Case Study: DIAN Manual QC Uploader

John Paulett

jpaulett@wustl.edu

June 30, 2010

XNAT

Overview

Case study on using the XNAT REST API to import externally-managed data into XNAT

Dominantly Inherited Alzheimer's Network (DIAN)

- Multi-center study storing data in the CNDA
- Mayo has existing system for performing MR Quality Control (QC)
- QC results must be present in the CNDA

Mayo Export

- QC data exportable in 2 Comma Separated Value (CSV) files
  - First file has session-level QC metrics (e.g. overall pass, payable)
  - Second file has scan-level QC metrics (e.g. scan pass, head coverage, head motion)

Session-Level CSV

```
patid,sdate,field,coil,pass,quarantine,rescan,pay_site,initials,
   comments

000101_MR1,20090126,3,HeadMatrix,1,1,0,1,gmp01,""
```

Scan-Level CSV

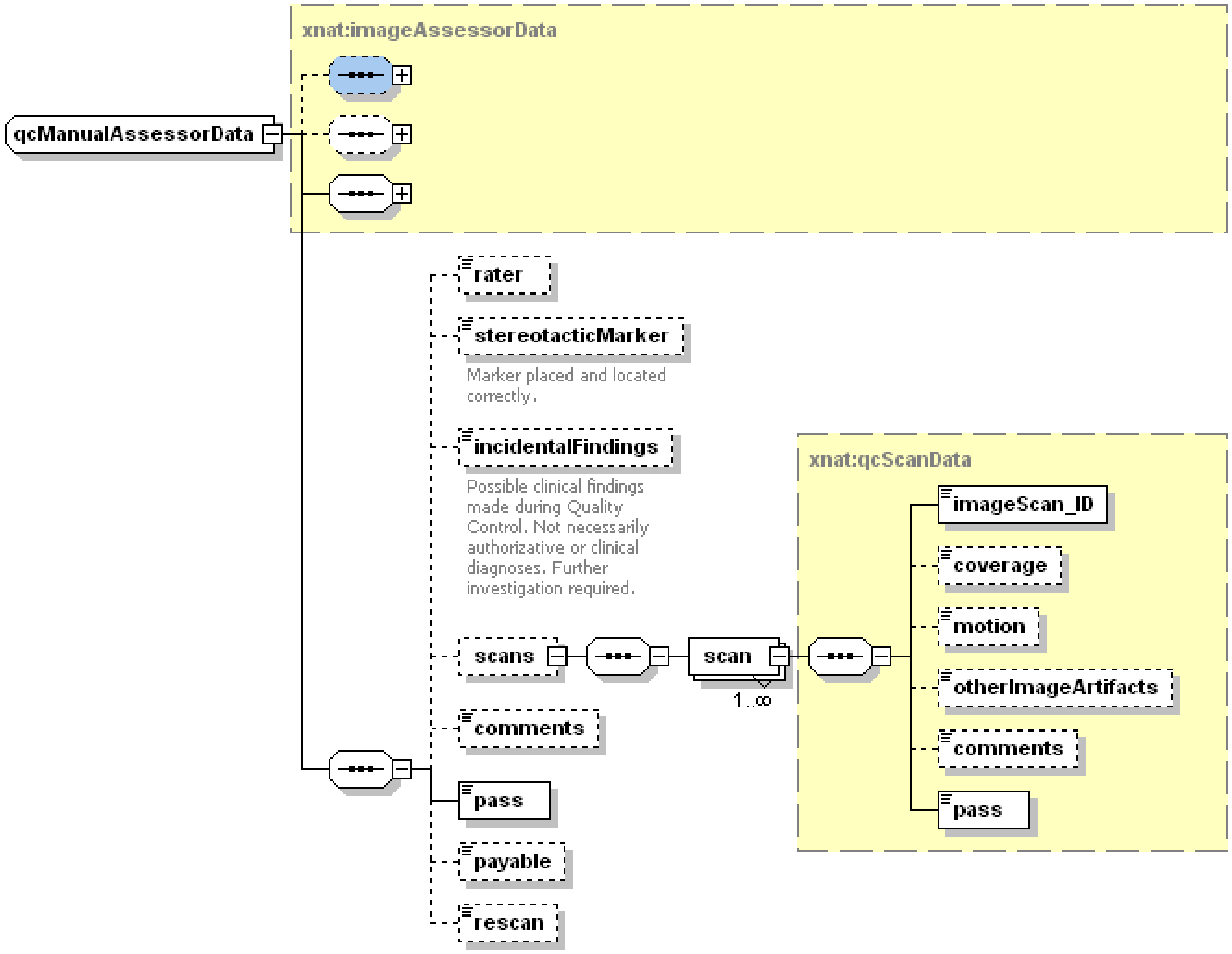
```
patid,sdate,seriesnumber,seriesdescription,in_bgr,in_flow,in_oth
er,wrap,headcoverage,susceptibility,head_motion,ip_motion,marker
,pass,comments

000101_MR1,20090126,9,"mIP_Images(SW)",0,0,0,0,0,-1,-1,-1,0,1,"“
000101_MR1,20090126,8,"Pha_Images",0,0,0,0,0,-1,-1,-1,0,1,"“
000101_MR1,20090126,7,"Mag_Images",0,1,0,0,0,-1,-1,-1,0,1,"“
000101_MR1,20090126,6,"Axial T2-FLAIR",0,0,0,0,0,0,-1,0,0,1,"“
000101_MR1,20090126,5,"MPRAGE GRAPPA2
repeat",1,0,0,0,0,,,,0,1,"“
000101_MR1,20090126,4,"MPRAGE GRAPPA2
repeat",1,0,0,0,0,,,,0,1,"“
000101_MR1,20090126,3,"MPRAGE GRAPPA2",1,0,0,0,0,,,,0,1,"“
000101_MR1,20090126,2,"MPRAGE GRAPPA2",1,0,0,0,0,,,,0,1,""
```

Development Process

- Added “Manual QC” image assessor to xnat.xsd
- Built command line tool in Groovy language that parses CSV files, builds Manual QC XML, and uploads XML to XNAT’s REST API

xnat:QCManualAssessment

- Needed for DIAN’s MR & PET QC
- Modeled on DIAN QC & several additional Quality Control projects
- Extension of xnat:imageAssessorData
  - Generic top-level element with unbounded list of modality-specific scan-level assessors


xnat:imageAssessorData
qcManualAssessorData
rater
stereotacticMarker
Marker placed and located correctly.
incidentalFindings
Possible clinical findings made during Quality Control. Not necessarily authorizative or clinical diagnoses. Further investigation required.
scans
scan
1..∞
xnat:qcScanData
imageScan_ID
coverage
motion
otherImageArtifacts
comments
pass
comments
pass
payable
rescan

Upload Process

for each row in the session-level file

- search for the Subject & Project using the Session ID via the REST API
- find the session’s scans in the scan-level file
- build the QCManualAssessment XML
- HTTP PUT the XML to the REST API

Get Subject ID & Project

HTTP GET:

```
/REST/experiments
  ?format=xml
  &xsiType=xnat:mrSessionData
  &project=DIAN_*
  &label=<Session ID>
  &column=ID,subject_ID,label,project,date
```

Put Assessment

HTTP PUT Assessor XML to:

```
REST/projects/<project ID>
  /subjects/<subject ID>
  /experiments/<session ID>
  /assessors/<generated assessor ID>
```

```xml
<?xml version="1.0"?>
<xnat:QCManualAssessment
    ID='0000001_v00_mr_mQC_2010-03-29'
    project='DIAN_011' >
    <xnat:date>2010-03-29</xnat:date>
    <xnat:imageSession_ID>CNDA_E000024</xnat:imageSession_ID>
    <xnat:scans>
      <xnat:scan xsi:type='xnat:mrQcScanData'>
        <xnat:imageScan_ID>10</xnat:imageScan_ID>
        <xnat:coverage>0</xnat:coverage>
        <xnat:pass>1</xnat:pass>
        …
      </xnat:scan>
       …
    </xnat:scans>
    <xnat:pass>1</xnat:pass>
    <xnat:payable>1</xnat:payable>
</xnat:QCManualAssessment>
```

XNAT

Upload Tool

nrg.github.com/dian-qc-uploader/

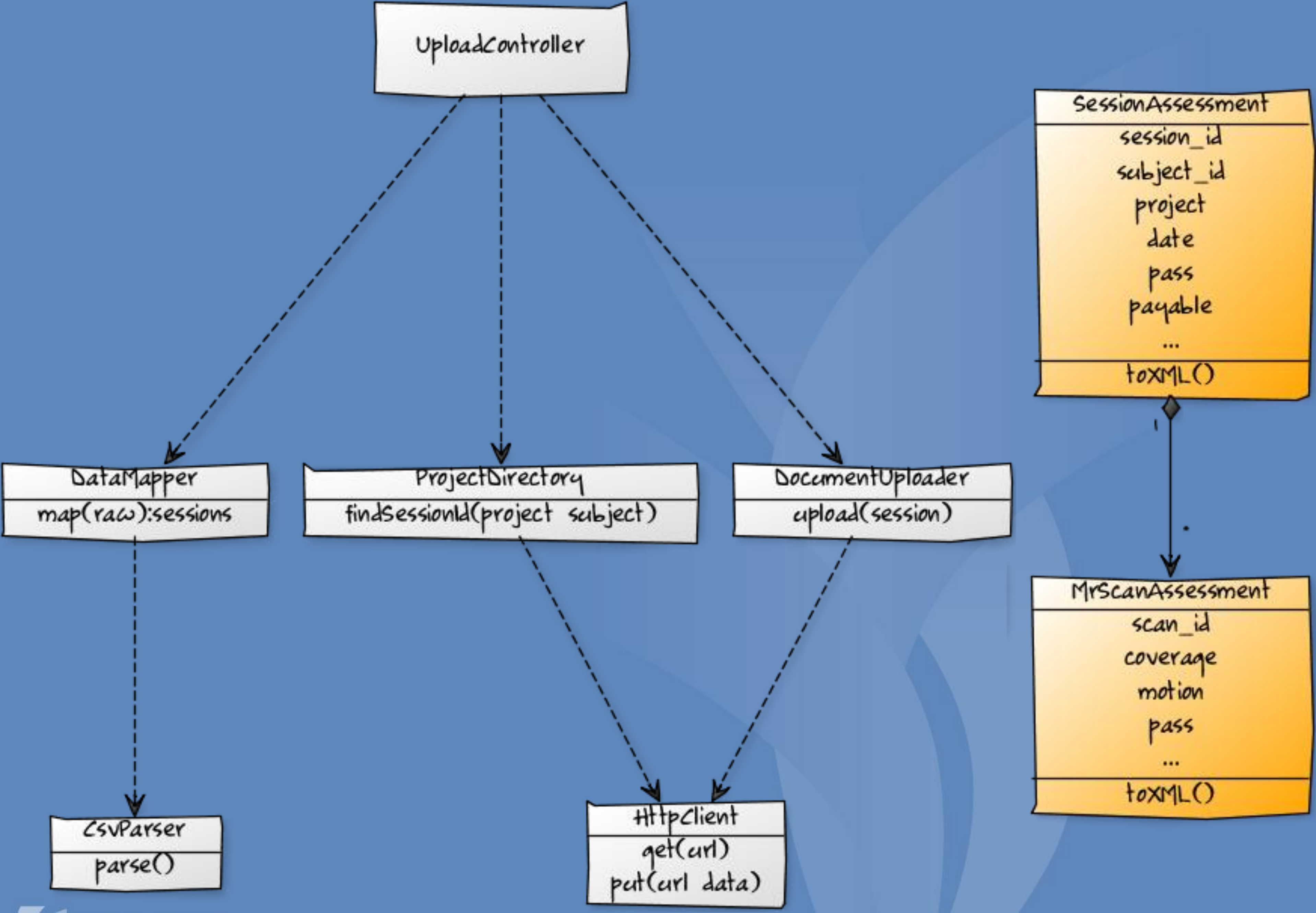
Written in Groovy

Command line tool takes username, password, server, and file names as arguments

XNAT

Upload Tool

```
$ java -jar dian-qc-uploader-0.4.jar -s
  https://cnda.wustl.edu -u admin -p admin scanqc.csv
  sessionqc.csv
2010-04-22 10:02:42,830 INFO UploadController -
  Processing 00001_v00_mr
2010-04-22 10:02:44,231 INFO UploadController -
  Processing 00002_v00_mr
```


UploadController
DataMapper
map(raw):sessions
ProjectDirectory
findSessionId(project subject)
DocumentUploader
upload(session)
CsvParser
parse()
HttpClient
get(url)
put(url data)
SessionAssessment
session_id
subject_id
project
date
pass
payable
...
toXML()
1
*
MrScanAssessment
scan_id
coverage
motion
pass
...
toXML()

Challenges

- Separating generalizable schema from DIAN-specific model
- CSV files lacked Subject & Project, requiring search before upload
- Subject IDs out of sync after ID format change

“Take Away” Points

- Errors from a single session should not prevent other sessions from being uploaded
- Logging
  - Progress & Errors to standard output
  - Debug info to log file
- Unit testing quickly isolates regressions
- Modular design (even in “simple script”) makes inevitable changes less hacky

Using Groovy

Pros

- Use familiar Java APIs and libraries
- Lacks Java’s verbosity, while still readable by Java developers
- Builder pattern makes XML creation very easy

Cons

- IDE support is still maturing
- Documentation & community are still small

Alternative Languages

Criteria

– CSV parsing, XML generation, HTTP client

- Python (PyXNAT)
- Clojure (xnat4clj)
- Java (xnat-beans.jar)

Questions?

XNAT