

*Modeling Frameworks, Workflows
and Community Modeling:
Where are we now and
where do we go from here?*

*Scott D. Peckham
Chief Software Architect for CSDMS
University of Colorado, Boulder
April 23, 2013*



csdms.colorado.edu



What are the Key Problems ?

Heterogeneity: Lots of resources (data, models, web services, etc.), but all different. How do you accommodate differences between resources? (names, data types, units, grids, timesteps, formats...)

Discovery: How do you find a resource of interest? (e.g. ontologies)

Linking/Connecting: Models have input and output variables, data has only “output”. How do you connect the users to the providers? How much data is involved? What is the *data transfer rate*?

Semantic issues: Perhaps the “*last frontier*” of interoperability. We need to boldly go there and develop *domain-independent* solutions.

Performance, speed & efficiency: The 80/20 Problem: 80% grunt work, 20% doing science. Automation and seamless integration can turn this around. Need to design around data transfer rates.

Some Key Concepts and Terms

Model: *State variables* for some system of interest are discretized in space (on a *computational grid*) and new values are computed from previous values by *marching forward in time* according to a set of rules (e.g. laws of physics).

Model Component: A model that has been specially prepared for plug-and-play reusability. (i.e. wrapped with a *standard interface* & compiled for *language interoperability*).

Interface: A standardized *set of functions*, which a caller uses to interact with a resource such as a model, database or service. *Heterogeneous resources* wrapped with a *standard interface* then operate in a “familiar” way.

Modeling Framework: A software container in which pre-compiled model components are instantiated, configured and *dynamically linked* to rapidly *create a customized model*. (With a “birds-eye view” of all components.)

Workflow: Software that allows a *chain of steps* to be saved, modified and re-executed. Each step uses an application to create an *intermediate product* that is passed along to the next step, *often as a file*. (e.g. Kepler, Pegasus)

Taming Heterogeneity with Interfaces

Before:

Each resource is unique.
Own ways of doing things.
Respond differently.
Can become unstable.
Difficult to control.



After:

Uniform outward appearance.
Respond to same commands.
Interchangeable units.
Have a chain of command.
Work as a team.

Local and Global Standards

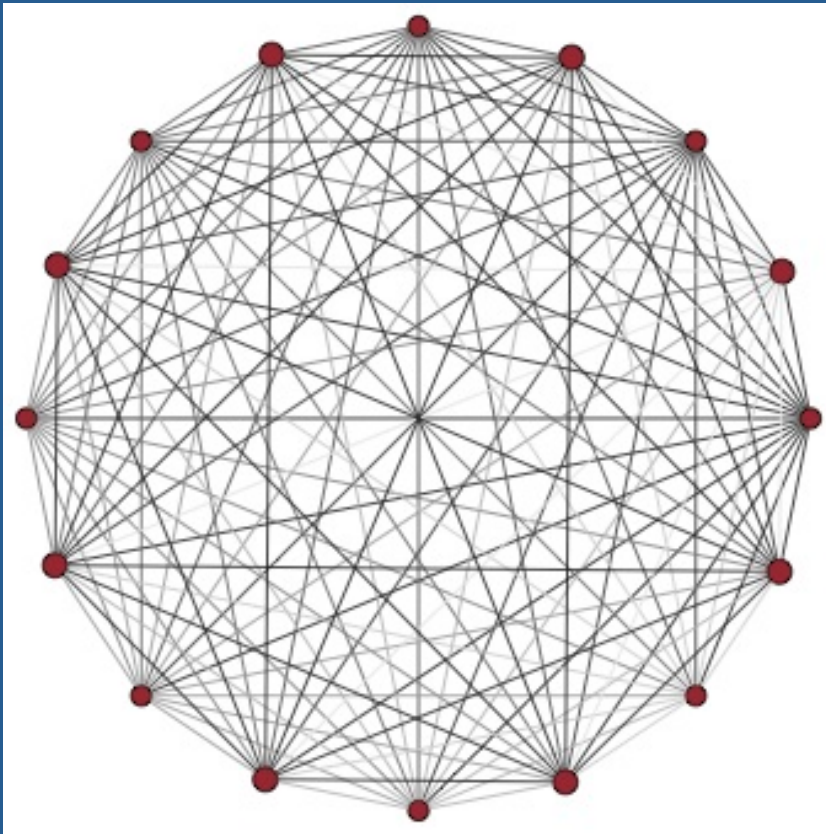


Building
bridges
between
Silos:

Semantics,
Formats,
Etc.

Different domains (e.g. disciplines) are like silos. They should be free to use their own internal or “**local standards**”, but need to agree on some common “**global standards**” when they want to start sharing resources.

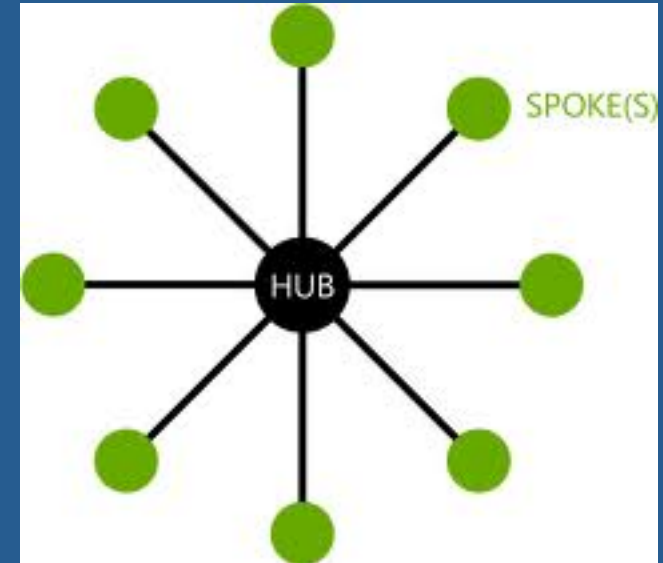
Standards Reduce Complexity



If we **reconcile differences** between the resources in a pairwise manner, the amount of work, etc. grows fast:

$$\text{Cost}(N) = N(N-1) / 2 \sim N^2.$$

VS.



Introduce a new, generic or **standard** representation (the “hub”), then map resources to and from it. The amount of work, maintenance, etc. drops to:

$$\text{Cost}(N) = N.$$

Typical Data Transfer Rates

Reading data from RAM: (DDR3 SDRAM, sequential) (Writing is about 40 times slower)	4000 MB/sec - 17,000 MB/sec
Reading data from Solid State Drive (SSD): (Write speeds not much slower.)	200 - 600 MB/sec
Reading data from a Hard Drive (HDD): (7200 rpm, seek + read): (SATA = 3 Gbps max= 375 MB/sec, not realized)	50 MB/sec
Transfer data over a <i>wired</i> network: (1 Gbps + protocol overhead)	100 MB/sec
Transfer data over a <i>wireless</i> network:	5 to 30 MB/sec

These largely determine the appropriate solution or tool for a given problem.

Modeling Frameworks:

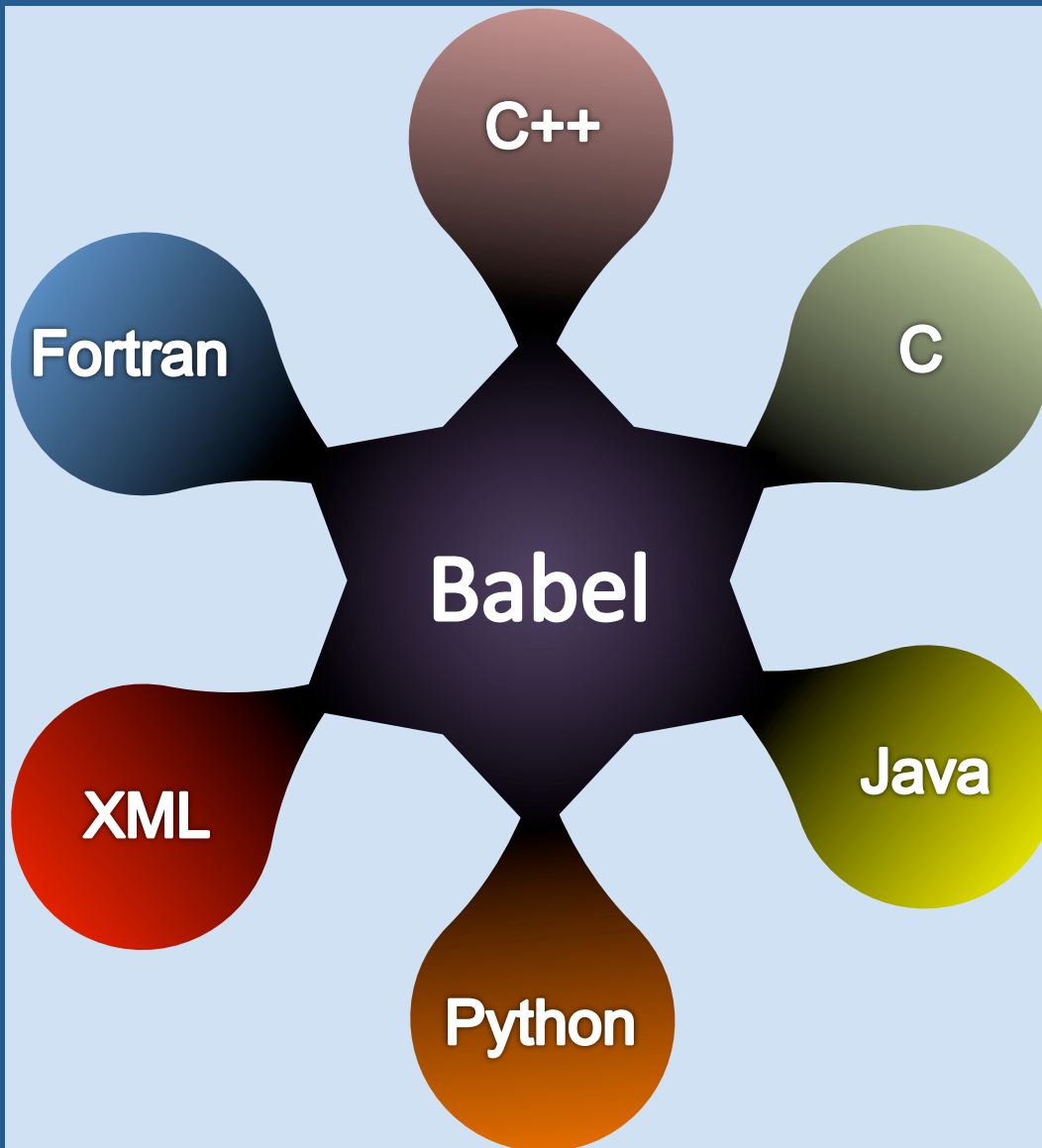
Build a new model from a collection of reusable, pre-compiled, plug-and-play components.

This new model could then become a node in a workflow.

Linking Component-based Models: How Can Two Models Differ?

- Programming language
(C, C++, Fortran, Java, Python, etc.)
Solution: Babel and Bocca (CCA toolchain)
- Computational grid
(triangles, rectangles, Voronoi, etc.)
Solution: ESMF regridder (parallel, spatial interpol.)
- Timestepping scheme
(fixed, adaptive, local)
Solution: Temporal interpolation tool
- Variable names
Need some means of “semantic mediation”
Solution: CSDMS Standard Names
- Variable units
Solution: UDUNITS (Unidata)

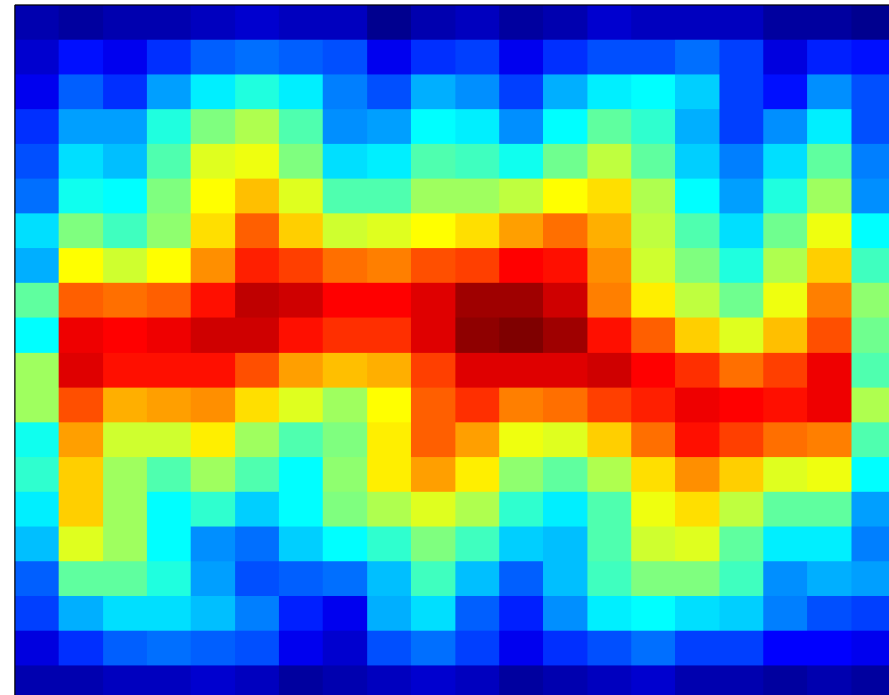
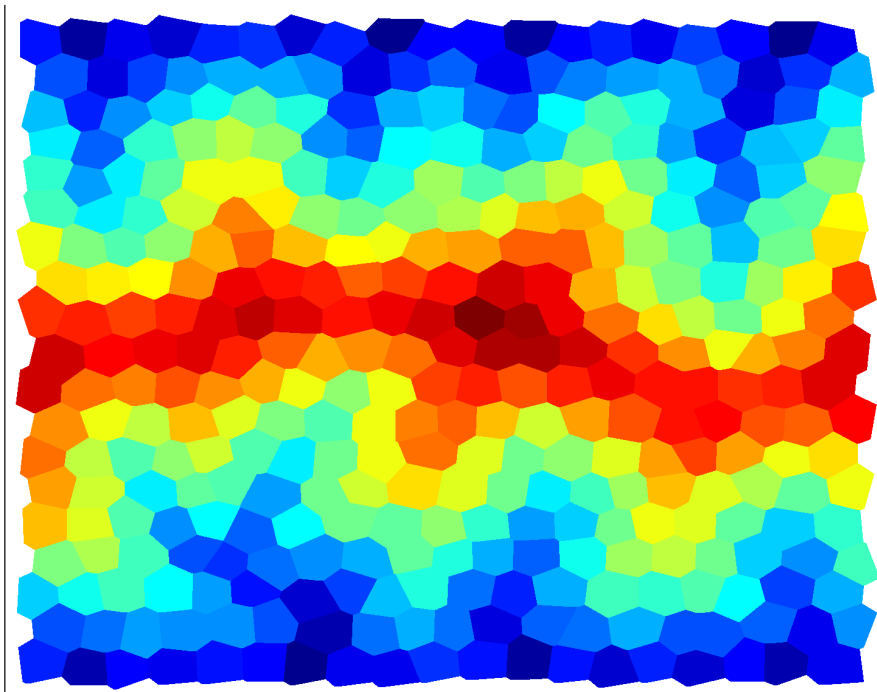
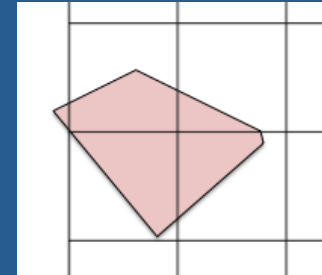
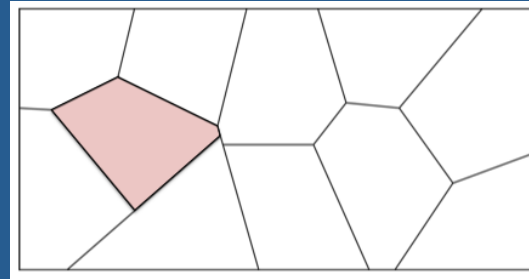
Language Interoperability with Babel



Language interoperability is a powerful feature of the CCA framework. Components written in different languages can be rapidly linked in HPC applications with hardly any performance cost. This allows us to “shop” for open-source solutions (e.g. libraries), gives us access to both procedural and object-oriented strategies (legacy and modern code), and allows us to add graphics & GUIs at will.

CSDMS Regridding Tools

- ESMF Regrid (multi-proc., Fortran)
- OpenMI Regrid (single-proc., Java)



The Basic Model Interface (BMI) Standard

BMI requires a developer to make some relatively simple, *noninvasive*, and *framework-independent* changes to his/her model source code, mostly by adding some new functions. Functions can be grouped into:

Model Control Functions (initialize, update, finalize)

Model Information Functions (e.g. time-stepping scheme)

Variable Information Functions (e.g. dimensions, data type, units)

Variable Getters and Setters

Grid Information Functions

CSDMS automatically wraps BMI-enabled models to provide them with a CMI interface. CMI makes function calls to the (1) *framework*, (2) *service components* (the “reconcilers”), (3) *the BMI of the wrapped model* and (4) *the CMI of other plug-and-play components*. CMI gives a model many new capabilities, including NetCDF output.

CSDMS Talking to a Model

CSDMS: So, tell me about yourself.

Model:

I need input variables A, B and C.

I provide output variables X, Y and Z.

I use adaptive time-stepping.

I use a computational grid of Voronoi polygons.

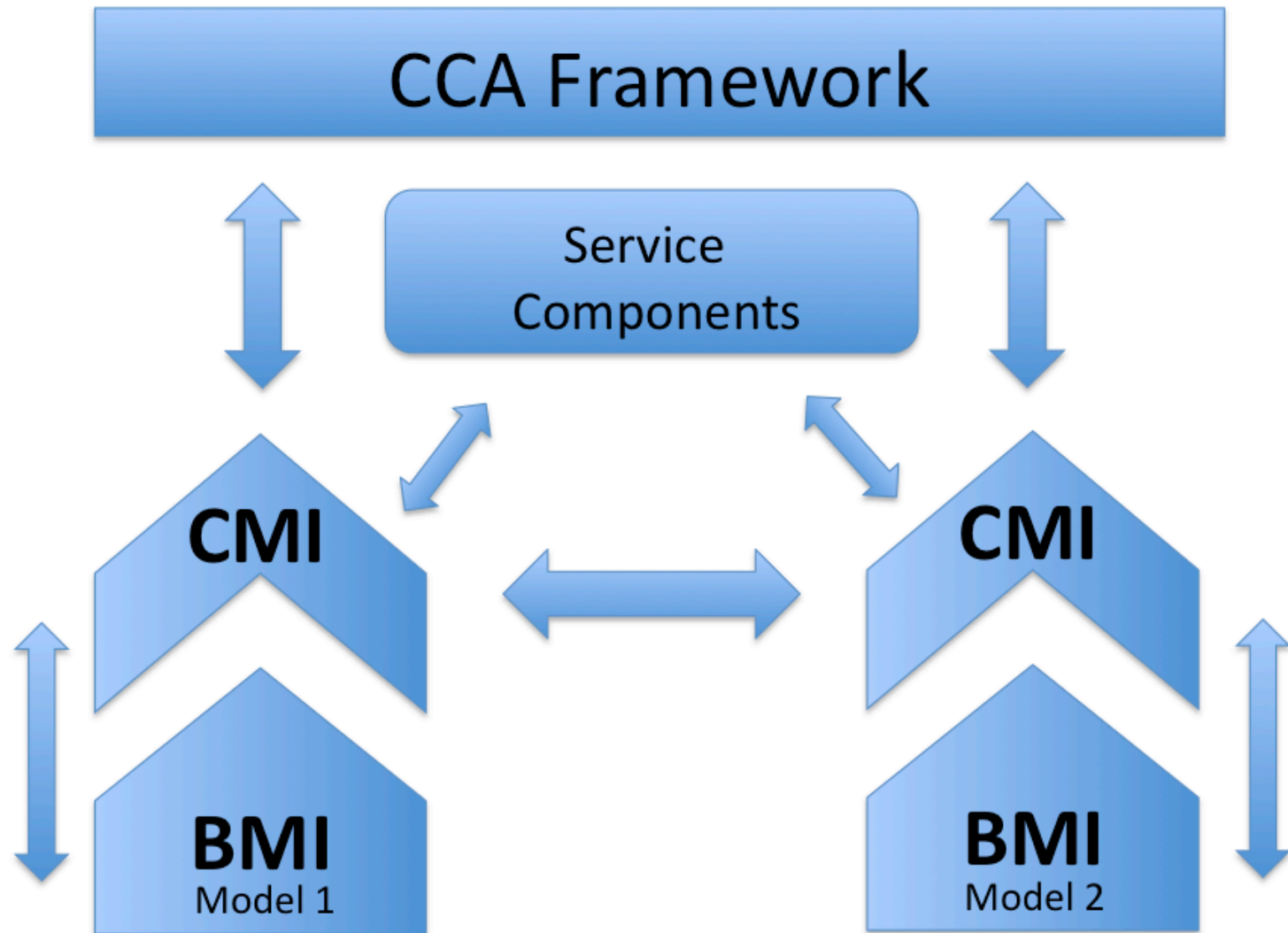
I like pina coladas and getting caught in the rain.

CSDMS: Those are great attributes. I think we can hook you up.

Model: Am I just an *object* to you ??

CSDMS: Yes. Your state variables are your *data members*. Your *member functions* consist of the BMI interface functions.

How BMI and CMI Work



The CSDMS Standard Names

Data Models like **RDF** and **EAV** use *triples* like:

Subject + Predicate + Object, and

Entity/Object + Attribute + Value (object-oriented)

CSDMS Standard Names use a similar pattern for creating **unambiguous** and **easily understood** standard *variable names* or “*preferred labels*” according to a **set of rules**. These are then used to retrieve/match values (and metadata). The pattern is:

Object name + [**Operation name**] + **Quantity name**

Examples:

atmosphere_carbon_dioxide__partial_pressure

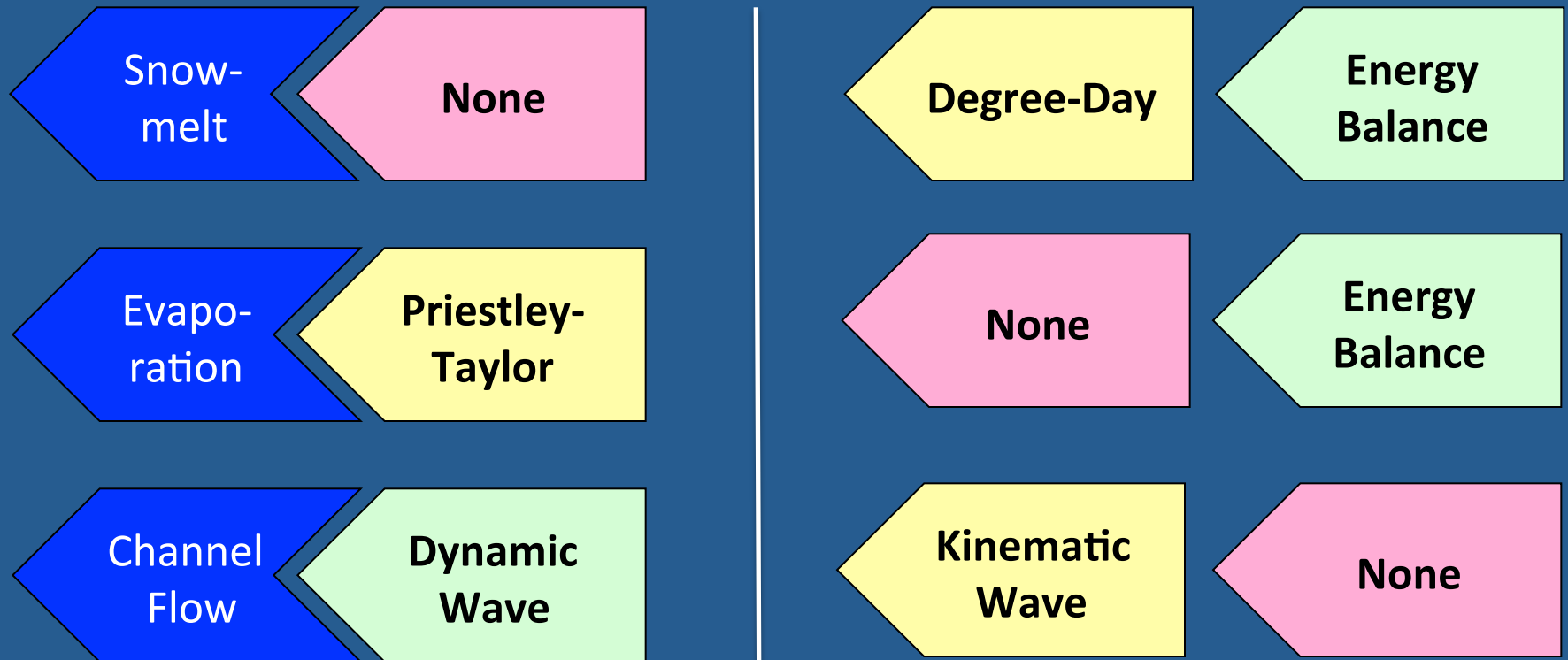
atmosphere_water__liquid_equivalent_precipitation_rate

earth_ellipsoid__equatorial_radius

soil__saturated_hydraulic_conductivity

We have also started building a set of standard Attribute and Process Names.

Multiple Methods per Process



For each physical process (blue), there are typically many different ways to parameterize it, ranging from **not at all** to **simple** (with few inputs) to **advanced** (with many inputs).

The CSDMS Modeling Tool (CMT)

The screenshot displays the CSDMS Modeling Tool (CMT) interface. The window title is "CSDMS Modeling Tool". The menu bar includes "File", "Edit", "View", "Tools", and "Help". The "Remote Working Directory" is set to "~/CMT_Output/". The "Working Project" is "TopoFlow + GC2D".

The interface is divided into several sections:

- Driver:** A dropdown menu showing "TopoFlow".
- Palette:** A list of components including ChannelsDiffWave, ChannelsDynamWave, ChannelsKinWave, DataHIS, Diversions, EvapEnergyBalance, EvapPriestleyTaylor, EvapReadFile, IceGC2D, InfilGreenAmpt, InfilRichards1D, InfilSmithParlange, Meteorology, SatZoneDarcyLayers, SnowDegreeDay, SnowEnergyBalance, TestService, and TopoFlow.
- Arena:** The main workspace showing a simulation setup. It includes a "Driver: TopoFlow" panel with buttons for "Run", "Configure", and "Hydro_model", and sub-components for "Meteorology", "Channels", "Snow", "Evap", "Infil", "Satzone", "Ice", and "Diversions". Several other component panels are visible, each with a "Configure" button and sub-components: "Component: ChannelsKinWave" (Channels, Snow, Evap, Infil, Satzone, Ice, Diversions), "Component: Meteorology" (Snow, Meteorology), "Component: Diversions" (Channels, Diversions), "Component: EvapPriestleyTaylor" (Evap, Channels, Snow, Infil, Satzone), "Component: IceGC2D" (Ice, Snow, Meteorology), "Component: InfilGreenAmpt" (Infil, Channels, Snow, Evap, Satzone), and "Component: SatZoneDarcyLayers" (Satzone, Infil, Channels).
- CMT Console:** A text area at the bottom showing the simulation output.

The CMT Console output is as follows:

```
Max(precip rate): 165.947006226 [mm/hr]

Channels component: Finished.
Snow component: Finished.
Meteorology component: Finished.
Evaporation component: Finished.
Infiltration component: Finished.
Groundwater component: Finished.
Ice component: Finished.
Diversions component: Finished.

=====
Simulation complete.
=====

Finished. (June_20_67)
```

For More Information

Peckham, S.D., E.W.H. Hutton and B. Norris (2012)
*A component-based approach to integrated modeling in the
geosciences: The design of CSDMS,*
Computers & Geosciences

Available online at:

<http://www.sciencedirect.com/science/article/pii/S0098300412001252>

Basic Model Interface (BMI):

http://csdms.colorado.edu/wiki/BMI_Description

CSDMS Standard Names:

http://csdms.colorado.edu/wiki/CSDMS_Standard_Names

Transit of Venus at Sunset: June 5, 2012



Thank you and happy modeling!

Building a Modeling Framework

CSDMS has integrated a variety of powerful, open-source tools to build its modeling framework, such as:

Babel – Language interoperability (C,C++,Java,Python,Fortran)

Bocca – Component preparation and project management

Ccaffeine – Low-level model coupling (parallel environ.)

ESMF Regrid – Multi-processor spatial regridding

OpenMI Regrid – Single-processor spatial regridding

OpenMI – Component interface standard (1.4 and 2.0)

NetCDF – Scientific data format (self-describing, etc.)

VisIt – Visualization of large data sets (multi-proc.)

We greatly extended the original *Ccaffeine GUI* to create our ***CSDMS Modeling Tool*** for interactive model coupling.

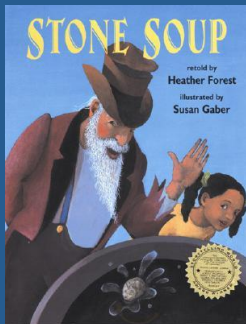
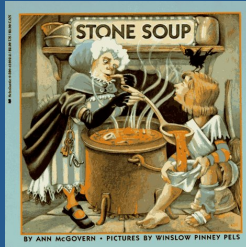
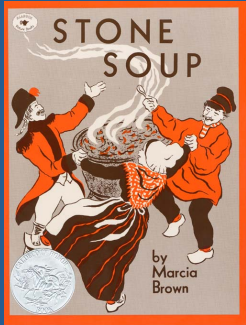
How Can a Model be Converted to a Reusable, Plug-and-Play Component?

CSDMS has developed two model *interfaces*, one called *Basic Model Interface (BMI)* and one called *Component Model Interface (CMI)* for this purpose.

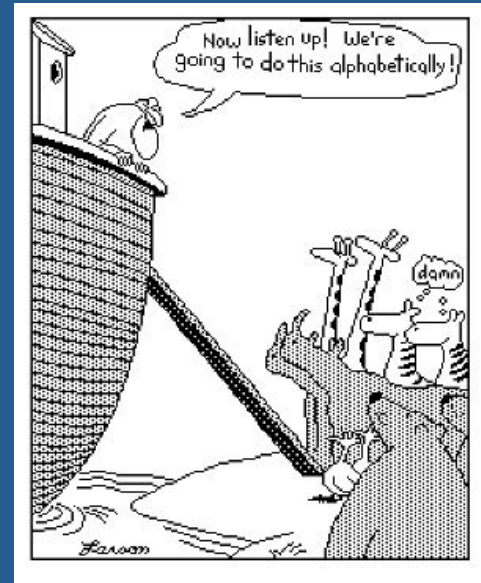
BMI requires a developer to make some relatively simple, *noninvasive*, and *framework-independent* changes to his/her model source code, mostly by adding some new functions. These provide a caller with 3 key things: (1) *fine-grained control* (i.e. *IUF*), (2) *variable getters and setters* and (3) *model metadata*. The model can still be used in “stand-alone mode,” just as before.

CSDMS automatically wraps BMI-enabled models to provide them with a CMI interface. CMI makes function calls to the (1) *framework*, (2) *service components* (the “reconcilers”), (3) *the BMI of the wrapped model* and (4) *the CMI of other plug-and-play components*. CMI gives a model many new capabilities, including NetCDF output.

One “Super Model” or Stone Soup?

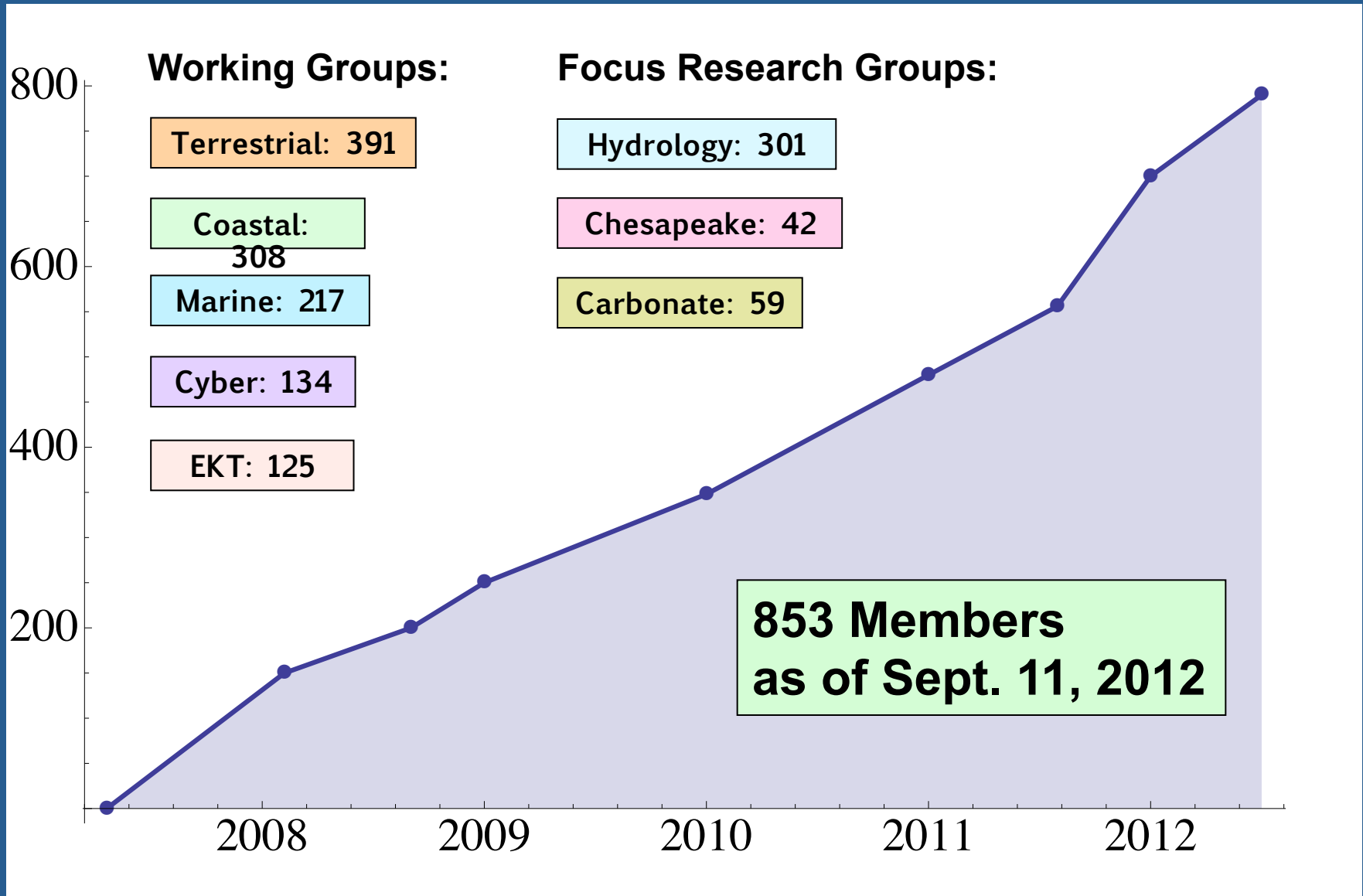


Collecting Earth Surface Process Models

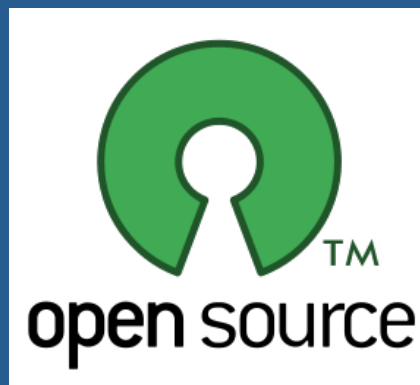
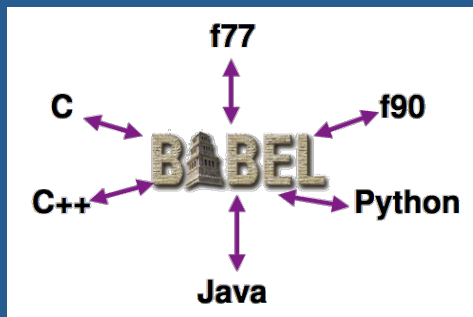


Open source:
164 Models
51 Tools
55 Components

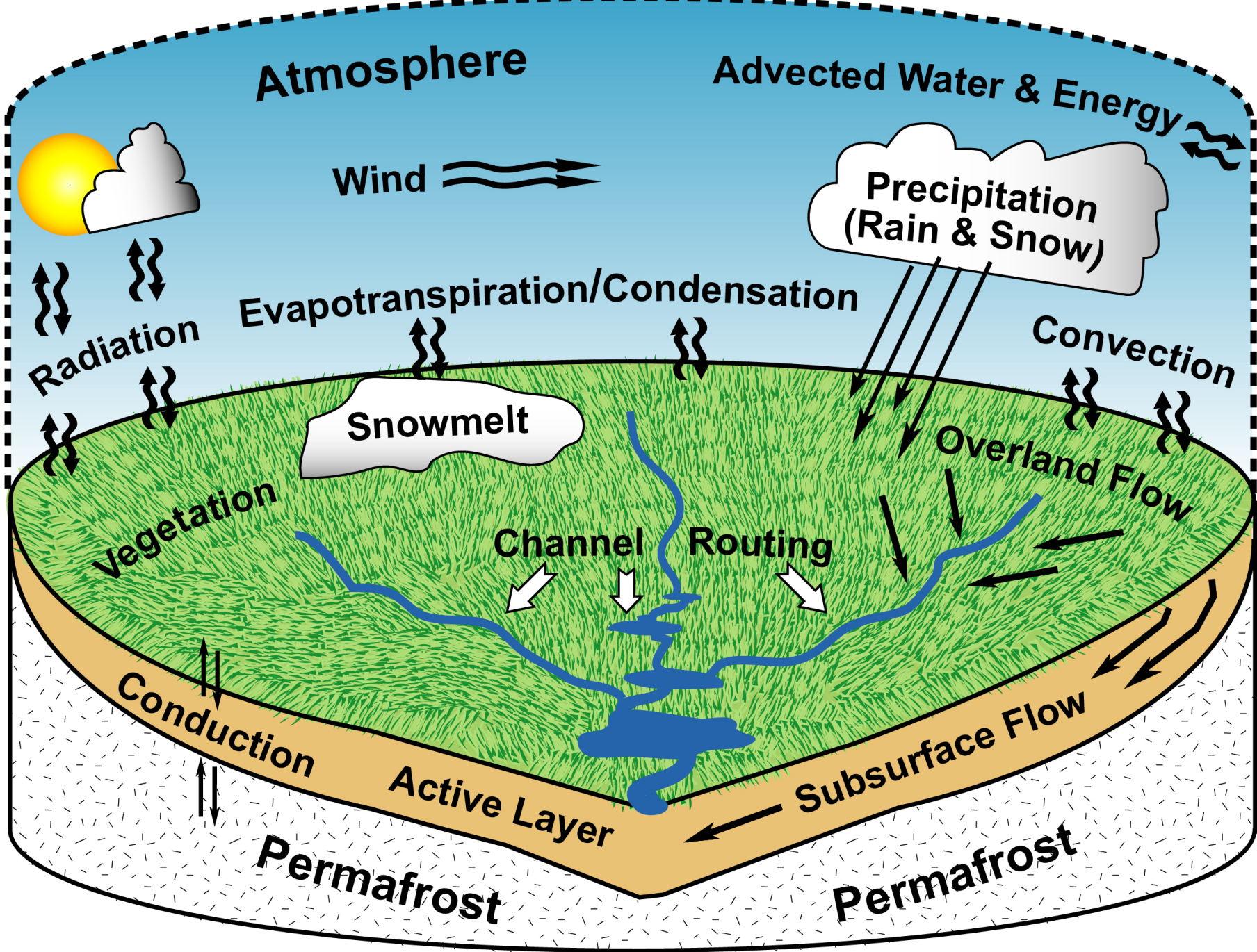
Number of CSDMS Members vs. Time



Powerful Tools used by CSDMS




*Building and Running a Spatial
Hydrologic Model with CMT*



Live Demo of CMT 1.6

CSDMS Modelling Tool (BETA) Version 1.6.1



Remote Host*:

Username*:

New to CSDMS HPCC? [Click to Register!](#)



Tabbed Dialog for Meteorology

Meteorology: Method 1 Parameters: Meteorology

Input 1 | Input 2 | Input 3 | Output 1 | Output 2 | About

Component status:	-	Enabled	?
Input directory:	-	/home/csdms/models/topoflow/3.1/share/data/treyno	?
Output directory:	-	~/CMT_Output/	?
Site prefix:	-	Treynor	?
Case prefix:	-	Case5	?
Number of steps:	{1, 10000000}	10	?
Time step:	{0.0, 1000000.0}	60	?
rho_H2O type:	-	Scalar	?
rho_H2O:	{0.0, 2000.0}	1000	?
Cp_air type:	-	Scalar	?
Cp_air:	{0.0, 2000.0}	1005.7	?
rho_air type:	-	Scalar	?
rho_air:	{0.0, 2.0}	1.2614	?
precip. rates type:	-	Time_Series	?
precip. rates:	-	[case_prefix]_rain_rates.txt	?
PRECIP_ONLY toggle:	-	No	?

Help | Restore Defaults | OK | Cancel

HTML Help for Meteorology Component

The screenshot shows a web browser window with the address bar displaying `csdms.colorado.edu/wiki/Model_help:TopoFlow-Meteorology`. The browser's toolbar includes icons for Gmail, Google, my.cu.edu, CSDMS, Wikipedia, YouTube, Apple, News, Popular, Google Maps, Mail, and Other Bookmarks. The CSDMS logo is prominently displayed at the top left, with the text "COMMUNITY SURFACE DYNAMICS MODELING SYSTEM" below it. A navigation menu at the top contains links for Models, CMT, Supercomputing, Education, Data, Community, Meetings, Help, and Wiki tools. The main content area features a question mark icon followed by the title "TopoFlow-Meteorology". Below the title, a paragraph describes the module as the meteorology process component for a DB-based, spatial hydrologic model. A section titled "Model introduction" follows, providing a detailed description of the component's functionality, including its use of input files and various atmospheric and land surface variables. Another section titled "Model parameters" is visible, containing a set of tabs for "Input 1", "Input 2", "Input 3", "Output 1", and "Output 2". Below these tabs is a table with three columns: "Parameter", "Description", and "Unit".

CSDMS
COMMUNITY SURFACE DYNAMICS MODELING SYSTEM

log in

Search

Models ▾ CMT ▾ Supercomputing ▾ Education ▾ Data ▾ Community ▾ Meetings ▾ Help ▾ Wiki tools ▾

? TopoFlow-Meteorology

The module is the meteorology process component for a DB-based, spatial hydrologic model

Model introduction

This component reads a variety of variables for the atmosphere and for the land surface from input files or as simple scalars. It then computes many additional variables, such as vapor pressure, e_{air} , and net shortwave (solar) radiation, Q_{SW} , using built-in shortwave and longwave radiation calculators that are based on celestial mechanics and widely-used empirical relationships. These additional variables are needed by the Snowmelt → Energy Balance and Evaporation → Energy Balance components. Direct, diffuse and back-scattered radiation fluxes are all modeled. Properties of the atmosphere (e.g. precipitation rate, P , air temperature, T_{air} , relative humidity, RH , and dust attenuation, γ), are used as well as surface/topographic properties (e.g. slope angle, aspect angle and surface albedo). The approach used here closely follows the one outlined in Appendix E of Dingman (2002). However, instantaneous vs. day-integrated radiation fluxes are used and the optical air mass is modeled using the widely used method of Kasten and Young (1989).

Model parameters

Input 1 Input 2 Input 3 Output 1 Output 2

Parameter	Description	Unit
Components status		Enabled/Disabled
Input directory		
Output directory		
Site prefix	file prefix for the study site	
Case prefix	file prefix for the model scenario	
Number of steps	number of time steps	
Time step	meteorology time step	sec
o-H ₂ O type	allowed input types (Scalar/Grid/Time Series/Grid Sequence)	

HTML Help for Meteorology Component

Uses ports

- Snow (Snowmelt)

Provides ports

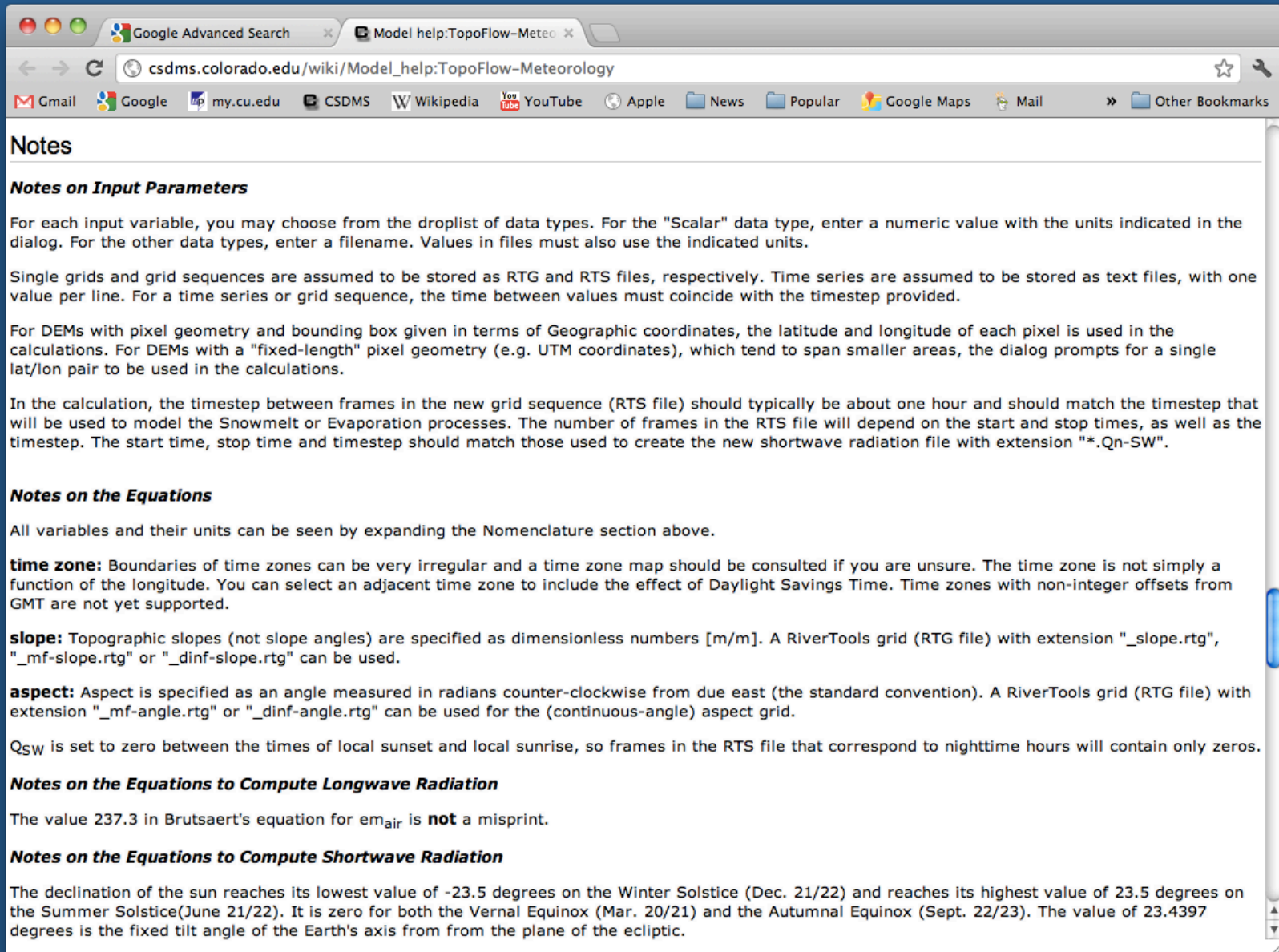
- Meteorology
- Configure (tabbed dialog GUI to change settings)
- Run (only if used as the Driver)

Main equations

Equations Used to Compute Longwave Radiation

- Net longwave radiation
$$Q_{LW} = (LW_{in} - LW_{out}) \quad (1)$$
- Incoming longwave radiation (using Stefan-Bolzman law)
$$LW_{in} = \varepsilon_{air} \cdot \sigma \cdot (T_{air} + 273.15)^4 \quad (2)$$
- Outgoing longwave radiation (using Stefan-Bolzman law)
$$LW_{out} = \varepsilon_{surf} \cdot \sigma \cdot (T_{surf} + 273.15)^4 + (1 - \varepsilon_{surf}) LW_{in} \quad (3)$$
- Saturation vapor pressure, Brutsaert (1975) method
$$e_{sat} = 0.611 \cdot \exp[(17.3 \cdot T) / (T + 237.3)] \quad (4)$$
- Saturation vapor pressure, Satterlund (1979) method
$$e_{sat} = \frac{10^{[11.4 - (2353 / (T + 273.15))]}{1000} \quad (4)$$
- Vapor pressure of air
$$e_{air} = e_{sat} \cdot RH \quad (5)$$

HTML Help for Meteorology Component



Google Advanced Search x Model help:TopoFlow-Meteo x

csdms.colorado.edu/wiki/Model_help:TopoFlow-Meteorology

Gmail Google my.cu.edu CSDMS Wikipedia YouTube Apple News Popular Google Maps Mail Other Bookmarks

Notes

Notes on Input Parameters

For each input variable, you may choose from the droplist of data types. For the "Scalar" data type, enter a numeric value with the units indicated in the dialog. For the other data types, enter a filename. Values in files must also use the indicated units.

Single grids and grid sequences are assumed to be stored as RTG and RTS files, respectively. Time series are assumed to be stored as text files, with one value per line. For a time series or grid sequence, the time between values must coincide with the timestep provided.

For DEMs with pixel geometry and bounding box given in terms of Geographic coordinates, the latitude and longitude of each pixel is used in the calculations. For DEMs with a "fixed-length" pixel geometry (e.g. UTM coordinates), which tend to span smaller areas, the dialog prompts for a single lat/lon pair to be used in the calculations.

In the calculation, the timestep between frames in the new grid sequence (RTS file) should typically be about one hour and should match the timestep that will be used to model the Snowmelt or Evaporation processes. The number of frames in the RTS file will depend on the start and stop times, as well as the timestep. The start time, stop time and timestep should match those used to create the new shortwave radiation file with extension "*.Qn-SW".

Notes on the Equations

All variables and their units can be seen by expanding the Nomenclature section above.

time zone: Boundaries of time zones can be very irregular and a time zone map should be consulted if you are unsure. The time zone is not simply a function of the longitude. You can select an adjacent time zone to include the effect of Daylight Savings Time. Time zones with non-integer offsets from GMT are not yet supported.

slope: Topographic slopes (not slope angles) are specified as dimensionless numbers [m/m]. A RiverTools grid (RTG file) with extension "_slope.rtg", "_mf-slope.rtg" or "_dinf-slope.rtg" can be used.

aspect: Aspect is specified as an angle measured in radians counter-clockwise from due east (the standard convention). A RiverTools grid (RTG file) with extension "_mf-angle.rtg" or "_dinf-angle.rtg" can be used for the (continuous-angle) aspect grid.

Q_{SW} is set to zero between the times of local sunset and local sunrise, so frames in the RTS file that correspond to nighttime hours will contain only zeros.

Notes on the Equations to Compute Longwave Radiation

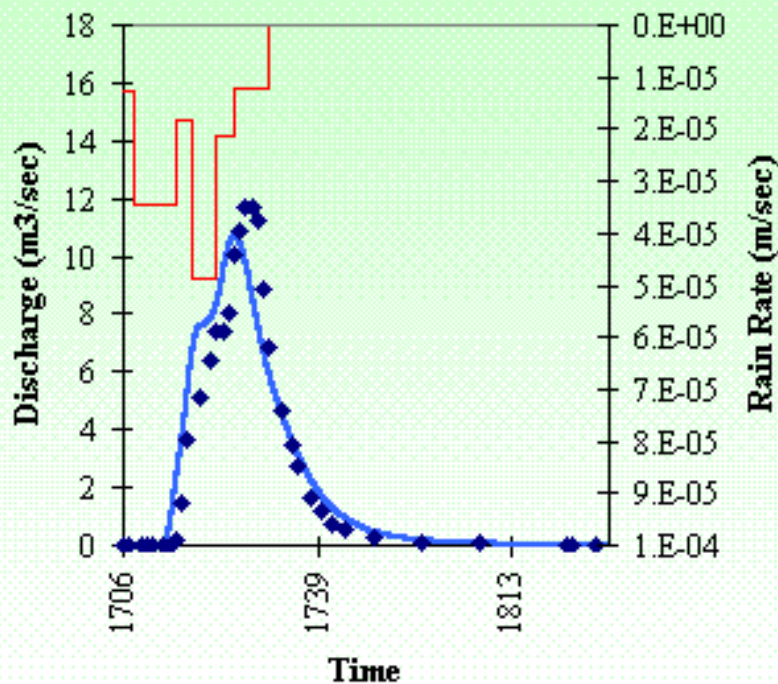
The value 237.3 in Brutsaert's equation for em_{air} is **not** a misprint.

Notes on the Equations to Compute Shortwave Radiation

The declination of the sun reaches its lowest value of -23.5 degrees on the Winter Solstice (Dec. 21/22) and reaches its highest value of 23.5 degrees on the Summer Solstice (June 21/22). It is zero for both the Vernal Equinox (Mar. 20/21) and the Autumnal Equinox (Sept. 22/23). The value of 23.4397 degrees is the fixed tilt angle of the Earth's axis from from the plane of the ecliptic.

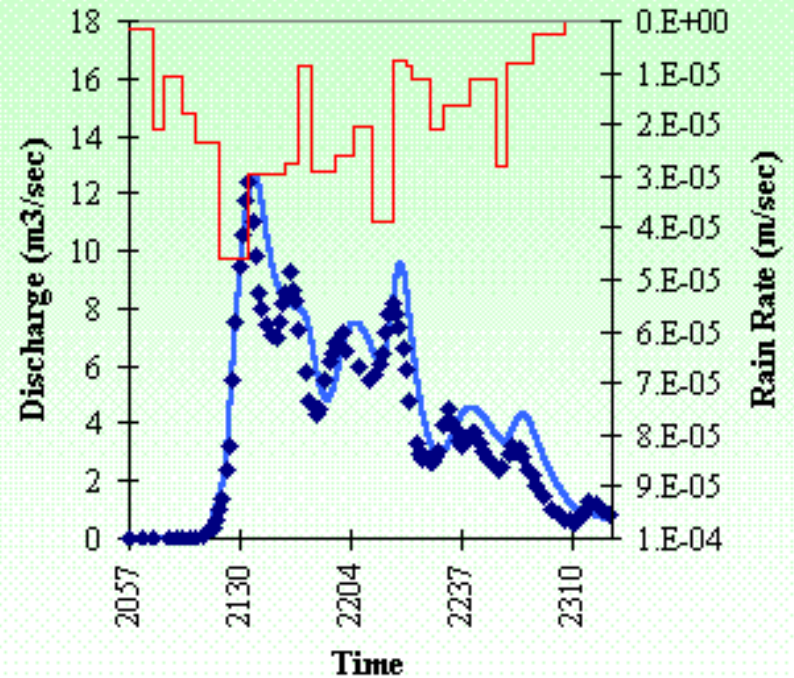
Modeled vs. Observed Hydrographs for Storms of June 7 and 20, 1967, Treynor Watershed, Iowa

Comparison of Observed and Simulated Hydrographs: Treynor Watershed 1, 6/7/67



— Simulated —◆— Observed — Storm Hyetograph

Comparison of Observed and Simulated Hydrographs: Treynor Watershed 1, 6/20/67



— Simulated —◆— Observed — Storm Hyetograph

*Save Model Results in netCDF Files,
Then Use the VisIt Visualization
Tool to Make Movies*



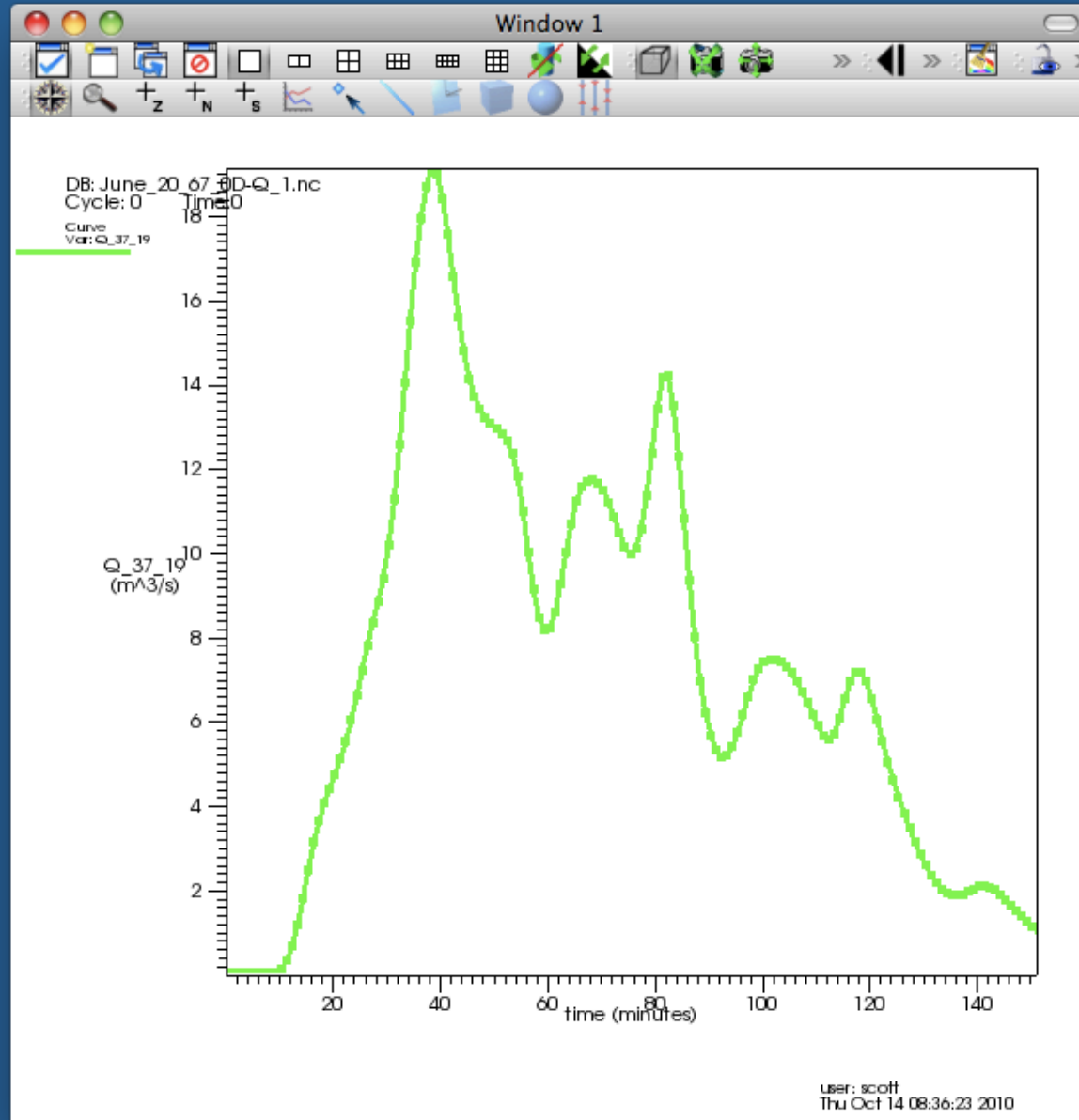
CSDMS Tools for Writing NetCDF

We used the Python package PyNIO to create classes for writing several standard types of output as netCDF files that use the CSDMS Standard Names.

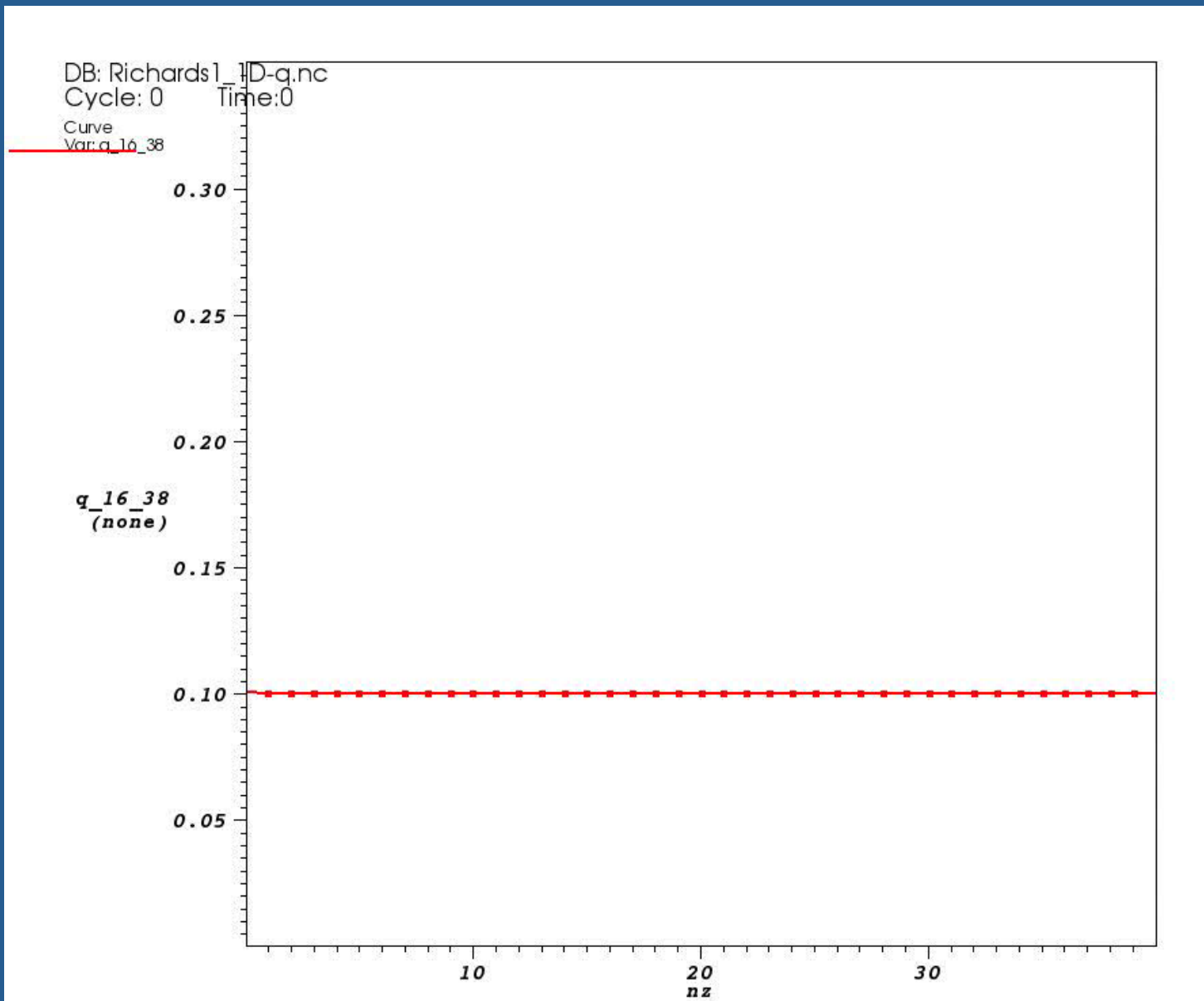
- Time series
- Profile series
- Grid stacks
- Cube stacks
- Scatter plot series (in future)

Regardless of what language your code is written in, CMI can link to this class via Babel to use the tools.

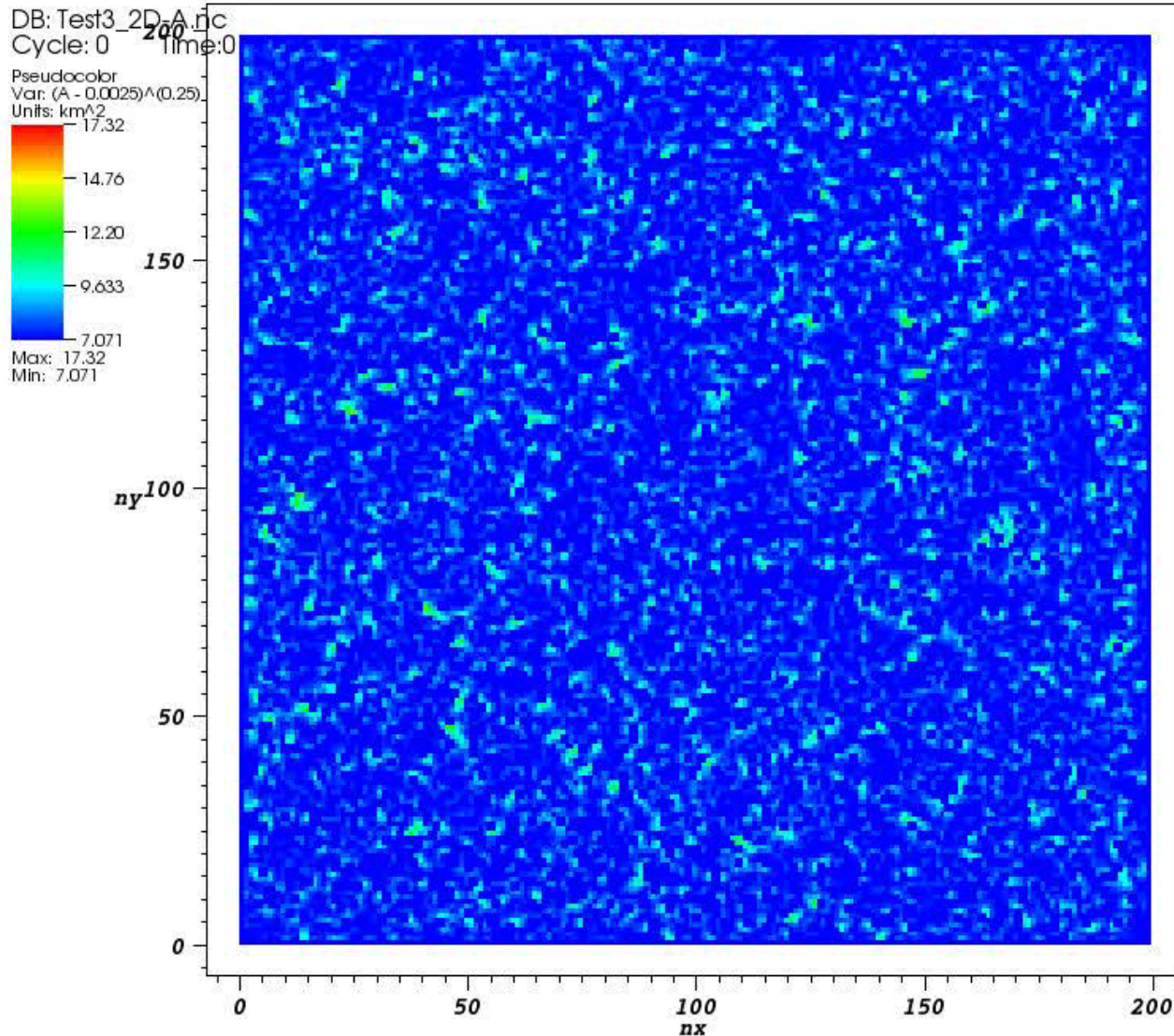
Writing Time Series to NetCDF



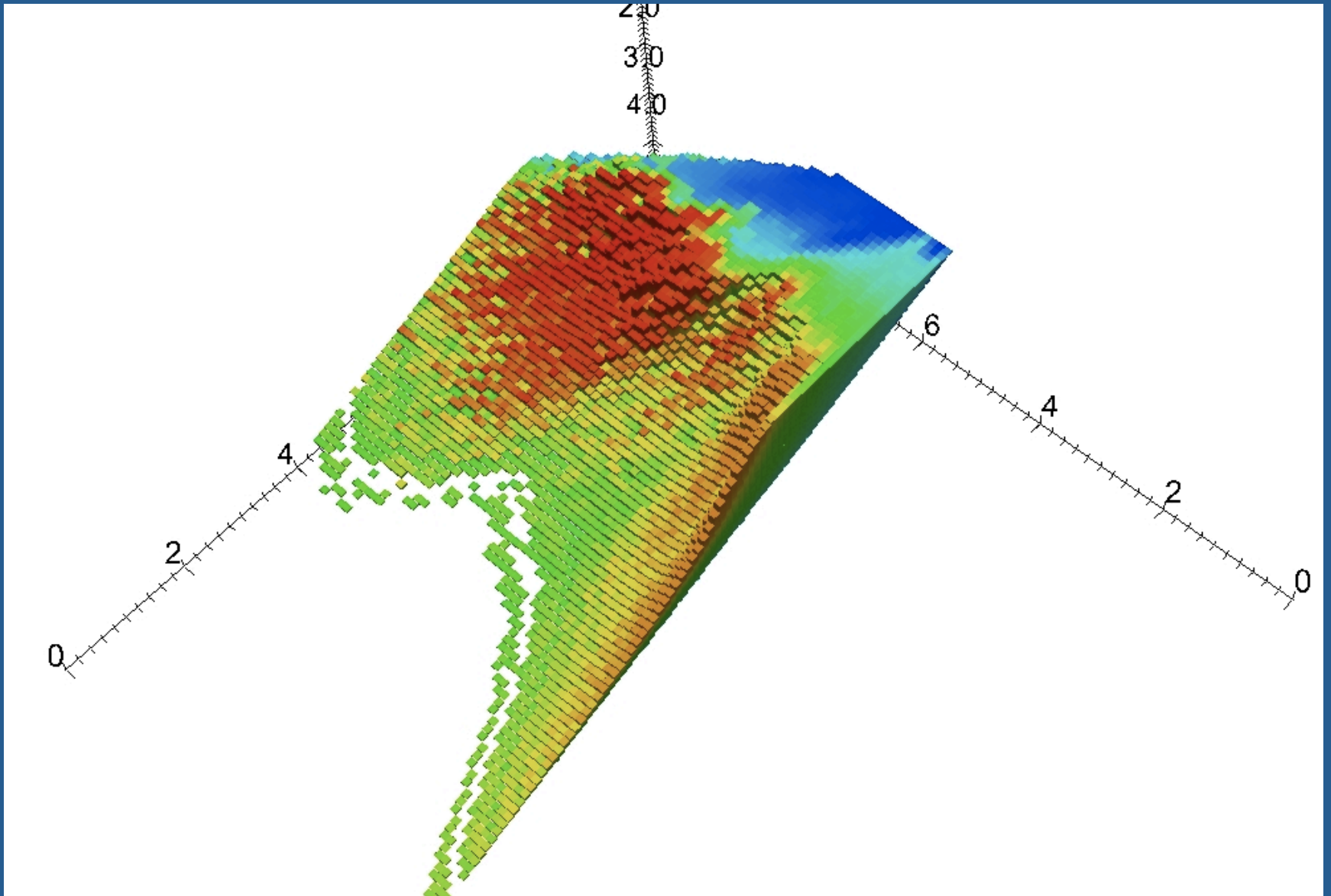
Writing "Profile Stacks" to NetCDF



Writing "Grid Stacks" to NetCDF



Writing "Cube Stacks" to NetCDF



Conclusions

- Any CSDMS member can download and install the CSDMS Modeling Tool (CMT) as an easy-to-install Java app.
- Choose model components from available palettes to create customized applications, and run them on our supercomputer. The CMT GUI is linked to VisIt to provide run-time visualization.
- “Process components” are the best level of granularity for model componentization. Processes represent the “scale” at which modelers are most likely to want to replace one approach with another, i.e. to compare different approaches and algorithms with respect to speed, accuracy, scalability or realism.
- Complete models represent the next higher level of granularity but provide limited opportunities for linking and re-use.

More Conclusions

- “Grid stacks” from spatial models can be saved as netCDF files, and then imported into VisIt as a “time-varying” database to allow animation and making movies.
- CSDMS has developed a general set of rules for constructing “CSDMS Standard Names” that address the problem of semantic mediation (model-to-model or model-to-data).
- CSDMS components are designed to be used as either “drivers” (i.e. stand-alone models) or as components.
- Each component has its own GUI for both input and output options as well as its own HTML help page. These pages have many possible uses, such as giving credit to the author of the model component.