TKSU
Experiences with a graphical front end to SU

Jeff Thorson
Henry Thorson Consulting
henrythorson.com

Elements essential for interactively building a processing flow:

- Help on selecting modules (programs)
- Help on setting a module's run-time parameters
- Help on linking modules together into a “flow”
- Simple job management to test/run the flow
- Simple logging of module error messages

Processing flow

- Broad definition: a set of component programs ("modules") run as a group, communicating via files, pipes, etc. Modules may run sequentially.
- Narrow definition: a set of modules running concurrently, communicating via one-way pipes.

TKSU is limited in scope:

- An independent package not bundled with SU.
- Relies on an underlying package of modules (SU) which may or may not be installed. TKSU's output is a shell script that executes the flow.
- No included modules apart from a few special ones. Seismic display is done with SU modules.
- No batch management (apart from executing the shell script).

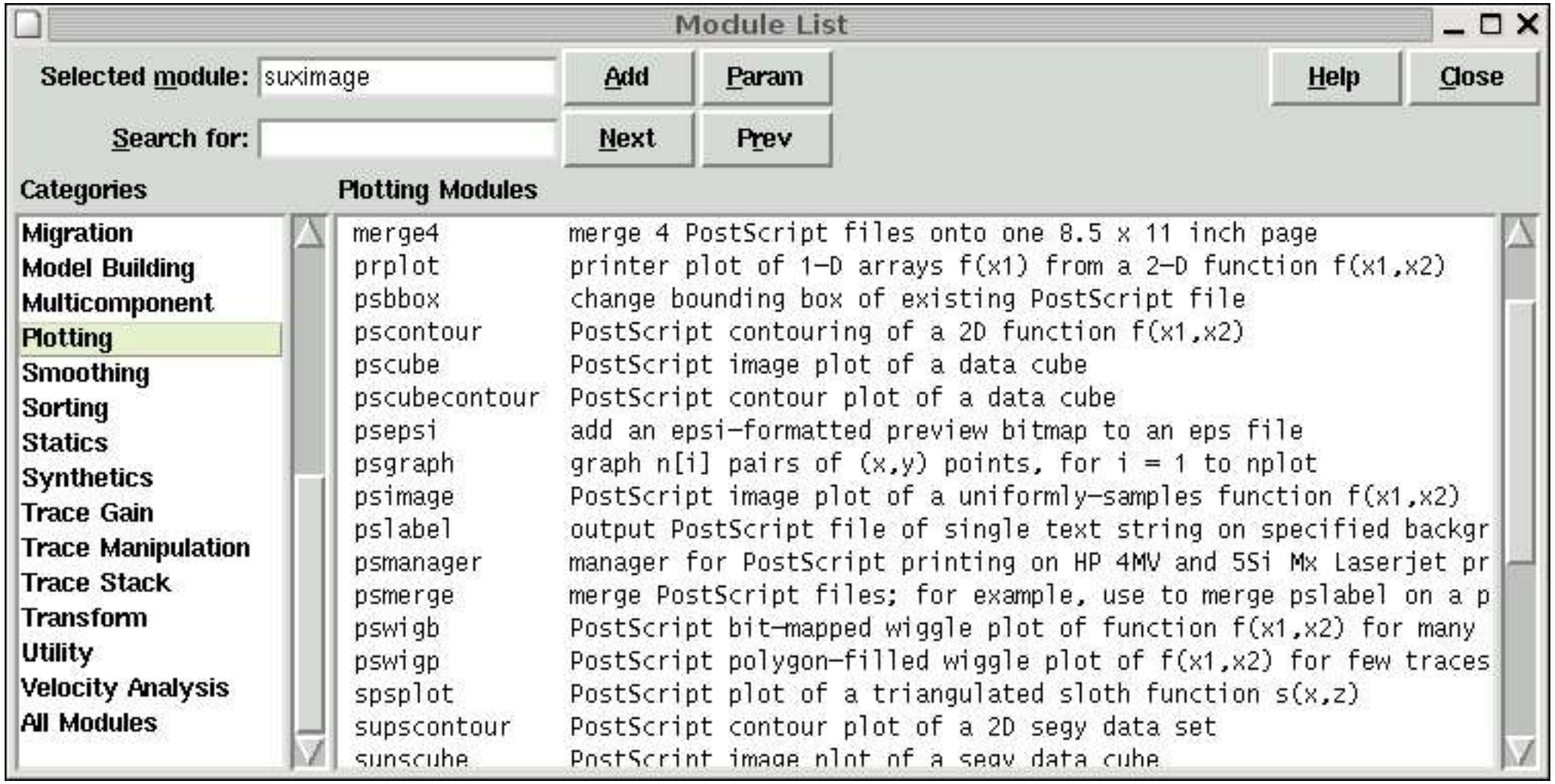
TKSU code:

- Written in Tcl script using Tk graphic elements.
- Even though Tcl has no concept of structures or classes, it was relatively easy to develop in it. The “debugger” is the interpreter itself.
- Application is interpreted, requires Tcl/Tk and TclX be installed.
- License is GPL.

TKSU elements:

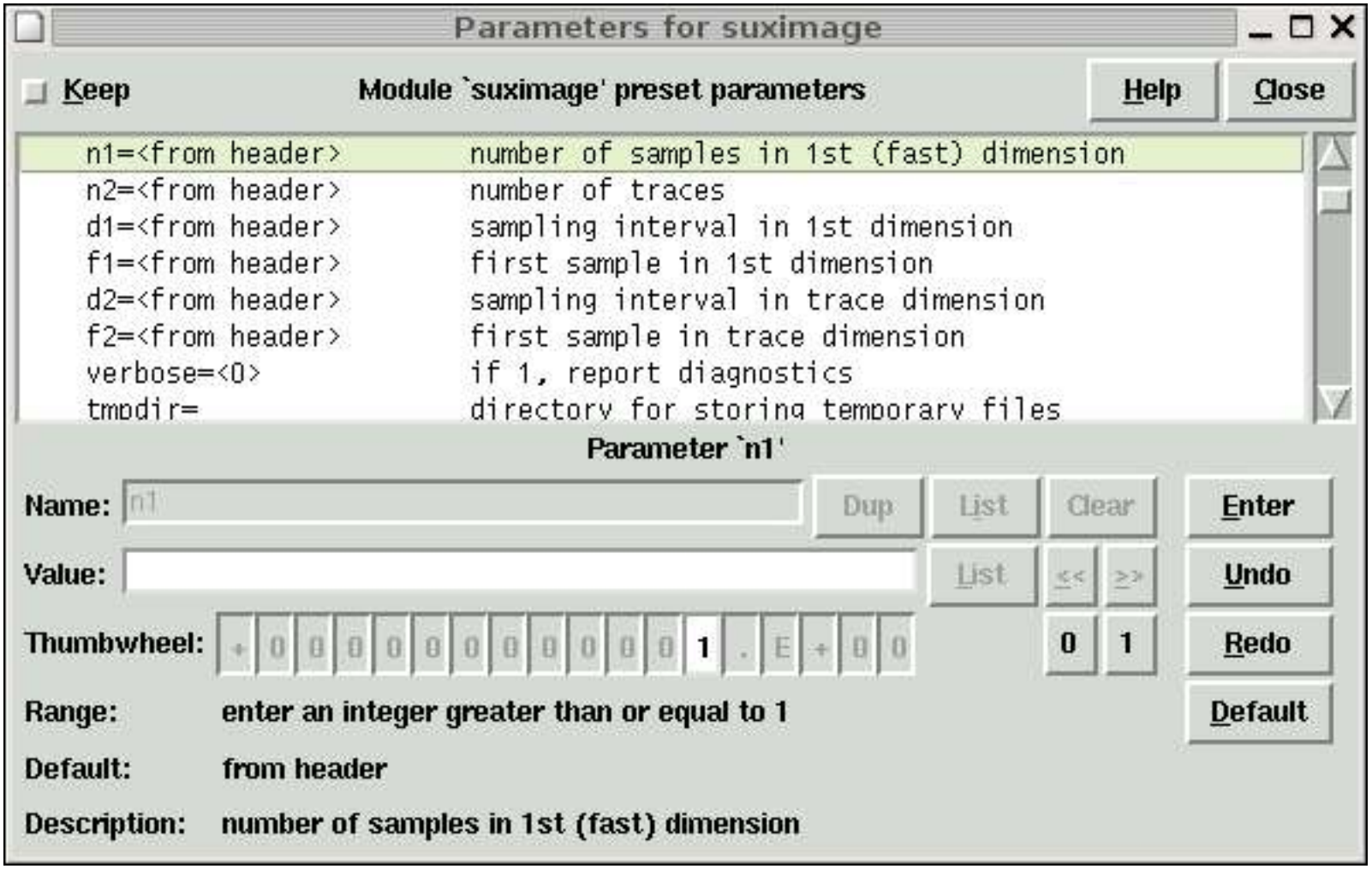
1. Module selection (module window)
2. Parameter management (parameter dialog and list)
3. Flow building (canvas window)
4. Flow testing/running (go command, log window)

Module selection window:



Wish list: search on entire module help database.

Parameter selection window:



Wish list: a more conventional entry dialog.

Two types of command line arguments:

1. Ports: file or pipe input and output, stdin, stdout.
2. Parameters: all other arguments.

- All arguments are in getpar (name=value) form.
- TKSU can map getpar arguments to other forms: positional, Gnu-style (--name value), etc.

Port attributes:

- Name and default value
- Read vs. write vs. bidirectional
- Pipe vs. file
- Mandatory vs. optional argument
- Data type (trace, binary, etc.)
- Special choices: write to /dev/null, read from /dev/zero.

Parameter attributes:

- Name and default value (no default: mandatory)
- Type: integer, float, string, enumeration (list)
- Range of acceptable values
- Whether parameter accepts a comma-delimited list of values
- Short one-line description of the parameter

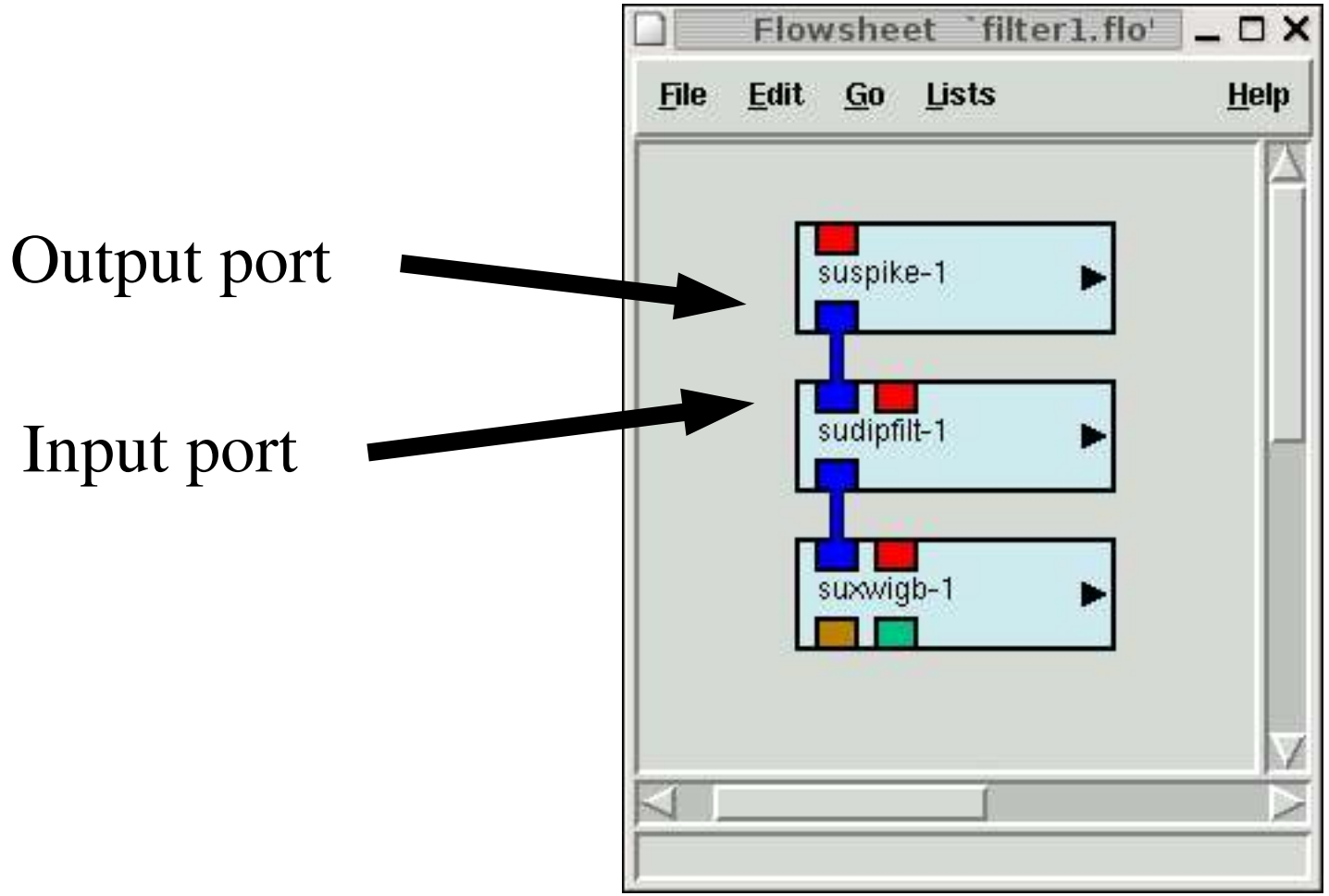
Parameter spec files

- TKSU loads the port and parameter attributes for all SU modules from “spec” files on startup.
- Spec files have a simple ASCII format.
- Where do spec files come from? They were laboriously created by us, often requiring descent into the module code to figure out what a parameter means.
- They SHOULD come directly from the module, in the form of self-documentation.

Wish list for parameter attributes:

- A formal interface to the module should exist that provides specs for all parameters.
- Possibilities:
  - A special module command delivers specs on stdout or stderr.
  - Special structure to the self-document is enforced.
  - Source file runner (least favorite).

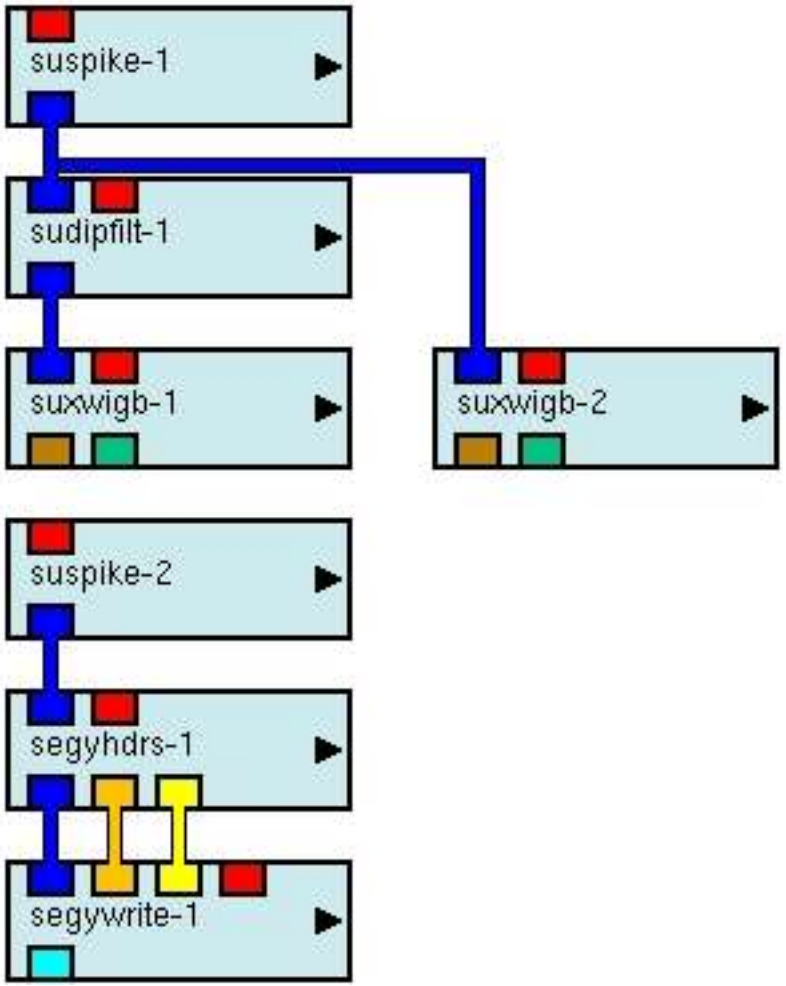
Building flows in the canvas window



Wish list: cut/copy/paste, and ability to import existing shell script into canvas.

Flow examples:

- Sequential modules vs. concurrent modules
- Inserting intermediate display module



TKSU output script:

- POSIX-compliant Bourne shell script
- All links between modules are implemented with named pipes (fifos)
- Script duties:
  - Setup (create fifos)
  - Execute modules in background, directing input & output to fifos
  - Catch signals and exit status of each module
  - Cleanup (remove fifos)
- TKSU canvas state held in the same file

Concurrent vs. sequential flows:

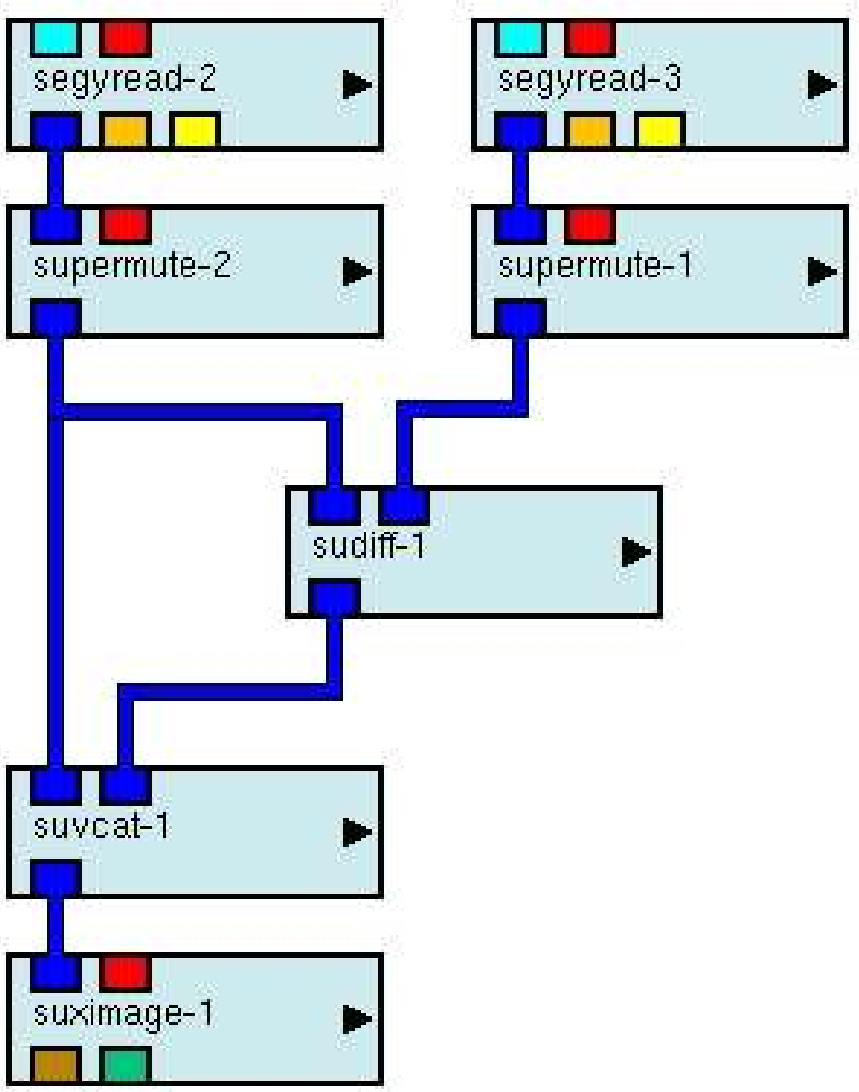
- If modules run sequentially, intermediate output must be stored in temporary files.
- If modules run concurrently with fifo linkages, no cleanup of temporary files is necessary. All fifos are zero-length files in /tmp/tksu.
- Not a big deal for linear flows connected with anonymous pipes (“|”).

Branching flows

- No restriction as to which output port can be connected to which input port (file type mismatch can be overridden).
- Output may be split by making multiple connections to the same output port (“tee”).
- Input may be merged with the “cat” module.
- Feedback is even allowed -- may possibly serve some purpose.

Problem: branching flows may deadlock

- Pipes have low capacity (4 KB).
- Deadlocks occur at the output tee when one leg gets 4 KB ahead of the other leg.
- Solution: module supipe, a pipe with a reservoir, replaces tee.



Advanced parameter management

- Command-line parameters may only be shared via parfiles now. All modules have parfile input ports.
- Wish list: specify which parameters are to be shared.
- Wish list: allow shared parameters to be input from the command line of the flow shell script. A new concept of subflow which can be embedded in other flows.

Interactive processing flows

- Components of an interactive flow:
  - Flow builder (TKSU)
  - Display server
  - Controls for manipulating flow parameters
- Role of the display module is special. It is desirable to have a display server running independent of the flow.
- Changing a parameter triggers execution of the flow. Updated output sent to display.

TKSU download summary

- In the 4 years that it has been available, there have been 3600 downloads of TKSU.
- Majority of downloads to sites worldwide (U.S. figures are uncertain). In order of frequency: Europe, Far East, N America, S America, SE Asia, Australia, S Asia, Middle East, and Africa.
- Roughly 2:1 split between institutional and commercial.
- These may be proxy statistics for SU. After fetching SU, users may tend to migrate to our site to “complete the package”.