Multiscale Materials Design: A model-to-model interface for multiscale materials modeling

Richard LeSar

Increasing energy use

Increasing impact on the environment

Increasing resource scarcity

Energy and environmental challenges

Need to use materials more effectively *within* energy systems

Materials have to be part of the design process

Energy and environmental challenges

Technology depends on material structures whose properties are governed by both the materials themselves and the interactions between them

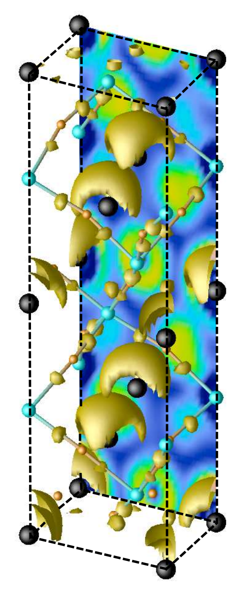
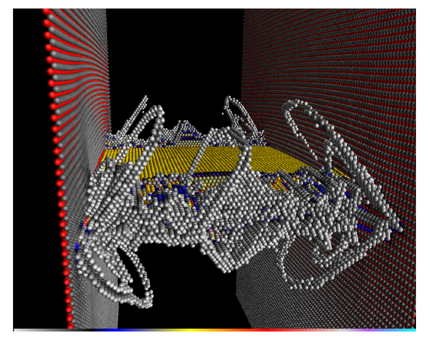
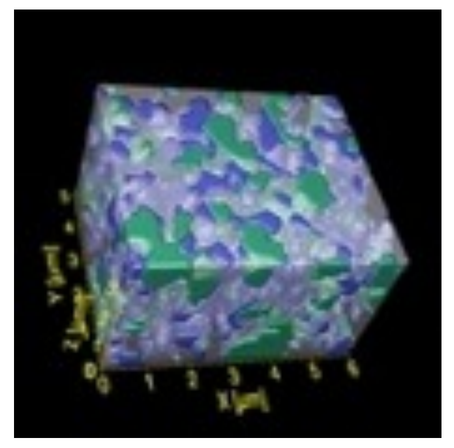
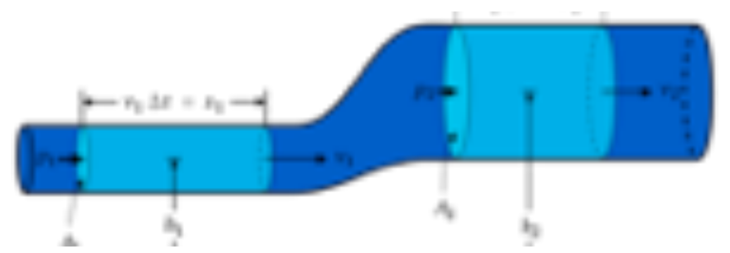
Materials are governed by physics/chemistry at many scales

Unit	Length Scale	Time Scale	Mechanics
Complex Structure	10^{3} m	10^{6} s	Structural Mechanics
Simple Structure	10^{1} m	10^{3} s	Fracture Mechanics
Component	10^{-1} m	10^{0} s	Continuum mechanics
Grain Microstructure	10^{-3} m	10^{-3} s	Crystal plasticity
Dislocation Microstructure	10^{-5} m	10^{-6} s	Micro-mechanics
Single Dislocation	10^{-7} m	10^{-9} s	Dislocation Dynamics
Atomic	10^{-9} m	10^{-12} s	Molecular Dynamics
Electron Orbitals	10^{-11} m	10^{-15} s	Quantum Mechanics

We have existing models at all scales of length and time.

The most challenging question in materials modeling is how to link information between those scales.

Length and time scales

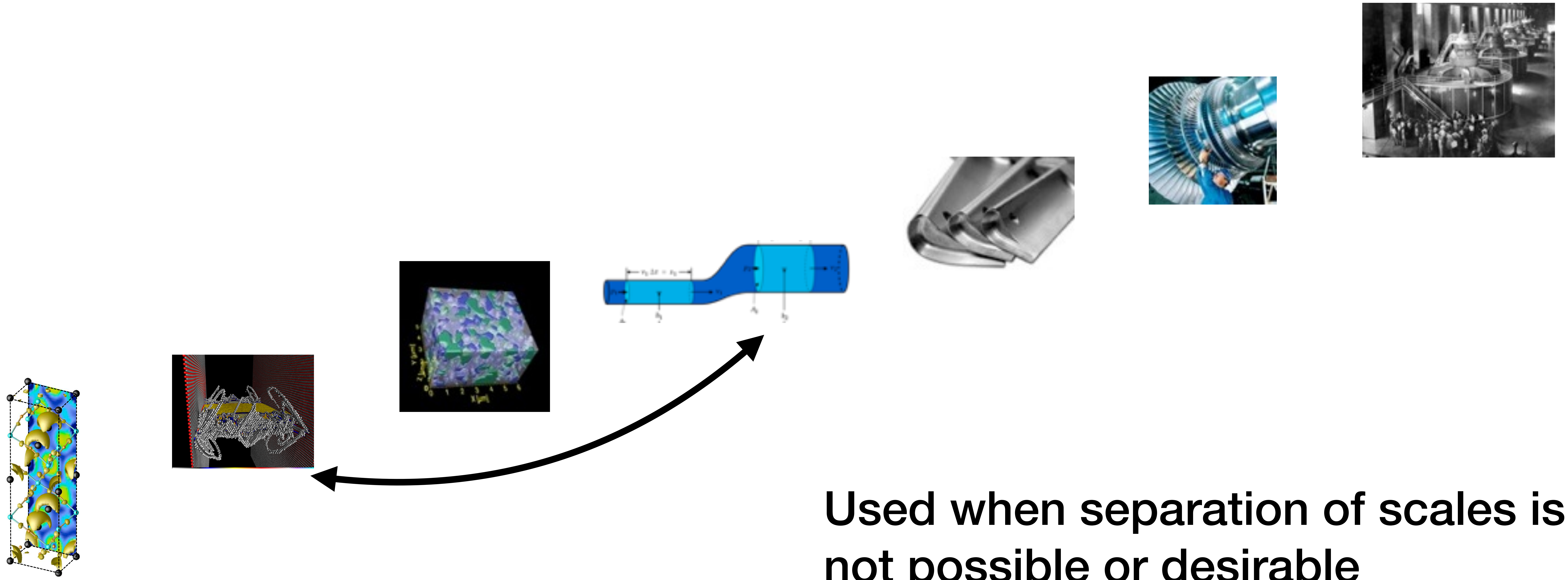
Information at each scale is passed as an input to a larger scale

- could be as parameters or models or …

Sequential multiscale or “information passing”

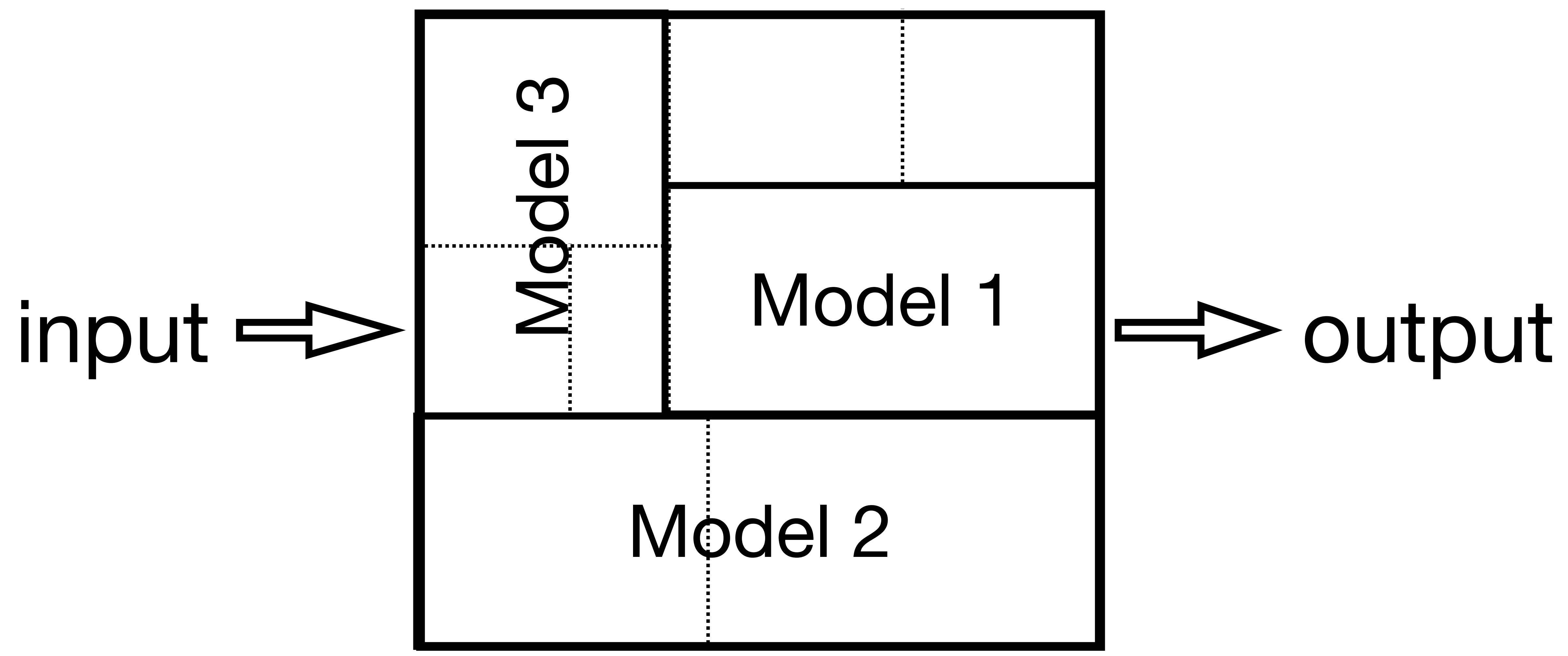
- Assumes a separation of time and length scales that is sometimes not true
- Generally passes mean quantities. How do we include distributions and what roles do their tails play?
- There are few theories to guide linking of scales.
- Information is passed in only one direction; we lack inverse models

Concurrent (coupled) multiscale



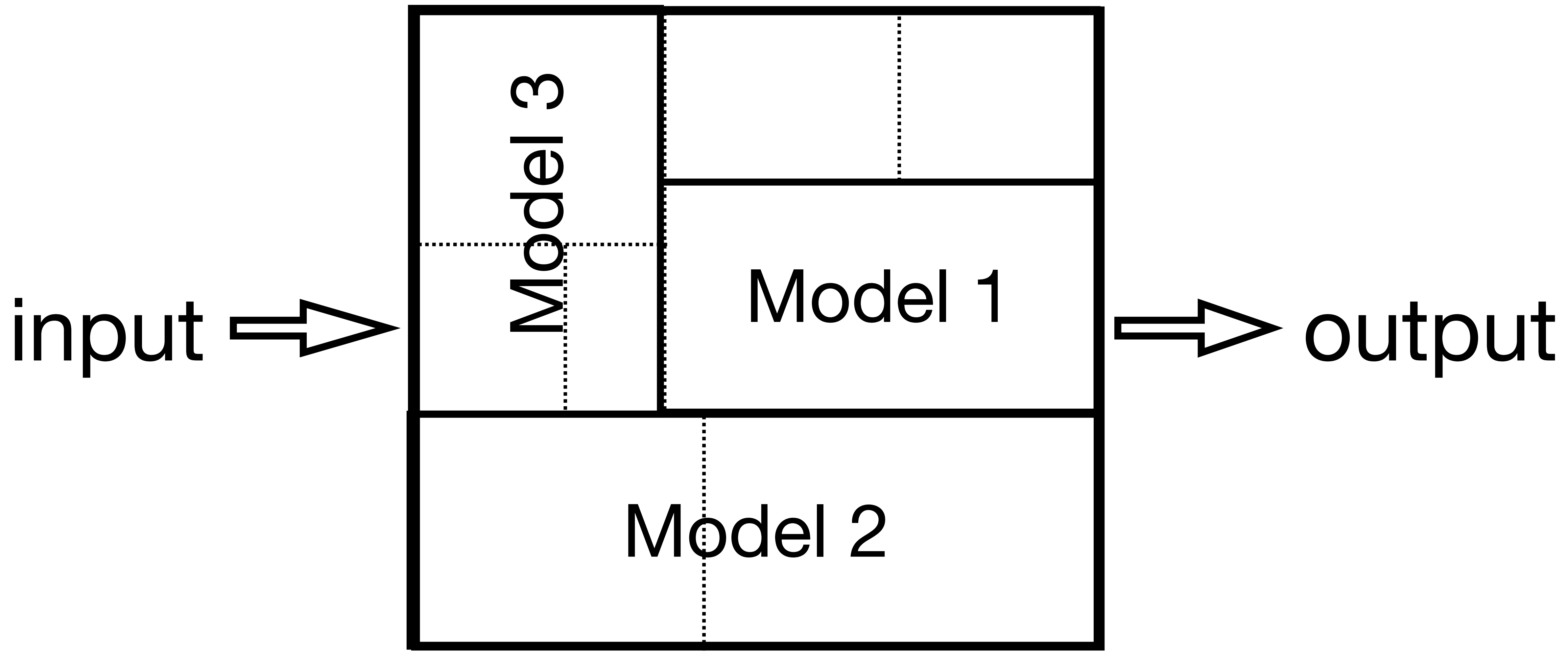
Used when separation of scales is not possible or desirable

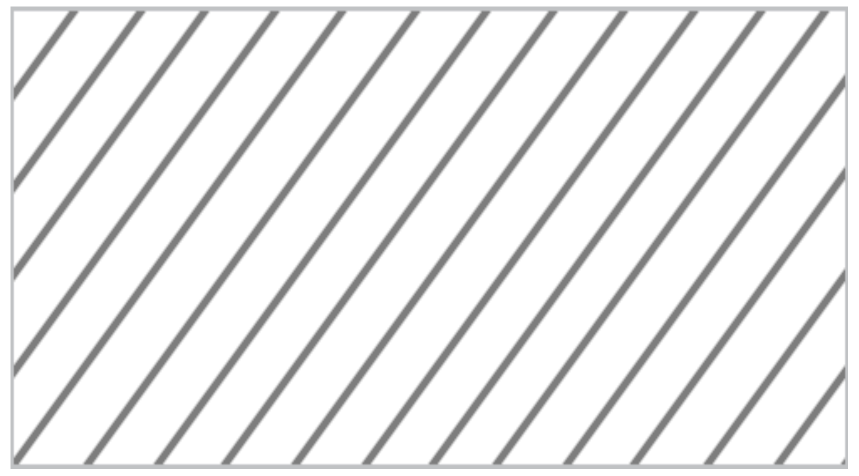
- monolithic codes consist of interconnected “subroutines” with a common data structure
- routines can be aggregated as specific models
- still must have specific interconnections and a common data structure



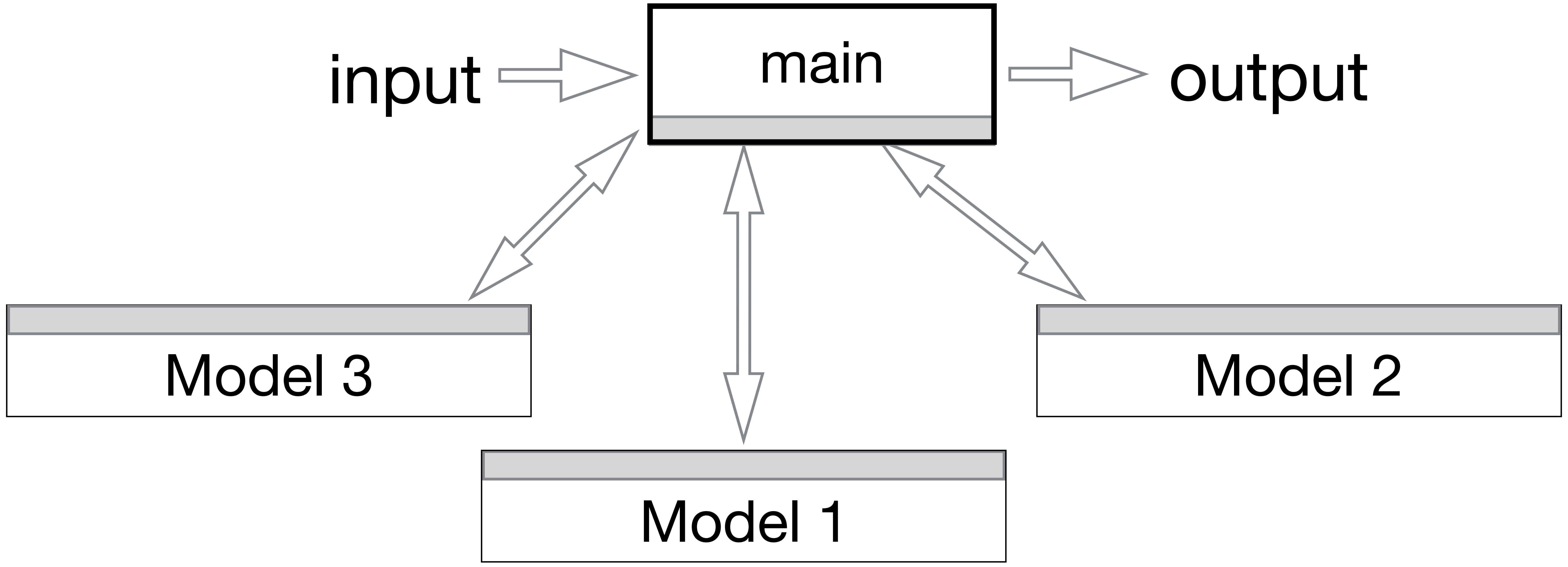
Monolithic codes

- including a new model would require that model to adopt the data structure of the rest of the code, to create linkages with other routines, …
- using the new model in some other code would require the same type of modifications
- inefficient and a barrier to adopting new models

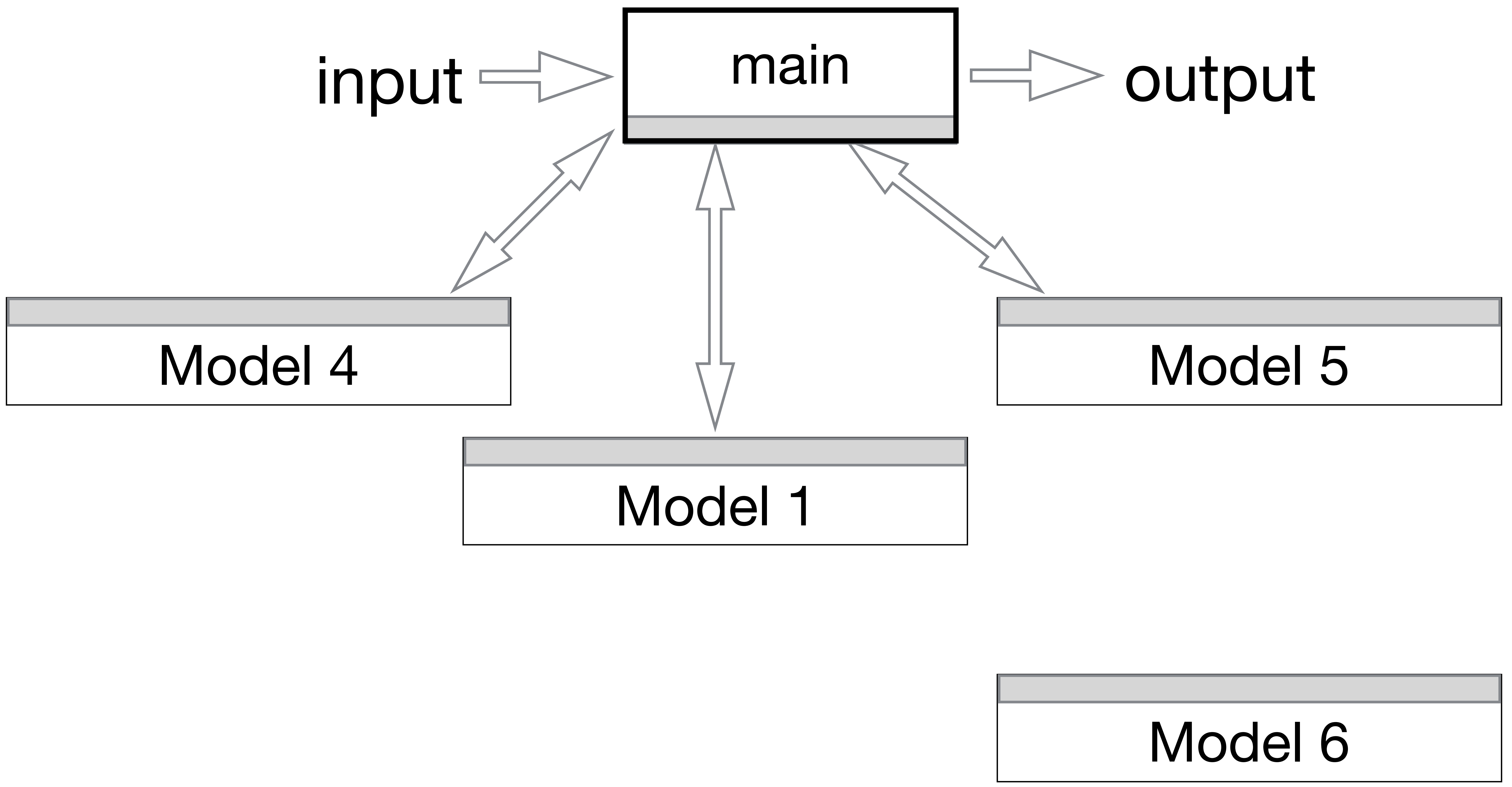




Monolithic codes are cumbersome to maintain and change

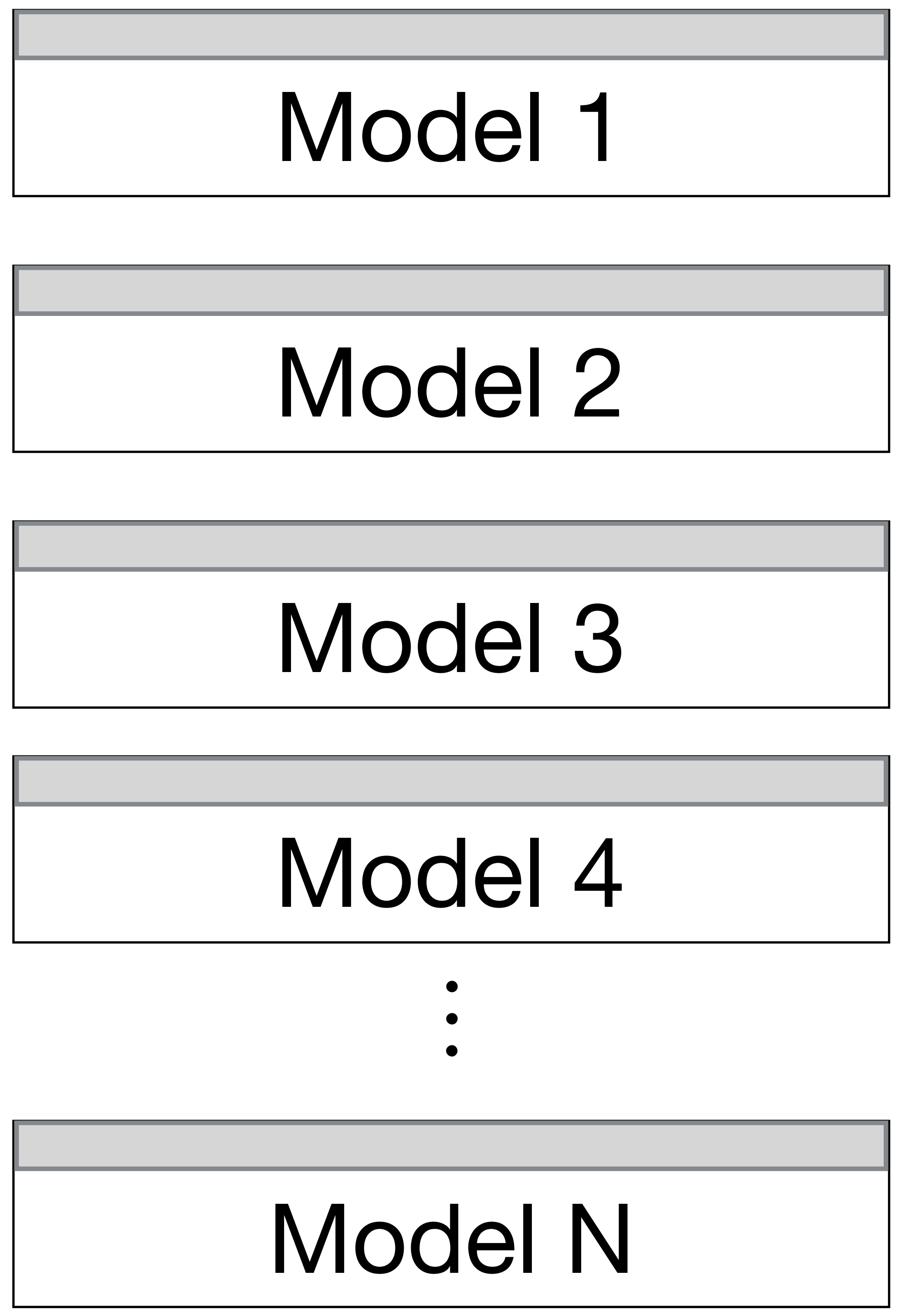


Our goal: models coupled via a standard interface



Our goal: reusable and interchangeable models

- all communication through a standard model-to-model interface
- models are autonomous, with their own data structures, grids, etc.
- models with this interface can be developed by disparate groups
- models can be in any computer language
- models are identified by a DOI with appropriate metadata



Our goal: to create a library of reusable models

The environmental modeling community faces many of the same issues as does the materials modeling community

- disparate models developed by disparate groups
- linking models

The approach taken by the Community Surface Dynamics Modeling System (CSDMS) centered at the University of Colorado seemed to meet our needs best.

They developed an approach by which their international set of collaborators could make their models available for coupling with other models.

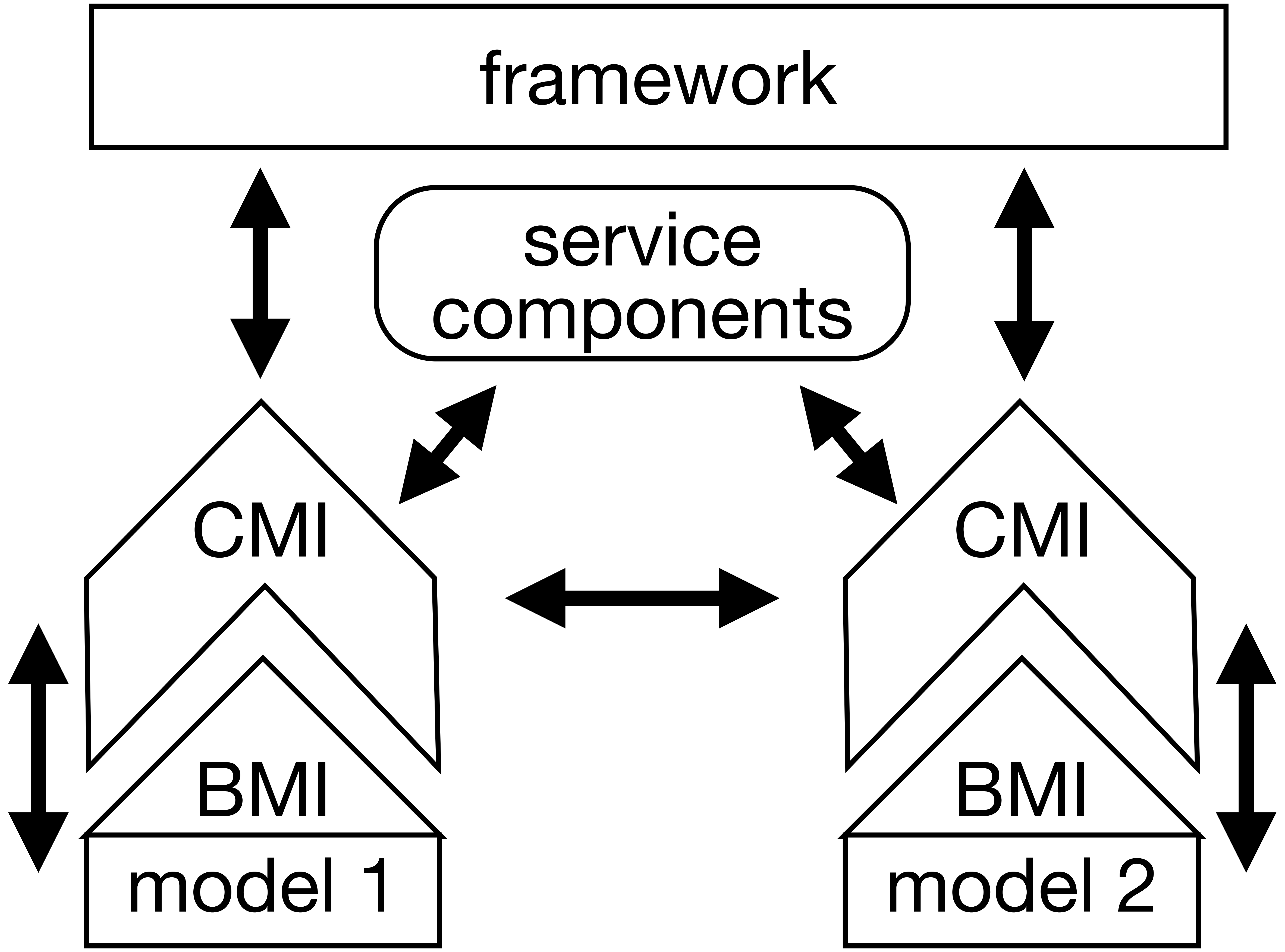
http://csdms.colorado.edu/wiki/Main_Page

Each model has added to it the Basic Model Interface (BMI)

The Common Model Interface (CMI) automatically handles conversions between languages (with Babel)

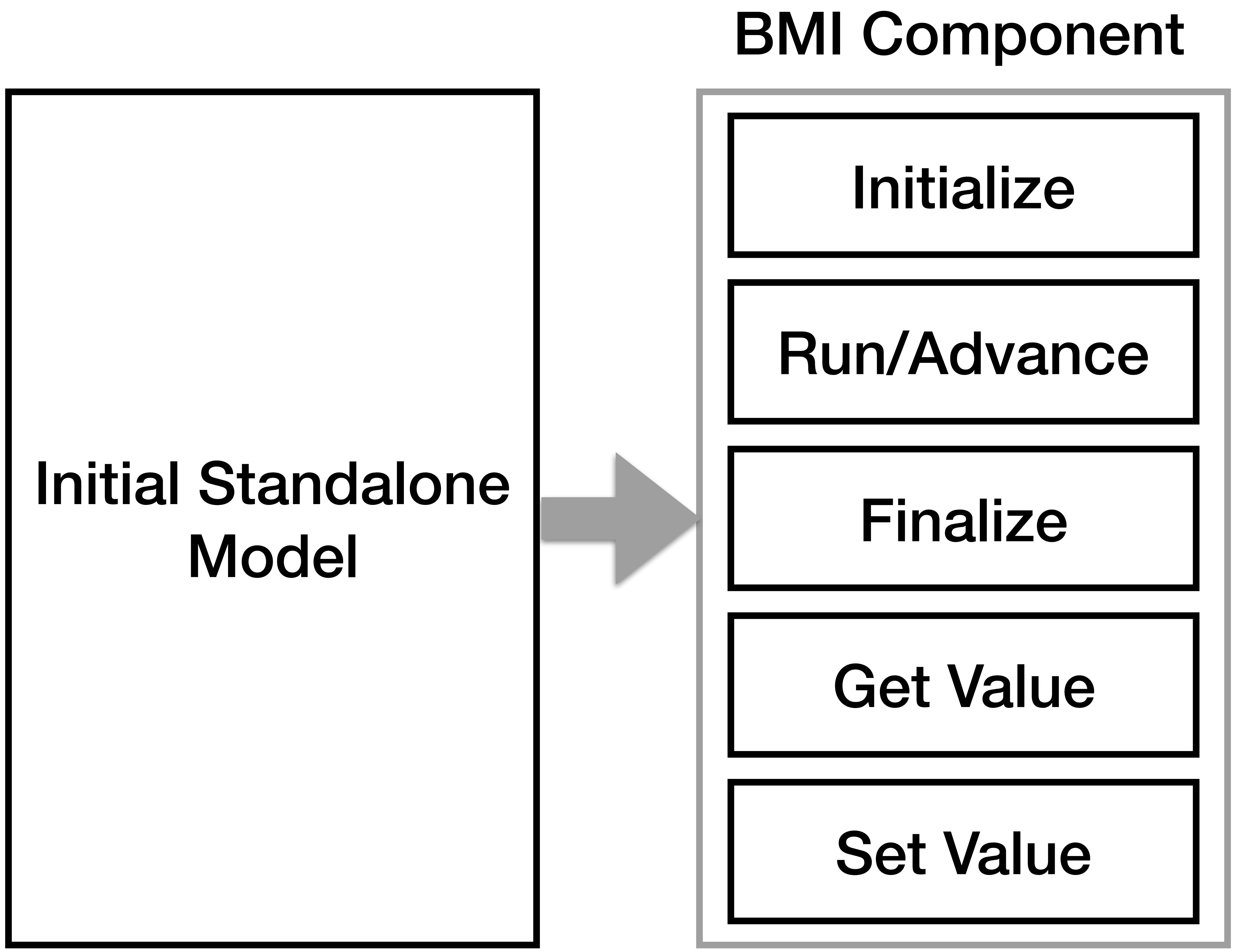
Service components handle such common activities as conversions between grids

The framework controls the calculation and the communication to/from models



The BMI is a set of common functions (available in a set of computer languages) that handles all communication with the model: input, instructions, output, …

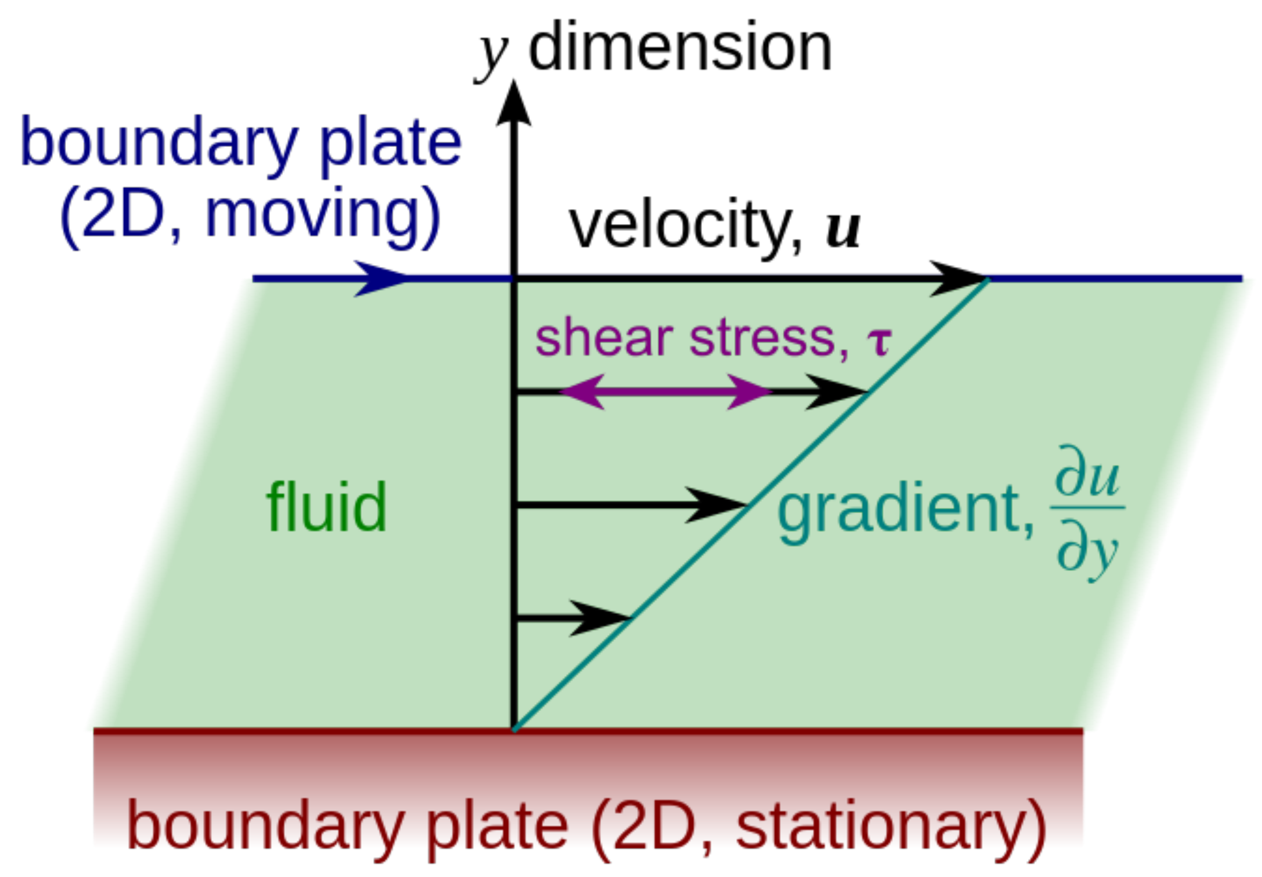
Creating a BMI for materials modeling was our first task.



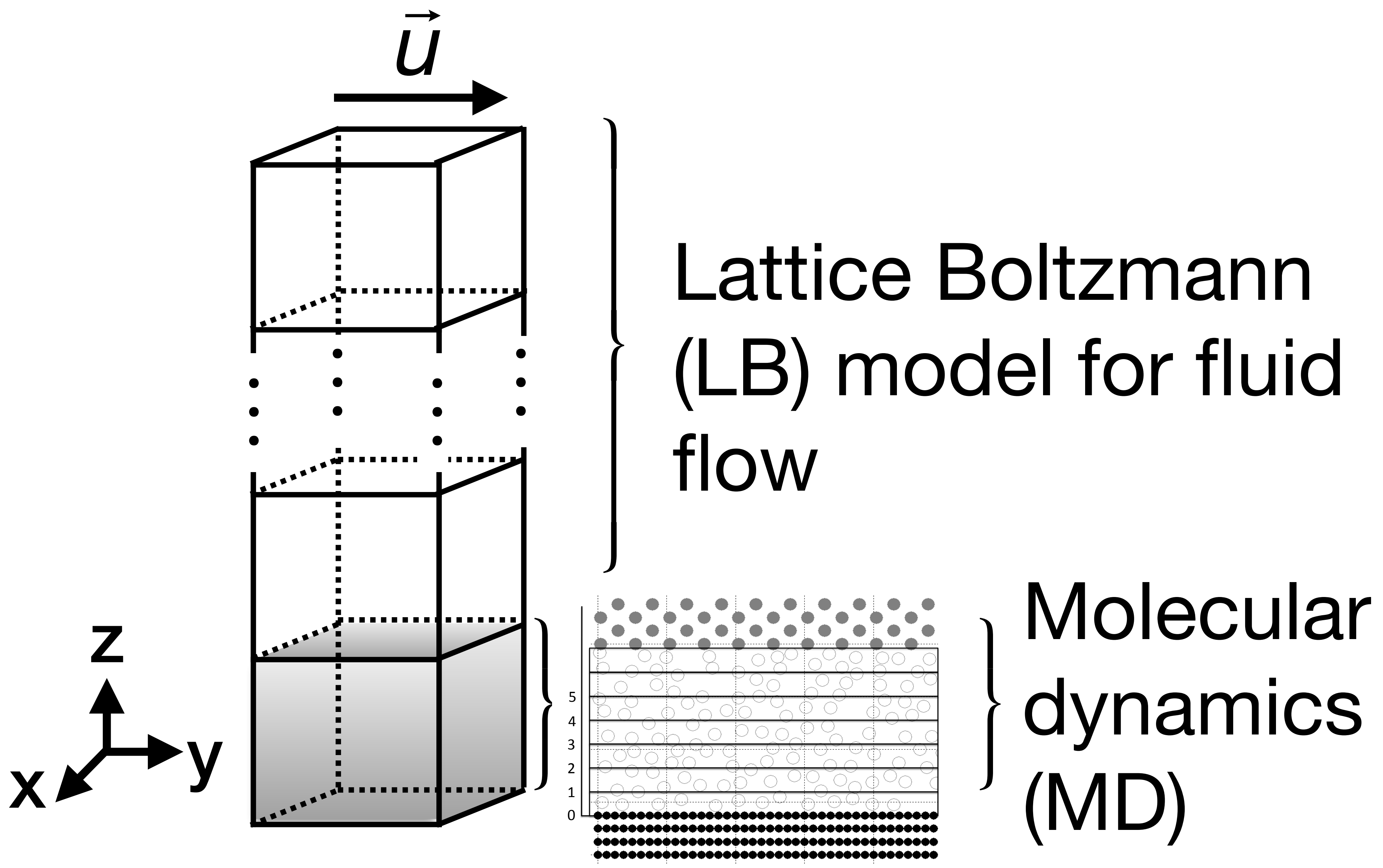
We linked fluid flow calculations to atomistics to calculate slip velocity at bottom surface in Couette flow

Created a BMI for each model and examined

- information transfer between models
- boundaries between models
- convergence of the solution
- stability
- …



y dimension
boundary plate (2D, moving)
velocity, $\boldsymbol{u}$
shear stress, τ
fluid
gradient, $\frac{\partial u}{\partial y}$
boundary plate (2D, stationary)



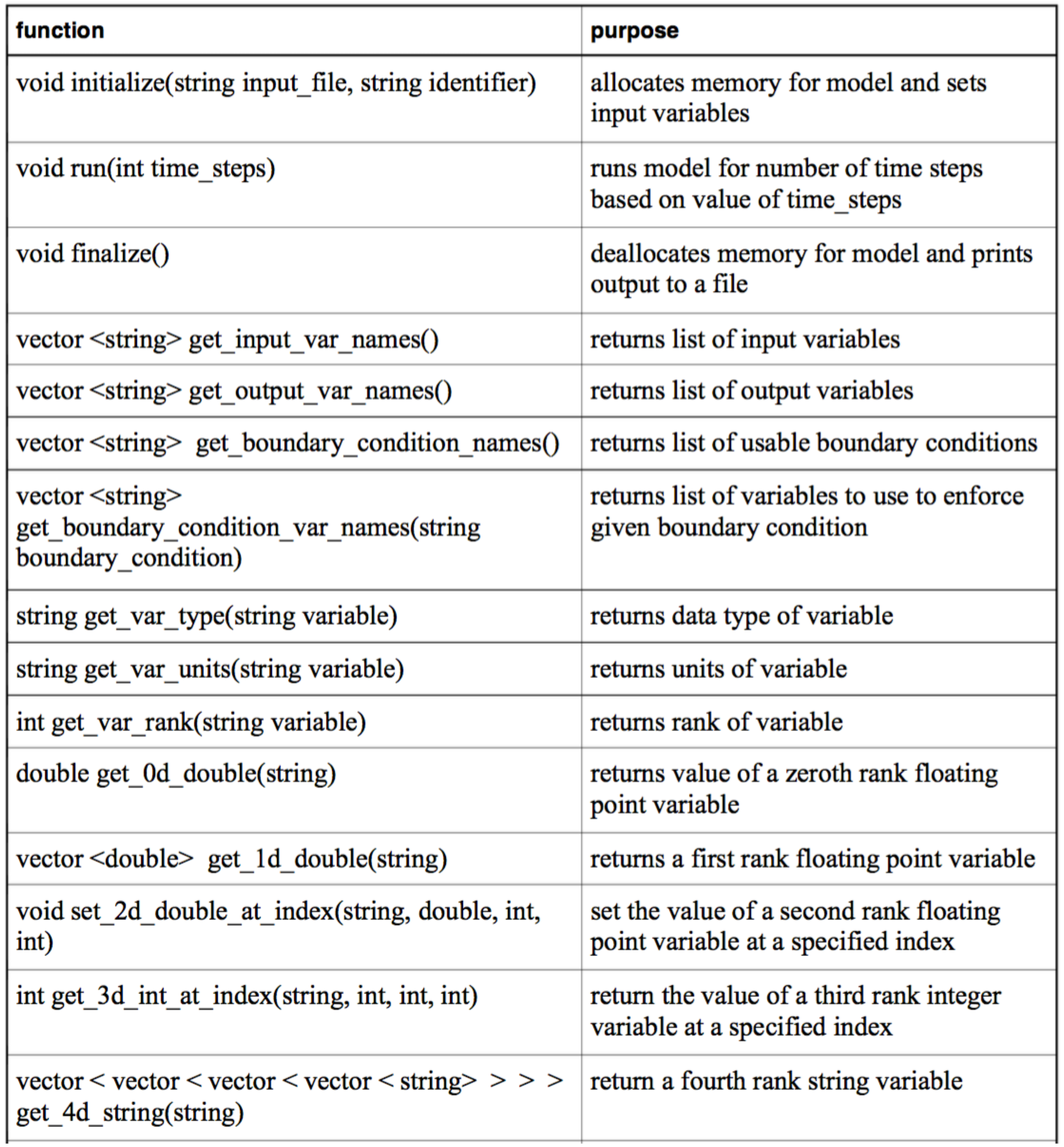
Create a hybrid system to examine key questions

- The Lattice Boltzmann (LB) and molecular dynamics (MD) models are autonomous with their own internal data structure
- Each model has its own internal units.
- Each model is solved with its own time step (very different in size).
- Each model has its own implementation of boundary conditions.
- Each model has its own requirements for convergence.

The BMI handles all communication with the models: input, instructions, output, … through a standard set of functions.

Information mediation between LB and MD

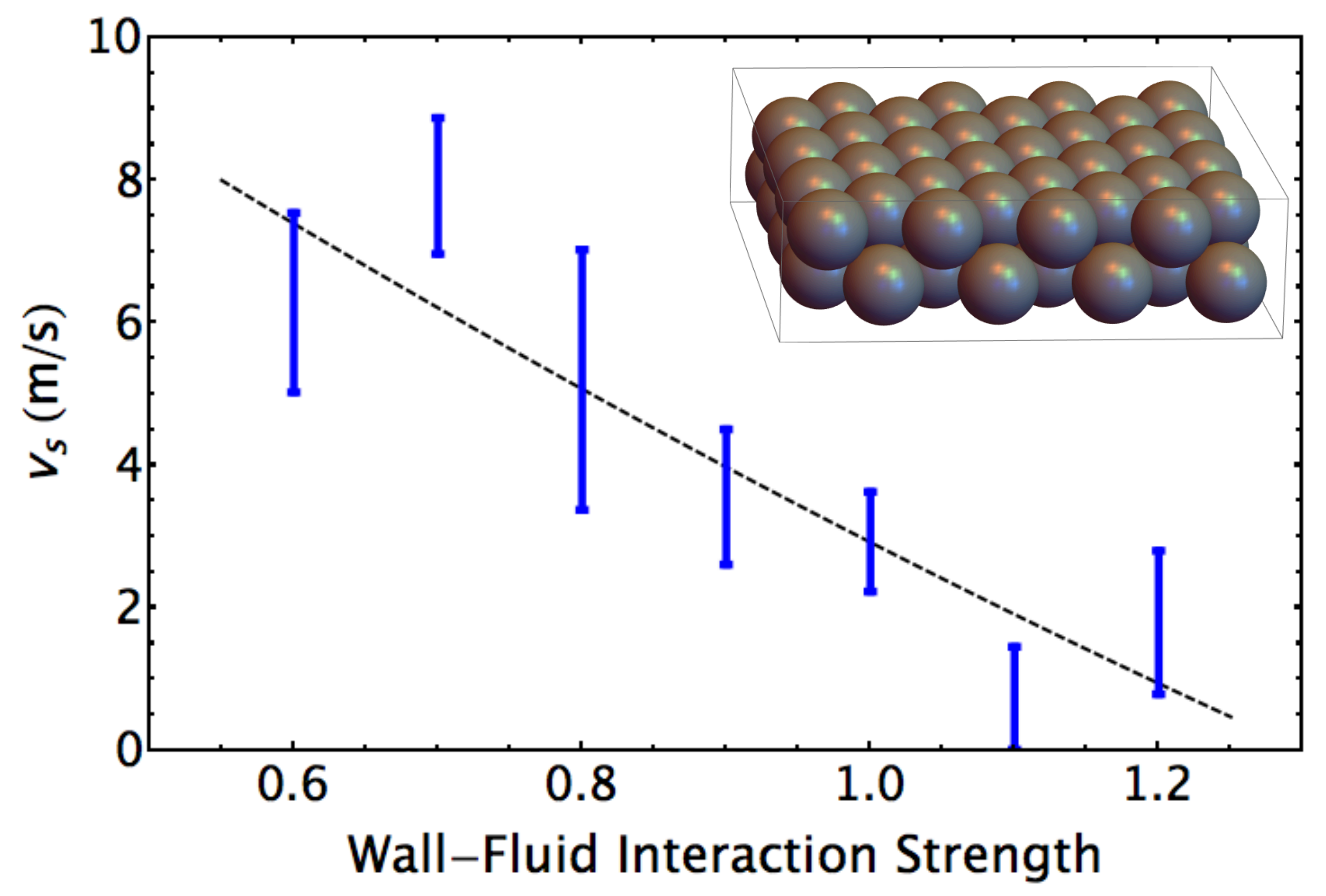
- Straightforward to implement BMI functions within code
- Originally we added specific BMI functions needed for materials (e.g., to ensure the same state)
- However, that would require modifying the CMI
- We now use only the BMI functions and include all materials specific information as models (with the BMI)

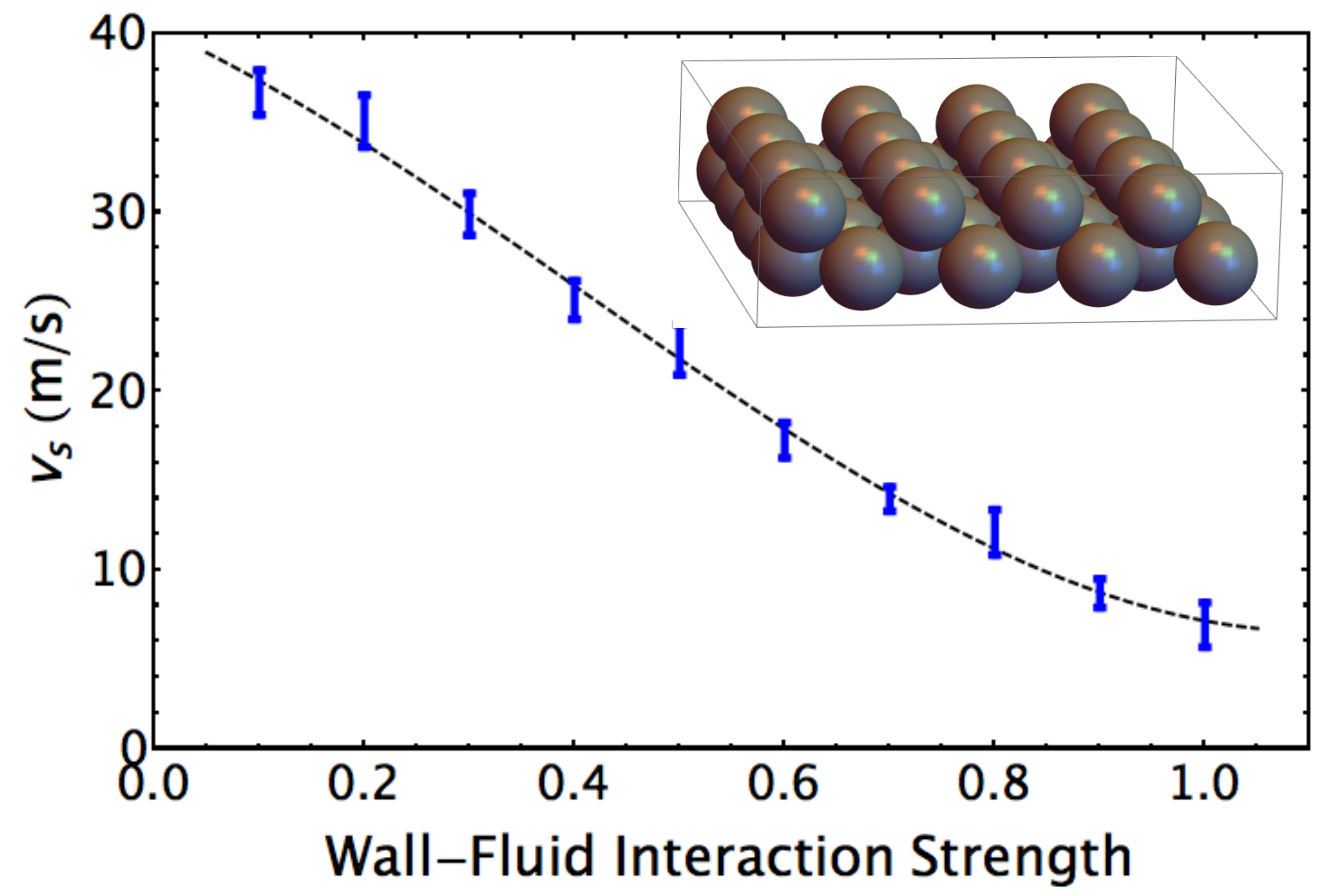


function	purpose
void initialize(string input_file, string identifier)	allocates memory for model and sets input variables
void run(int time_steps)	runs model for number of time steps based on value of time_steps
void finalize()	deallocates memory for model and prints output to a file
vector <string> get_input_var_names()	returns list of input variables
vector <string> get_output_var_names()	returns list of output variables
vector <string> get_boundary_condition_names()	returns list of usable boundary conditions
vector <string> get_boundary_condition_var_names(string boundary_condition)	returns list of variables to use to enforce given boundary condition
string get_var_type(string variable)	returns data type of variable
string get_var_units(string variable)	returns units of variable
int get_var_rank(string variable)	returns rank of variable
double get_0d_double(string)	returns value of a zeroth rank floating point variable
vector <double> get_1d_double(string)	returns a first rank floating point variable
void set_2d_double_at_index(string, double, int, int)	set the value of a second rank floating point variable at a specified index
int get_3d_int_at_index(string, int, int, int)	return the value of a third rank integer variable at a specified index
vector < vector < vector < vector < string> > > > get_4d_string(string)	return a fourth rank string variable

BMI functions: lessons learned

- Examined slip velocity as a function of solid surface crystal structure and the interactions between fluid atoms and solid atoms
- We saw no change in computational efficiency with the BMI when compared to an optimized monolithic code

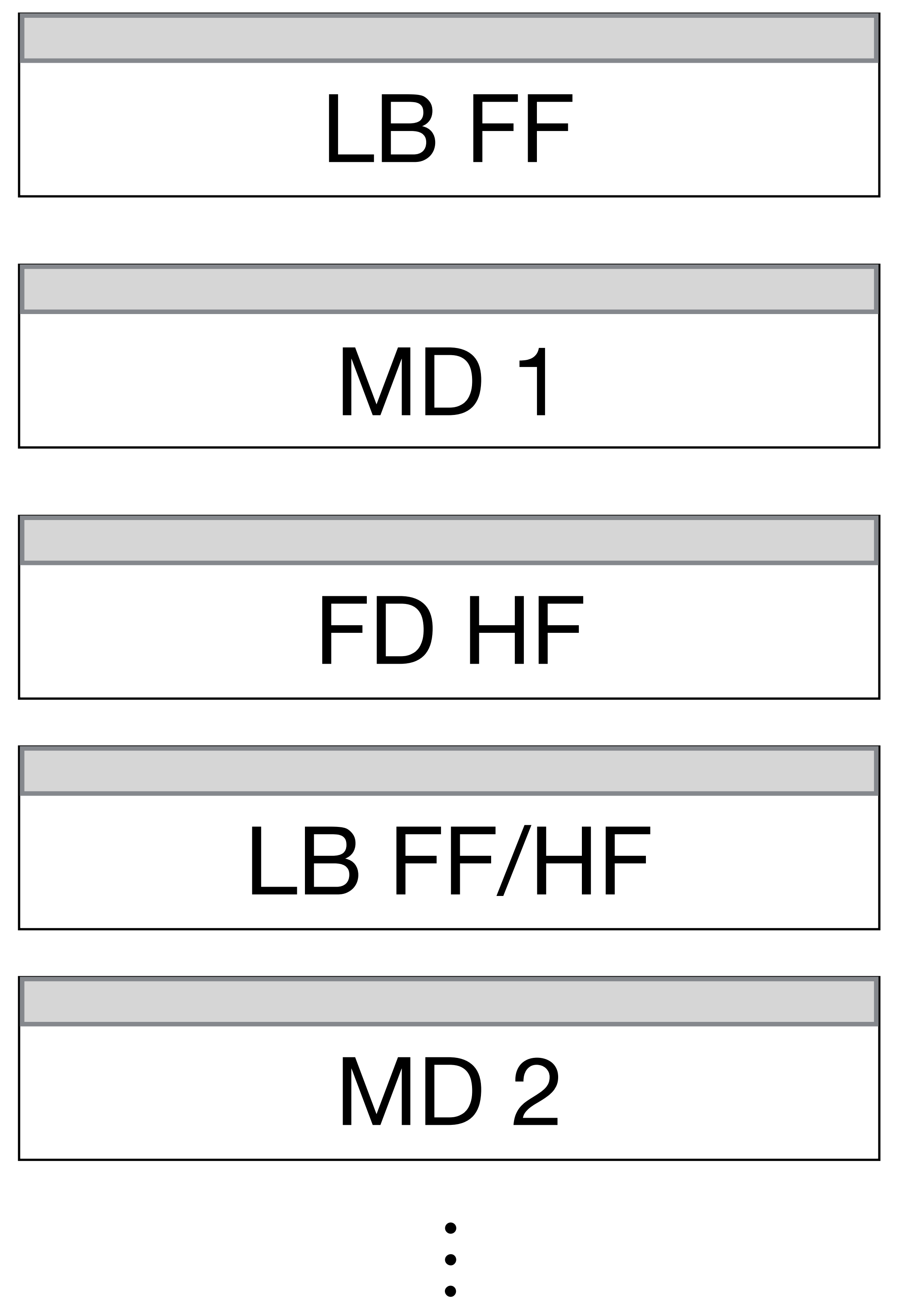




Results

"A model-to-model interface for multiscale materials simulations, P. E. Antonelli, K. M. Bryden, and R. LeSar, submitted to Computational Materials Science.

- Have a BMI version of a finite difference heat flow code (linked successfully through the BMI with our Lattice-Boltzmann fluid flow code)
- Have a BMI version Lattice Boltzmann fluid flow/heat flow code (in final stages)
- Currently creating a molecular dynamics code to model heat flow from a fluid to a solid (eventually would like to develop a BMI for an all-purpose MD code such as LAMMPS)

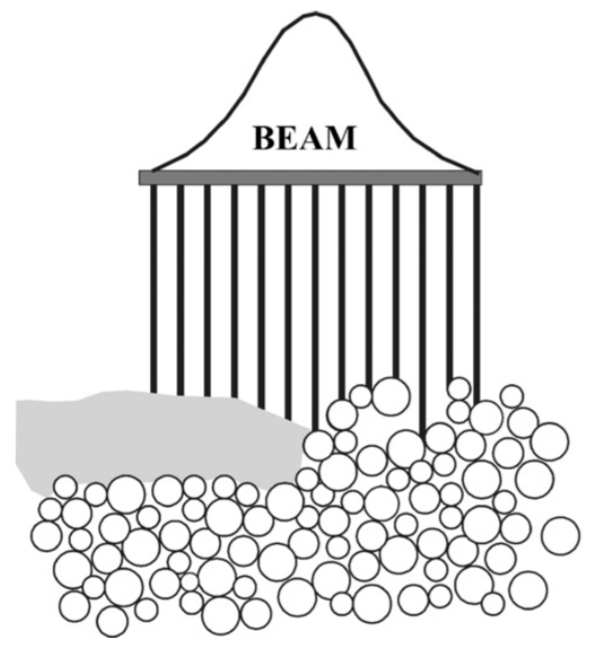


The first stages of our library of models

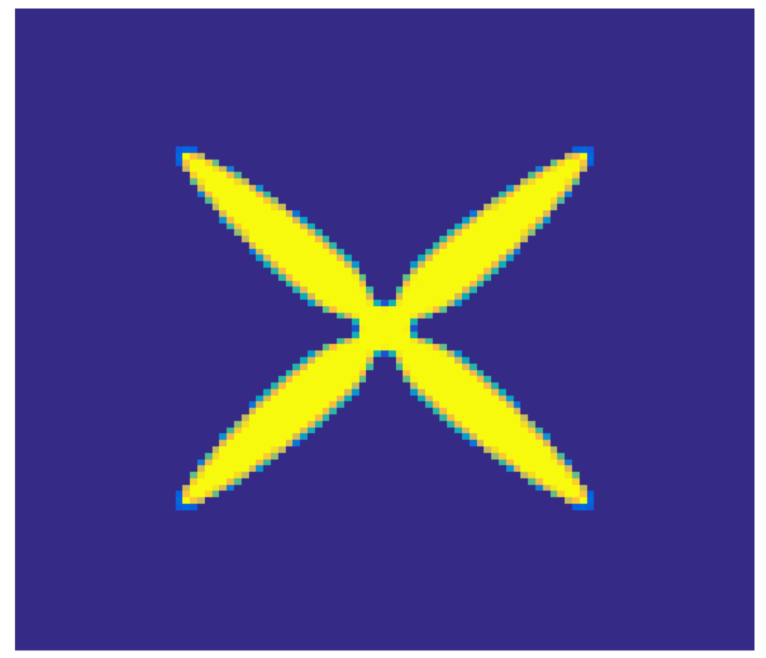
Current work

Future directions: additive manufacturing

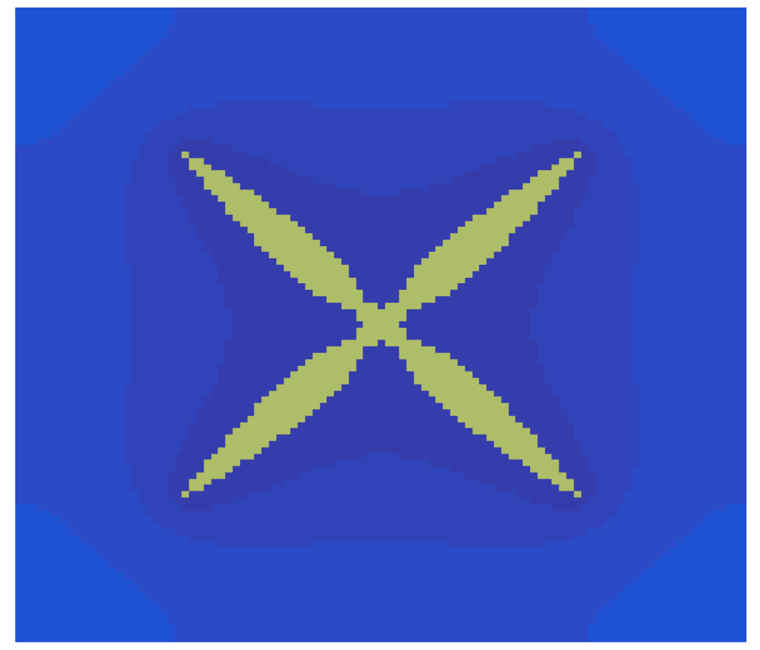
- Integrated program to develop new alloys for additive manufacturing (supported by NSF DMREF)
- Modeling couples fluid flow, heat flow, and solute flow with models of solidification to predict microstructures
- Goal would be to implement the BMI in these models to create an easy-to-modify AM simulation package for FE



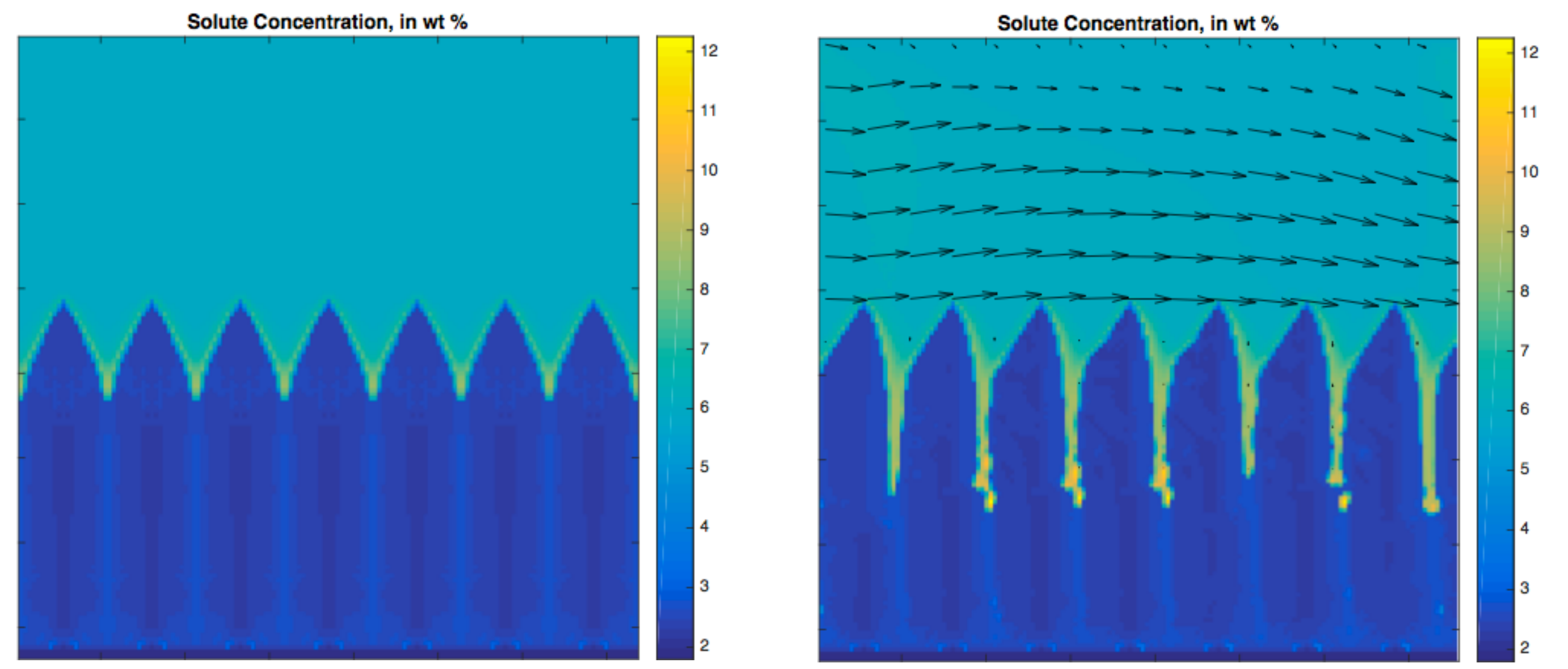
solid fraction



solute concentration

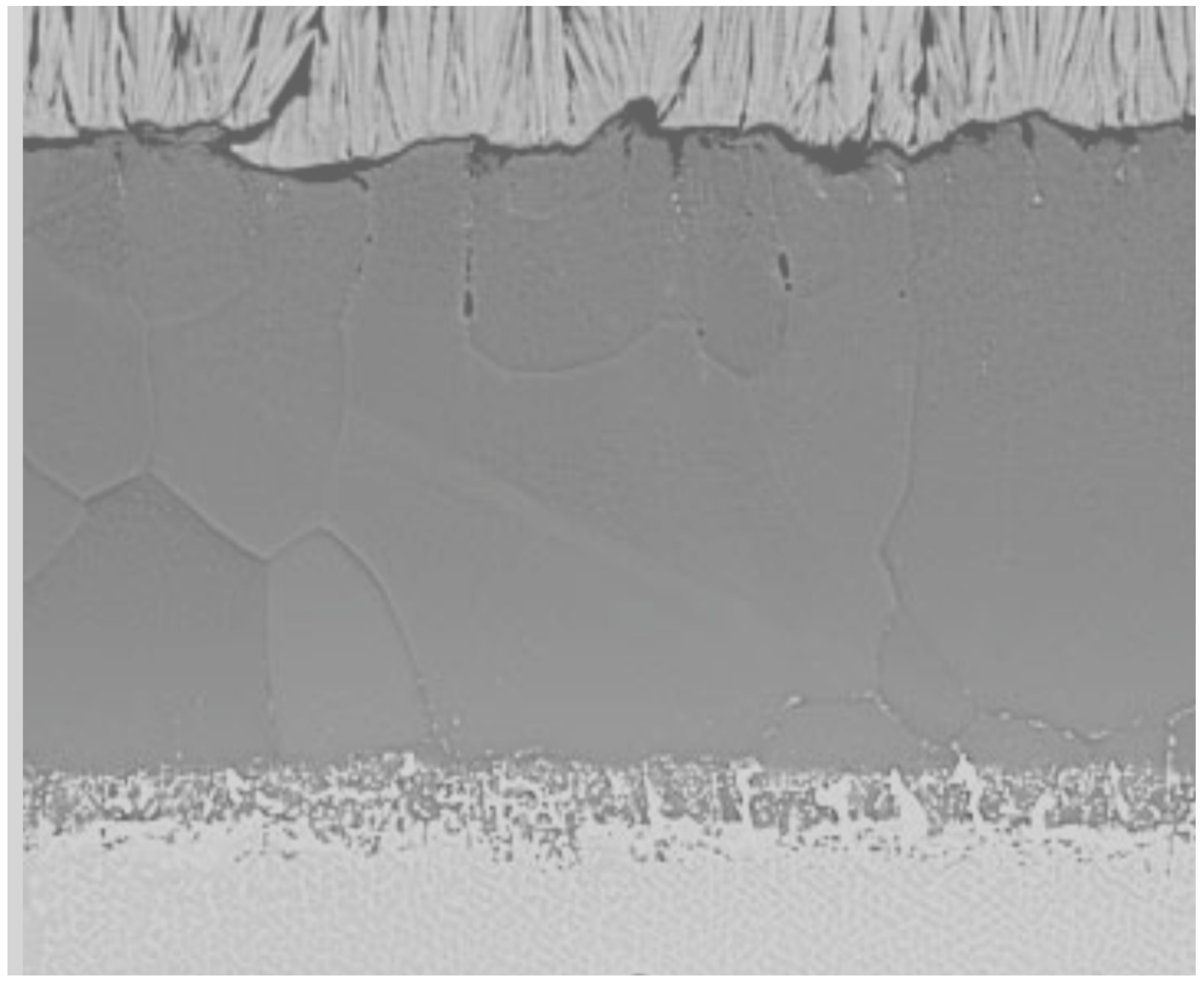


solute concentration



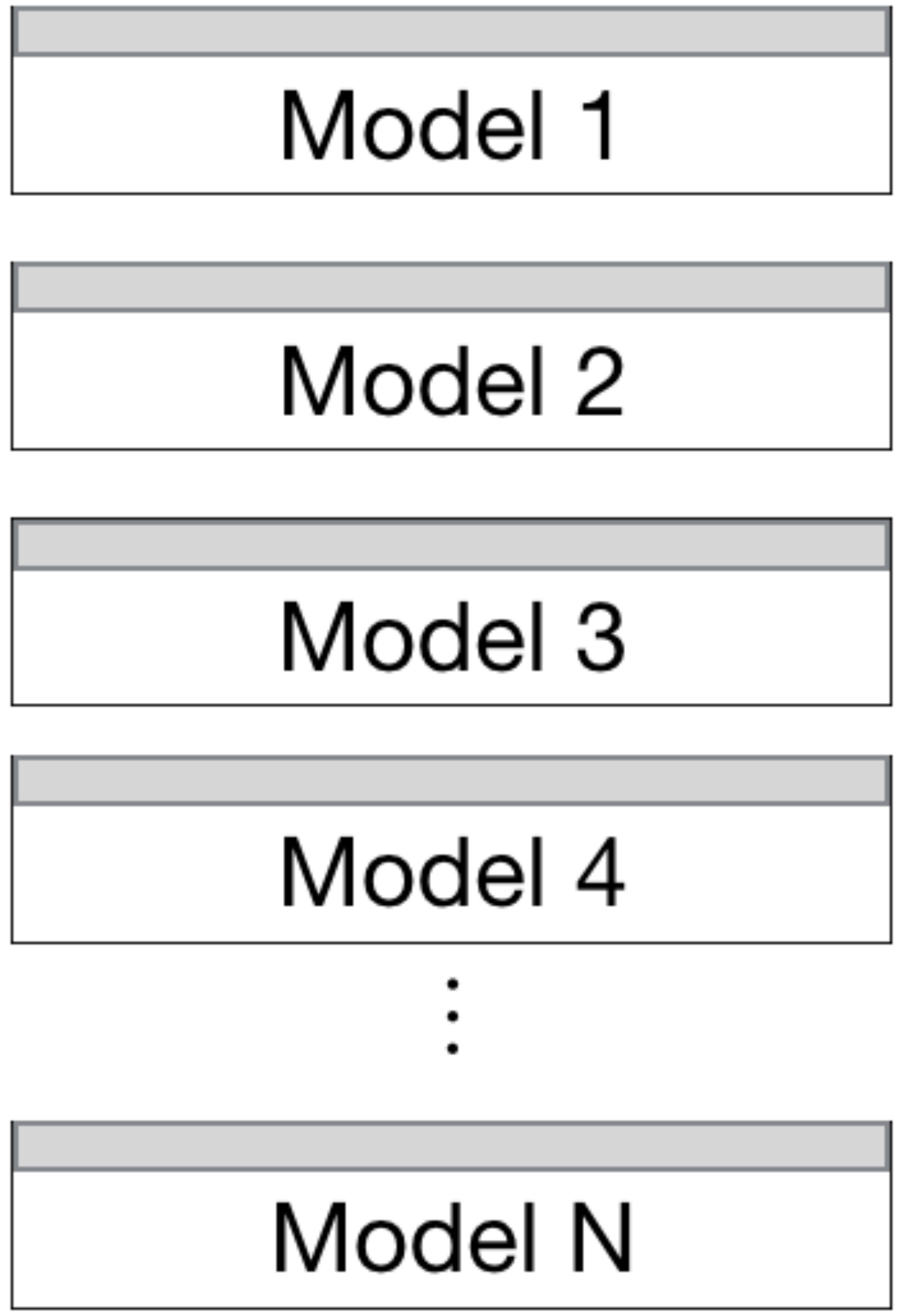
A FE supported project (T. I.-P. Shih, Purdue) is developing physics-based analysis tools to examine heat transfer issues in turbine components to support the development of advanced turbines.

Our goal would be to do multiscale modeling across the turbine blade that includes material properties, geometries, convective heat transfer, and all other tools needed for the analysis and design of a turbine blade



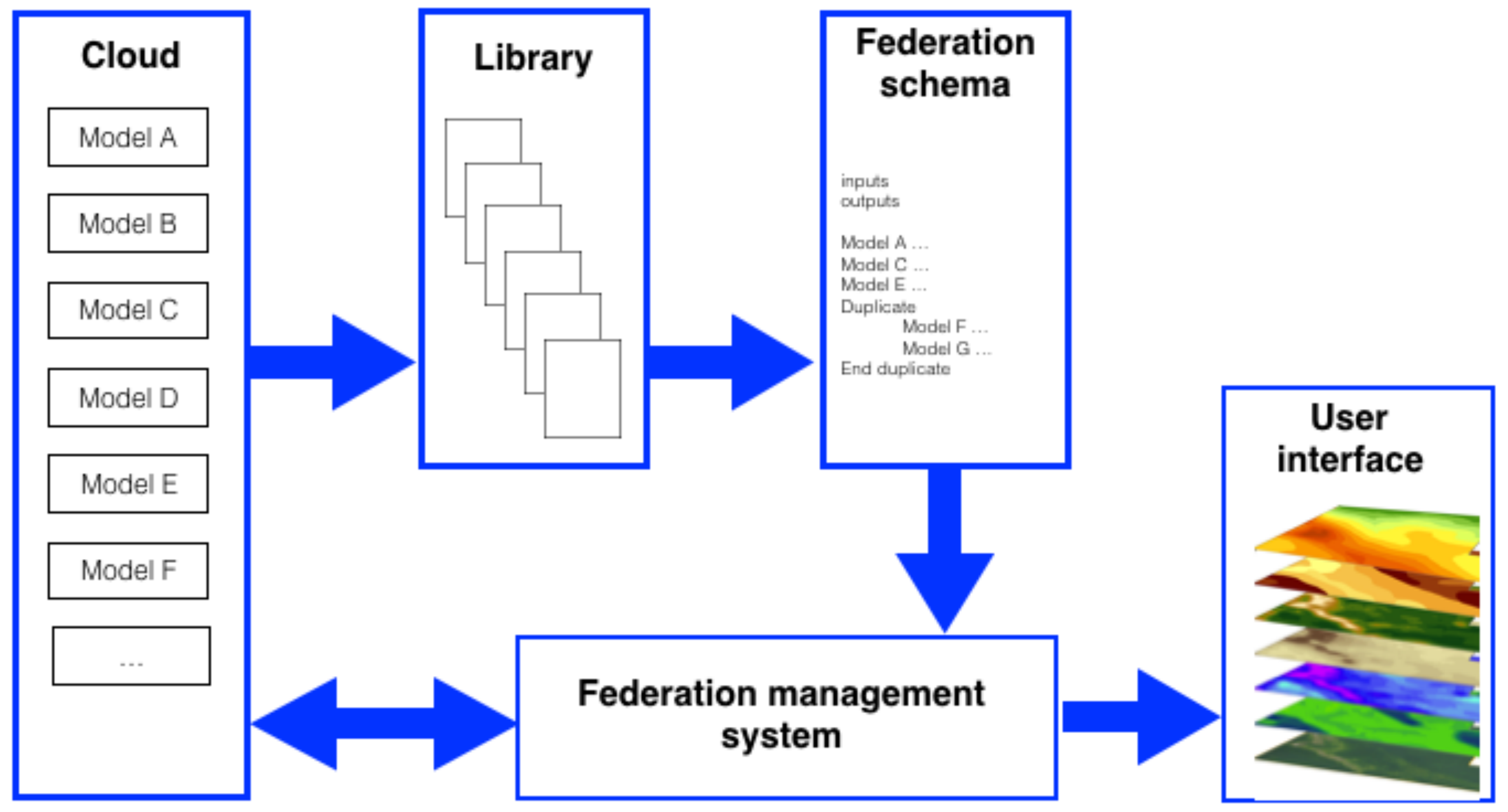
Future directions: design of turbine blades

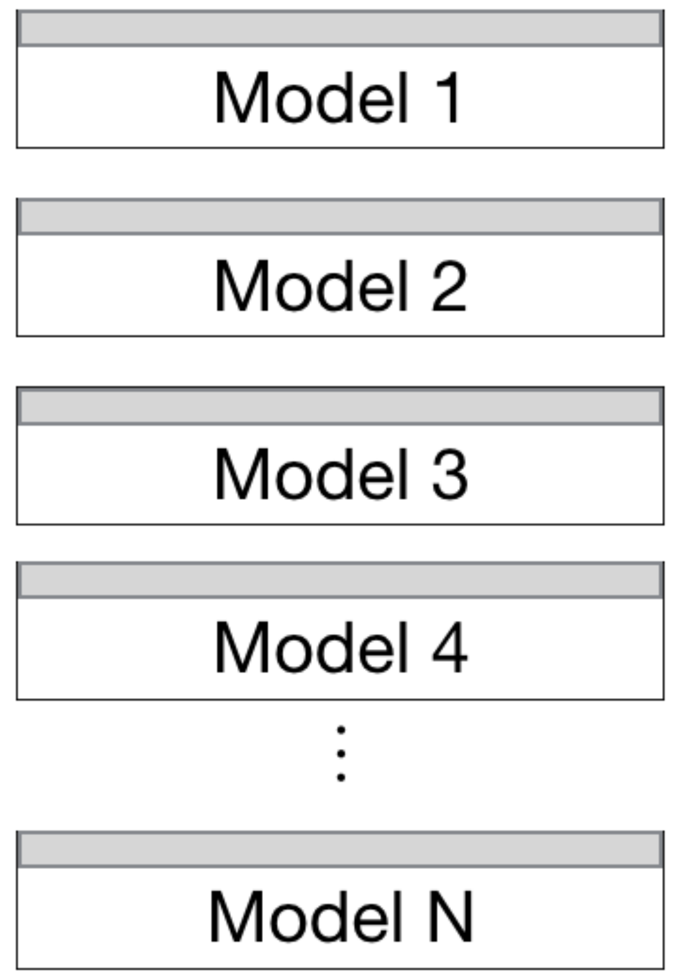
Develop a library of multiscale materials models, created by us and by the FE modeling community, to enable the DOE to create dynamic simulation tools in support of affordable, low carbon, high efficiency, advanced power systems.



Vision

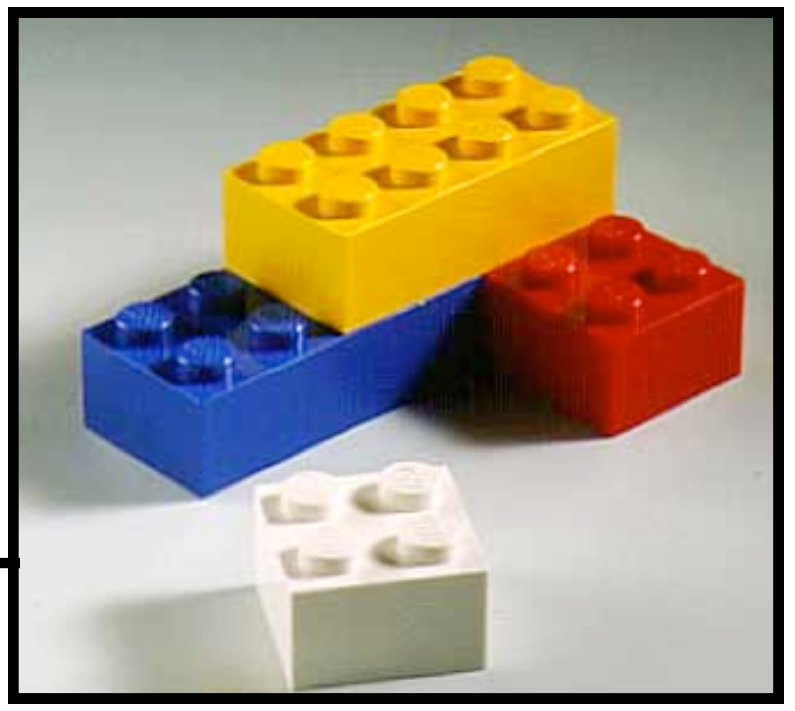
- Goal of the ECS program is to create a cloud-based system that acts as a web service for engineering models
- The MMD program will develop a library of materials models that will able to be linked and run on a single computer system to solve specific problems
- Our programs will converge over time



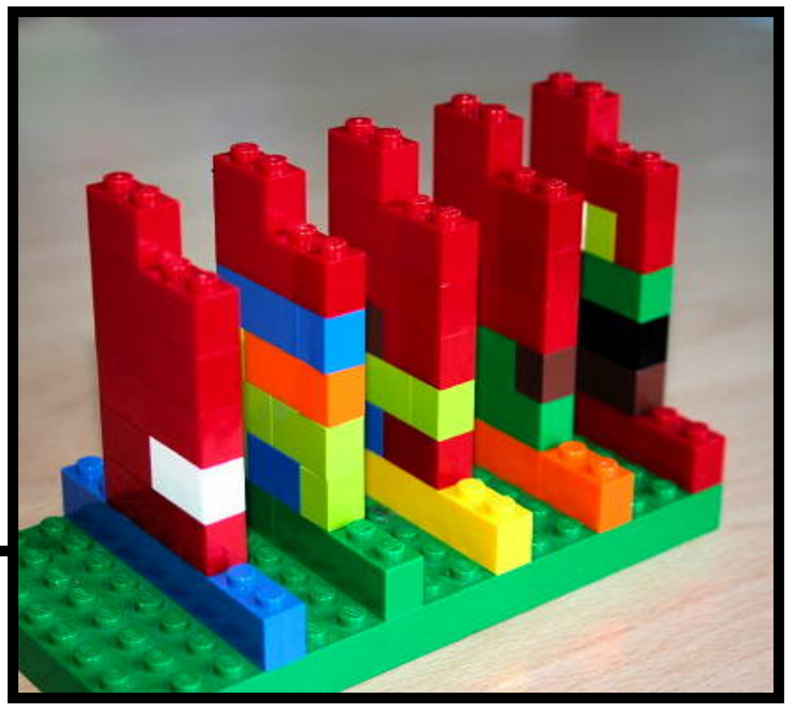


Connection to Engineering of Complex Systems program

Snap



Build



Do

New modeling paradigm

Richard LeSar
515-294-1841
lesar@iastate.edu

Contact info