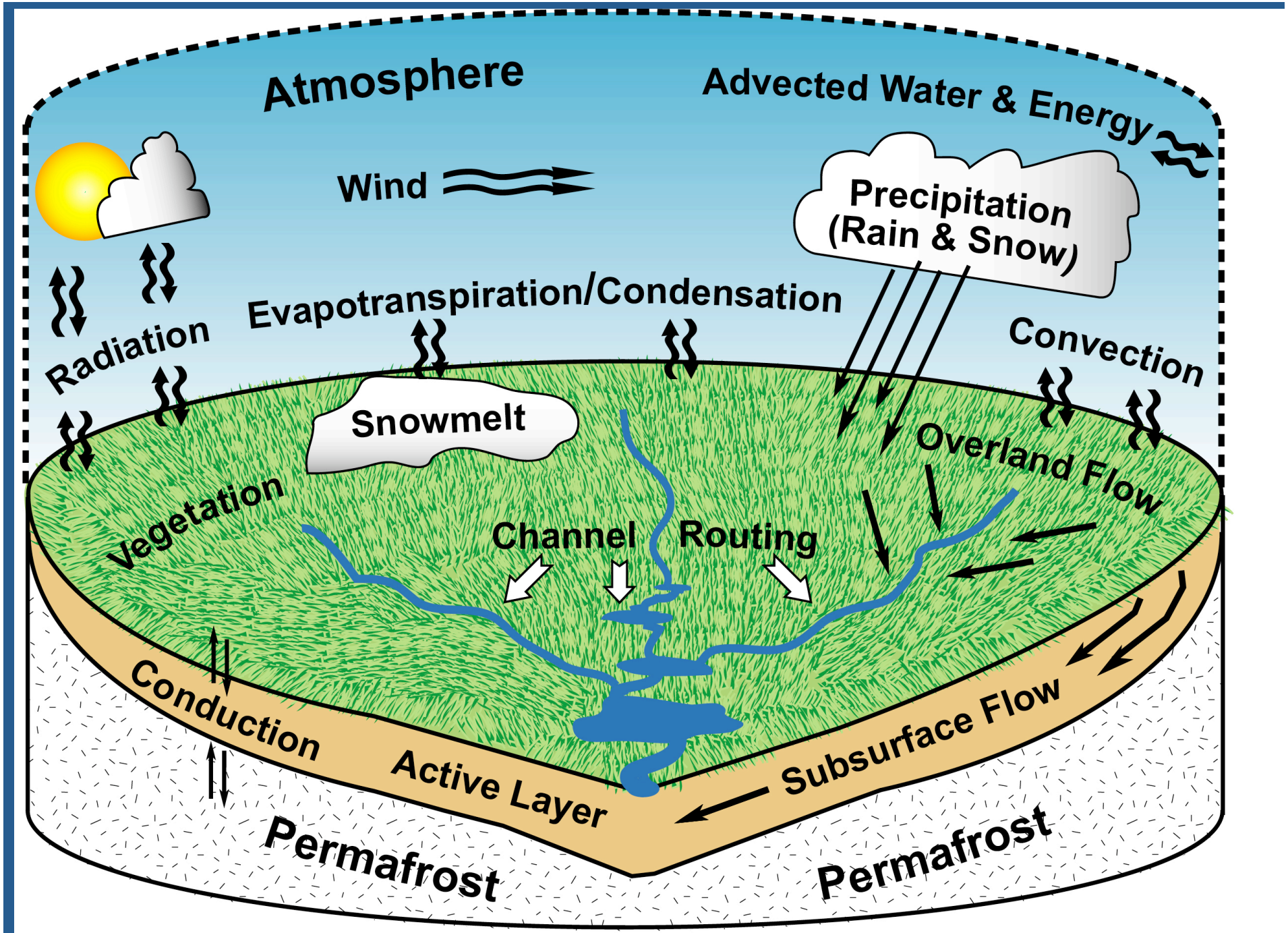
http://csdms.colorado.edu/wiki/Model:TopoFlow
http://csdms.colorado.edu/wiki/Model:gc2d

Coupling the TopoFlow and GC2D Models as CCA Components

Scott D. Peckham
University of Colorado, Boulder

October 5, 2009

Brief Overview of the TopoFlow Hydrologic Model


Atmosphere
Advected Water & Energy
Wind
Precipitation
(Rain & Snow)
Radiation
Evapotranspiration/Condensation
Convection
Snowmelt
Overland Flow
Vegetation
Channel Routing
Conduction
Active Layer
Subsurface Flow
Permafrost
Permafrost

What is TopoFlow 1.5 ?

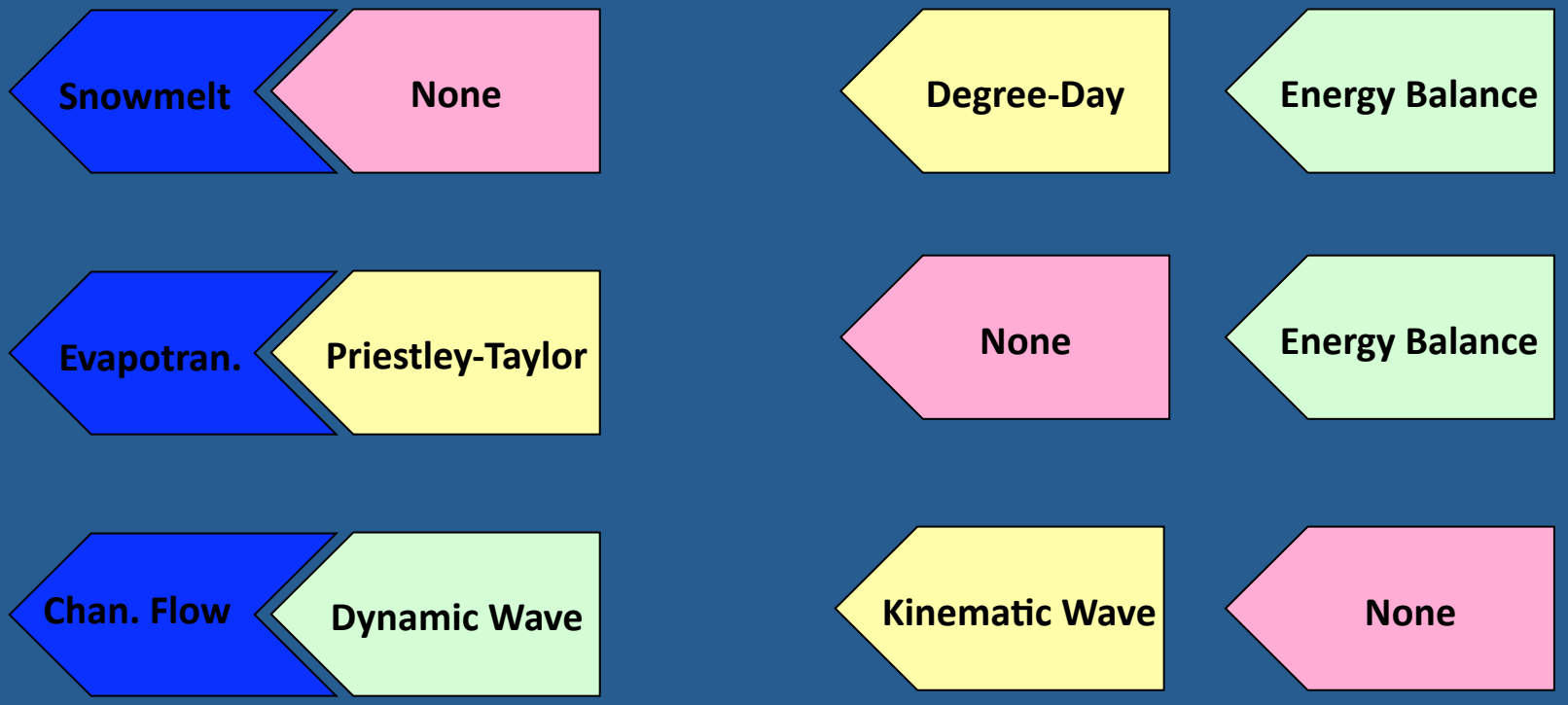
✓ A fully spatial hydrologic model with multiple methods for modeling a variety of physical processes in watersheds, including:

* Precipitation (uniform or varying in space and time
* Snowmelt (simple Degree-Day or full Energy Balance)
* Evaporation (Priestley-Taylor or full Energy Balance)
* Infiltration (Green-Ampt, Smith-Parlange or Richards 1D)
* Channel/Overland Flow (Kinematic, Diffusive or Dynamic Wave)
* Shallow Subsurface Flow (Darcy, multiple uniform layers)
* Flow Diversions (sources, sinks and canals)

✓ 37,434 lines (749 pages) of IDL code. (Interactive Data Language)
✓ Complete, point-and-click GUI with HTML Help System.
✓ Any input variable can be: Scalar, Time Series, Grid or Grid Sequence.
✓ Any computed variable can be saved as Time Series or Grid.
✓ Not object-oriented (but almost)

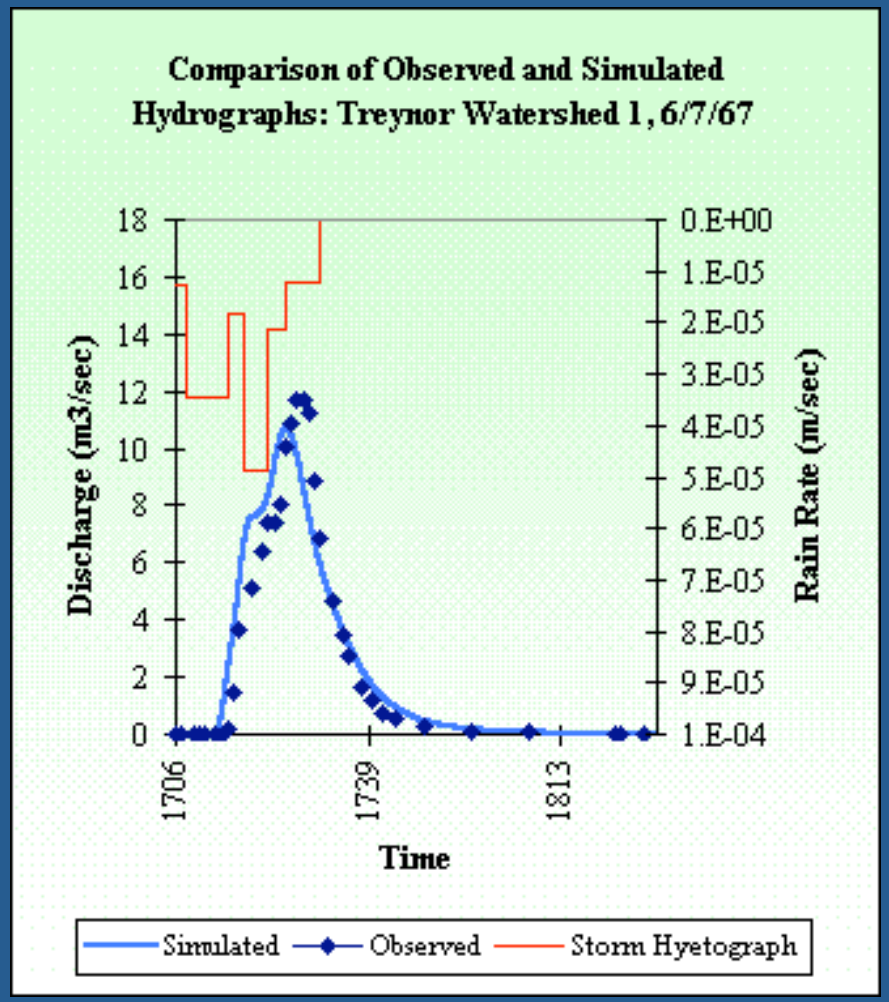
But all components were designed for use within TopoFlow's own internal coupling mechanisms.

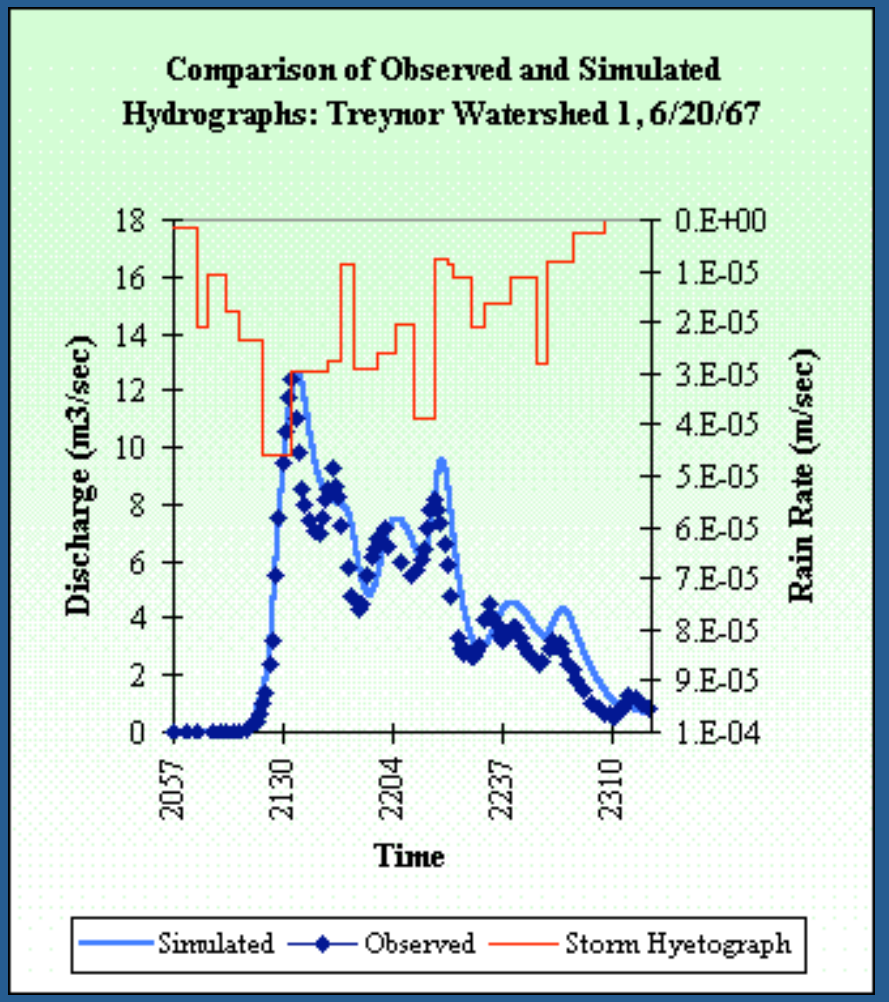
Multiple Methods per Process



Each method has a similar set of dialogs to specify or collect input and output variables. Any process can be turned off.

Treynor Hydrographs vs. Data



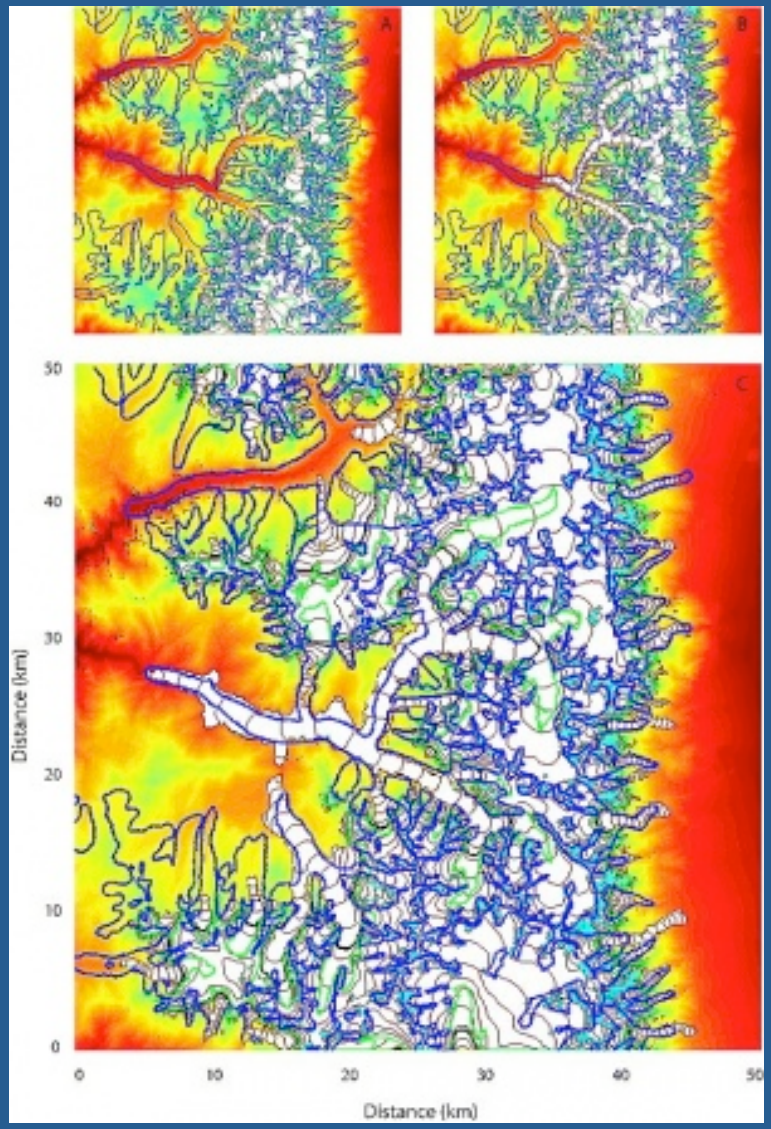


Brief Overview of the GC2D
Valley Glacier & Ice Sheet Model

What is GC2D 1.0 ?

- ✓ A 2D Valley Glacier and Ice Sheet model.
- ✓ Finite-difference, explicit time-stepping
- ✓ Simulates flow of ice via Glen's Law
- ✓ (nonlinear stress-strain) and a rule to
- ✓ compute basal sliding velocity from
- ✓ basal shear stress.
- ✓ Input consists of a DEM and prescribed
- ✓ ELA (as scalar or time series).
- ✓ Precipitation and icemelt processes are
- ✓ not modeled directly – "net mass
- ✓ balance" is prescribed.

- ✓ 1495 lines (30 pages) of MatLab code.
- ✓ Not intended for plug-and-play use – no OpenMI-style interface.
- ✓ All input parameters hard-wired in code (no input or configuration file)
- ✓ Limited ability to save computed variables to output files.



New, Componentized Versions of TopoFlow and GC2D

Components that implement an OpenMI-style interface (IRF, getters, setters, etc.) and are linkable within a CCA framework such as Ccaffeine.

What is TopoFlow 3.0 ?

✓ Now consists of 17 separate components. Each component has:

(1) Ability to be used as a model (driver) or as a component.
(2) An OpenMI-style interface (incl. IRF, getter, setters, etc.)
(3) A wrapper to make it a CCA component (CCA "impl" file)
(3) Its own, separate input file (*.cfg)
(4) A GUI dialog to change its parameters, with HTML help.
(5) Its own output options.

✓ Written in Python & uses Numerical Python. (33,058 lines)
Code was converted from IDL to Python using I2PY 2.0.
✓ Completely object-oriented.
✓ All components inherit from a "CSDMS component" base class.
✓ Computed variables can be saved as before, now as BOV or netCDF.
✓ Precipitation component merged into Meteorology component.

But, pre-processing tools (8500 lines) not yet converted from IDL and GUI is limited as compared to the original.

What is GC2D 2.0 ?

This new version can be used as either a component that provides meltwater runoff to a spatial hydrologic model (TF) or as a stand-alone Model/Driver. As a Driver, it can now optionally be driven by TopoFlow's physically-based process components. That is, the Meteorology/Precip and Snow components could be used to provide snowfall and icemelt rates directly to GC2D.

- ✓ Written in Python & uses Numerical Python. (1966 lines)
- ✓ Wrapped in "ice_base" class to provide OpenMI-style interface and additional capabilities (880 additional lines, Python)
- ✓ Wrapped again (IceGC2D_Impl.py) to be a CCA component.
- ✓ Reads all input parameters from a "configuration file" (*.cfg).
- ✓ Computed variables can now be saved as BOV or netCDF.
- ✓ Can now output a grid of "meltrates" for use by other models.
- ✓ Will soon have a "ParameterPort" dialog for user input.
- ✓ Will eventually have an HTML help page.

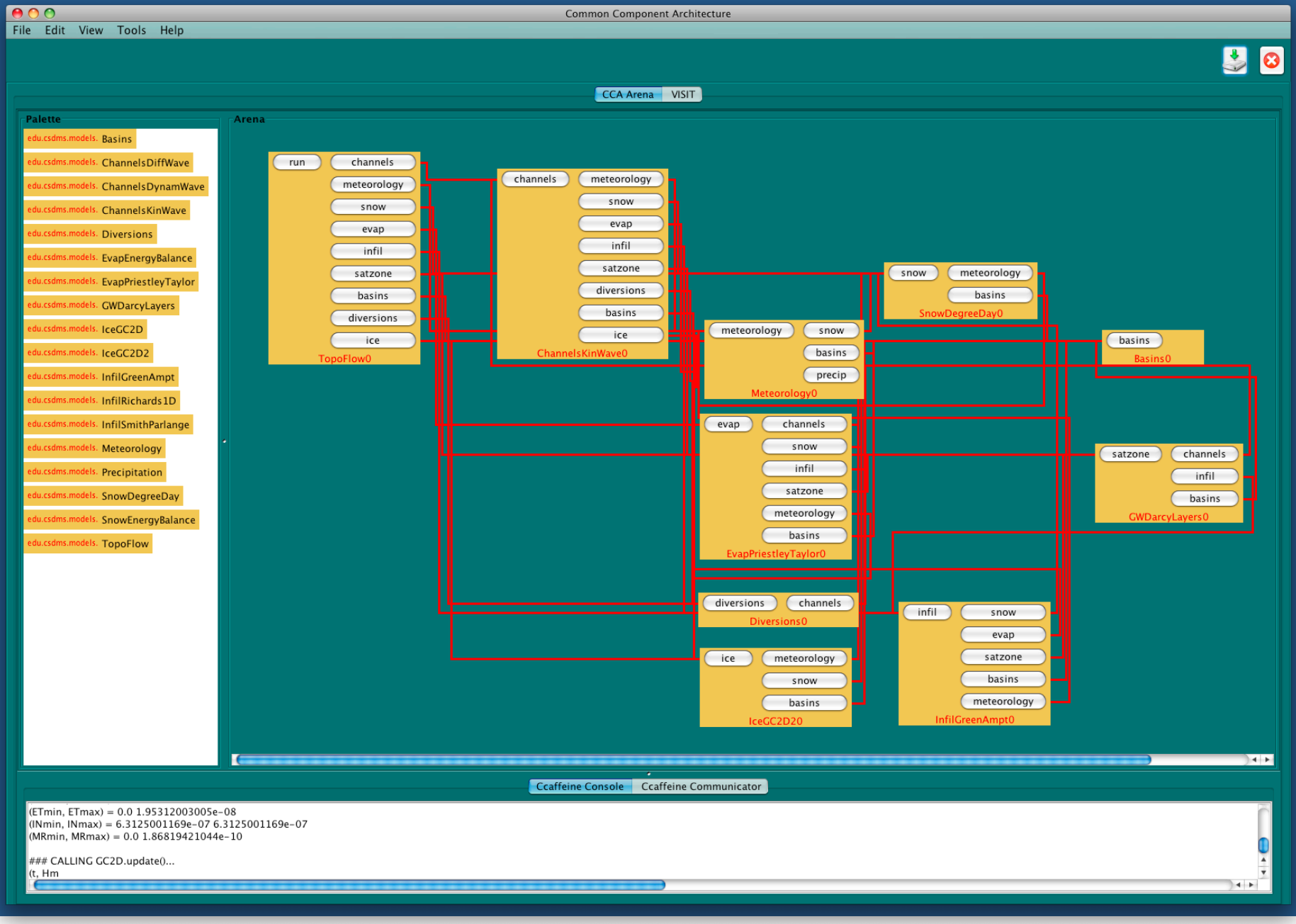
Added Value for TopoFlow

Can now be used as a Component or stand-alone Model/Driver.
Can be run by any CSDMS member, remotely, on beach.
Can use VisIt, integrated into CCAFFE-GUI, to visualize output.
Can easily be linked to components written in other languages.
Can now utilize meltrate output from an ice model.

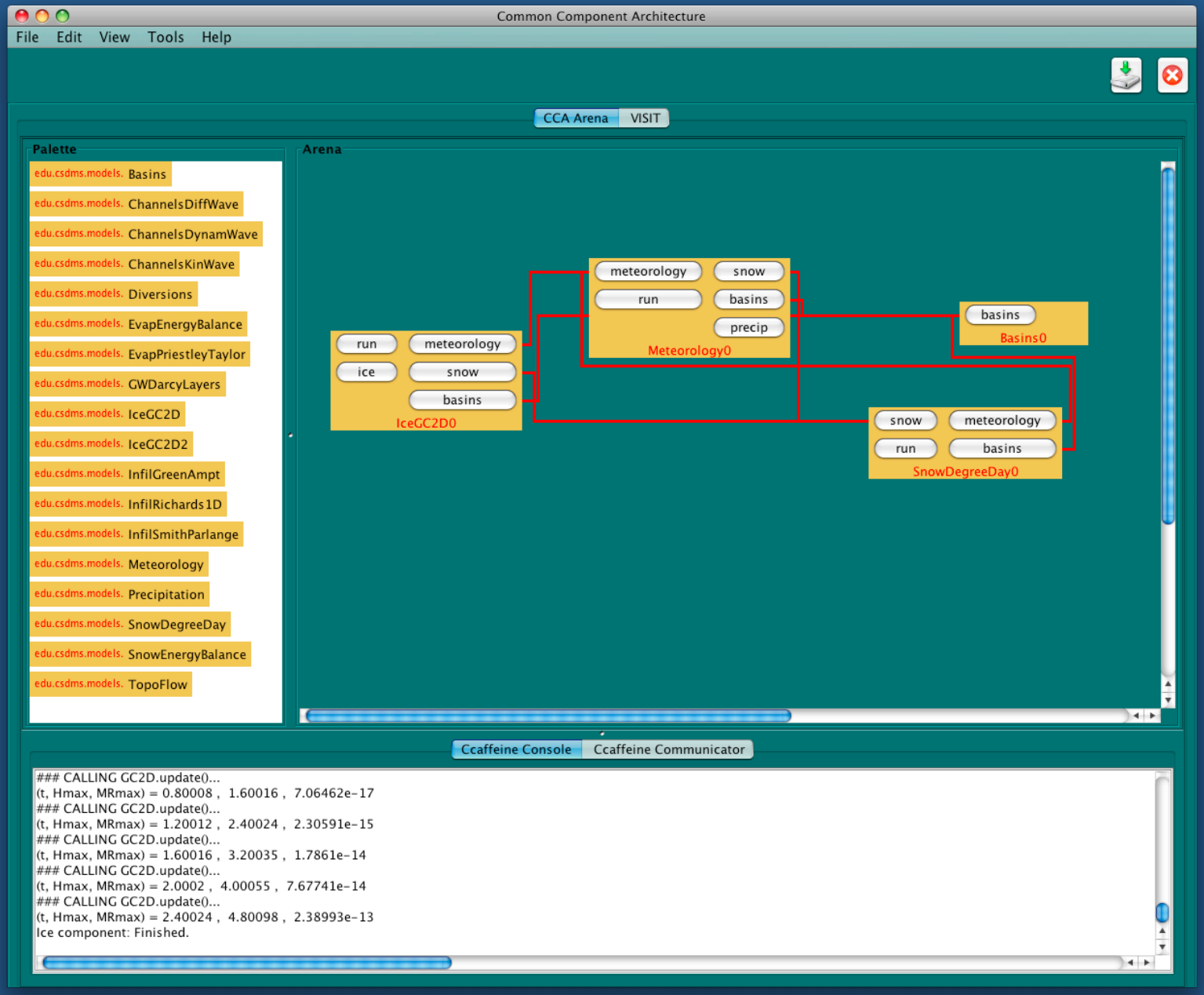
Added Value for GC2D

Can now be used as a Component or stand-alone Model/Driver.
As a Driver, can be driven by Meteorology & Snow components.
GUI dialog and input files for changing model parameters.
Can be run by any CSDMS member, remotely, on beach.
Can use VisIt, integrated into CCAFFE-GUI, to visualize output.
Improved output options (Time Series and/or Grid Sequence).
Can easily be linked to components written in other languages.

CCAFE-GUI Wiring Diagram for TopoFlow using GC2D as Component



CCAFFE-GUI Wiring Diagram for GC2D using TopoFlow Components



Conclusions

Now, any CSDMS member can download and install the CSDMS version of the Ccaffeine GUI as a small, easy-to-install Java app.

Using this GUI, they can choose components from available palettes to create their own, customized applications, and then run them on our supercomputer, “beach”. We have linked our GUI with VisIt to provide run-time visualization.

“Process components” seem to be the most natural level of granularity for model componentization. “Processes” (such as infiltration) represent the “scale” at which modelers are most likely to want to replace one approach with another. For example, modelers very often want to compare different approaches and algorithms with respect to speed, accuracy, scalability or realism.

Complete models represent the next higher level of granularity and provide only limited opportunities for linking and re-use.

More Conclusions

Converting models from one language to another is a complex task that should be avoided whenever possible. Conversion tools usually cannot fully automate the conversion process.

Our community has a large number of models written in Very High-Level Languages (VHLLs) like MatLab and IDL.

MatLab and IDL are proprietary languages and are not supported by Babel. Python, however, is supported by Babel, and with the Numerical Python package (and other packages) provides a very powerful, open-source alternative with similar performance.

For MatLab, IDL and Python, there are commercial products that allow code to take advantage of multiple processors. These products require virtually no changes to the original code, as long as it was written properly so as to replace all “loops over space” with array operators.