Case Study: DIAN Manual QC Uploader

John Paulett (jpaulett@wustl.edu)

Audience: Focus on coders, but useful information to anyone who can fill out a bug report

Overview: Demonstrate 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). This QC data must be imported into the CNDA on a regular basis

Development Process

- Build new image assessor XSD element to hold QC data (*xnat:QCManualAssessment*)
- Build tool to covert CSV export of Mayo data into *xnat:QCManualAssessment* XML and upload to CNDA via the REST API

xnat:QCManualAssessment

- Added to xnat.xsd (anyone can use it as of XNAT 1.4)
- Designed to cover both MR & PET QC (PET QC is directly entered via a form on the CNDA)
- QC requirements modeled from DIAN & several other projects. Obtained input from radiologists on the modeled schema
- Extension of *xnat:imageAssessorData*
  - Generic top-level element with unbounded list of modality-specific scan-level assessors

Mayo CSV Export

- QC data exported 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)

Upload Tool[1]

Groovy command line tool parses CSV files, builds XML and uploads to XNAT's REST API. Takes username, password, server, and file names as arguments

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 xnat:QCManualAssessment XML
  - HTTP PUT the XML to the REST API
```

Challenges

- Separating generalizable schema from DIAN-specific model
- CSV files lacked Subject & Project, requiring search before upload
- Having multiple primary data stores can become fragile when data inconsistently changed

"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

Should I use 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 (used Eclipse Groovy plugin)
  - Documentation & community are still small

Alternative Languages

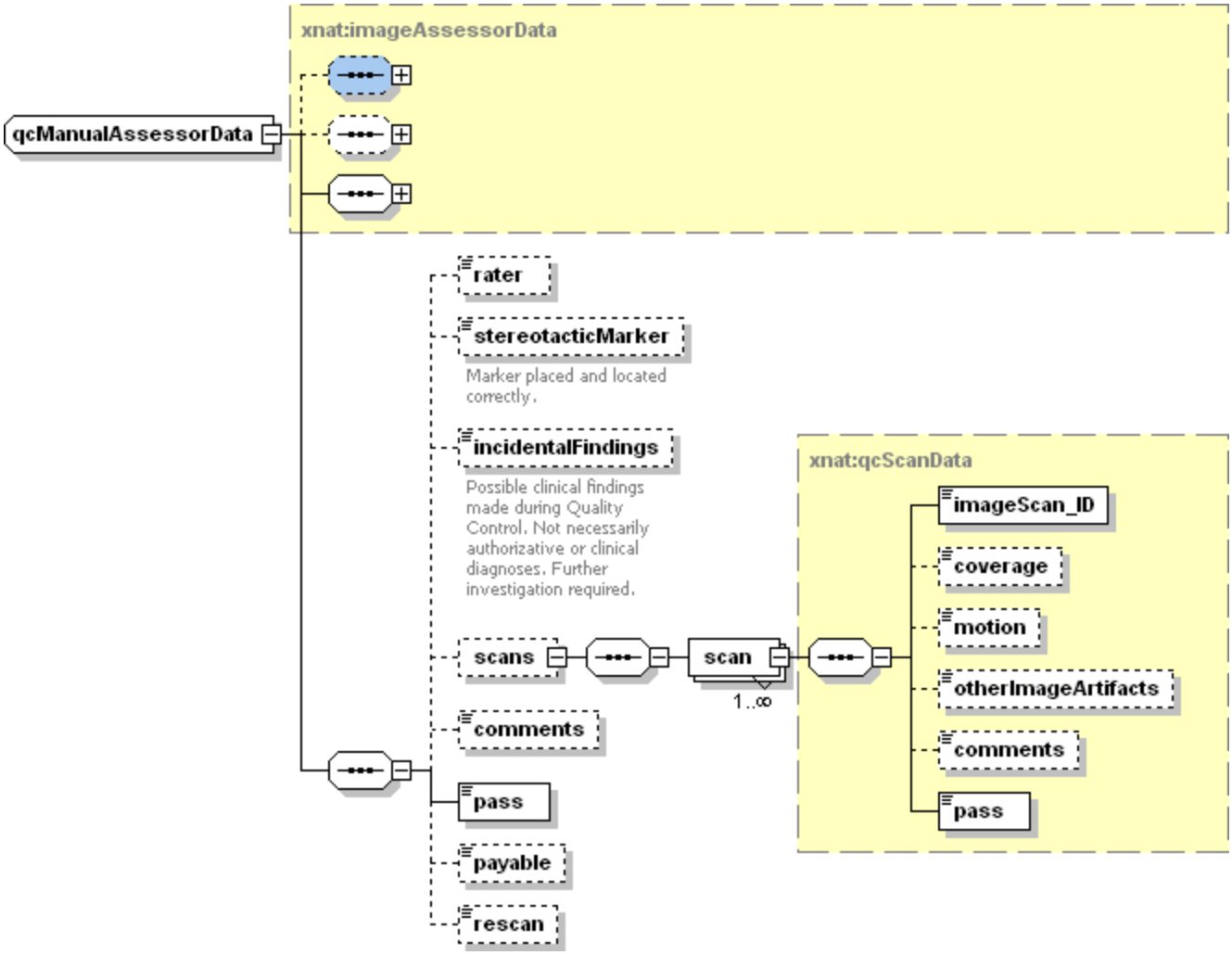
For this task, we need built-in language support or good libraries for CSV parsing, XML generation, and HTTP communication.

- Python (PyXNAT[2])
- Clojure (xnat4clj[3])
- Java (xdat-beans)

[1] http://nrg.github.com/dian-qc-uploader/

[2] http://packages.python.org/pyxnat/

[3] http://bitbucket.org/karchie/xnat4clj



HTTP Requests

Search for Project & Subject ID:

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

Upload *xnat:QCManualAssessment*:

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

```
<?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>
```