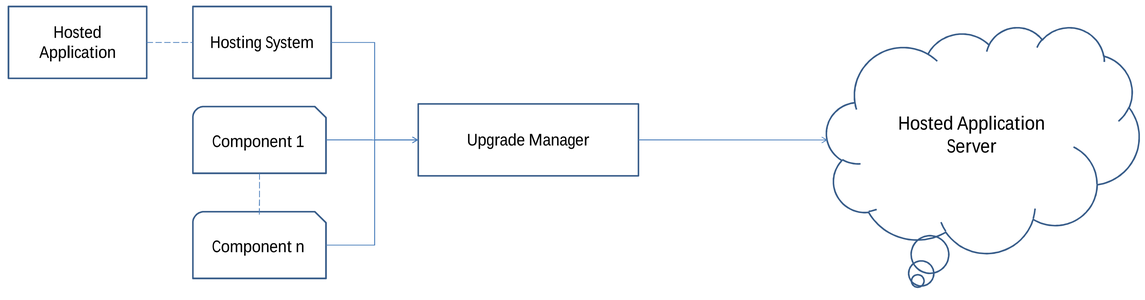
Hosting API Extensions (Shared Services):

- SendToPrint (printObjects: DataSet[]) : bool
- CheckLicense(applicationUID: UID) : bool
- UpdateStatus(data: string[]) : void
- BurnToMedia(location: string) : bool
- SendToReport (findings: DataSet[]) : bool
- SavePreferences(preferences: object) : bool

Software Upgrading: As in any cloud based application, software up gradation happens in automated fashion for TMIP platform as well. TMIP manages upgrading of all the components that are hosted by TMIP. Hosted application should communicate with the hosting system as where the upgrades are published. Plug-in manages the automatic download and installation of components. Each of the components can upgrade independently and should be backward compatible with other dependent modules.

Hosted Application API Extensions (Upgrade Services):

- CheckForUpgrades() : URL



Future Work:

- ✓*Design adapters for non-compliant third part plug-ins*
- ✓*Design appropriate abstract models for data exchange*
- ✓*Extend framework for distributed execution*
- ✓ *Achieve Platform Independence*