Overview of DICOM and the open source ecosystem

Why use DICOM?

- Universal standard for medical imaging
- Don't lose data
    - Get all the details from the scanner
    - Well-defined representations with documentation

How to think about DICOM

- Each "dataset" is an instance of a "class" with strongly typed instance variables (called "elements")
    - Instances can be stored as files (called Part-10 files after the section of the standard describing them)
    - Instances can be grouped when they share unique IDs
- The sequence of instances are like a logfile of what the scanner generated and it's up to the application to sort through them to determine the relationships and map them into useful constructs like Volumes, Segmentations, etc.
- To create DICOM instances the application populates the elements to link it with the other instances as appropriate

Some of the more useful DICOM classes

- Imaging: CT, MR, PET, US...
    - Orginal scan data
- Segmentation: SEG
    - Image based labeling of structures
- Structured Reporting: SR
    - Vector annotations, quantifications, qualitative findings
- Radiotherapy: RT
    - Doses, plans, structures...
- Parametric Maps: PM
    - Images with defined quantities and units
- Spatial Registration: SRO
    - Linear and nonlinear with explicit frames of reference
- Whole Slide Images: WSI
    - Microscopy images, possibly multichannel with annotations in SR

DICOM networking

- DIMSE is tradional "PACS" networking used worldwide
    - Both endpoints need custom configuration
    - Best for use within controlled firewalls
- DICOMweb is uses modern REST API concepts
    - Better suited to internet and security
    - Introduced JSON header encoding

DICOM Implementations: Java, C#

- PixelMed toolkit, open source, but intended for reference not for community use
- FairOaks
- probably others...

DICOM Implementations: C++

- GDCM: traditional implementation used in ITK
- DCMTK: also widely used in ITK and many other places
- CommonTK (CTK)
    - DCMTK + Qt * SQLite
    - Core of Slicer's DICOM module
- dcmqi: convenience interface over DCMTK to support encoding of analysis results in DICOM

DICOM Implementations: Python

- pydicom
  - Widely used, bundled with Slicer
  - Maps instances to python objects and numpy arrays
- pydicomnet
  - Implements DIMSE with pydicom
- dicomweb-client
  - Implements DICOMweb with pydicom
- highdicom (new)
  - Adds SEG, SR, etc on pydicom

DICOM Implementations: JavaScript

- dicomParser, cornerstone, OHIF
  - Layers of the Open Health Imaging Foundation stack
- dcmjs.org
  - dcmjs: maps instances to/from JavaScript classes
    - original: emscripten cross-compiled DCMTK
    - current: pure JavaScript (browser/server)
  - dicomweb-client/dicomweb-server: DICOMweb on dcmjs
  - dcmjs-dimse (new): DIMSE on dcmjs (server only)
- Can be used in qSlicerWebWidget

DICOM in Slicer

- DICOM module supports local database and DIMSE networking
- DICOM Plugins examine related instances to propose mappings to Slicer datatypes, export Sicer data to DICOM
- DICOMwebBrowser query/retrieve/store and support Google DICOMweb stores securely
- DICOM Plugins provided by SlicerRT, QuantitativeReporting, PET...

Summary

- Supporting all of DICOM is a huge task
- Community is very active tools are becoming very capable
- Interoperability is improving
    - Slicer-generated segmentations in OHIF
    - OHIF structured report annotations in Slicer
    - highdicom encoded machine learning results in Slicer and OHIF