
# Open-source software usage in a geophysical software and services company

Ioan Vlad and Charles Kovacs

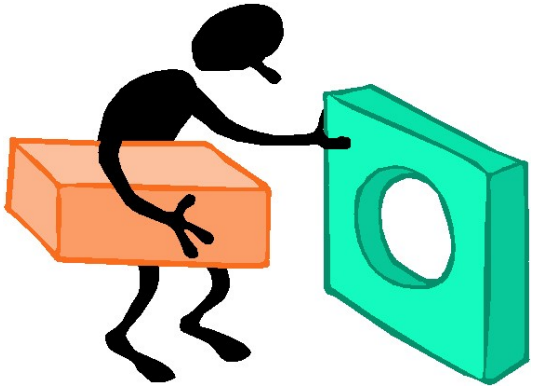
Fusion Petroleum Technologies, Inc.

PTTC Workshop, Houston, TX, 2011-06-17

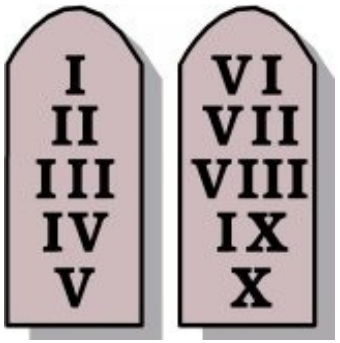
# Background

Common misconceptions:

- Open-source software (OSS) is *incompatible* with the activity of a commercial software and services company at high levels of the solutions stack



- Open-source package <insert name here> currently provides a complete solution and should be used exclusively



# Overview

We will:

- Show that open-source software can be *highly useful* in a commercial software company
- Discuss the fundamental relative advantages and the "ecological niche" of SU, SEPlib, Madagascar and CPSeis in Fusion's software and services environment
- Discuss architecture improvements that can benefit some packages

# OSS usage

- The “hand tools” of the software developer and of the command-line user
- Most of the time used as standalone programs
- When used as programming framework it is most often for prototyping
- In production only in a few particular cases

# Why is it used at all?

(1) Because people are usually already familiar with it when they start in a new job

- Then, they teach their co-workers how to use their tools

- Retraining is expensive
- Learning a new set of tools is more than reading a manual or reference
- “The young man knows the rules, the old man knows the exceptions”
- Smart people value portable knowledge

# Why is it used at all?

(2) Because it has usually fewer bugs than custom proprietary frameworks

- Why? # of users (and developers) vs. # of lines of code

- "Many eyes make all bugs shallow": actually works when most users are also coders

# Why is it used at all?


(3) Because it has usually better documentation than custom proprietary frameworks

- Again: # of users (and developers) / # of lines of code

- Google beats any in-house search engine

- Wikis make it easy for users to contribute to documentation

# Why is it used at all?

(4) Because its portability means easier technology transfer from external collaborators who use it

- No more porting software

- Interns and new hires can be productive from day 1

# Is it OK to mix and match?

OSS and proprietary software?

- ... BSD-style licensed packages (SU, SEPlib): Yes, minimal issues
- ... GPL-ed packages (Madagascar): Yes, as long as all:
  - Calls take place over the public interface (no linking occurs)
  - No code copying occurs
- There is no point to re-write in-house a commodity tool, if a satisfactory, stable OSS one exists
- Cheaper to contribute improvements upstream than to maintain a fork or to write an in-house version

# Why is it not used more?

- Because of its limitations.
- **SU/SEPlib/Madagascar**: optimized for ease of use and flexibility, not data volume
  - This is not a bad thing! Just a different animal
- **CPSeis**: its "*assign IP rights to CoP if you want to participate in public development*" Contributor Agreement clause precludes the existence of a large, active open-source community

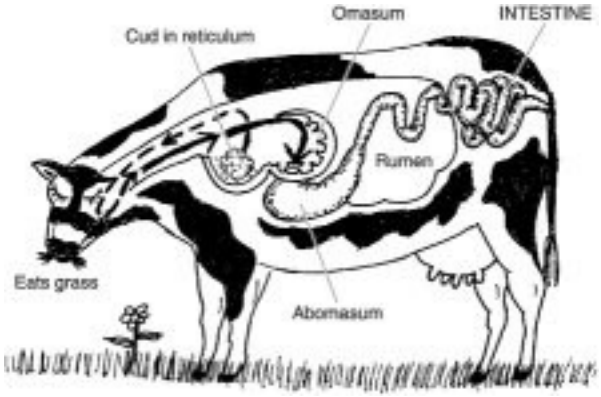
# The data volume issue

- When input data is 20Tb, it becomes crucial to avoid making copies of the data and/or re-sorting it

Academic OSS packages

Big data

# Large datasets discussion




Filter-type processing program structure:

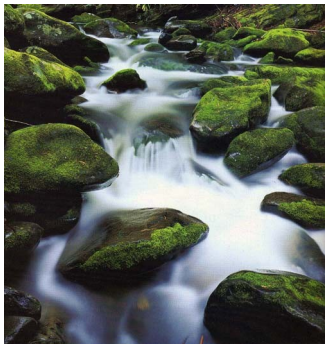
- "p" step: I/O parameters check
  - Preempts dimension / axis type / data type mismatches
- "a" step: allocation
  - A "work array" ensemble gets allocated
- "b" step: Actual computation
  - Loop reading from input into the work array, process, output
- "c" step: cleanup
  - deallocation, closing files, etc.

# Large datasets discussion

When chaining together multiple filters, the flow can proceed as:

- p1-a1-b1-c1 | p2-a2-b2-c2 | p3-a3-b3-c3
  - SU/SEPlib/Madagascar (“standalone-style”)
  - Easy debugging – just redirect output to file / graphics
- p1-p2-p3 -> a1-a2-a3 -> b1-b2-b3 -> c1-c2-c3
  - “Flow-based” style
  - Arco Benchmarking Suite/JavaSeis/CPSeis/GeoPRO
  - “p” step ensures no I/O mismatch error will occur midway through the flow
  - “a” step is done only once
  - “b” step can take days -- no need to keep pipes open for days and to access disk outside the I/O to the flow

# Flow-based architectures



- Superior for very large-scale *preprocessing*
- Work style: try various flow parameters on a data subset, then apply to entire volume
- Very suited for "canned" flows, in which the sequence of steps is the same, and some steps may be skipped, parameters may be varied, but sequence remains the same
- Attempts to use standalone-style (SU/SEPlib/m8r) architecture for preprocessing converge into the writing of one large monolithic utility that implements a flow... without the maintainable modularity of a flow-based architecture.
- Flows rarely used for expensive imaging algorithms, for which collect-and-QC steps are usually needed both before and after.

# Standalone-style architectures

- Thankfully, many problems in seismic imaging are data-parallel
- Standalone-style works well with external parallelization managers, which can also take care of queuing systems, restart in case of failure, etc:
  - Madagascar's Flow(split=), sfomp, sfmpi
  - CWP iTeam's Fork/Join
  - UBC's SlimPy
  - Fusion's Overlord
  - Apache Pig
  - Many others

# Standalone-style architectures

For large problems, need:

- Parameters to act on a subset of input, instead of running a separate windowing program
- Utilities that are flexible about data ordering in input, in order to eliminate re-sorting of data just to fit a program written a certain way.
- Domain decomposition across nodes (MPI) for problems too large to fit in memory
- Exploiting all the CPUs on a node with OMP
- Checkpointing
- Multiple levels of logging verbosity

# Standalone-style suites – relative differences


## Seismic Unix

- Many utilities for preprocessing, especially irregularly-sampled data handling
- Few bugs (large user base)
- Stable
- Data format more suited to preprocessing than imaging

## Madagascar

- Few bugs (test-driven!), maintainable, portable, active community
- Still evolving towards better functionality and even higher robustness, portability and maintainability (this is good, but less stable)

## SEPlib

- Stable, but buggy (small user base)
- Irregularly-sampled data handling in RSF-compatible format
- Backwards compatibility with legacy imaging workflows

# Dual architectures

- Is it possible to use the same codebase in both flow-mode and standalone-mode?
- Yes, if the code is structured in p-a-b-c subroutines, which can be placed in a library, then either:
  - (A) Organized into flows as needed by a driver automatically written by the flow builder, which then gets compiled and executed
  - (B) Called by an already-written standalone driver
  - The standalone driver is boilerplate code that many similar programs share. A lot of non-geophysics in it (OMP parallelization, for example)
  - Write only one driver, and pass it the name of the procedure to be run and its specific arguments. This can cut down dramatically on the # of executables (~800 in Madagascar!!). Less boilerplate code == good thing.
- Example: CPSeis Front End (CFE)

FUSION

# Other dual-architecture benefits

The “p” gatekeeper forces registration of parameters and I/O axes/dimensions (where this makes sense)

- The I/O dimensions and the parameter list (for self-doc) can be obtained by executing “p” with a certain flag, instead of having to parse the driver (m8r-style) and to hope the user did not move reading of parameters into some subroutine, for code reuse
- The user can still reuse code and put parameter reading in shared procedures, as long as they pass to “p” the resulting object
- The parameters are read only once, with a single default value. Allowing the parameter table to be read from anywhere, SEPlib-style, can lead to some interesting bugs
- Example of parameter registration that results in guaranteed-complete self-doc: Python's argparse module

# Other dual-architecture benefits

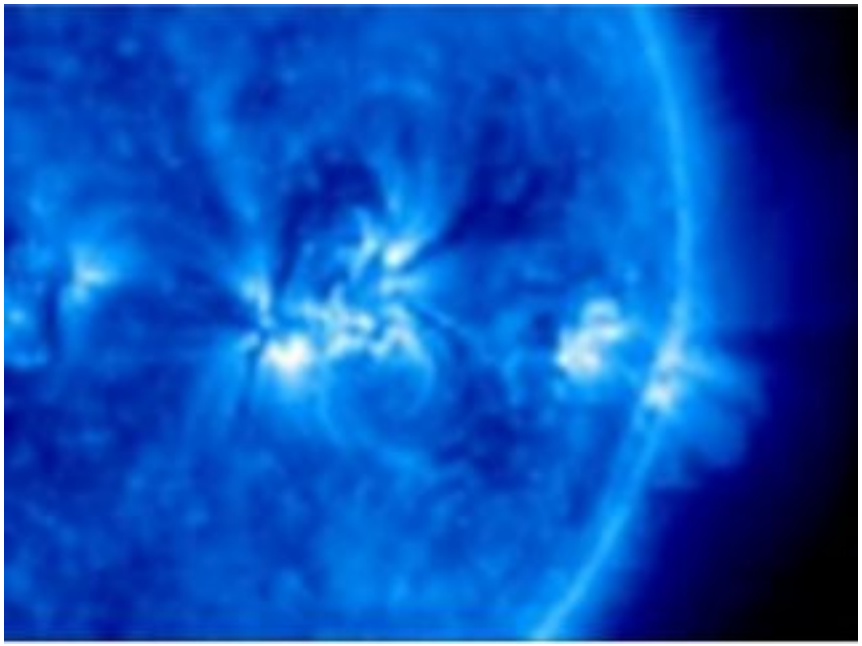
Forces separation of concerns

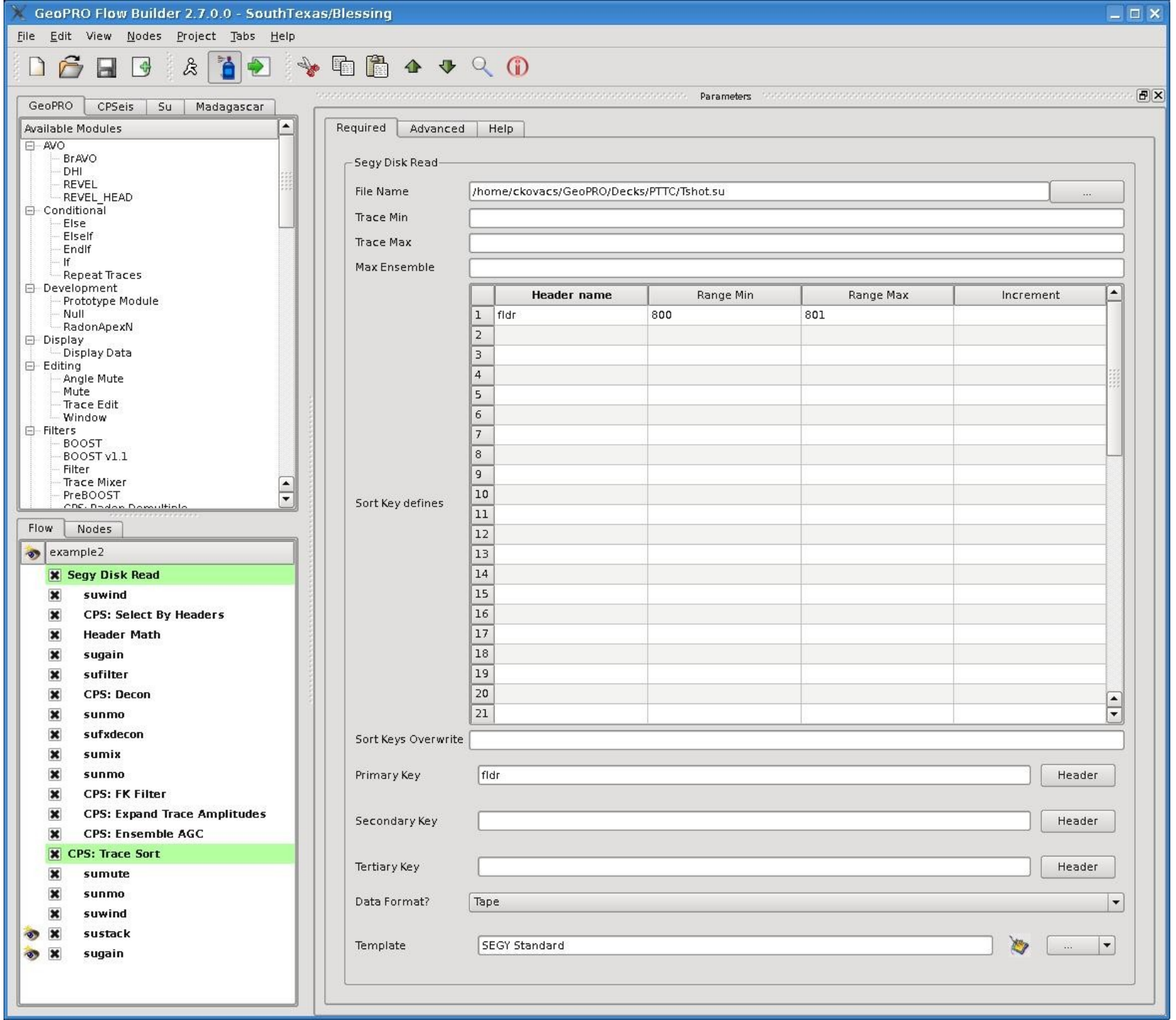
- Domain logic (i.e. geophysics) “lives” in the ”b” procedure
- The “b” procedure is independent from input data format
  - Inter-package code reuse, by simple linking with another library!
- Procedures are forced to live in libraries
  - Code reuse done right, instead of
    - copy/paste
    - copying source code files between directories
    - creating Unix symlinks

# Fusion's solution

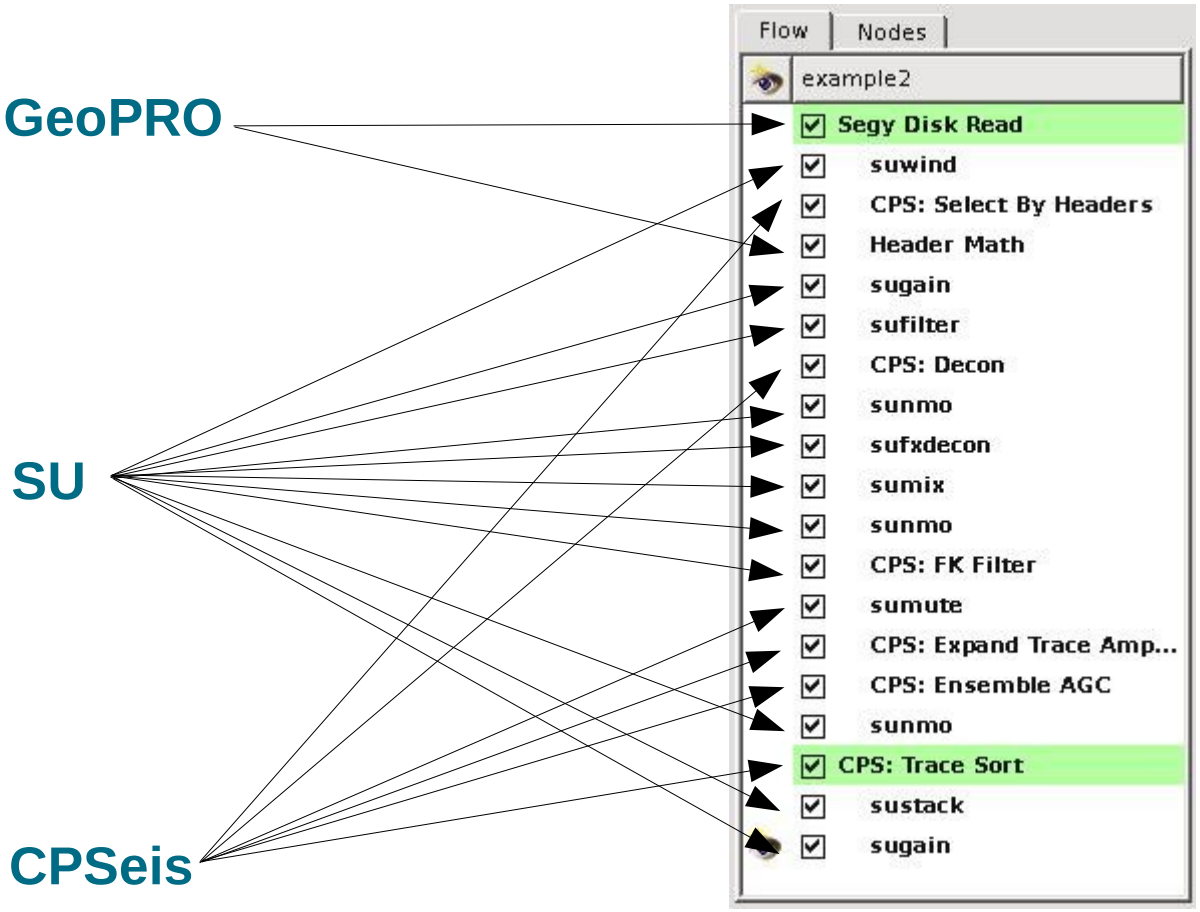
- Use SU/SEPlib/Madagascar/CPSeis:
  - As “hand tools” according to individual preferences
  - Integrated into GeoPRO's Flow Builder
  - Mainly for preprocessing
- Use standalone imaging programs managed by a fault-tolerant, PBS-compatible, Python-based parallelization manager (Overlord)
- Some of these standalone programs can be from Madagascar, SEPlib or SU.
  - Example: **`sffkamo`** parallelized by a wrapper script called by Overlord

# GeoPRO Flow Builder mix & match example




GeoPRO Flow Builder 2.7.0.0 - SouthTexas/Blessing
File Edit View Nodes Project Tabs Help
GeoPRO CPSeis Su Madagascar
Available Modules
AVO
BrAVO
DHI
REVEL
REVEL_HEAD
Conditional
Else
ElseIf
EndIf
If
Repeat Traces
Development
Prototype Module
Null
RadonApexN
Display
Display Data
Editing
Angle Mute
Mute
Trace Edit
Window
Filters
BOOST
BOOST v1.1
Filter
Trace Mixer
PreBOOST
Flow Nodes
example2
Segy Disk Read
suwind
CPS: Select By Headers
Header Math
sugain
sufilter
CPS: Decon
sunmo
sufxdecon
sumix
sunmo
CPS: FK Filter
CPS: Expand Trace Amplitudes
CPS: Ensemble AGC
CPS: Trace Sort
sumute
sunmo
suwind
sustack
sugain
Parameters
Required Advanced Help
Segy Disk Read
File Name
/home/ckovacs/GeoPRO/Decks/PTTC/Tshot.su
Trace Min
Trace Max
Max Ensemble
Sort Key defines
Header name
Range Min
Range Max
Increment
1 fldr 800 801
Sort Keys Overwrite
Primary Key
fldr
Header
Secondary Key
Header
Tertiary Key
Header
Data Format?
Tape
Template
SEGY Standard

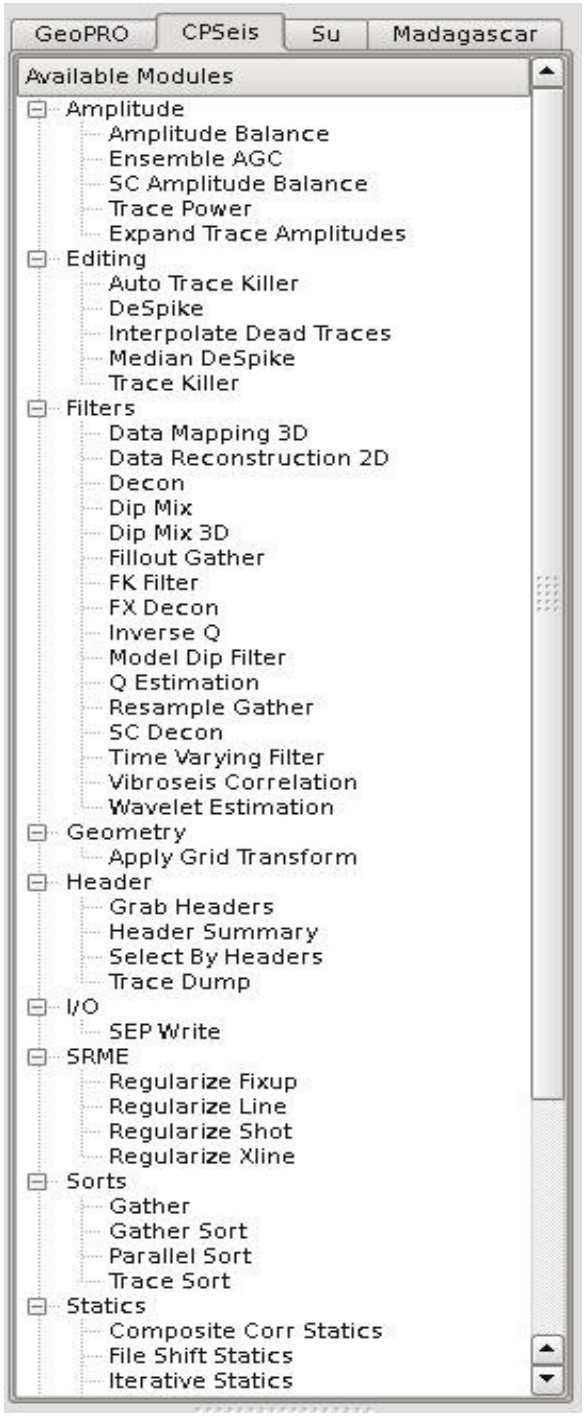
# Mix processing modules as needed!



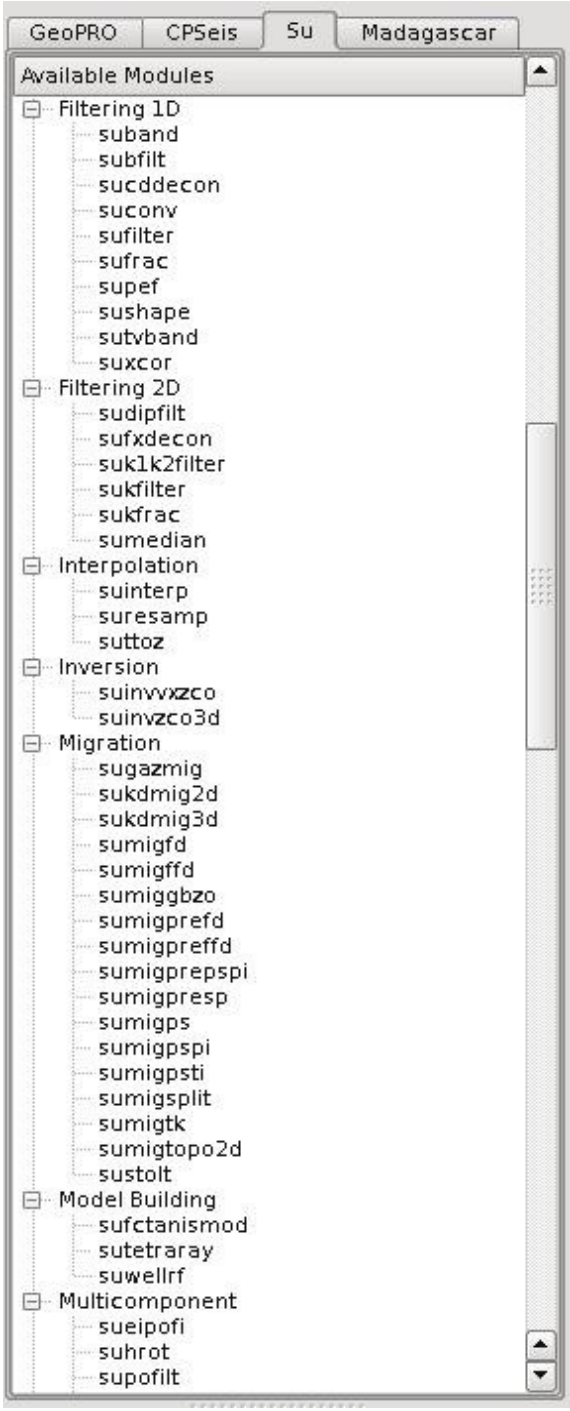



FUSION

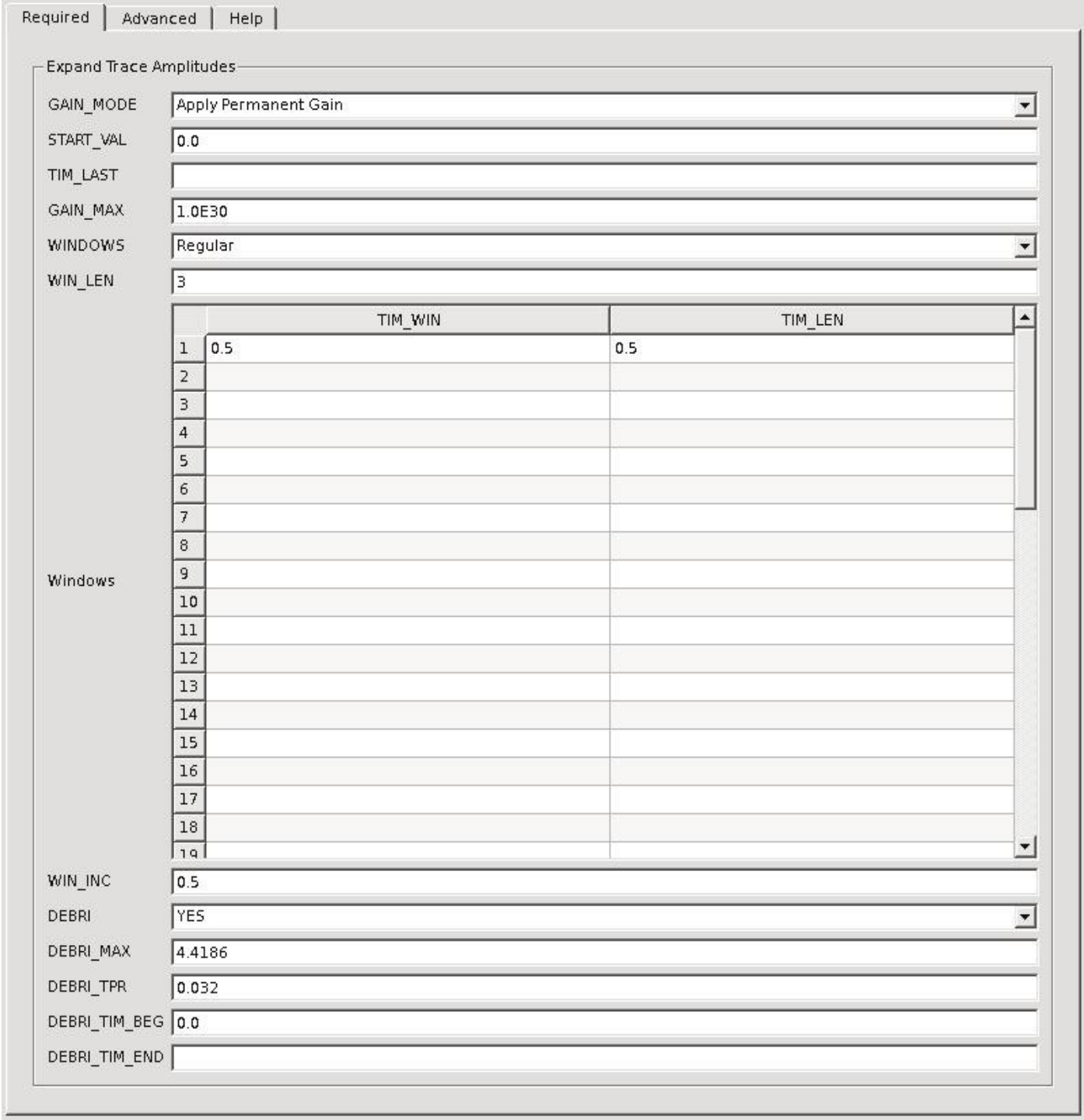
# CPSeis Modules




# SU Modules



# CPSeis Module Parameter UI



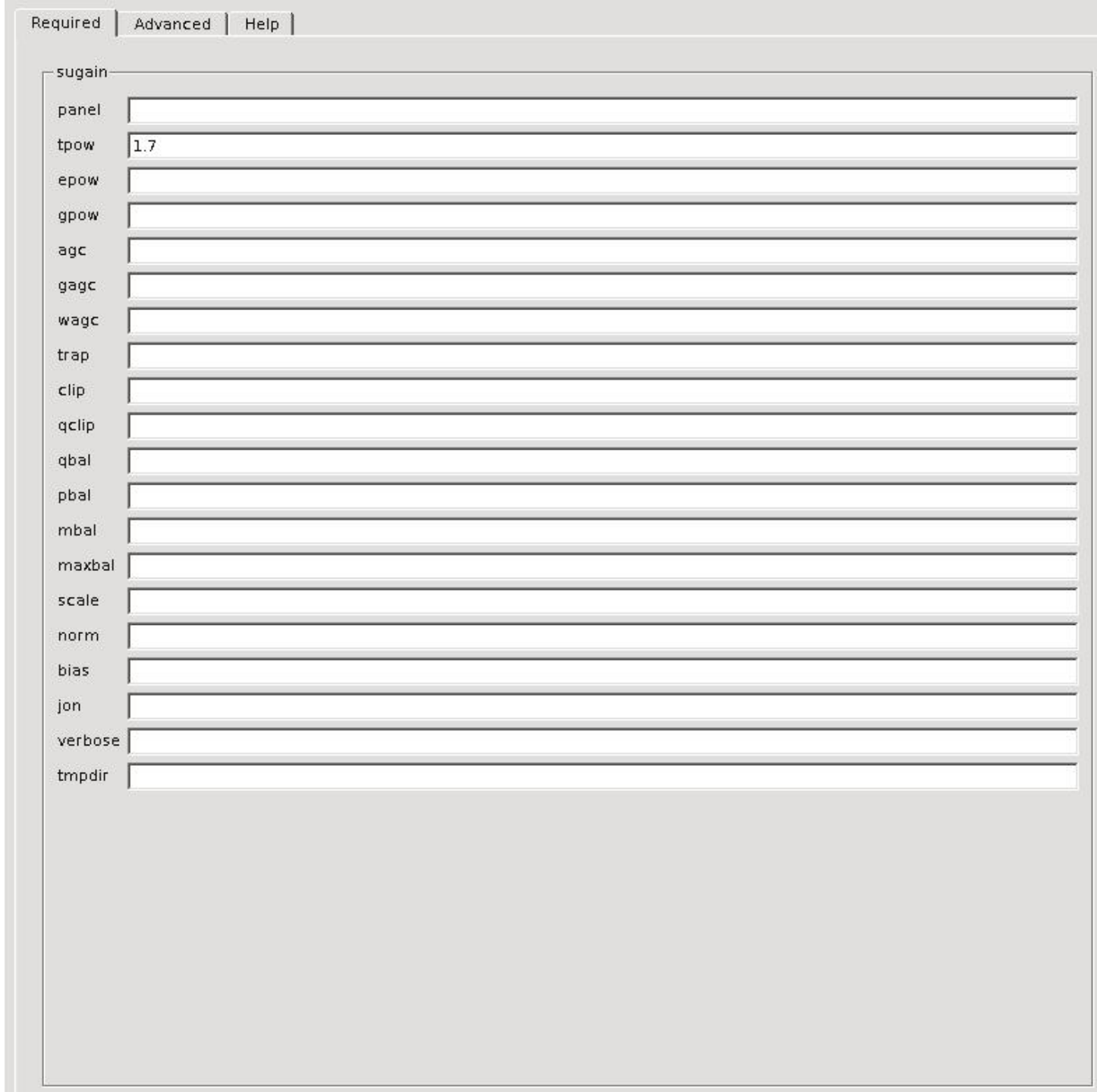
# CPSeis Module Help

Required | Advanced | Help

## Introduction

```
------------------------------------------------------------------------------
                        C P S    P R O C E S S

 Process name: XP   (eXPand amplitudes)
 Category    : Amplitude_Mod
 Written     : 1986-07-01  by: Bob Baumel and Richard Day
 Revised     : 2006-10-16   by: D. Glover
 Maturity    : production
 Purpose     : XP rescales trace samples to balance amplitudes.
 Portability: No known limitations.
 Parallel    : Yes

------------------------------------------------------------------------------
------------------------------------------------------------------------------
                           GENERAL DESCRIPTION

 XP is a single trace process that balances trace amplitudes and is the CPS
 version of industry standard AGC.  It is a two-step process, amplitude
 balance followed by an optional debrighten.  XP is normally used in
 structural processing. It does not preserve amplitudes.

 This process calls the XPUTIL primitive to do all the work.  See the XPUTIL
 primitive for further information.

 Amplitude Balance:

  XP calculates a gain function by first calculating the L1 NORM (mean
  absolute value) of the non-zero trace samples in each window.
  The gain function at the center of each window is the reciprocal of that
  window s mean and is linearly interpolated between window centers. (The gain
  function is constant above the first window center and below the last window
  center.)

  XP reduces the trace amplitude variation on a time-scale greater than the
  window length, while allowing some amplitude variation on a time-scale
  smaller than the window length.

 Debrighten:

  If you set DEBRI=YES, then debrightening is performed AFTER the XP amplitude
  balance.  The choice of debrightening threshold (DEBRI_MAX) does not depend
  on the absolute scale of the input data, as the amplitude balance attempts to
  scale the trace so that the average absolute value in each window is 1.

  Debrightening is performed as follows:  If any absolute values in the trace
  exceed the DEBRI_MAX threshold, then the largest peak is reduced to
  DEBRI_MAX, tapering the amount of reduction for distance DEBRI_TPR to either
  side of this peak.  This process is repeated until no values exceed
```

# SU Module Parameter UI



# SU Module Help

Required | Advanced | Help

## Introduction

```
SUGAIN - apply various types of gain to display traces

sugain < stdin > stdout [optional parameters]

Required parameters:
      none (no-op)

Optional parameters:
      panel=0         =1 gain whole data set (vs. trace by trace)
      tpow=0.0        multiply data by t^tpow
      epow=0.0        multiply data by exp(epow*t)
      gpow=1.0        take signed gpowth power of scaled data
      agc=0           flag; 1 = do automatic gain control
      gagc=0          flag; 1 = ... with gaussian taper
      wagc=0.5        agc window in seconds (use if agc=1 or gagc=1)
      trap=none       zero any value whose magnitude exceeds trapval
      clip=none       clip any value whose magnitude exceeds clipval
      qclip=1.0       clip by quantile on absolute values on trace
      qbal=0          flag; 1 = balance traces by qclip and scale
      pbal=0          flag; 1 = bal traces by dividing by rms value
      mbal=0          flag; 1 = bal traces by subtracting the mean
      maxbal=0        flag; 1 = balance traces by subtracting the max
      scale=1.0       multiply data by overall scale factor
      norm=1.0        divide data by overall scale factor
      bias=0.0        bias data by adding an overall bias value
      jon=0           flag; 1 means tpow=2, gpow=.5, qclip=.95
      verbose=0       verbose = 1 echoes info


 tmpdir=  if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Operation order:

out(t) = scale * BAL{CLIP[AGC{[t^tpow * exp(epow * t) * ( in(t)-bias )]^gpow}]}

Notes:
The jon flag selects the parameter choices discussed in
Claerbout's Imaging the Earth, pp 233-236.

Extremely large/small values may be lost during agc. Windowing
these off and applying a scale in a preliminary pass through
sugain may help.
```

# Visualization in VizPRO (movie)

# How about other OSS packages?

- We only talked about what we were familiar with
- This workshop discusses other packages as well
- Wikipedia's *List of free geophysics software* lists many more packages yet
- We are always looking forward to learning new things
- This is a big reason why we are attending this workshop

# Conclusions

- Open-Source Software is compatible and useful in the activity of a services and software company

- Various architectures and packages provide specific benefits
  - It is up to the user to mix, match and adapt according to his needs

- Dual-style architectures offer the convenience of standalone programs combined with the ability to process industrial-sized data volumes

# Acknowledgments



- Doug Hanson for details about CPSeis
- Tom Stoeckley for details about CPSeis and presentation feedback
- Fusion Petroleum Technologies for allowing the presentation of this material
- Fotolia, Wikimedia Commons and other image sites for illustrations