

EVLA Data Processing PDR

Pipeline design

Tim Cornwell, NRAO



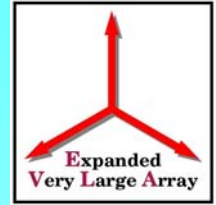
Background



- Pipeline processing automates by-rote processing from *e.g.* *AIPS* or *AIPS++* cookbooks
- Many different radio astronomy pipelines now being developed
 - BIMA, ATCA, WSRT, MERLIN, EVN, *etc.*
- *AIPS++* ideally suited to pipeline development
 - No accident!
 - Rich scripting language, tool-based applications
- ALMA strongly committed to pipeline reduction
 - NRAO contracted to provide pipeline framework
- Known processing background important for use of radio data within the National Virtual Observatory



Run Through



1. The user requests pipeline processing of a project AB973
2. An appropriate processing script (AB973.g) is constructed based upon the meta-data in the archive for project AB973, some standard choices, and information about calibrators in the calibrator archive.
3. The resulting processing script is executed in a queue by an executive.
4. During execution, the script requests data from the archive
5. The results are passed through quality assessment to derive numerical measures of the quality of the result
6. Results from the processing (calibrated data, images, calibration tables, processing logs, diagnostic plots, processing script, source catalogs) are archived.
7. The user receives notification of the result of the processing.
8. The user downloads information from the designated web page.



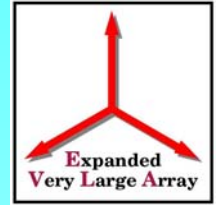
Requirements



- Requirements
 - Process EVLA observations from original observed data through to reference image or spectrum *without human intervention*
 - Adaptable to different calibration and imaging strategies
 - Provide standard log of processing
 - Run on non-specialized hardware
 - Be operable automatically or manually
 - Each result will have a quality measure attached to it
 - Required success rate not yet specified but expect ~ 50 – 70%
- Assume rest of system is designed to support pipeline reduction
 - e.g. observation scripting, monitor and control, archive



Design



- Looked at other pipeline systems
 - ORAC-DR
 - OPUS
 - BIMA Pipeline
 - ACSIS
- Major issues are:
 - Determining observation meta-data
 - Capturing heuristics
 - Generating processing scripts appropriate to actual observation
 - Running scripts robustly, allowing for failure, stalling, pre-emption, *etc.*
 - Publishing results
- Since we expect to build on top of AIPS++, none of the above systems were judged worth adopting directly.
 - Could change with experience!



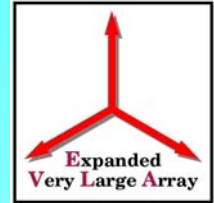
Pipeline toolkit



- Prototype is implemented entirely in AIPS++
- Use standard AIPS++ tools for filling, editing, calibration, imaging
- How to capture processing rules?
 - Use production rules software (make) to capture dependencies and processing rules
 - Rules for processing (“heuristics”) expressed as glish/AIPS++ script fragments
 - Encapsulated in makefiles
- Prototype framework exists and works end-to-end
 - Reads VLA archive tapes (from disk) and produces HTML pages
 - *No human interaction needed*
 - ~ 1000 lines of Glish code for entire framework
 - Working on two examples currently
 - VLA A-configuration, 8 GHz continuum including self-calibration
 - VLA D-configuration, HI synthesis
 - Expect further significant evolution of framework as experience accumulates



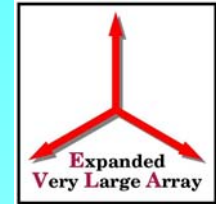
Production rules!



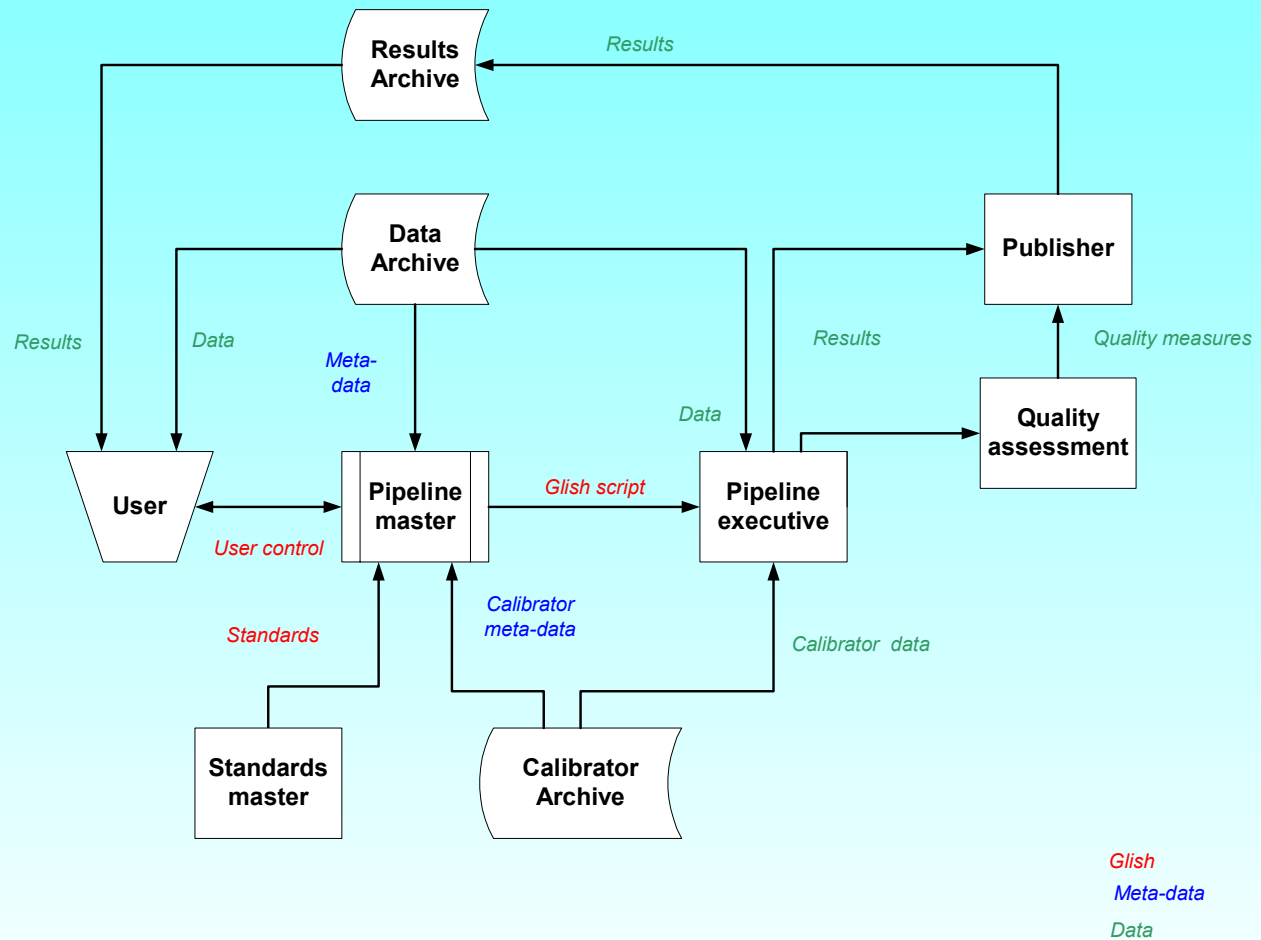
Method	Advantages	Disadvantages
<i>Glish scripts</i>	Familiar, Integration is easy	Not a natural fit for production rules
<i>Text e.g. ORAC_DR</i>	Simple	Limited in expressiveness, Only sequences easy
<i>Make system</i>	Familiar, Very powerful, Widely used, Available in AIPS++ (make)	Can be hard to debug
<i>XML e.g. BIMA pipeline</i>	Expressive, Tools available	Production rules must be imposed
<i>CLIPS or equivalent</i>	Very powerful, Widely used	Learning curve Binding to Glish required
<i>OPUS e.g. STScI pipeline</i>	Powerful, Widely used (e.g. Celera) Machine-aware	Production rules implicit rather than explicit



Prototype pipeline software design

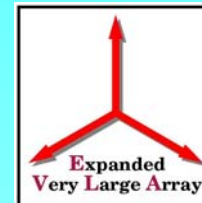


```
include
"e2pipeline.g"
myp := e2pipeline()
myp.project('AB973')
```





e2e/AIPS++ pipeline results



End-to-end processing results for project : AB973

Processing information

- Processing script
- Processing log

MeasurementSet

UVFITS file [AB973.ms.uvfits](#)

Images

- Image 116288+392
- Image 1635+381
- Image 116382-061
- Image 1651+014
- Image MRK915
- Image 2236-145
- Image N7213
- Image 2230-397
- Image AKN564
- Image 2236+284
- Image 103106-025
- Image 0339-017
- Image 103125+011

End-to-end processing results for project : AB973, source : 0339-017

Generated as part of project AB973

Clean image : 0339-017.clean.restored : In neighborhood of brightest point

Clean image : 0339-017.clean.restored : Full image

Coordinate system

Axis	Coord Type	Length	Coord value	Pixel
1	Right Ascension	1024	0.957817619	513
2	Declination	1024	-0.0310077328	513
3	Stokes	2	1	1
4	Frequency	1	8.4601e+09	1

Statistics

Statistic	Value	Sum
Number points	2097152	2617.03369
Flux density	79.0168457 Jy/beam	
Mean	0.00124789891 Jy/beam	Median -0.00692834193 Jy/beam
RMS	0.72200191 Jy/beam	Std. Dev. 0.722000957 Jy/beam
Med Abs Dev Med	0.418486387 Jy/beam	Quartile 0.418538928 Jy/beam

Positions of maximum and minimum

	Value	Pixel coordinates	Sky coordinates
Minimum	-4.66335344 Jy/beam	[5 305 1 1]	03:39:32.700, -01:46:46.620, 1, 8.460100e+09Hz
Maximum	16.7389145 Jy/beam	[528 510 1 1]	03:39:30.886, -01:46:35.960, 1, 8.460100e+09Hz

FITS image

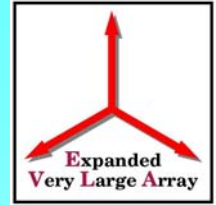
[0339-017.clean.restored.fits](#)

This page was generated by the e2e project using the AIPS++ package. Please send comments to: [e2e_Manager](#)

Generated by tcornwel at 2002/04/02/15:39:23



Difficulties



- No person dedicated to this development – I did about 2 – 3 months work
- Working from VLA historical archive but EVLA will be designed to support pipelined reduction
 - Most glaring omission is the intention of the observer
 - Have tried to work around this limitation
- Very limited heuristics used in this version
 - Need much more work with scientific staff in capturing best practices.
 - Expect to get help from DM Scientific Working Group during next development cycle



In next cycle...



- Lindsey Davis will work mostly on this development, starting in September
- Second prototype will have limited goals
 - Explicitly record observing intent in first generation observing script
 - Construction of processing scripts determined by observing script
 - Expand range of test projects
 - Implement connections to calibrator database, and quality measurement
 - Improved publication of results, perhaps by VOTables (XML-based format from the NVO)
- Parallel development of simulator
 - To enable end-to-end testing (from observing script to simulator to archived results)
- Evaluate again at end of second development cycle