## class diagram for refinement upload

appionScript.AppionScript()

#### reconUploader.ReconUploader()

base class for uploading 3D refinements; performs:

#### uploadExternalRefine.py generic uploader for any external package refinement

requires: (1) mrc file, (2) particle parameter file optional inputs: FSC file, class averages & reprojections, refined averages & reprojections, variance images

## uploadXmippRefine.py uploader for

uploader for xmipp\_protocol\_projmatch.py

requires: N completed iterations from xmipp projection matching protocol

## uploadEMANRefine.py uploader for EMAN refinement

requires: N completed iterations from EMAN projection matching protocol. Iterations can be mixed and matched w/ and w/o SPIDER Coran or message passing

#### uploadXmippML3DRefine.py uploader for xmipp\_protocol\_ml3d.py

requires: N completed iterations from xmipp ML3D protocol

## how to construct a refinement uploader: required input is red

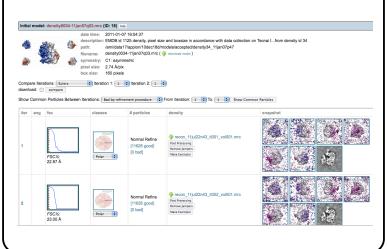
refinement procedure

- 1. write function to check for successful refinement completion
- 2. define a new table in the database for all package-specific parameters
- 3. write package-specific converters for volume files, particle parameters, FSC files, etc.

### input for upload:

- 1. 3D map (mrc format)
- 2. particle data file
- 3. (optional) FSC file: 1st column = inverse pixels, 2nd column = FSC
- 4. (optional) class averages & re-projections file
- 5. (optional) variance images
- 6. (optional) other results

# refinement summary page is automatically generated



# given input map & parameters, the following operations are performed in the background

- general error-checking
- untarring results and creating necessary directories
- querying database for stack / model parameters
- output parameter parsing
- reading particle data file
- determining upload iterations
- inserting all metadata.
- creating analysis outputs: Euler plots, Euler jumper calculations, FSC insertions, etc.
- verifies the number of inserted iterations
- other