Overview and Demonstration of the Community Surface Dynamics Modeling System

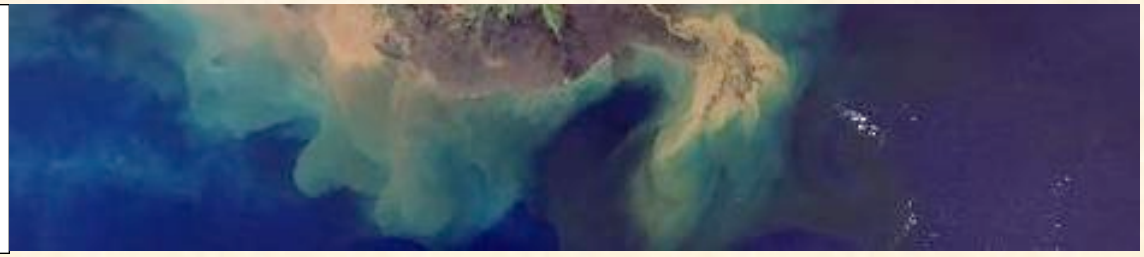
Scott Peckham

CSDMS Terrestrial Working Group Meeting

February 2, 2009. Boulder, CO.

CSDMS

Community Surface Dynamics Modeling System



A Request (or Your Homework)

On our website, please have a look at our:

CSDMS Handbook for Code Contributors : A Guide to Concepts and Protocols

It was written specifically to help code contributors understand the CSDMS approach to model coupling and to provide overviews and examples of key concepts. (But we don't expect anyone to absorb all of it on a first reading.)

It also contains links to a large number of useful articles and papers on component technology.

It is a work in progress and we welcome your feedback.

Two Powerful Tools for Component-Based Modeling

A ***component architecture standard*** (or framework standard) that supports high-performance, scientific computing.

Provides a language-interoperability tool called ***Babel***, RPC support, a component project management tool called ***Bocca***, and a graphical interface for connecting components within the ***Ccaffeine*** framework.

A ***component interface standard*** designed for numerical models of the type where arrays of values march forward in time.

Provides tools for passing data between models that may use ***different computational grids***, ***different dimensionality***, units, etc.

"Underemployed" Models

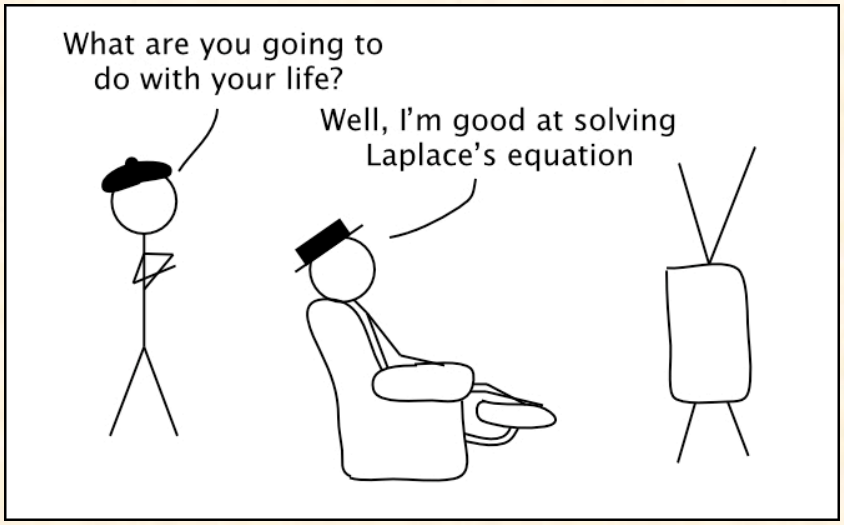
Many surface dynamics models have been developed by our community but they tend to be underutilized. They differ in a number of ways, which makes it difficult for them to be linked together or used interchangeably.

For example, they may:

- be written in ***different languages***
- use ***different grids*** (triangles, rectangles)
- have ***different dimensionality*** (1D, 2D, 3D)
- use ***different formats*** for input/output

The idea behind ***component-based*** modeling is to transform or wrap these models so that they can be used in a ***plug-and-play*** manner.

But how do we deal with these differences?



Components are like people, in that they:

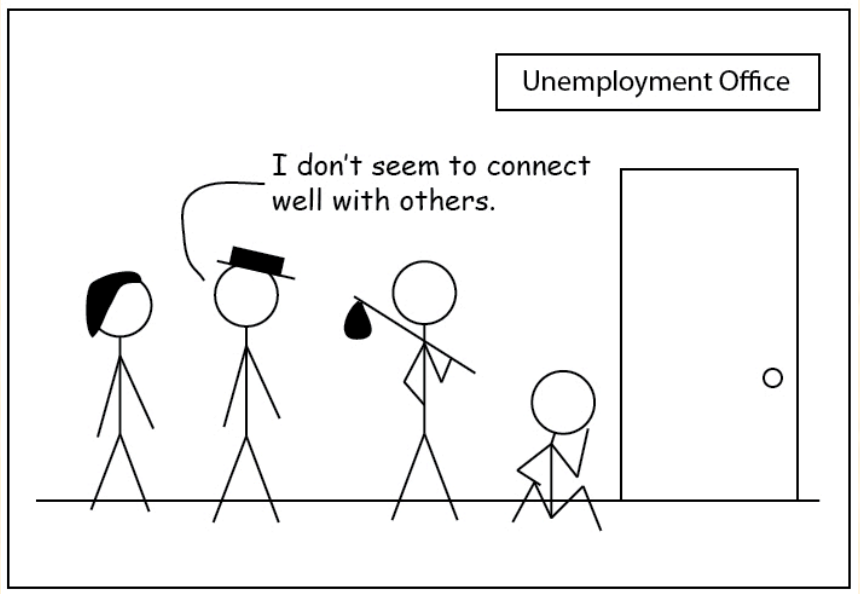
- have particular sets of skills or abilities
- speak some language
- communicate with other components
- work together to solve problems
- make requests of one another
- access and/or require certain services
- need to be "connected" to work together

The Need for Interface Standards

We require people who do "mission critical" jobs such as electricians, lawyers, doctors and airline pilots to be ***certified*** or ***licensed***.

This requires standardized training and a certification that they have can perform certain required skills. Similarly, to be viewed as ***interchangeable*** in a given context, two components must be able to perform the same named set of functions.

Without some kind of standardized interface, models cannot easily be used in other settings and tend to remain "underemployed".



Just as employers require evidence that a person has a particular skillset, software developers require components that meet certain ***interface standards***.

OpenMI is an open source interface standard for models that allows them to request data from one another according to specific rules.

The Basic "IRF" Interface

In the context of componentized software, **an *interface* is a named set of member functions** that provide a caller with access to its capabilities. (That is, names and data types of all arguments & return values are completely specified.

A basic "IRF interface" is something that virtually all model coupling efforts have in common (e.g. ESMF, OMS and OpenMI). IRF stands for "Initialize, Run_Step, Finalize". **We want contributors to provide this interface.**

Run_Model()
    Initialize()
    while not(DONE): **Update**()
    Finalize()

Initialize() => Open files, read/compute initial values, allocate memory, etc.
Run_step() or Update() => Update all of the computed values (one step forward)
    Can call other functions, e.g. Update_Q, Update_v.
Finalize() => Close files, print report, free memory.

Recruiting Models to CSDMS

CSDMS seeks to bring open-source models together and to repackage them as components that can be easily reused and connected to other components in order to solve a broad range of surface dynamics problems.

CSDMS requests that code contributors **make a few, relatively small changes to their model's structure** so that we can more easily provide it with a standard interface.

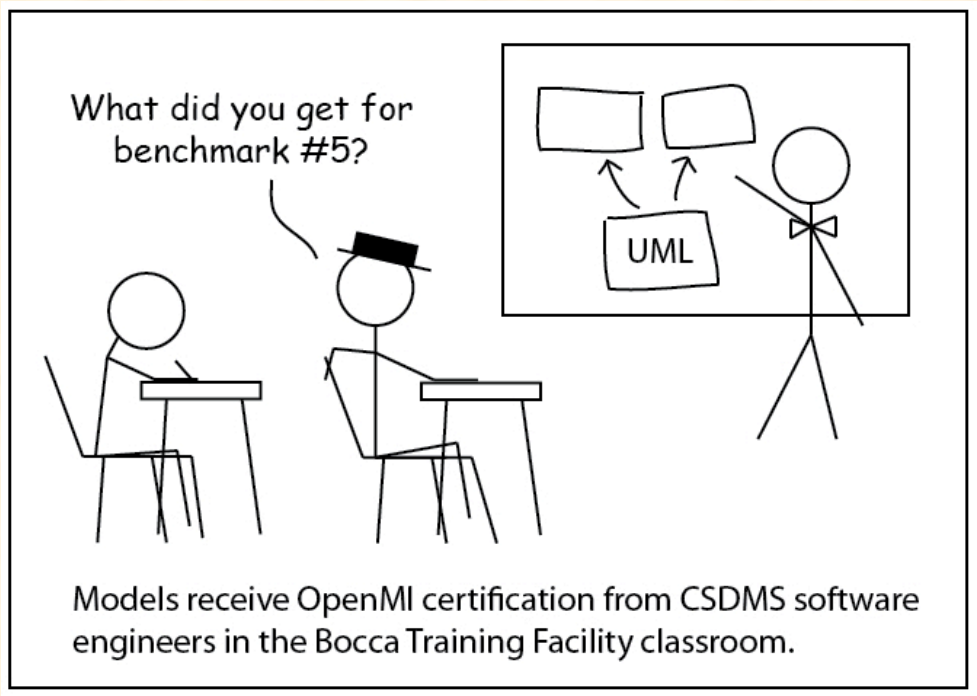
CSDMS also **requires that contributed models can be compiled with a standard, open-source compiler**. We don't have the resources to support all compilers.

Training Models with Bocca

Bocca is a tool in the CCA tool chain that the software engineers can use in order to ***prepare components*** for deployment in a CCA framework.

This preparation has mainly to do with providing components with standard interfaces (e.g. OpenMI) so they are interchangeable and can work together.

Bocca relies on another CCA tool called Babel to ***create language bindings*** that are necessary so that components written in different languages can communicate or call one another.

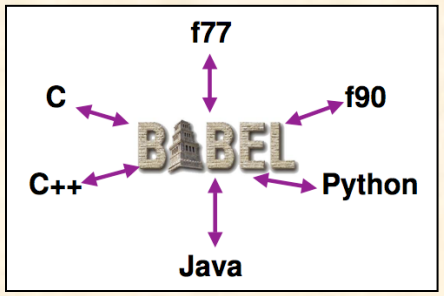


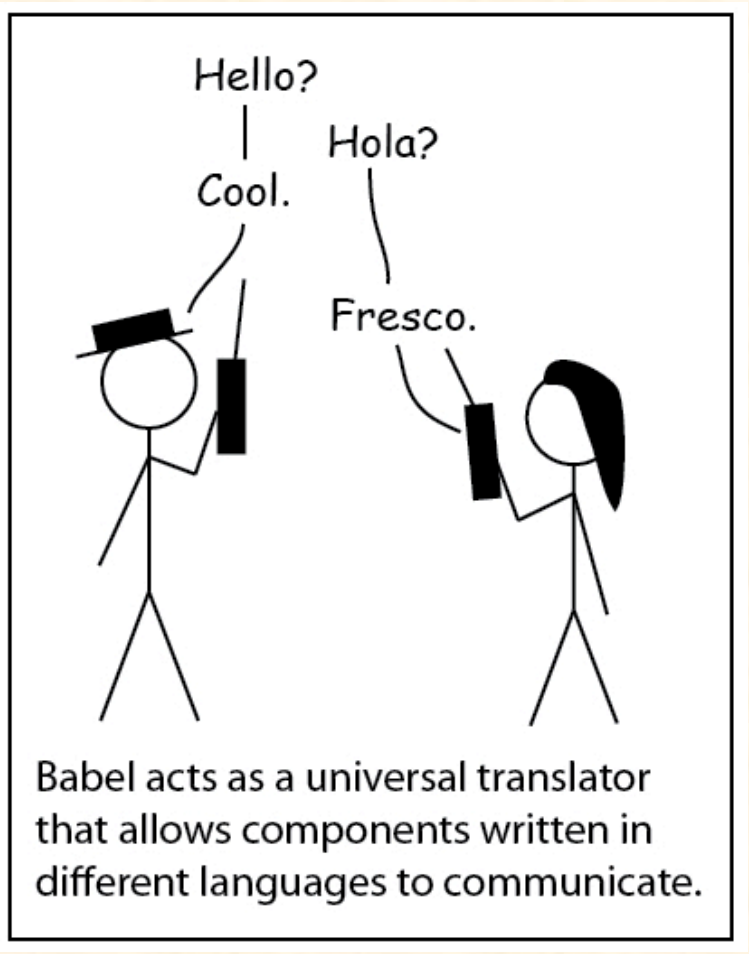
Models receive OpenMI certification from CSDMS software engineers in the Bocca Training Facility classroom.

Helping Components Communicate

Babel is a powerful tool that can provide ***language interoperability*** for components in a CCA framework. This means that there is no need to convert all of our models to a common language. Contributors can keep working in whatever language they want.

Components written in different languages can be rapidly linked in HPC applications with hardly any performance cost. This also allows us to "shop" for open-source solutions (e.g. libraries), lets us mix procedural and object-oriented strategies, and allows us to add graphics & GUIs.





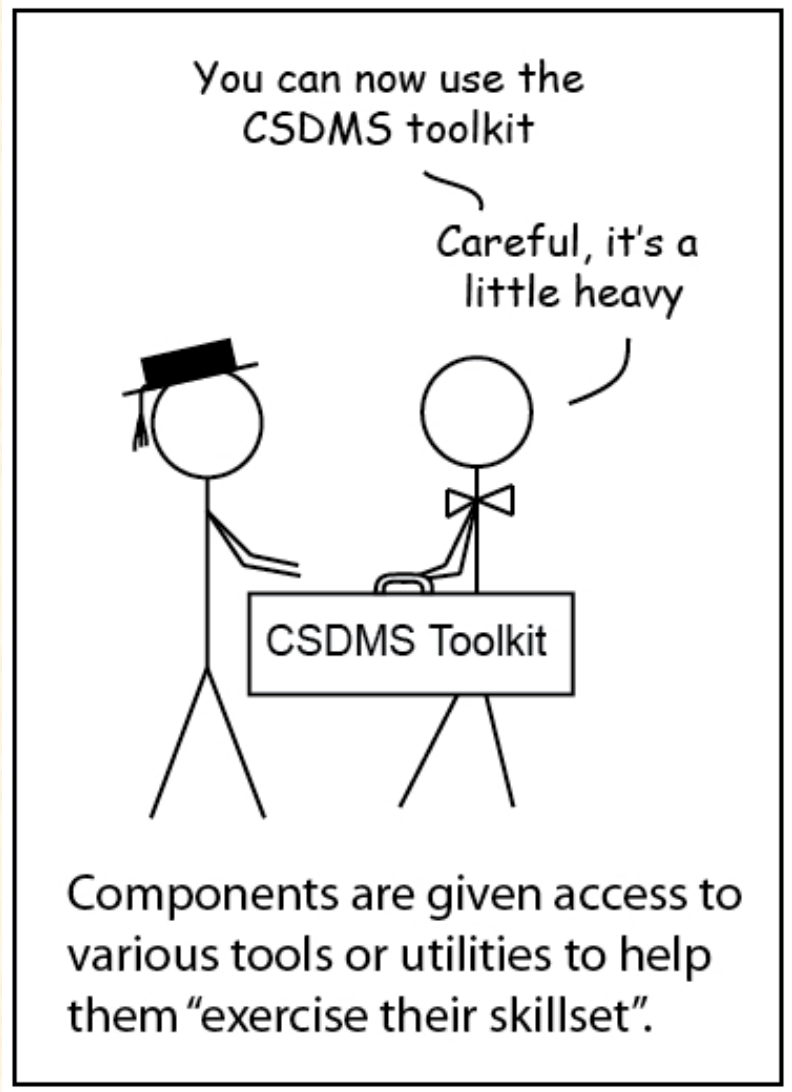
Babel acts as a universal translator that allows components written in different languages to communicate.

Providing Components with Tools

Components may also be provided with "toolkits" or utilities that are needed in order to "***exercise their skillset***."

This is similar to a licensed electrician or doctor needing a tool bag that contains the "tools of their trade."

An interface standard, like OpenMI, may include a ***software development kit*** or ***SDK*** that performs the low-level tasks that are necessary in order to provide a component with an implementation of that interface.

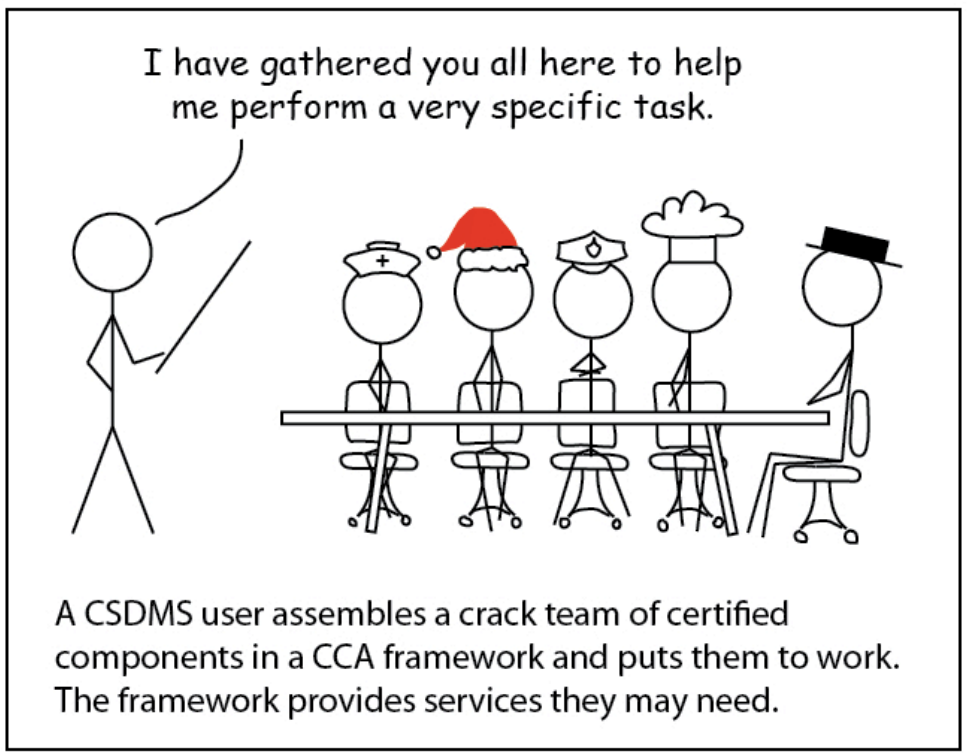


Deploying Models in a CCA Framework

The main difference between subroutines and components is that components are deployed and connected within a ***framework***.

If components are people, then a framework is a venue where people work together on a common goal, such as a boardroom, concert hall, sports arena, battlefield or construction site.

The framework provides a place to work but also typically provides a set of ***services*** that are accessible to every component. It may also provide ***language interoperability*** with a tool like Babel.



A CSDMS user assembles a crack team of certified components in a CCA framework and puts them to work. The framework provides services they may need.

Framework services are analogous to services that the venue provides, such as wireless internet, electricity, projectors, telephones, heat, light and catering.

Requirements for Code Contributors

1. Code must be in a Babel-supported language.
2. Code must compile with a CSDMS-supported, open-source compiler (e.g. gcc, gfortran, etc.)
3. Refactor source code to have a basic ***IRF interface***
4. Provide descriptions of all input & output ***exchange items***
5. Include suitable ***testing procedures*** and data
6. Include a user's guide or at least basic documentation
7. Specify what ***open-source license*** applies to your code
8. Use standard or generic file formats whenever possible for I/O
9. Apply a CSDMS automated wrapping tool

CSDMS
Community Surface Dynamics Modeling System

Non-requirements for Contributors

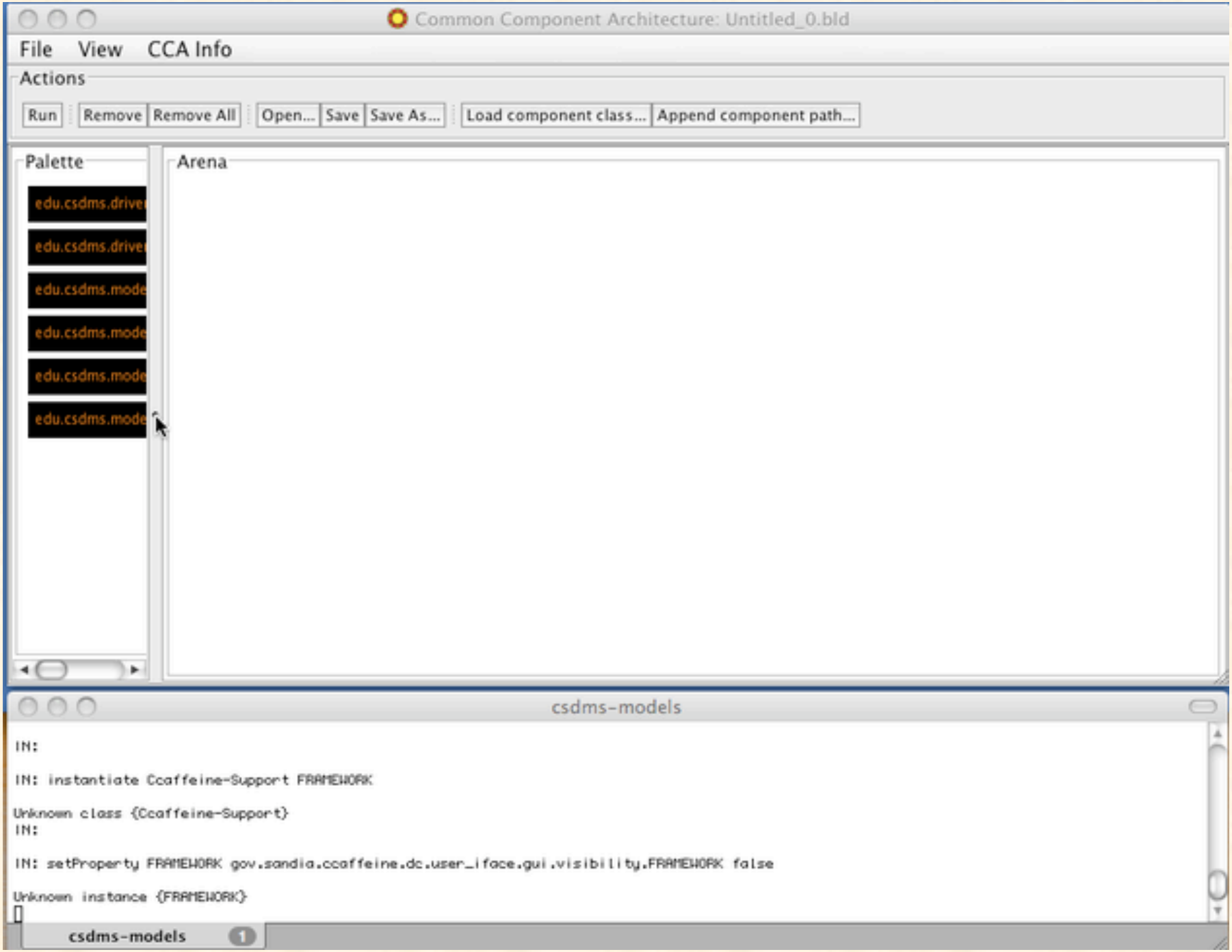
1. You don't need to learn how to use CCA's Babel tool.
2. You probably don't need to learn how to use CCA's Bocca tool, but if you decide to, it is pretty user-friendly.
3. You don't need to learn the details of the OpenMI interface standard, but code contributors should know why a basic "IRF interface" is required for linking models or components.
4. You will need to learn how to use the Ccaffeine GUI or how to write simple Ccaffeine scripts if you want to link CSDMS components to create new models. We are working on a client version of the Ccaffeine GUI that does not require you to install the full set of tools in the CCA tool chain. This GUI will let you build and run models on the new CSDMS supercomputer.

CSDMS
Community Surface Dynamics Modeling System

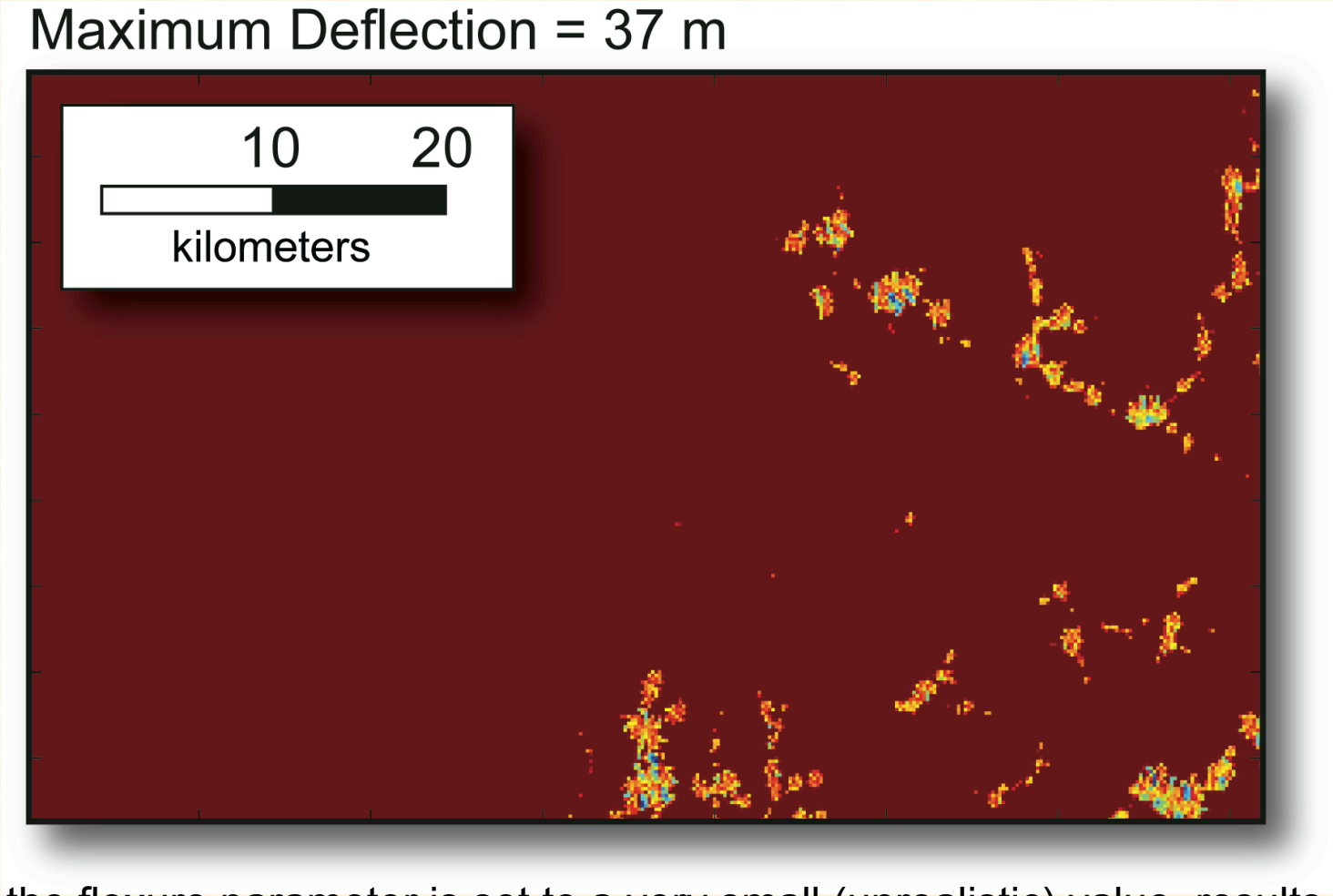
Example 1: Using the GUI for a CCA framework called Ccaffeine to create a new model from GC2D and Flexure components

GC2D is a glacier dynamics model developed by Mark Kessler and Bob Anderson.

An Ice Model called GC2D

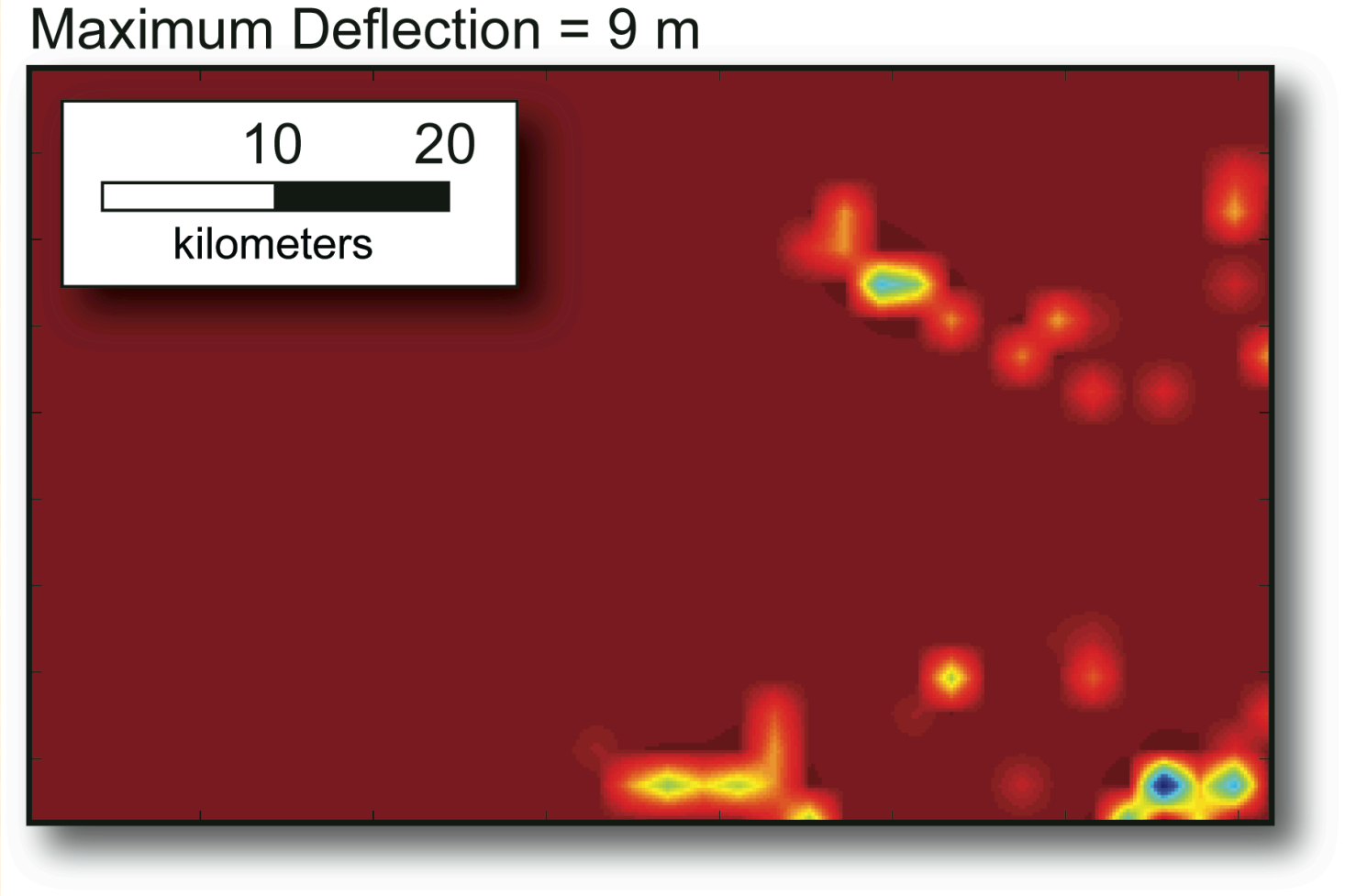

Common Component Architecture: Untitled_0.bld
File
View
CCA Info
Actions
Run
Remove
Remove All
Open...
Save
Save As...
Load component class...
Append component path...
Palette
edu.csdms.drive
edu.csdms.drive
edu.csdms.mode
edu.csdms.mode
edu.csdms.mode
edu.csdms.mode
Arena
csdms-models
IN:
IN: instantiate Ccaffeine-Support FRAMEWORK
Unknown class {Ccaffeine-Support}
IN:
IN: setProperty FRAMEWORK gov.sandia.ccaffeine.dc.user_iface.gui.visibility.FRAMEWORK false
Unknown instance {FRAMEWORK}
csdms-models
1

Results of GC2D + Flexure

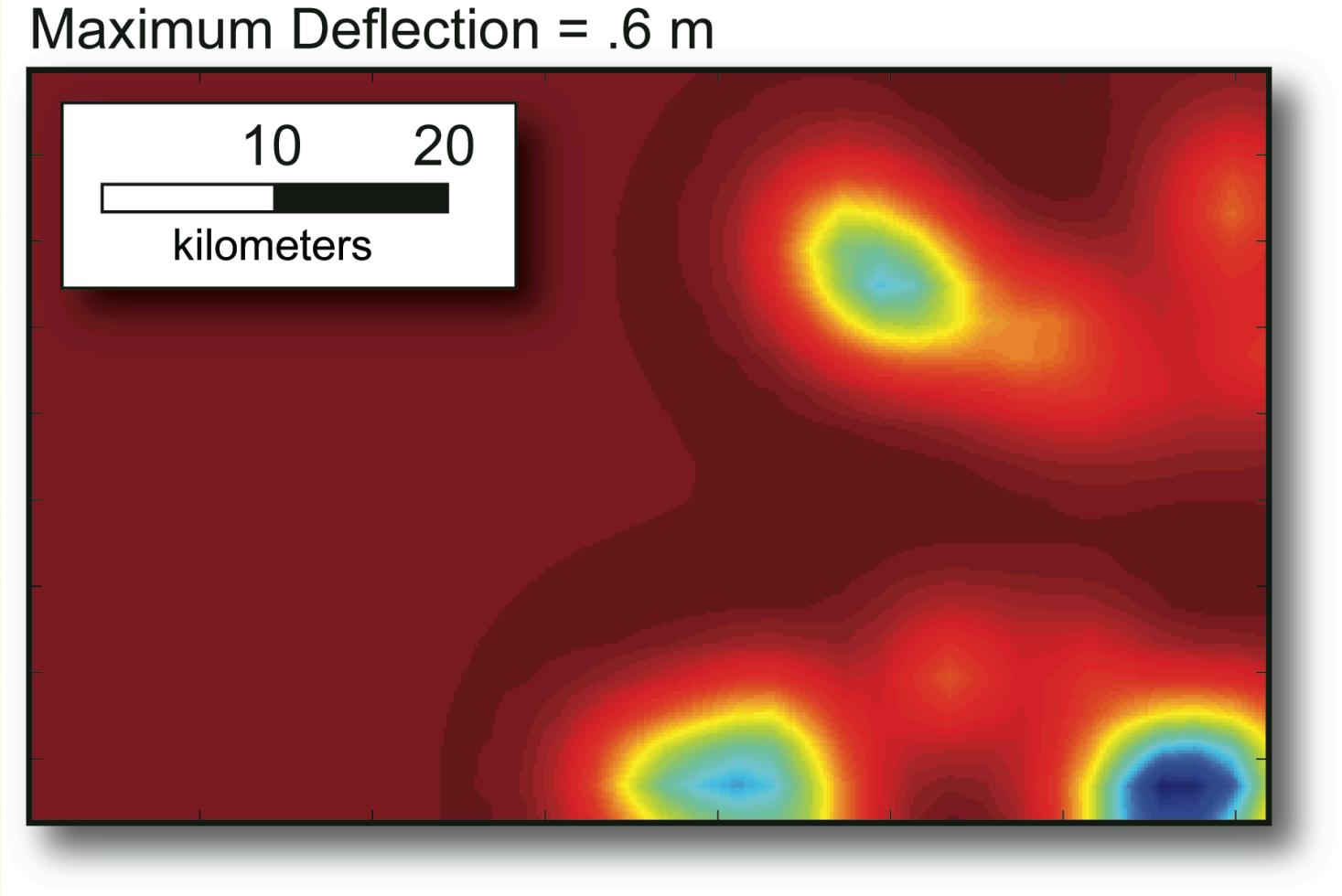


If the flexure parameter is set to a very small (unrealistic) value, results converge to those of Airy subsidence where cells subside independently.

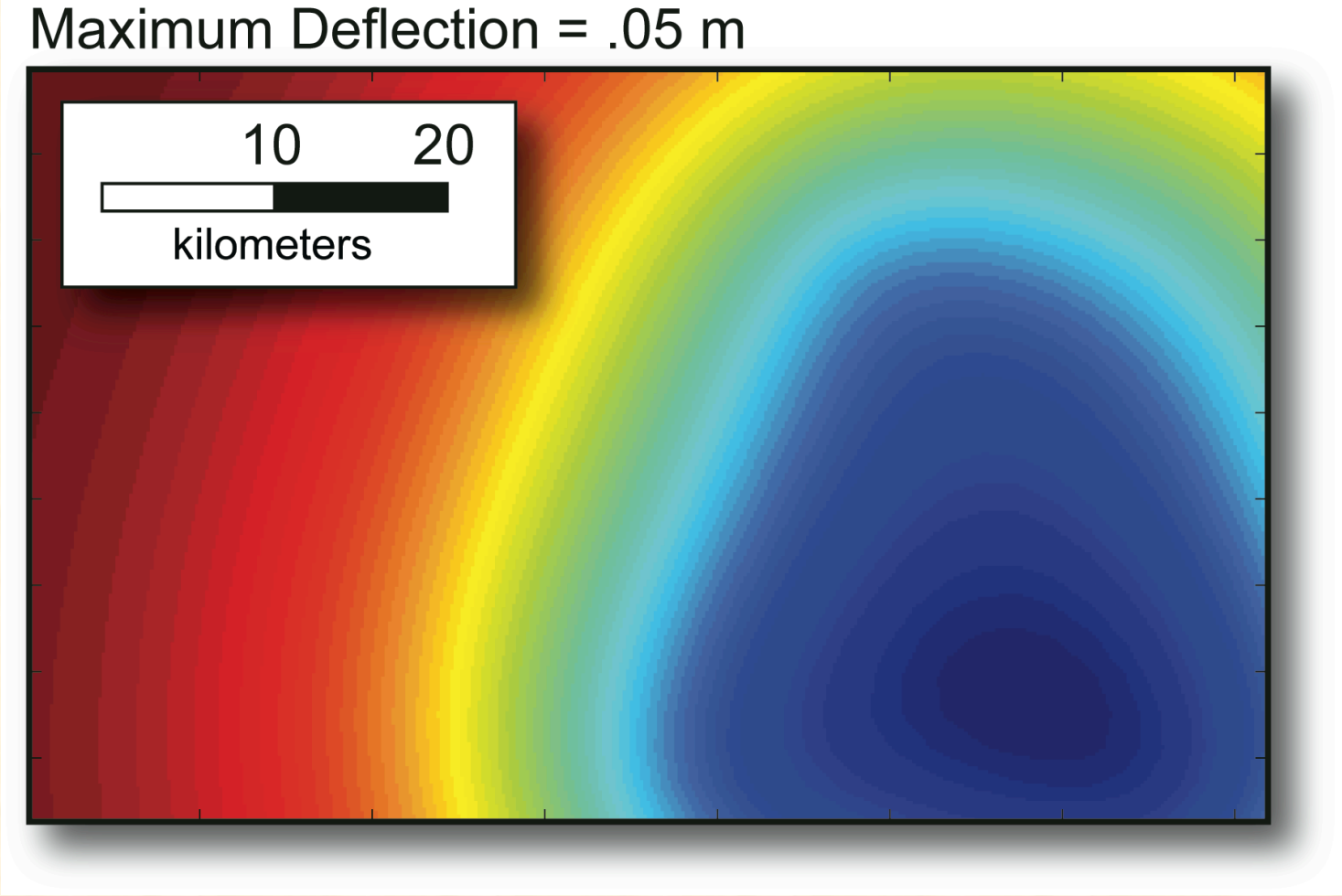
Results of GC2D + Flexure



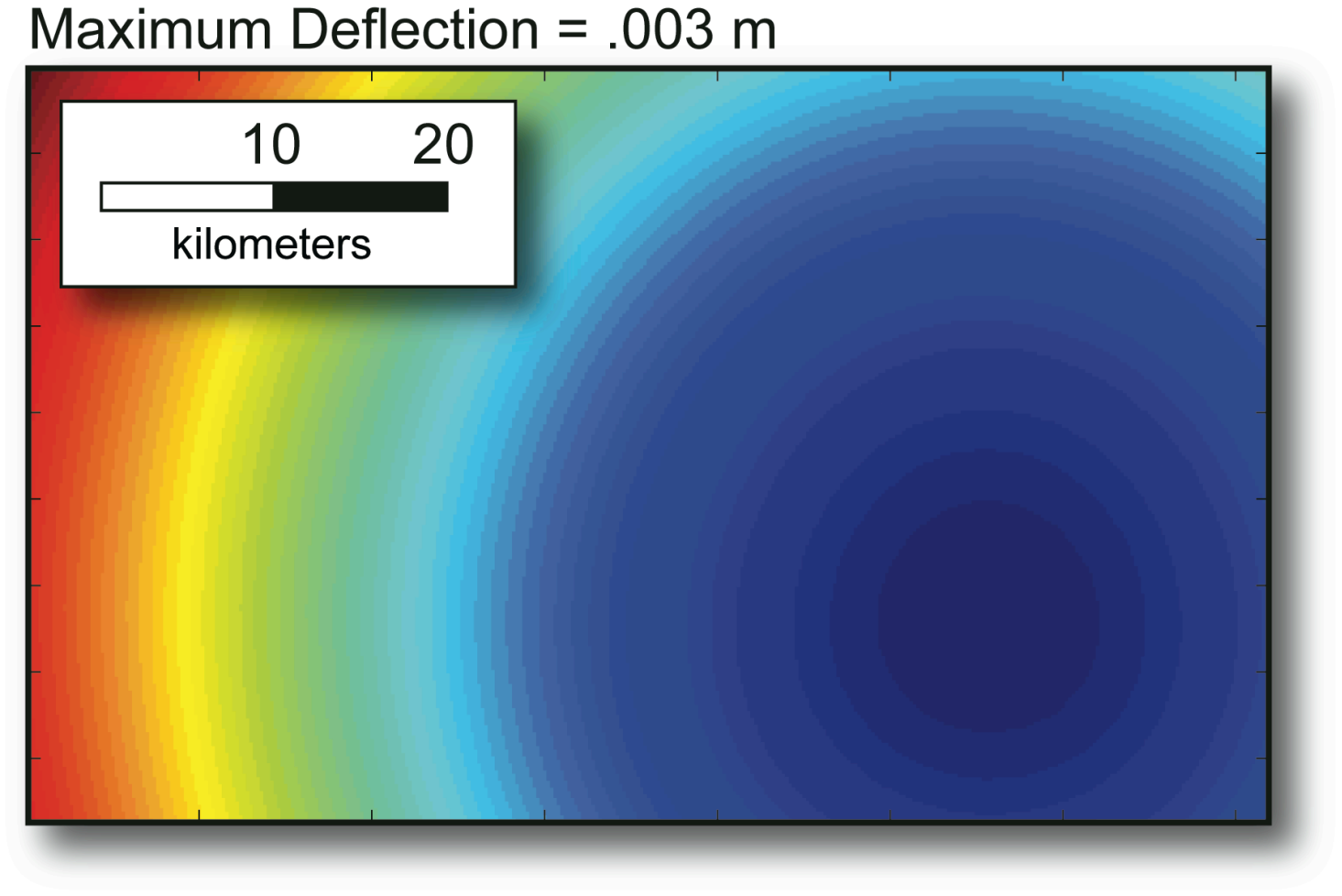
Results of GC2D + Flexure



Results of GC2D + Flexure

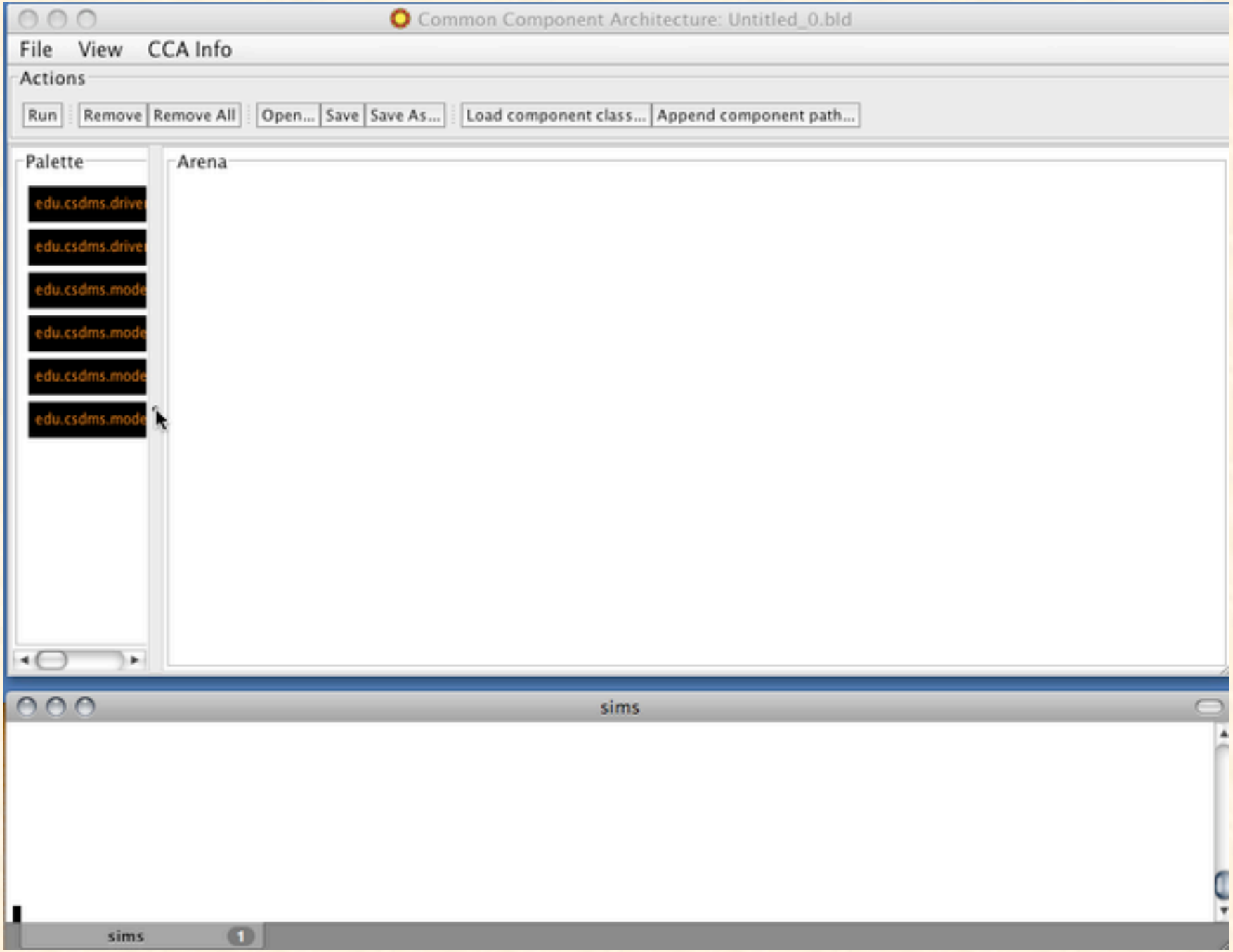


Results of GC2D + Flexure

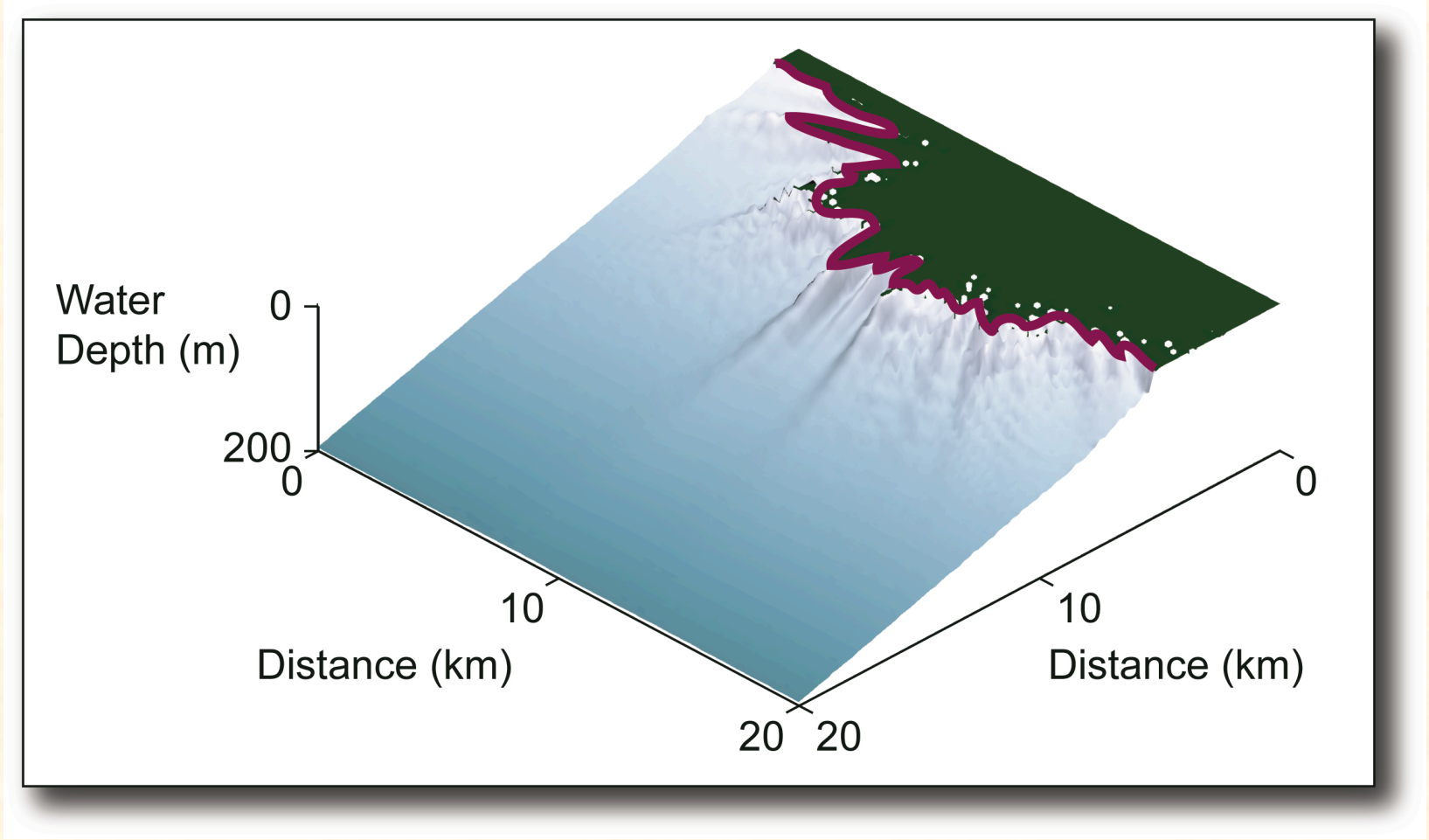


Example 2: Using the GUI for a CCA framework called Ccaffeine to create a new model from Sedflux and Airy subsidence components

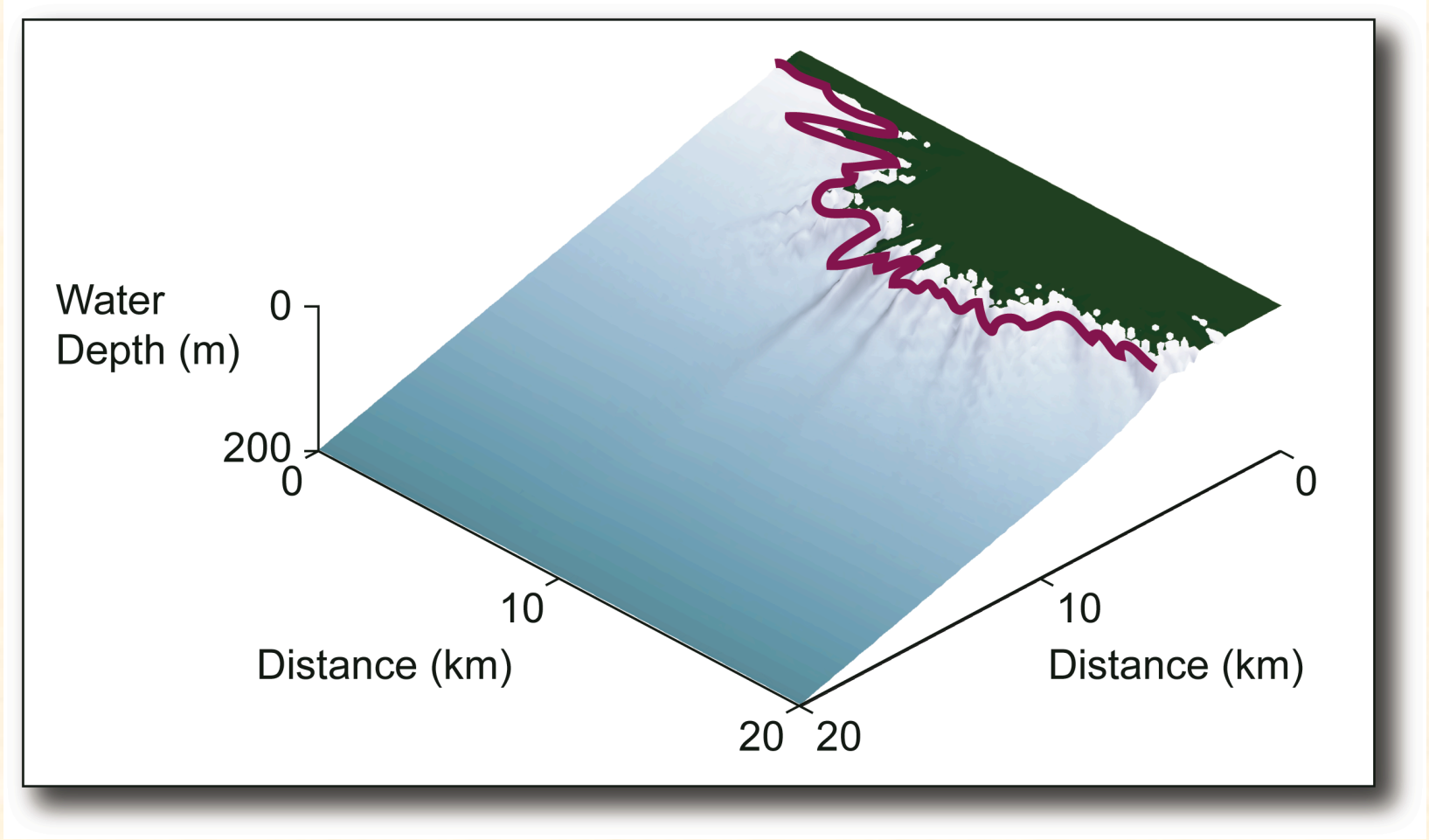
SedFlux is a seafloor stratigraphy model developed by Eric Hutton, James Syvitski and others.


Common Component Architecture: Untitled_0.bld
File
View
CCA Info
Actions
Run
Remove
Remove All
Open...
Save
Save As...
Load component class...
Append component path...
Palette
edu.csdms.drive
edu.csdms.drive
edu.csdms.mode
edu.csdms.mode
edu.csdms.mode
edu.csdms.mode
Arena
sims
sims
1

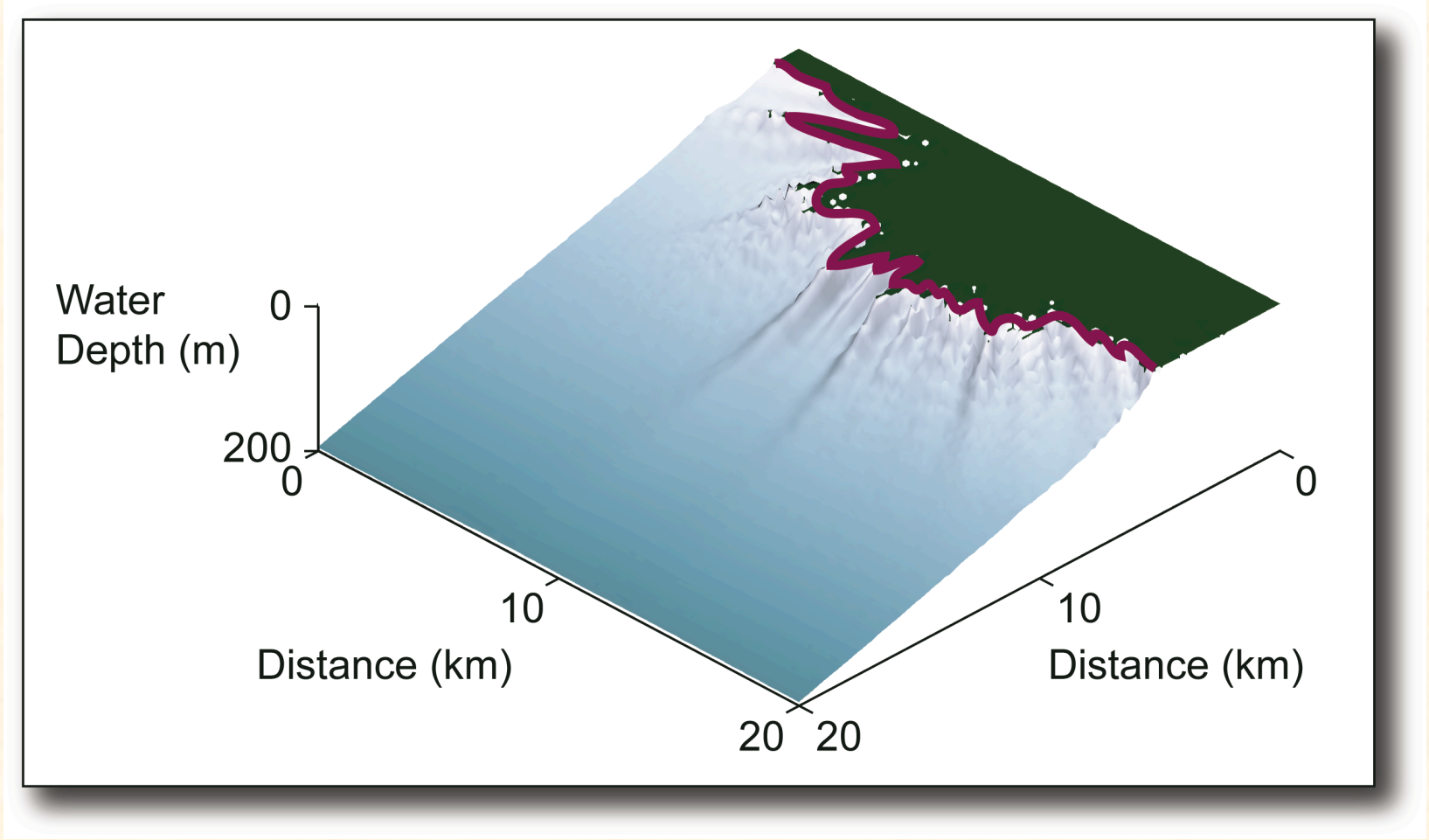
Without Subsidence, the delta progrades approximately 7 km



With Airy subsidence, the delta progrades approximately 5 km



With Flexure, the delta is similar to the case with no subsidence at all



Summary

CSDMS employs a component-based modeling approach.

CSDMS authors contribute models with an “IRF interface”, and in a Babel supported language (C, C++, Fortran, Python, Java)

CSDMS staff assists with converting models to plug-and-play components that have a standardized OpenMI interface.

CSDMS users can then assemble new models from these components in a CCA-compliant framework. The Ccaffeine GUI can be used locally to assemble a model and then the model can be run remotely on the CSDMS supercomputer.

CCA Babel takes care of language issues, while the OpenMI SDK takes care of other differences between models such as grid type and dimensionality.

This CSDMS approach has been demonstrated with prototypes.