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!

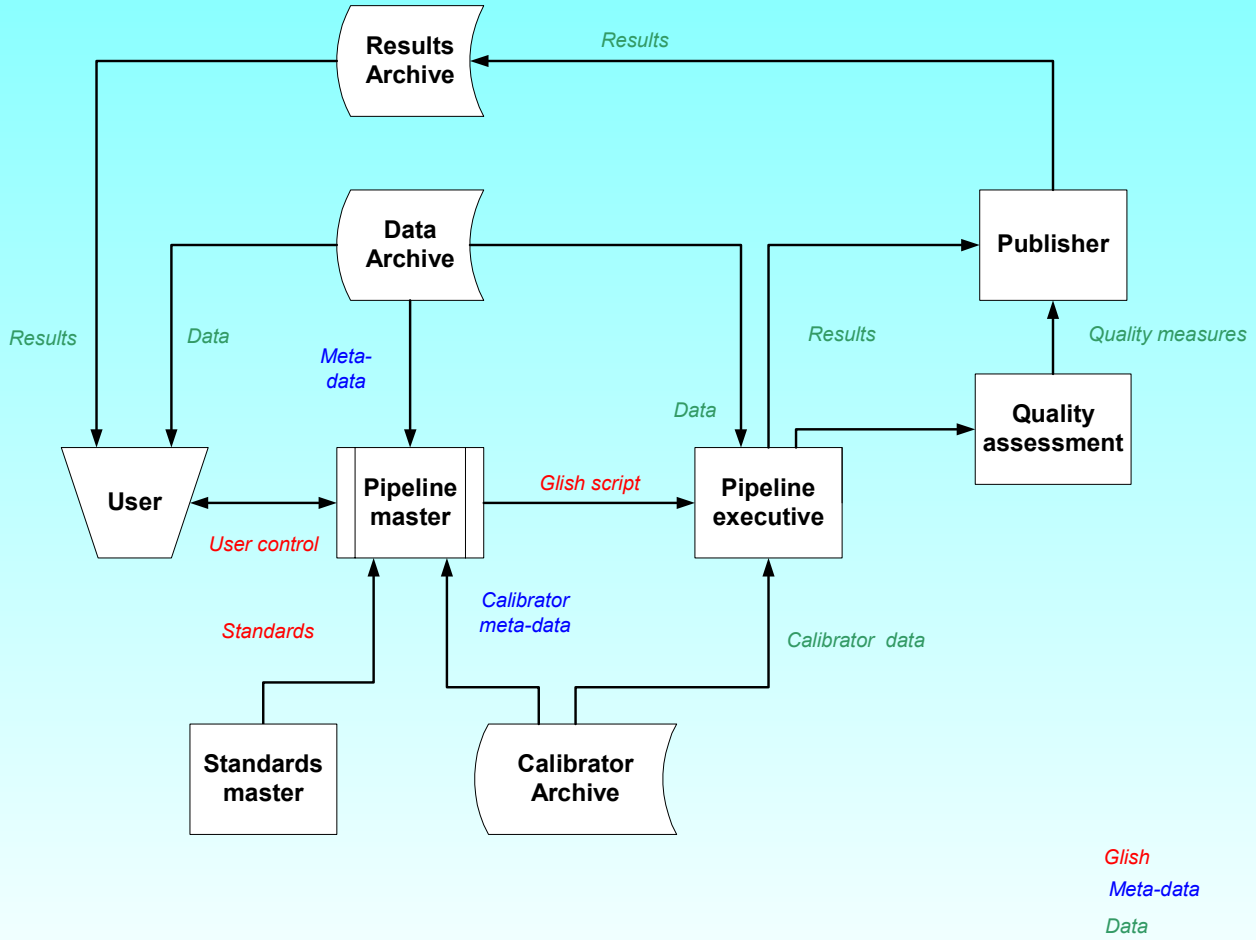
Method	Advantages	Disadvantages
Glish scripts	**Familiar,** **Integration is easy**	**Not a natural fit for production rules**
Text e.g. ORAC_DR	**Simple**	**Limited in expressiveness,** **Only sequences easy**
Make system	**Familiar,** **Very powerful,** **Widely used,** **Available in AIPS++ (make)**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

Prototype pipeline software design

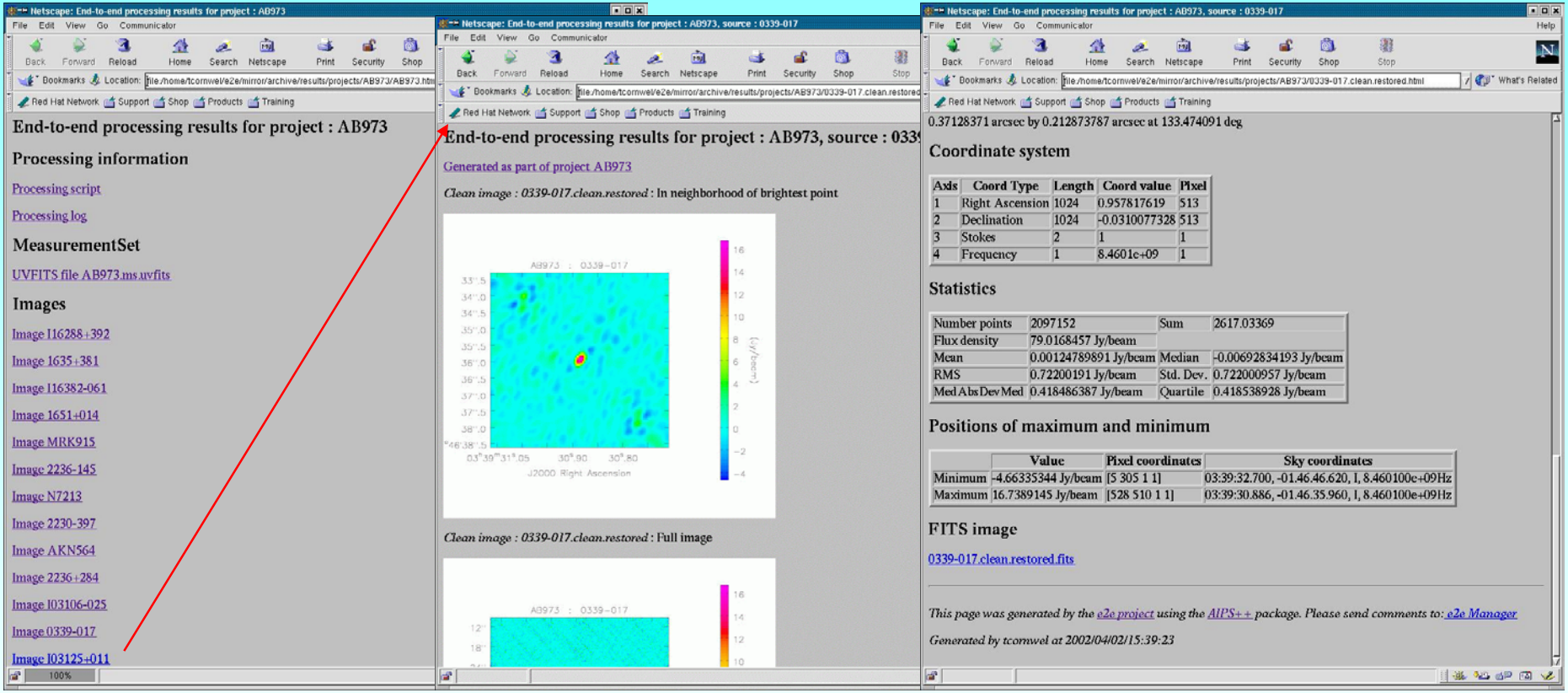
include
“e2epipeline.g”

myp := e2epipeline()

myp.project(‘AB973’)



e2e/AIPS++ pipeline results



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