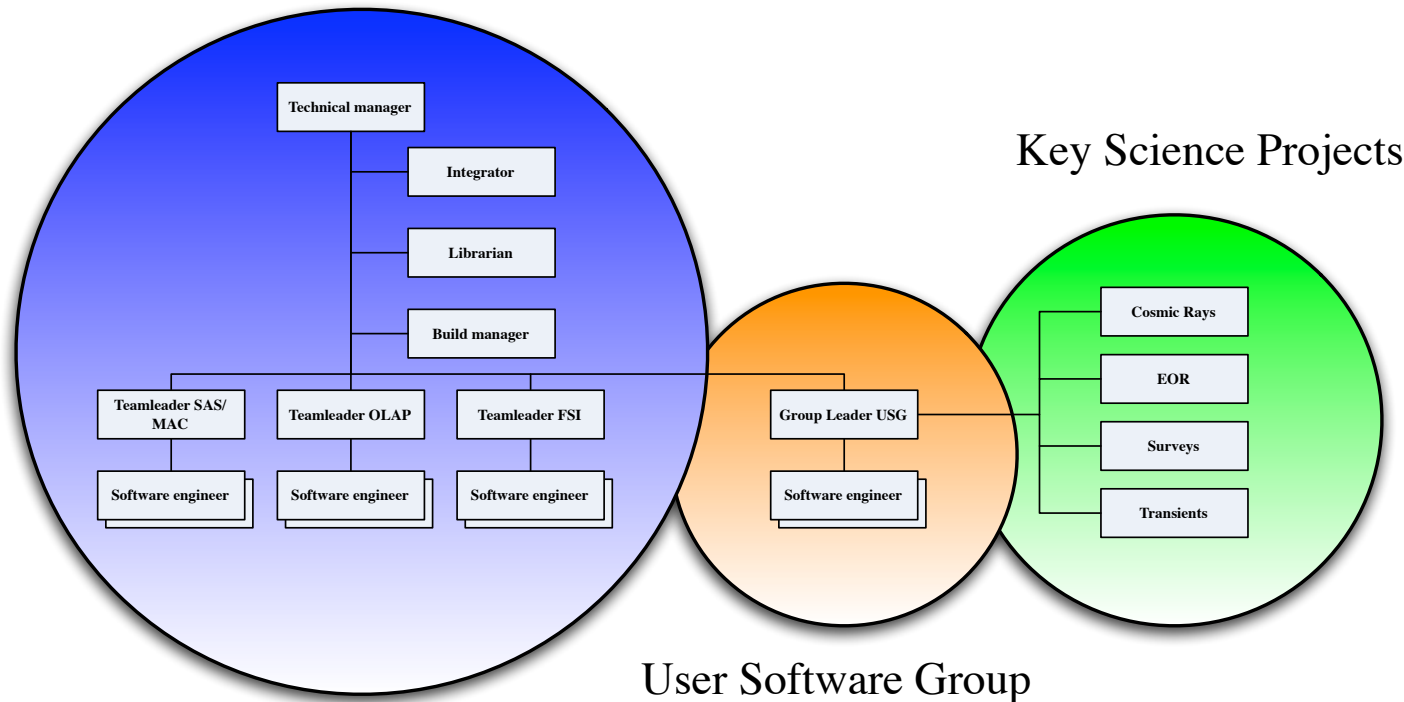

User Software Overview

LOFAR DCLA Project Meeting
26 June 2007

Michael Wise

- Group Overview
- Science Pipelines
 - Designs and module breakdowns
 - Implementation and roll-out
 - KSP status summary
- Infrastructure and Tools
 - Data formats
 - Core libraries
 - Offline analysis tools
- Support Activities
- Demos

LOFAR Project Software Team



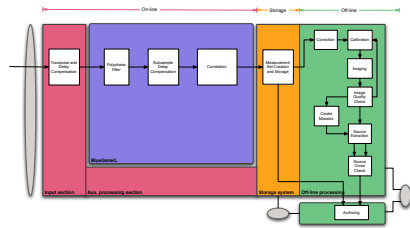
- Coordination *KSP developments, LST, E-LOFAR consortium, partner organizations*
 - Development *Requirements, algorithms, infrastructure, tools, and testing*
 - Support *Development, testing, documentation, code repository*

Science Pipelines

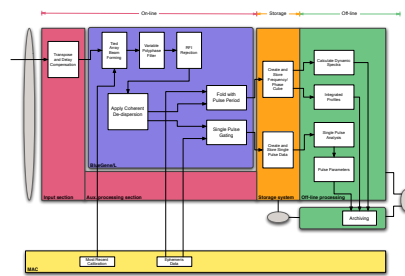
- Based on KSP requirements
- Mix of LOFAR project and USG/KSP development
- Includes online and offline components
- Pipeline and interactive components

⇒ *Preliminary integrated software plan*
9 observing modes
~ 60-70 FTE-yrs of effort
Phased roll-out of capability

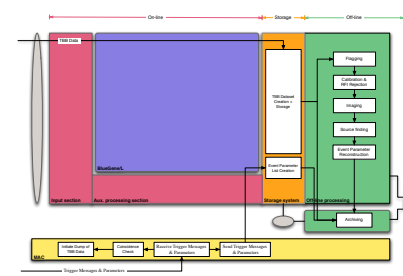
Survey mode



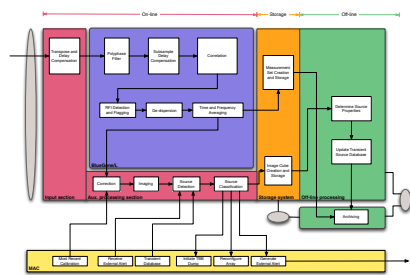
Known pulsars mode



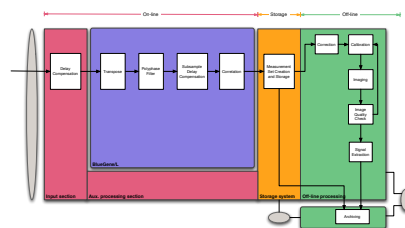
VHECR mode



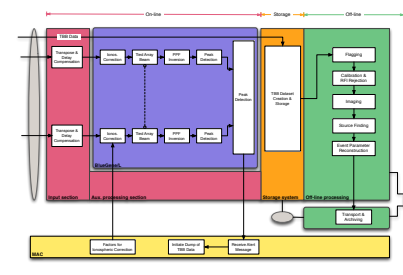
Transient detection mode



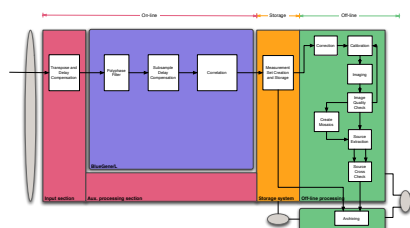
EoR mode



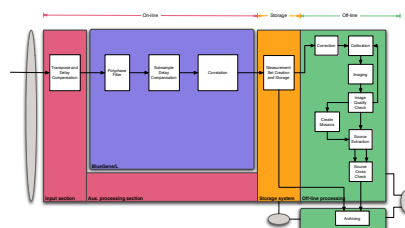
UHEP mode



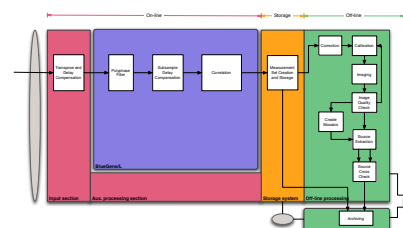
Pulsar Survey mode

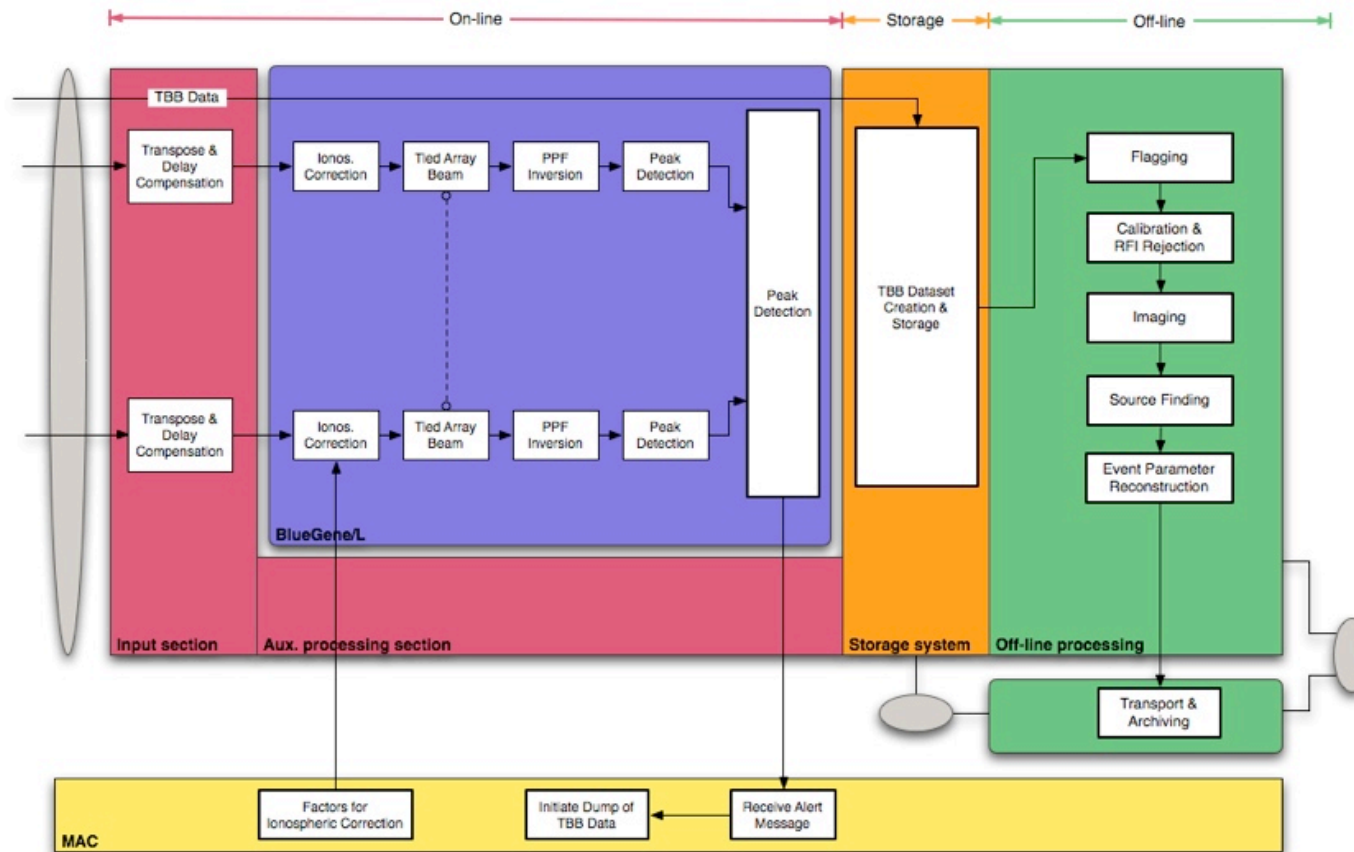


HECR mode

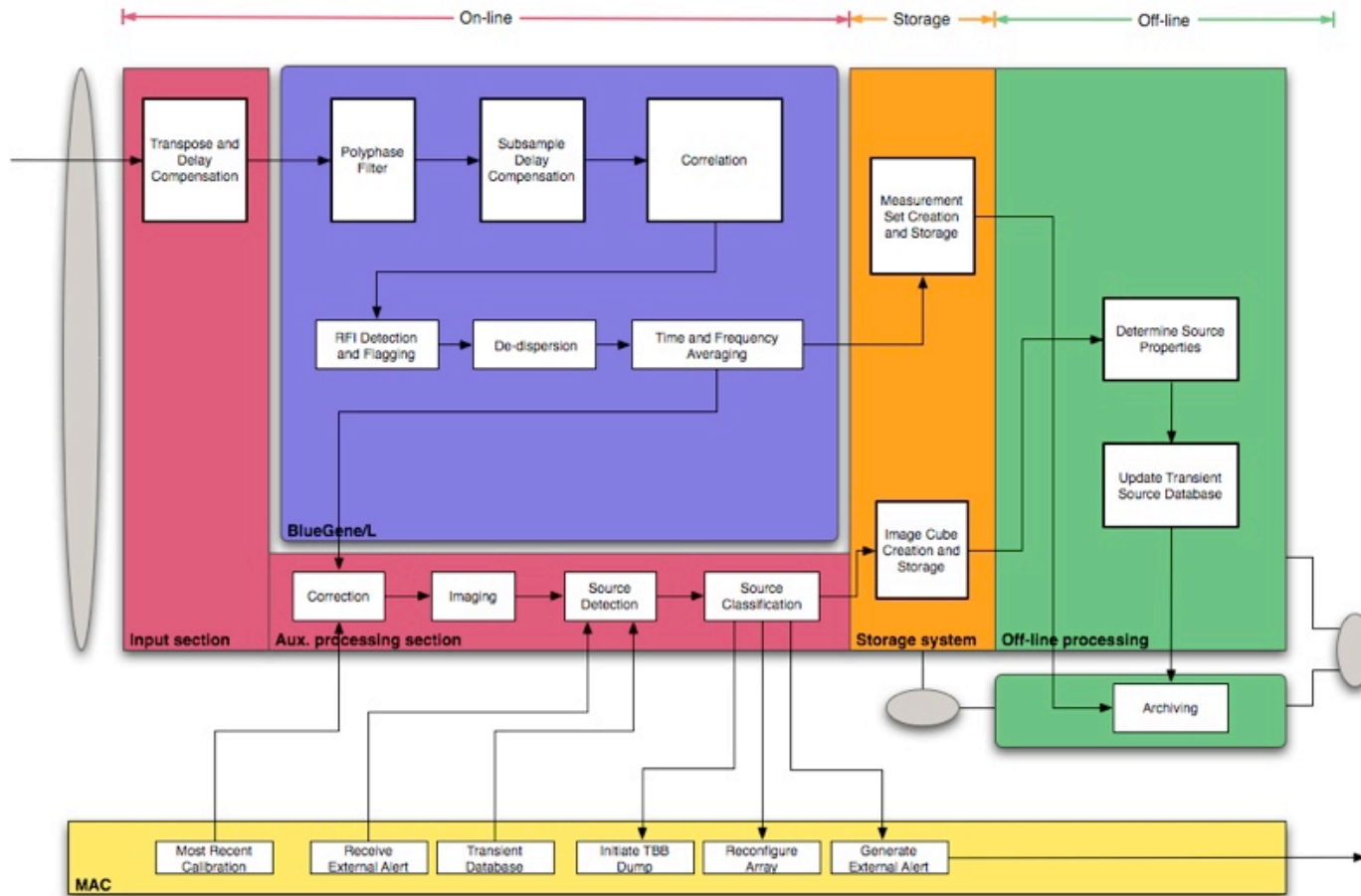


TS mode

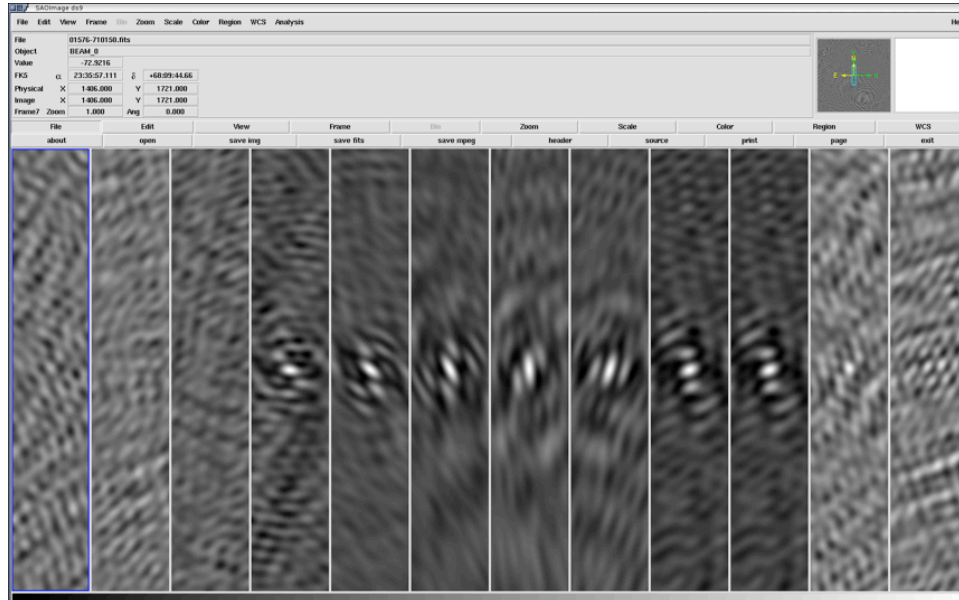




UHEP mode



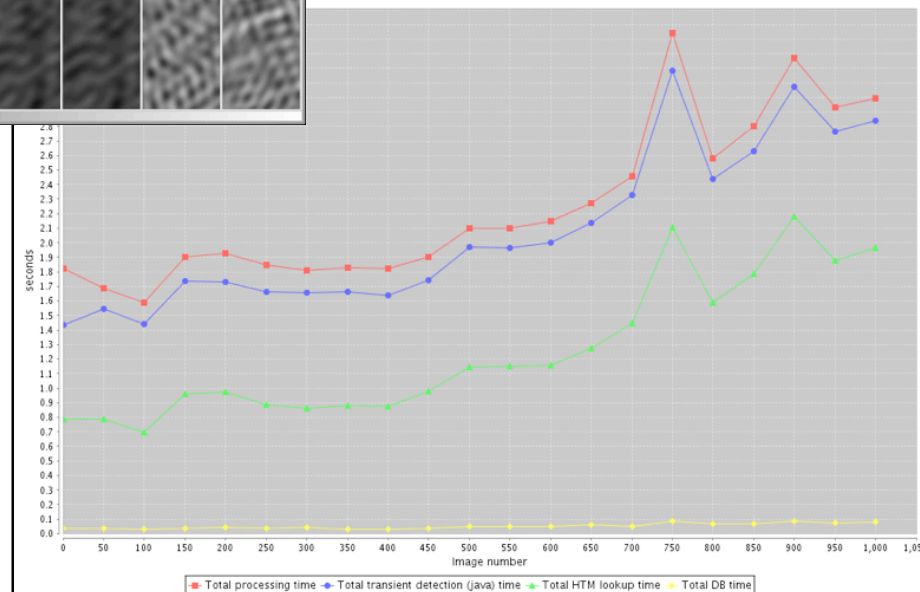
Transient detection mode



- *Real-time pipeline*
- *Rapid detection*
- *Source classification*
- *Trigger generation*
- *Transient database*

Detection and storage performance tests using simulated datasets

LOFAR Transient Pipeline - Processing times
ata on 3GHz processor 1GB RAM, 7.5 σ detection threshold



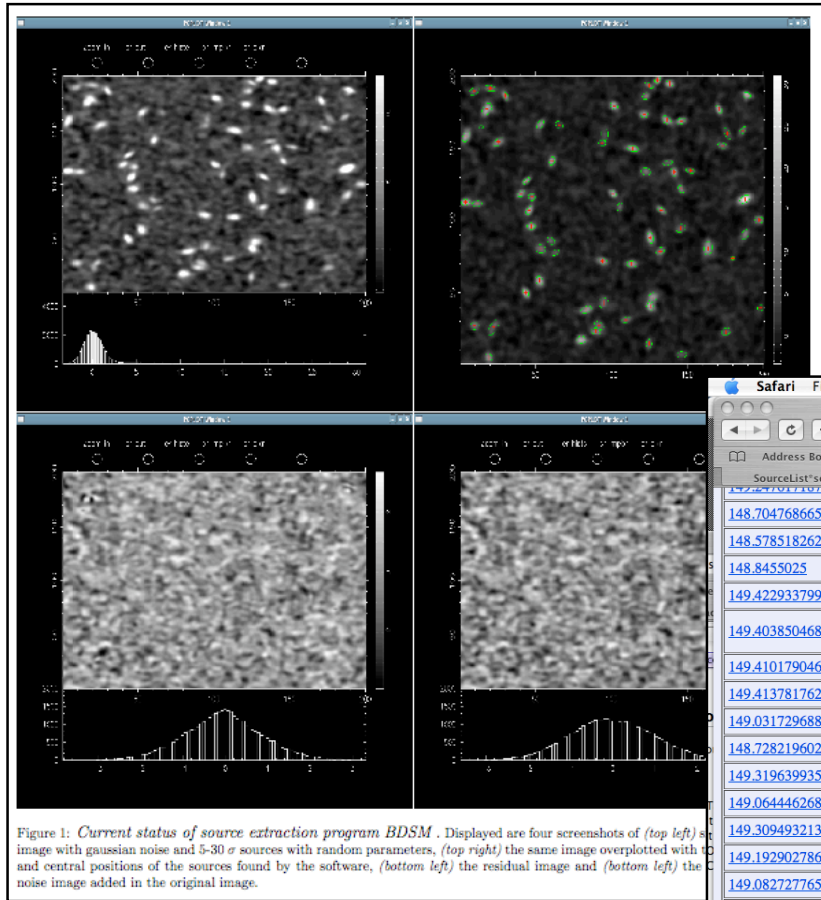
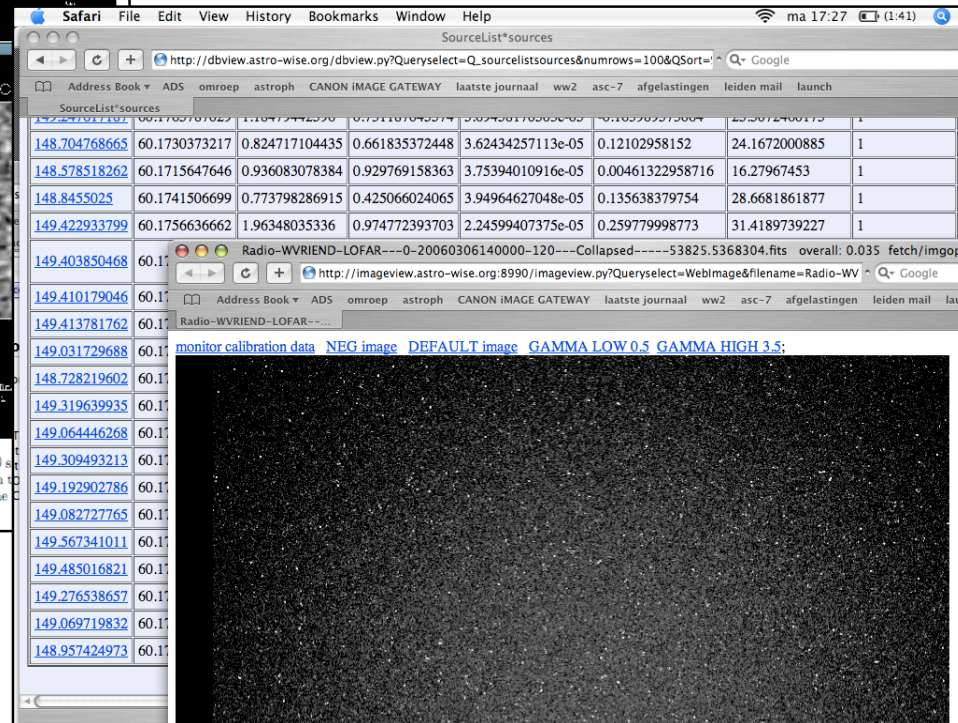






















Figure 1: Current status of source extraction program BDSM. Displayed are four screenshots of (top left) s image with gaussian noise and 5-30 σ sources with random parameters, (top right) the same image overlotted with and central positions of the sources found by the software, (bottom left) the residual image and (bottom left) the noise image added in the original image.

- *Quality control*
- *Mosaicing*
- *Source detection*
- *Catalogs*



The screenshot shows a Safari browser window displaying a source catalog table. The table has columns for source ID, RA, Dec, and other parameters. Below the table, there is a zoomed-in image of a source.

Source ID	RA	Dec	Other Parameters
148.704768665	60.1730373217	0.824717104435	0.661835372448
148.578518262	60.1715647646	0.936083078384	0.929769158363
148.8455025	60.1741506699	0.773798286915	0.425066024065
149.422933799	60.1756636662	1.96348035336	0.974772393703
149.403850468	60.1756636662	1.96348035336	0.974772393703
149.410179046	60.1756636662	1.96348035336	0.974772393703
149.413781762	60.1756636662	1.96348035336	0.974772393703
149.031729688	60.1756636662	1.96348035336	0.974772393703
148.728219602	60.1756636662	1.96348035336	0.974772393703
149.319639935	60.1756636662	1.96348035336	0.974772393703
149.064446268	60.1756636662	1.96348035336	0.974772393703
149.309493213	60.1756636662	1.96348035336	0.974772393703
149.192902786	60.1756636662	1.96348035336	0.974772393703
149.082727765	60.1756636662	1.96348035336	0.974772393703
149.567341011	60.1756636662	1.96348035336	0.974772393703
149.4885016821	60.1756636662	1.96348035336	0.974772393703
149.276538657	60.1756636662	1.96348035336	0.974772393703
149.069719832	60.1756636662	1.96348035336	0.974772393703
148.957424973	60.1756636662	1.96348035336	0.974772393703

	Design	Prototyping	Porting	Testing
<ul style="list-style-type: none"> • Cosmic Rays <ul style="list-style-type: none"> – Prototype pipeline near completion – Porting to USG framework underway 				
<ul style="list-style-type: none"> • EoR <ul style="list-style-type: none"> – Preliminary pipeline design – Component module inventory needed – No developer available 				
<ul style="list-style-type: none"> • Pulsars <ul style="list-style-type: none"> – Designs and prototypes exist – Data products and formats need definition – Work begun on de-dispersion module 				
<ul style="list-style-type: none"> • Surveys <ul style="list-style-type: none"> – Prototype source detection pipeline exists – Remaining modules require specification 				
<ul style="list-style-type: none"> • Transients <ul style="list-style-type: none"> – Prototype pipeline near completion – Performance testing about to begin – Classifier module needs better design 				

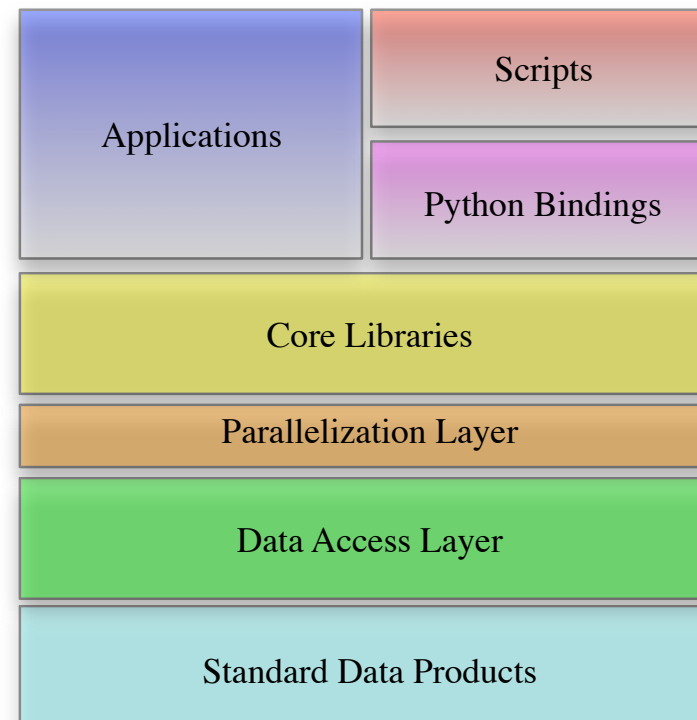
Infrastructure and Tools

Requirements

- Support HPC (*Large files, fine and coarse parallelism, GRID, etc.*)
- Compatibility with other major packages (*CASA, IRAF, ROOT, CIAO, etc.*)
- Support multiple platforms
- Support custom analysis
- Rapid development
- Supportable

Design

- Python scripting layer
- Native APIs will be in C/C++
- Python bindings for all libraries
- Support (MPI+PVM)
- Use or adapt existing software
- Open-source software

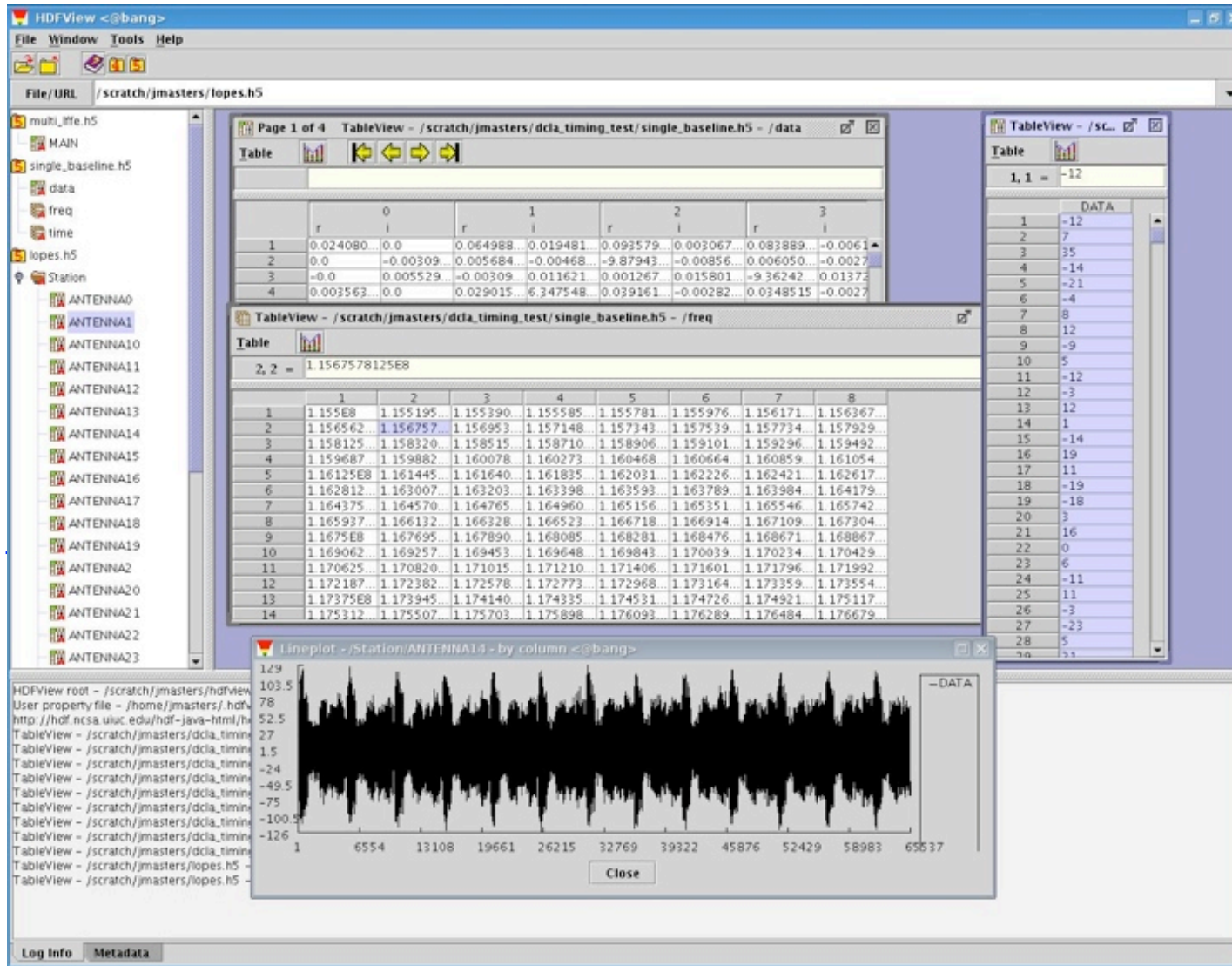


Data	Formats	ICD	DAL I/O	Availability
Time series	HDF5	1.0	R+W	Now
Beam-formed	HDF5	0.5	R+W	Soon
UV data	MS/HDF5	0.2	R/R+W	Q3 07
Image cubes	FITS/HDF5	0.2	--/R+W	Q4 07

Supporting several formats

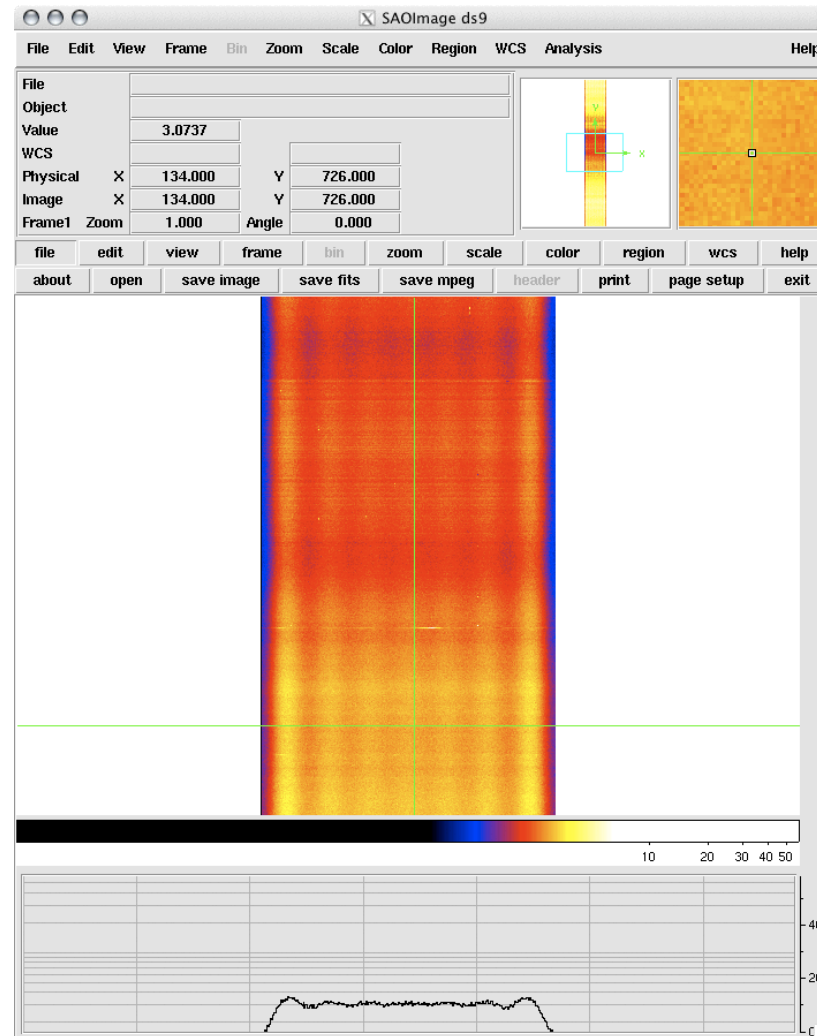
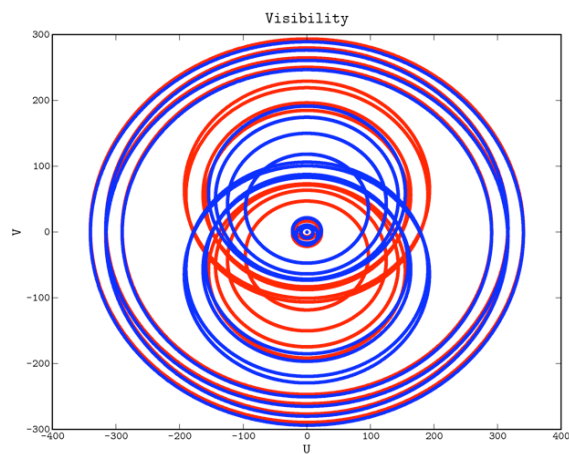
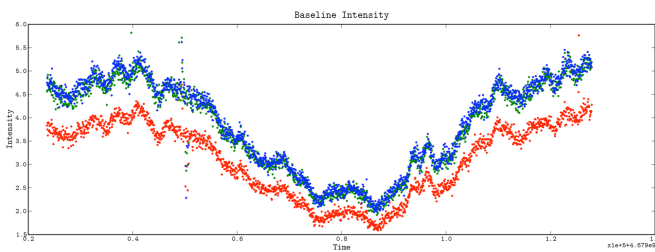
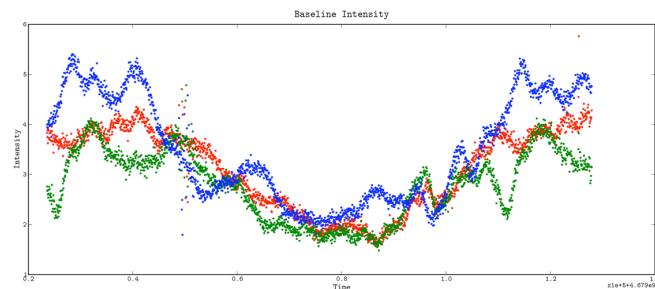
- AIPS++ measurement sets, tables (*CASACORE*)
- FITS images, tables (*CFITSIO*)
- HDF5 tables, image cubes (*HDF5IO*)
- Raw telemetry formats (*TBB, beam-formed, etc..*)
- LOPES, ROOT, etc...

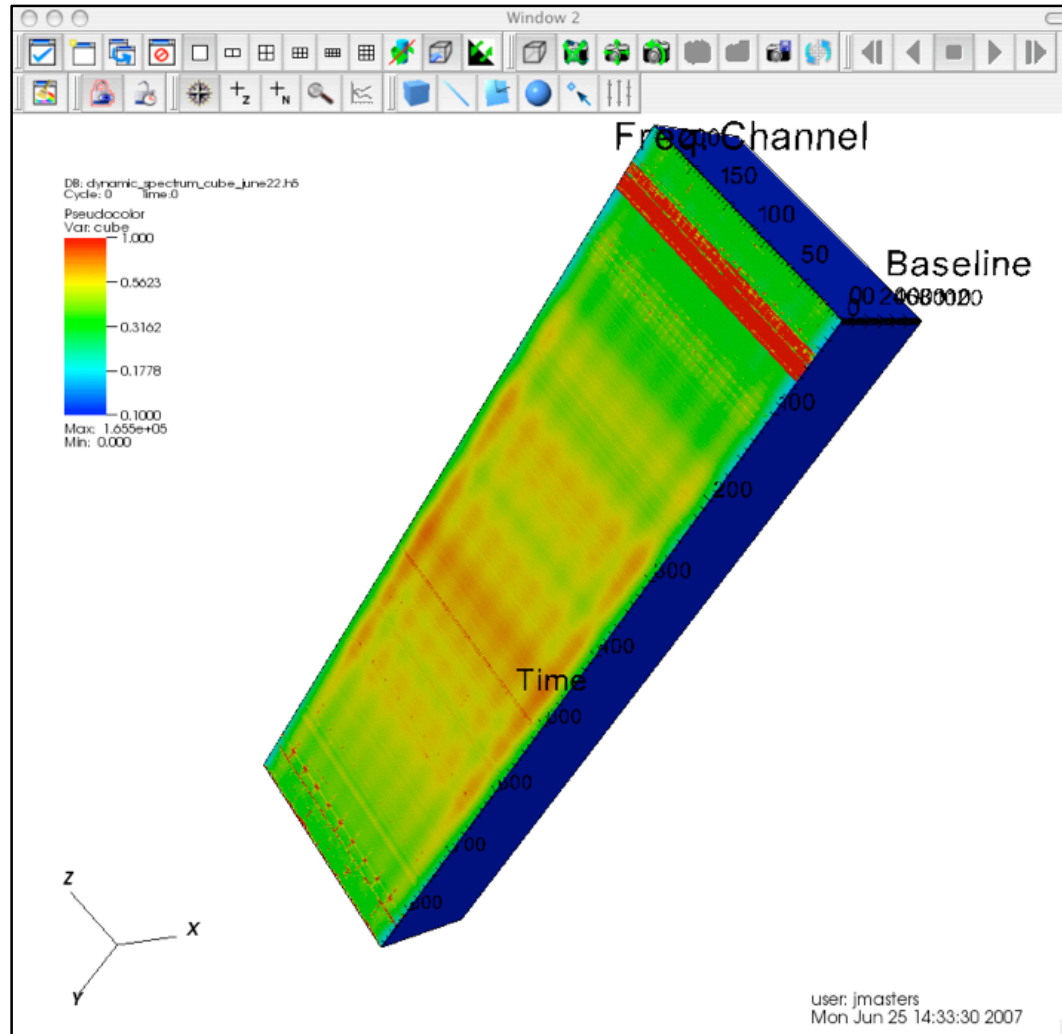
*Need formats
for metadata
and calibration
products*

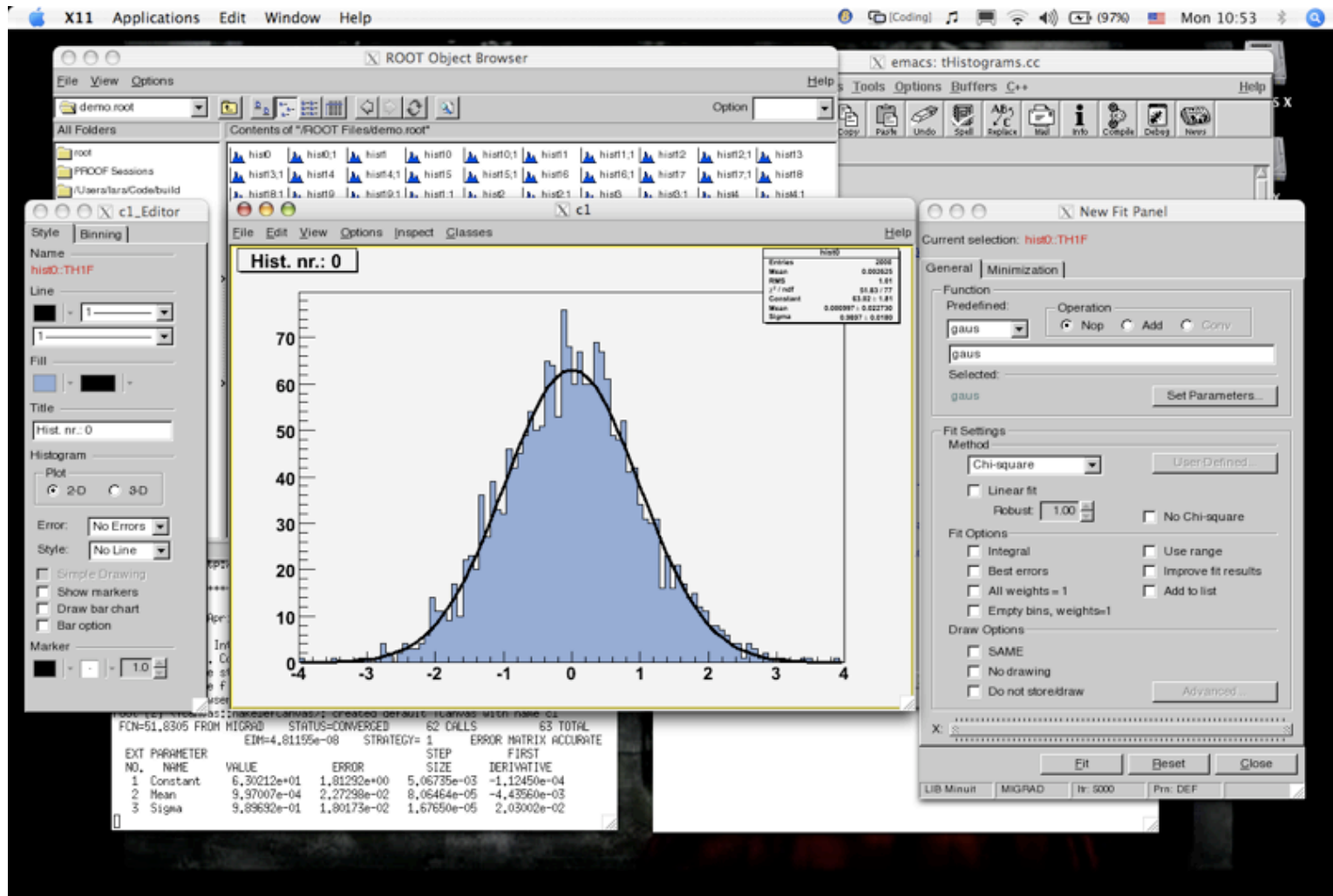


- DAL under development
 - Beta version of HDF5 I/O ~Now
 - Full I/O capabilities ~Q3 07
 - Slicing/Collapsing ~Q4 07
 - Streaming, Parallel I/O ~Q1/Q2 08

⇒ *Used for CS1 TBB time series data this summer*
- Additional core libraries
 - Parallelization Layer (OpenMP-aware)
 - Data visualization (DVL), General math library (GML)
 - Need better requirements (especially for DVL)
 - Beta version(s) ~Q4 07/Q1 08
- Tools
 - Reprocessing (metadata inspection, flagging, visualization)
 - Analysis (source detection+characterization, mosaicing, visualization)
 - Currently compiling specs. (inputs welcomed!)







The screenshot displays the ROOT software environment. The main window shows a histogram titled "Hist. nr.: 0" with a Gaussian fit overlaid. The histogram has a blue fill and a black line. The x-axis ranges from -4 to 4, and the y-axis ranges from 0 to 70. A small table in the top right corner of the histogram window provides fit statistics:

Entries	Mean	RMS	χ^2/NDF	Constant	Mean	Sigma
2000	0.00325	1.81	51.81177	61.82 ± 1.81	0.00397 ± 0.022730	0.9897 ± 0.0103

Below the histogram, a terminal window shows the output of the MIGRAD minimization algorithm:

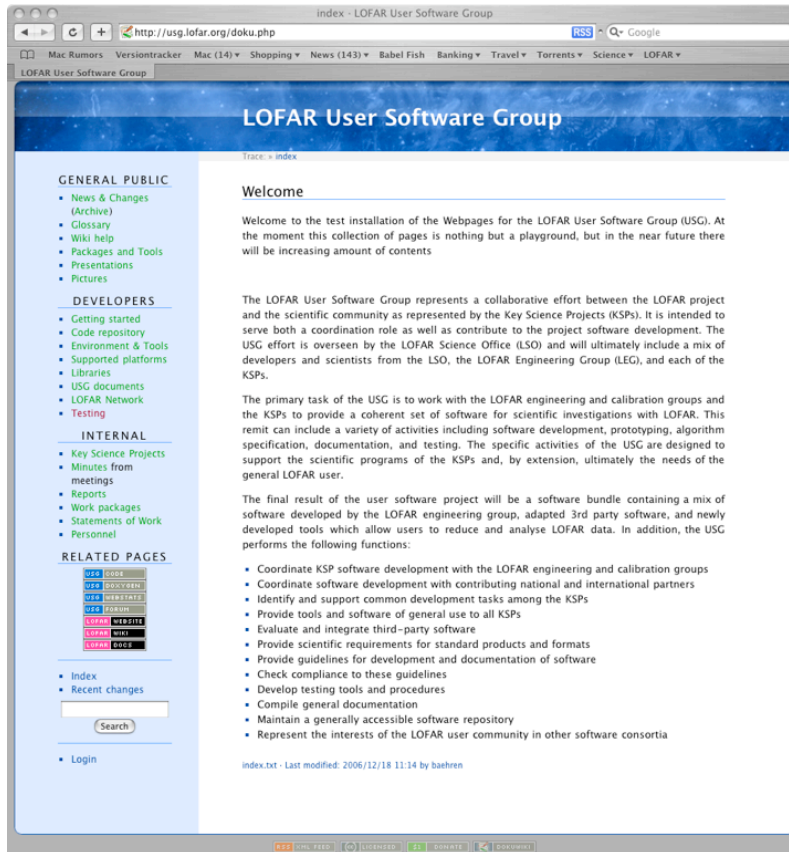
```

FCN=51.8305 FROM MIGRAD STATUS=CONVERGED 62 CALLS 63 TOTAL
           EIM=4.81155e-08 STRATEGY= 1 ERROR MATRIX ACCURATE
EXT PARAMETER          VALUE      ERROR      STEP      FIRST
NO.  NAME              VALUE      ERROR      SIZE      DERIVATIVE
 1  Constant            6.30212e+01  1.81292e+00  5.06735e-03  -1.12450e-04
 2  Mean                 9.97007e-04  2.27298e-02  8.06464e-05  -4.43560e-03
 3  Sigma                9.89692e-01  1.80173e-02  1.67650e-05  2.03002e-02
    
```

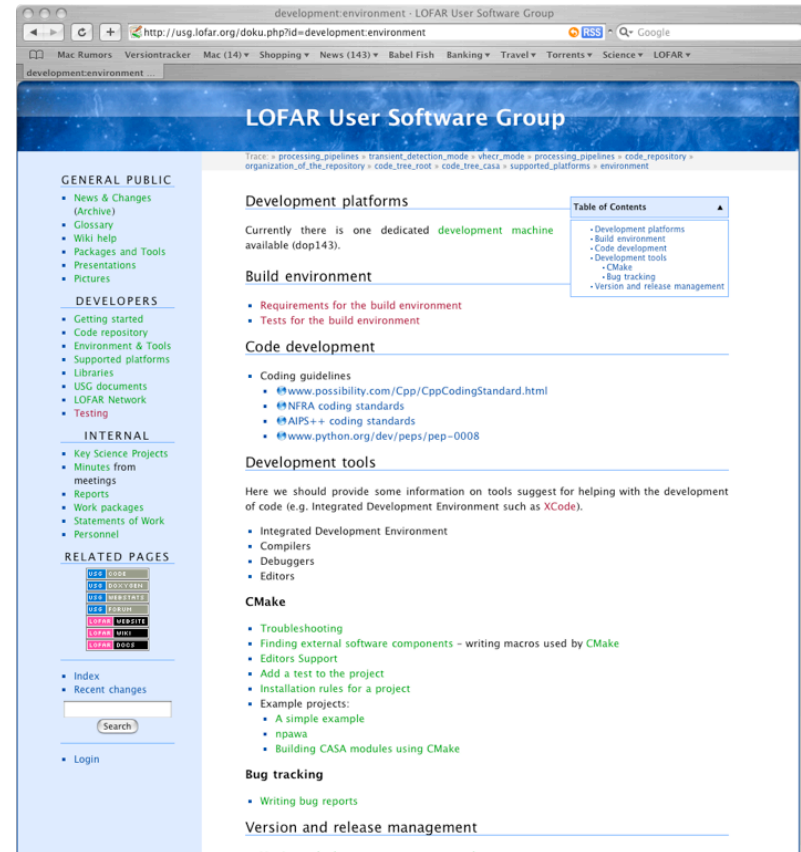
On the right, the "New Fit Panel" is open, showing the "gaus" function selected for fitting. The "Fit Settings" section shows "Chi-square" as the method, and "Robust" is set to 1.00. The "Fit Options" section includes checkboxes for "Integral", "Best errors", "All weights = 1", "Empty bins, weights=1", "Use range", "Improve fit results", and "Add to list". The "Draw Options" section includes checkboxes for "SAME", "No drawing", and "Do not store/draw".

Support Activities

- Established USG web server
 - Online now
 - Some developer's documentation available
- Development and porting platforms
 - USG development machine available (SUSE 10.1 +VMWare)
 - 3 Porting machines for Debian, RedHat, and Fedora available
- Software repository available
 - Preliminary build environment defined
 - Preliminary code tree defined
 - Available to USG/KSP developers now
- Coming soon
 - User contributed code area
 - Bug tracking, Discussion forum
 - Build and testing framework



The screenshot shows the homepage of the LOFAR User Software Group website. The browser address bar displays `http://usg.lofar.org/doku.php`. The page features a blue header with the group name and a navigation menu on the left. The main content area includes a 'Welcome' message, a description of the group's collaborative effort, and a list of 'RELATED PAGES' such as 'USG CODE', 'USG DOCUMENTS', and 'USG LIBRARIES'. A search bar and a 'Login' link are also visible at the bottom.



The screenshot shows the 'development:environment' page of the LOFAR User Software Group website. The browser address bar displays `http://usg.lofar.org/doku.php?id=development:environment`. The page is structured with sections for 'GENERAL PUBLIC', 'DEVELOPERS', and 'INTERNAL'. Key sections include 'Development platforms', 'Build environment', 'Code development', 'Development tools', 'CMake', 'Bug tracking', and 'Version and release management'. A 'Table of Contents' sidebar is visible on the right side of the page.

<http://usg.lofar.org>

- Focus on real-time pipelines
- Finalize standard data products
- Define metadata formats and interfaces
- Refine library and tool requirements
- Produce development timelines
- Preliminary release schedule

Demos

John Swinbank ⇒ *Transient detection pipeline*

Joe Masters ⇒ *Data Access Library*