Herschel Data Analysis Guerilla Style:
Keeping flexibility in a system with long development cycles

Bernhard Schulz

NASA Herschel Science Center

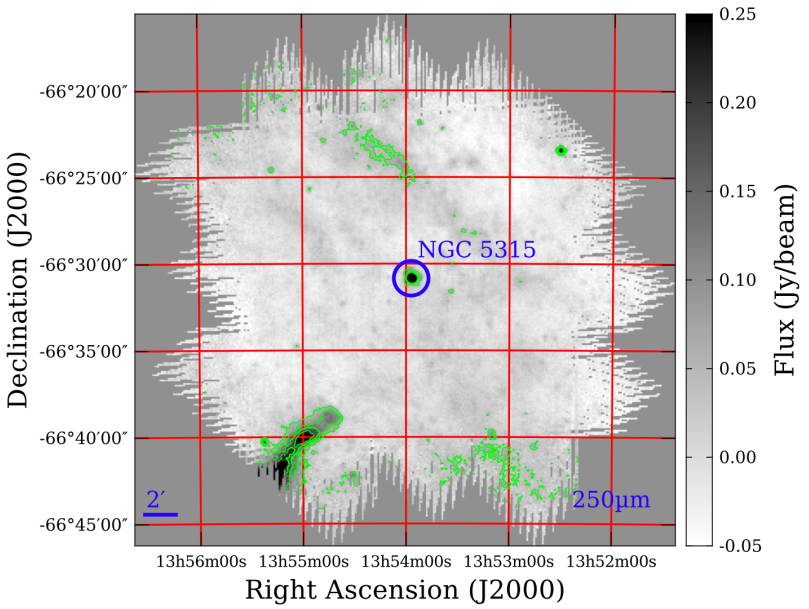
HCSS

- The Herschel Common Science System (HCSS) software accompanied the development of the Herschel mission through all phases of development.
- Large multinational development effort in Java with >5*10^5 lines of code and >200 contributors
- Functions:
  - scientific proposal submission
  - observation scheduling,
  - spacecraft and instrument commanding
  - data downlink
  - data processing
  - instrument calibration
  - data archival

esa HiFi ICC nhsc PACS SPIRE

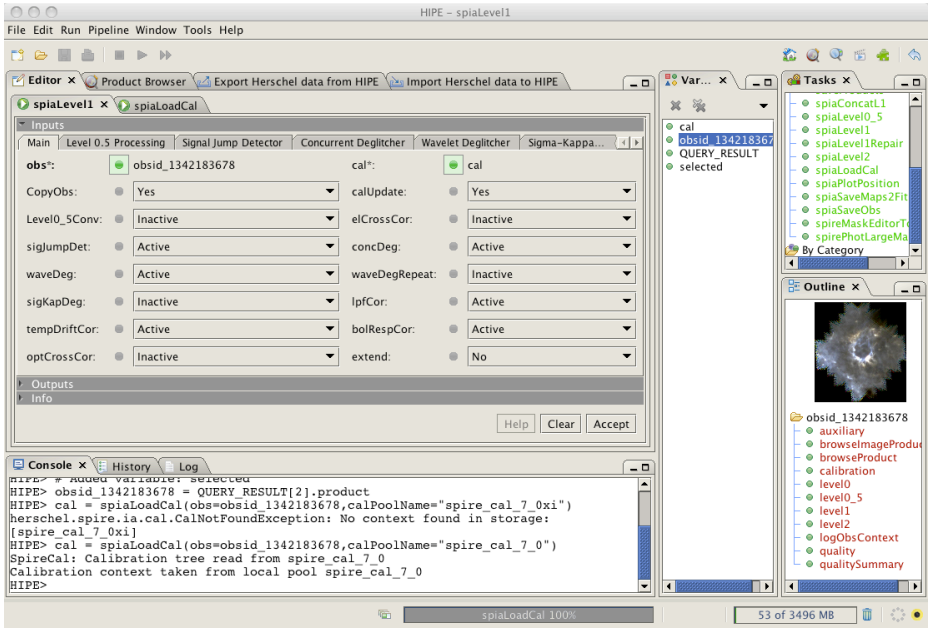
Data Processing

- Jython is used as script language to invoke Java modules
- Many libraries:
  - Numerics Package with up to 5 dim arrays, fitting etc.
  - Plotting Package
  - Object Oriented Database Access La
  - Toolboxes: Mapping, Spectroscopy, FITS IO, etc…
  - Flight dynamics library (coordinates etc.)
- Other cool features
  - Task Framework
  - Data Products have FITS file equivale
  - Calibration Source Database
  - Debugger
- HIPE: Interactive programming environment similar to Eclipse

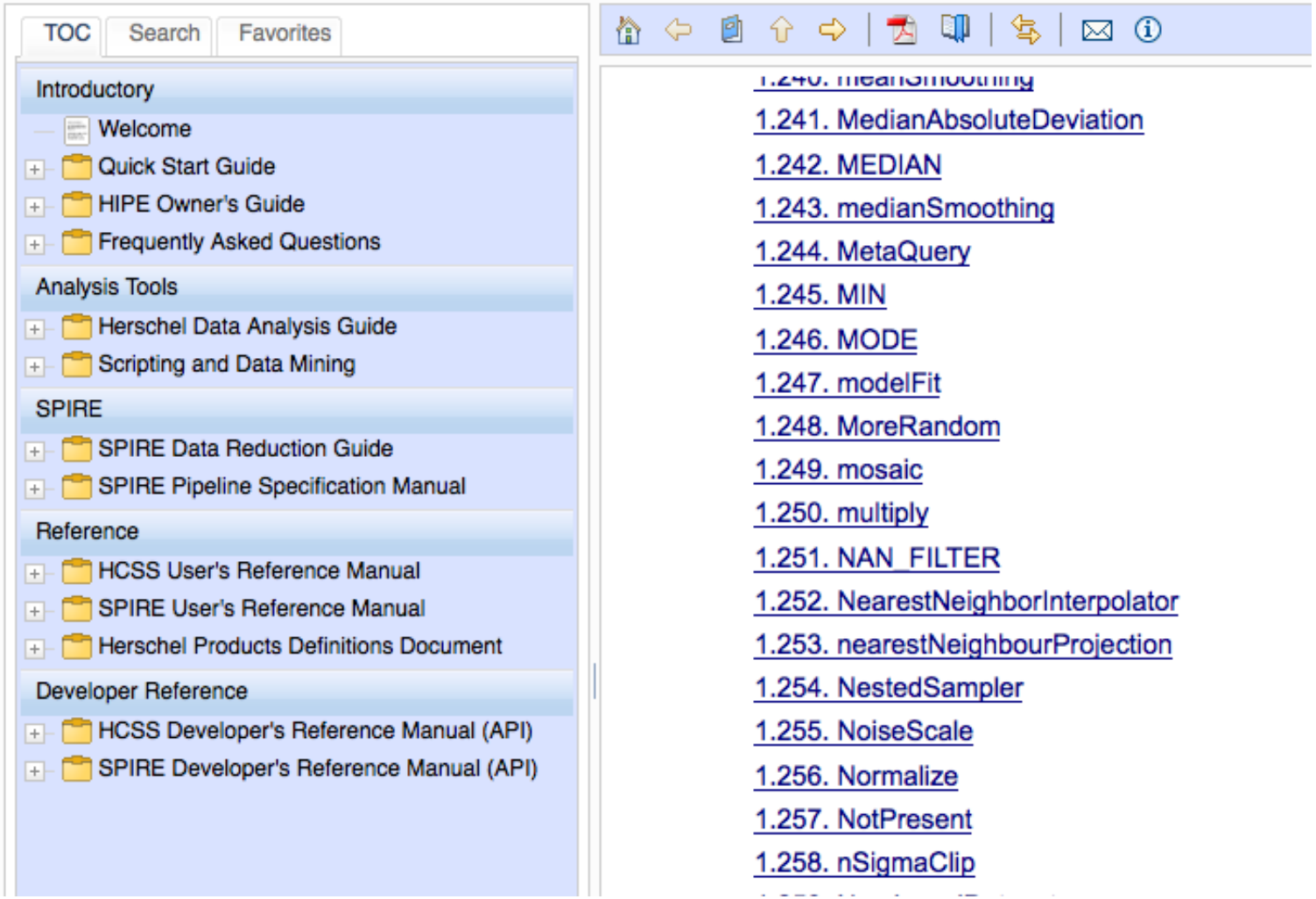


System Development

- The Herschel Science Centre (ESA), the Instrument Control Centres (HIFI, PACS and SPIRE) and the NHSC jointly manage and contribute to the Herschel Data Processing System
- All releases can be downloaded via
  - http://herschel.esac.esa.int/HIPE_download.shtml
- Latest HIPE developer releases are available via
  - http://herschel.esac.esa.int/CIB_disclaimer.html
- Many intermediate developer builds on a daily basis
- Major new version so far every 3 months
- Now changing to 6 months cycle



Libraries/Documentation



Early Issues

- In the beginning the learning curve was really steep
- Hard to keep track of changes
- Highly nested data structures produced very complex Jython expressions to access data in memory `print obs.refs["level1"].product.refs[2].product["signal"]["PSWE8"].data[0]`
- Object oriented database system relatively complex
- Need for instrument builders to use a system that is in the process of development
- No time to make things “user friendly”

The three data analysis choices

- Pipeline processing
  - Easy and straight forward
  - No flexibility
- Editing and running a script
  - Sophisticated and long learning curve
  - Full flexibility
- Interactive analysis with GUIs
  - Relatively easy to learn
  - Limited flexibility
- Straight pipeline results may be good enough for science analysis at a future date, but not yet.
- For astronomers with limited resources to learn the system, the GUI IA seems to be the optimal choice

The Dilemma

- Need for rapid results during instrument performance verification and calibration
- Pipelines only available as Jython scripts
- Need to simplify the system for “normal” astronomers
- Need to reduce potential for human error during data processing
- Turnaround cycles slow and decision making slow with many contributing parties
- Ideological divide between “Command line fundamentalists” and “GUI Geeks”

Guerilla Tactics

- If you are impatient: Do it yourself!
- At some point one realizes:
  - All the building blocks are there!!
- Take a weekend or two and have fun!

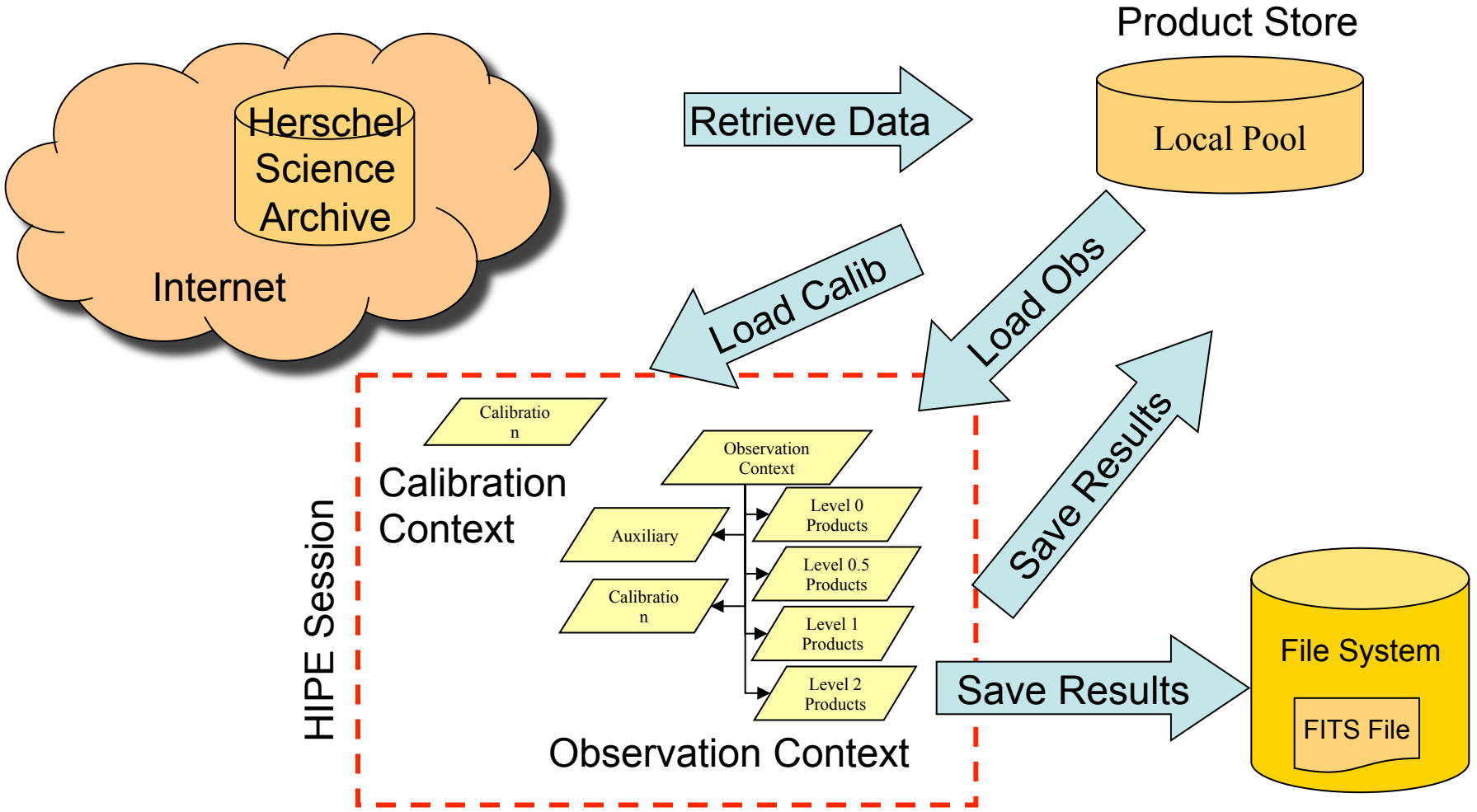
  ….provided you have such to spare…

- Result: SPIA (SPIRE Photometer Interactive Analysis)

Tasks in SPIA

Task Name	Category	Description
spiaCalCopyHsa	SPIRE	Loads specified calibration context from HSA into session and saves it to local store as default calibration context.
spiaLoadCal	SPIRE	Loads calibration context into session. Option to get newest calibration from own local store or potentially older calibration that came originally with the data.
spiaLoadUrn	SPIRE	Loads observation context from local store identified by URN.
spiaLoadObs	SPIRE	Loads SPIRE observation context from local store identified by obsid. Optional path to different local store directory can be given (somewhat outdated, use product browser instead).
spiaCopyHsa	SPIRE	Copy observation from HSA to local store.
spiaLevel0_5	SPIRE	Reduce SPIRE photometer scan map data from Level 0 to Level 0.5 and run jump detectors and deglitchers without the “repair” option. Most parameters are GUI accessible. The masks can be inspected and corrected afterwards using the spireMaskEditorTool.
spiaLevel1Repair	SPIRE	Reduce SPIRE photometer data from Level 0.5 to Level 1, assuming that spiaLevel0_5 was used before to mask glitches and jumps in the Level 0.5 data. This module only “repairs” previously flagged glitches and completes processing to Level 1.
spiaLevel1	SPIRE	Reduce SPIRE photometer scan map data from Level 0 to Level 1. Many parameters especially of the deglitchers are GUI accessible. This reflects the standard pipeline processing if parameters are kept at their default settings. This module detects and repairs glitches in one go on Level 1 but doesn’t set the masks at Level 0.5 nor allows for intermediate inspection at Level 0.5. To inspect and potentially correct glitch and jump detections you can run the tasks spiaLevel0_5 and spia_Level1Repair instead.
spiaLevel2	SPIRE	Reduce SPIRE photometer scan map data from Level 1 to Level 2 maps. Most parameters are GUI accessible. Allows combination of up to 5 Level1 contexts into one map and does allow for different baseline removal algorithms.
spiaSaveObs	SPIRE	Save observation to pool. Options for different path to local store directory and saving of Level 1 or Level 2 only stores.
spiaSaveMaps2Fits	SPIRE	Save Level 2 maps to FITS files.
spiaConcatL1	SPIRE	Concatenate selectable Level1 signal timelines from all building blocks into one table dataset that can be easily inspected with TableOverPlotter.
spiaPlotPosition	SPIRE	Plots the Level 1 track of a detector over one of the Level 2 maps of an observation context. The context must contain both Level 1 and Level2 products.

SPIA Workflow

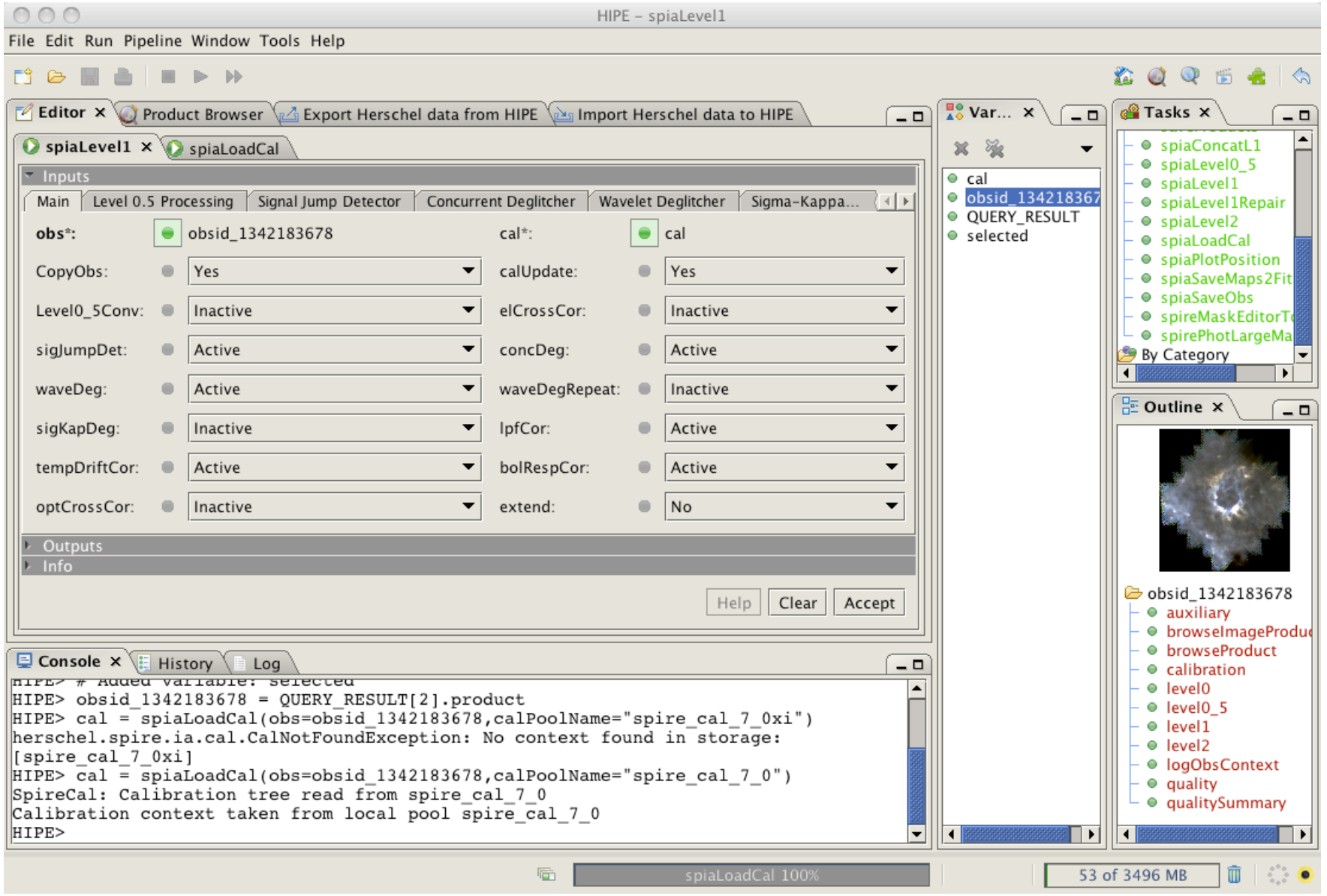


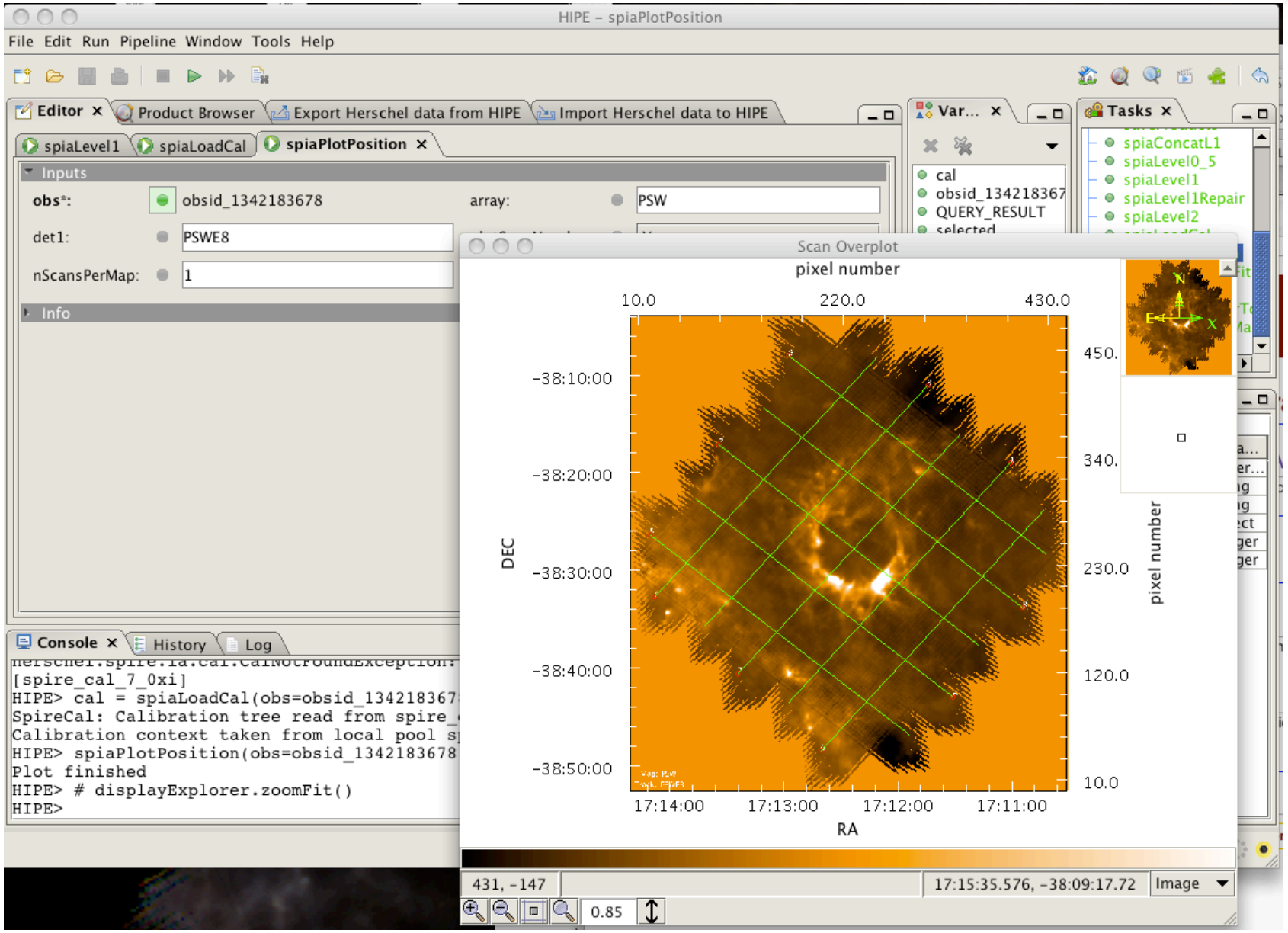
Task Framework

- Clear structure requiring definition of input and output parameters with type, default values, tooltips, help
- Integrated into HIPE (registering)
- Common look default GUI
- Records history
- Tasks can be executed as subroutines in other tasks or scripts for batch processing
- Disadvantage: static GUI that can not change based on user input

```
# Import task framework classes.
from herschel.ia.task.all import *
# Import data structures from java classes needed
from herschel.ia.dataset.all import *
from herschel.ia.numeric.all import *
# Import our algorithm
from average import average

class Average(JTask):
    def __init__(self):
        # This is the constructor of the Task. Initialize the superclass:
        JTask.__init__(self, "averageTask")
        # Define the *Signature* using calls to the addTaskParameter method
        # The following lines define one TableDataset input parameter, called table
        # And one output parameter of type DoubleId.
        p = TaskParameter("table",valueType = TableDataset, mandatory = 1)
        self.addTaskParameter(p)
        p = TaskParameter("result",valueType = DoubleId, type = OUT)
        self.addTaskParameter(p)
    # Here we define the execute method:
    def execute(self):
        # Define the code to be *executed*: Call our average method here
        # The input parameter 'table' is passed as the argument of the function
        self.result = average(self.table)
```


HIPE – spiaLevel1
File Edit Run Pipeline Window Tools Help
Editor
Product Browser
Export Herschel data from HIPE
Import Herschel data to HIPE
spiaLevel1
spiaLoadCal
Inputs
Main
Level 0.5 Processing
Signal Jump Detector
Concurrent Deglitcher
Wavelet Deglitcher
Sigma-Kappa...
obs*:
obsid_1342183678
cal*:
cal
CopyObs:
Yes
calUpdate:
Yes
Level0_5Conv:
Inactive
elCrossCor:
Inactive
sigJumpDet:
Active
concDeg:
Active
waveDeg:
Active
waveDegRepeat:
Inactive
sigKapDeg:
Inactive
lpfCor:
Active
tempDriftCor:
Active
bolRespCor:
Active
optCrossCor:
Inactive
extend:
No
Outputs
Info
Help
Clear
Accept
Var...
cal
obsid_134218367
QUERY_RESULT
selected
Tasks
spiaConcatL1
spiaLevel0_5
spiaLevel1
spiaLevel1Repair
spiaLevel2
spiaLoadCal
spiaPlotPosition
spiaSaveMaps2Fit
spiaSaveObs
By Category
Outline
obsid_1342183678
auxiliary
calibration
level0
level0_5
level1
level2
logObsContext
quality
qualitySummary
Console
History
Log
HIPE> obsid_1342183678 = QUERY_RESULT[2].product
HIPE> cal = spiaLoadCal(obs=obsid_1342183678,calPoolName="spire_cal_7_0xi")
herschel.spire.ia.cal.CalNotFoundException: No context found in storage:
[spire_cal_7_0xi]
HIPE> cal = spiaLoadCal(obs=obsid_1342183678,calPoolName="spire_cal_7_0")
SpireCal: Calibration tree read from spire_cal_7_0
Calibration context taken from local pool spire_cal_7_0
HIPE>
spiaLoadCal 100%
53 of 3496 MB


HIPE – spiaPlotPosition
File Edit Run Pipeline Window Tools Help
Editor
Product Browser
Export Herschel data from HIPE
Import Herschel data to HIPE
spiaLevel1
spiaLoadCal
spiaPlotPosition
Inputs
obs*:
obsid_1342183678
array:
PSW
det1:
PSWE8
nScansPerMap:
1
Info
Var...
cal
obsid_134218367
QUERY_RESULT
selected
Tasks
spiaConcatL1
spiaLevel0_5
spiaLevel1
spiaLevel1Repair
spiaLevel2
Scan Overplot
pixel number
10.0
220.0
430.0
-38:10:00
-38:20:00
-38:30:00
-38:40:00
-38:50:00
DEC
450.
340.
230.0
120.0
10.0
pixel number
17:14:00
17:13:00
17:12:00
17:11:00
RA
Console
History
Log
[spire_cal_7_0xi]
HIPE> cal = spiaLoadCal(obs=obsid_134218367
SpireCal: Calibration tree read from spire_
Calibration context taken from local pool s
HIPE> spiaPlotPosition(obs=obsid_1342183678
Plot finished
HIPE> # displayExplorer.zoomFit()
HIPE>
431, -147
17:15:35.576, -38:09:17.72
Image
0.85

Availability

- Distribution from NHSC Wiki
  - HIPE Plugin
  - User's Manual
  - Video Tutorials
  - Publication ADASS 2010 proceedings
    - http://arxiv.org/abs/1101.1284

- https://nhscsci.ipac.caltech.edu/sc/index.php/Spire/SPIA

Conclusive Remarks

- HCSS has become a system with huge potential
- Parallel development of software system and hardware is a successful concept
- Needs of three communities must be balanced
  - Developers, Instrument scientists, Astronomers
- You can not start early enough in a mission with planning for the software environment.
- Flexibility in a system is very useful
  - Helps to make up for omissions in the plan
  - Better: if a clear vision for limits of scope and need dates of functionalities are established early.
- A user friendly environment that hides most complexities must be part of the plan