CSDMS and the Terrestrial World: Looking Back, Looking Forward

CSDMS annual meeting, March 2013

Original goals of CSDMS

From the 2008 Strategic Plan:

2. CSDMS Long Range Goals and the CSDMS Cyber-Infrastructure

CSDMS has two overriding long range goals. The first long-range goal is to

- *Develop a modular modeling environment capable of significantly advancing fundamental earth-system science* (see section 3).

The second long-term goal of CSDMS is to:

- *Develop fully functional and useful repositories for CSDMS data, for CSDMS models and numerical tools, and for educational use.*

Model and Educational Repositories

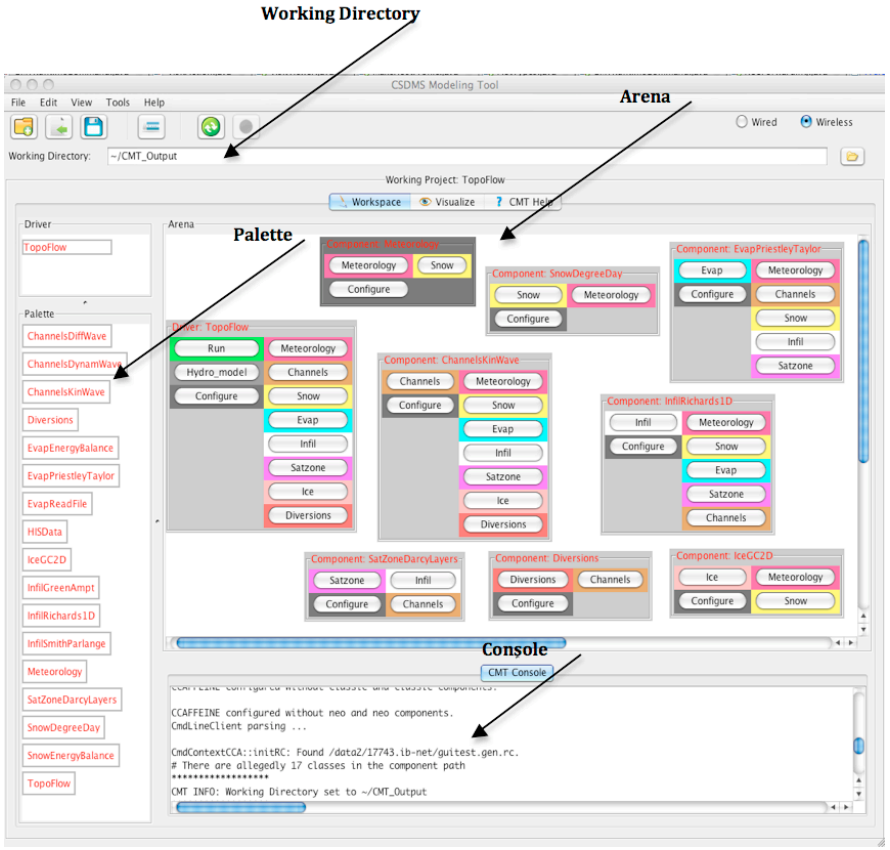
- Model repository: 166 models, 54 tools
  - http://csdms.colorado.edu/wiki/Models_all
- Education repository: movies, labs, lectures, images:
  - http://csdms.colorado.edu/wiki/Movies_portal
- Data repository:
  - http://csdms.colorado.edu/wiki/Data_download

Supercomputer: Beach

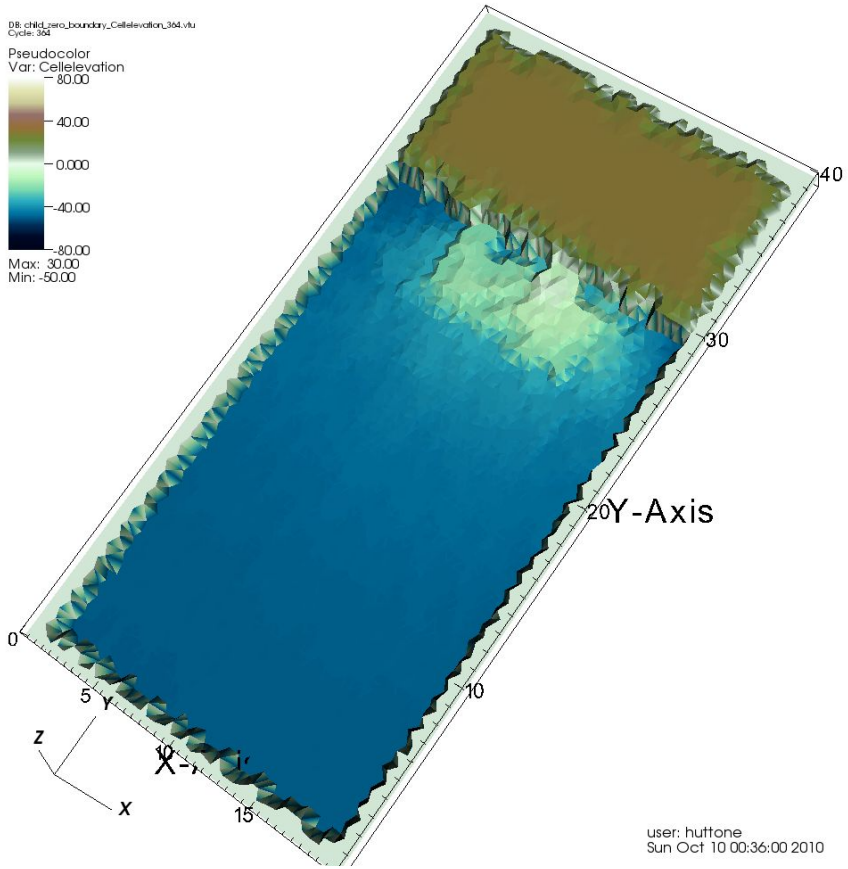
CSDMS Modeling Framework

- Design, architecture, and cyber-infrastructure for plug-and-play coupling of models
  - Different languages
  - HPC support
- GUI interface: CMT

Rapid idea generation, exploration, and hypothesis testing



Example of CSDMS modeling framework: coupling CHILD (landscape evolution) and SedFlux (deltaic deposition)



Source-to-sink model combining CHILD and SedFlux components (courtesy of Eric Hutton)

Making “plug and play” a reality: the Basic Model Interface (BMI)

Model Control Functions

```
void initialize (in string config_file)
void update (in double dt) //  Advance model variables by time interval, dt (dt=-1 means use model time step)
void finalize ()
void run_model (in string config_file) //  Do a complete model run. Not needed for CMI.
```

- These BMI functions are critical to plug-and-play modeling because they allow a calling component to bypass a model's own time loop. They also provide the caller with ***fine-grained control*** over the model, similar to a TV remote control.
- The ***initialize()*** function accepts a string argument that gives the name (and path) of its "main input file", called a ***configuration file***. This function should perform all tasks that are to take place before entering the model's time loop. Models should be refactored, if necessary, to read their inputs (which could include filenames for other input files) from a configuration file (a text file). CSDMS does not impose any constraint on how configuration files are formatted, but a "template" of your model's config file (with placeholder values) is used when the CSDMS-provided GUI creates a config file for your model.
- The ***update()*** function accepts a time step argument, "dt". If (dt == -1), then the model should use its own (internal) timestep; otherwise it should use the value provided. This function should perform all tasks that take place during one pass through the model's time loop. It does not contain the time loop. This typically includes incrementing all of the model's state variables. If the model's state variables don't change in time, then they can be computed by the initialize() function and this function can just return without doing anything.
- The ***finalize()*** function should perform all tasks that take place after exiting the model's time loop. This typically includes deallocating memory, closing files and printing reports.
- The ***run_model()*** function is not needed by CSDMS but provides a simple method to run the model in "stand-alone mode". (It is often used by the developer; it is basically the model's "main".) It would simply call "initialize()", start a time loop that only calls "update()" and then calls "finalize()".

Model Information Functions

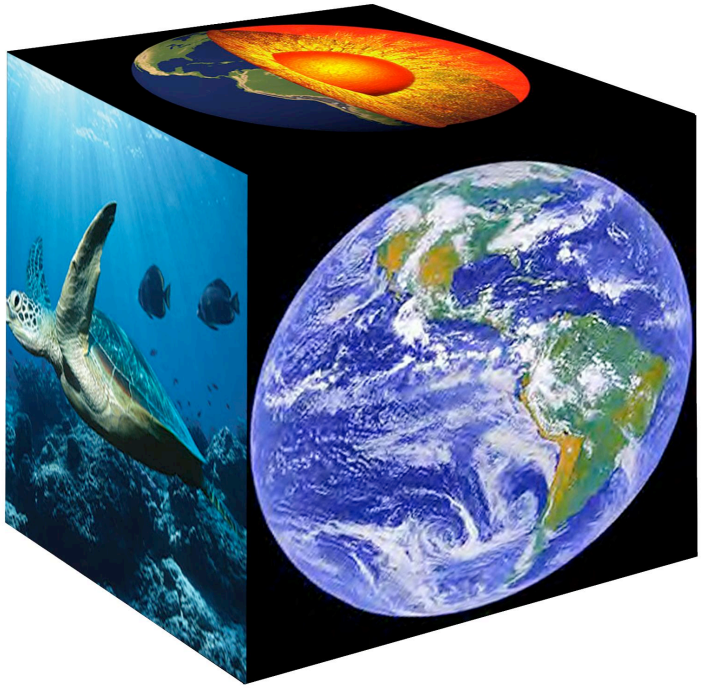
```
array<string> get_input_var_names()
array<string> get_output_var_names()
```

Building a Culture of Practice: Community Moving Toward "Best Practices" such as:

- Making open source code
- Sharing
- Using version control
- Using model identification (DOI's)
- Writing good documentation
- Implementing standard interfaces (BMI)
- Using unit tests and regression tests

Funding agencies supporting computational science; examples:

- NSF EarthCube
- NSF Software Infrastructure for Sustained Innovation (SI2)



Some needs and next steps

- Building and contributing fully-compatible components: “growing the library”
- Benchmarking, testing, calibrating, and comparing models
- Using CSDMS technology to do cutting-edge science

Role of Terrestrial Working Group

- ***GUIDE*** CSDMS
- ***CONTRIBUTE*** to CSDMS
- ***USE*** CSDMS

Tasks for this meeting

- Formulate long-term, short-term, and medium-term goals for strategic plan
- Nominate model(s) for full inclusion in CSDMS modeling framework
- Discuss “theme teams”

Agenda and google document for our breakout

- https://docs.google.com/document/d/15xy2mkn-Yu7g0wRDtTHYMQDGOHpfKZNqG_A355dywxw/edit?pli=1