

Overview and Demonstration of Linking Models with the Community Surface Dynamics Modeling System

By: Eric Hutton & Scott Peckham

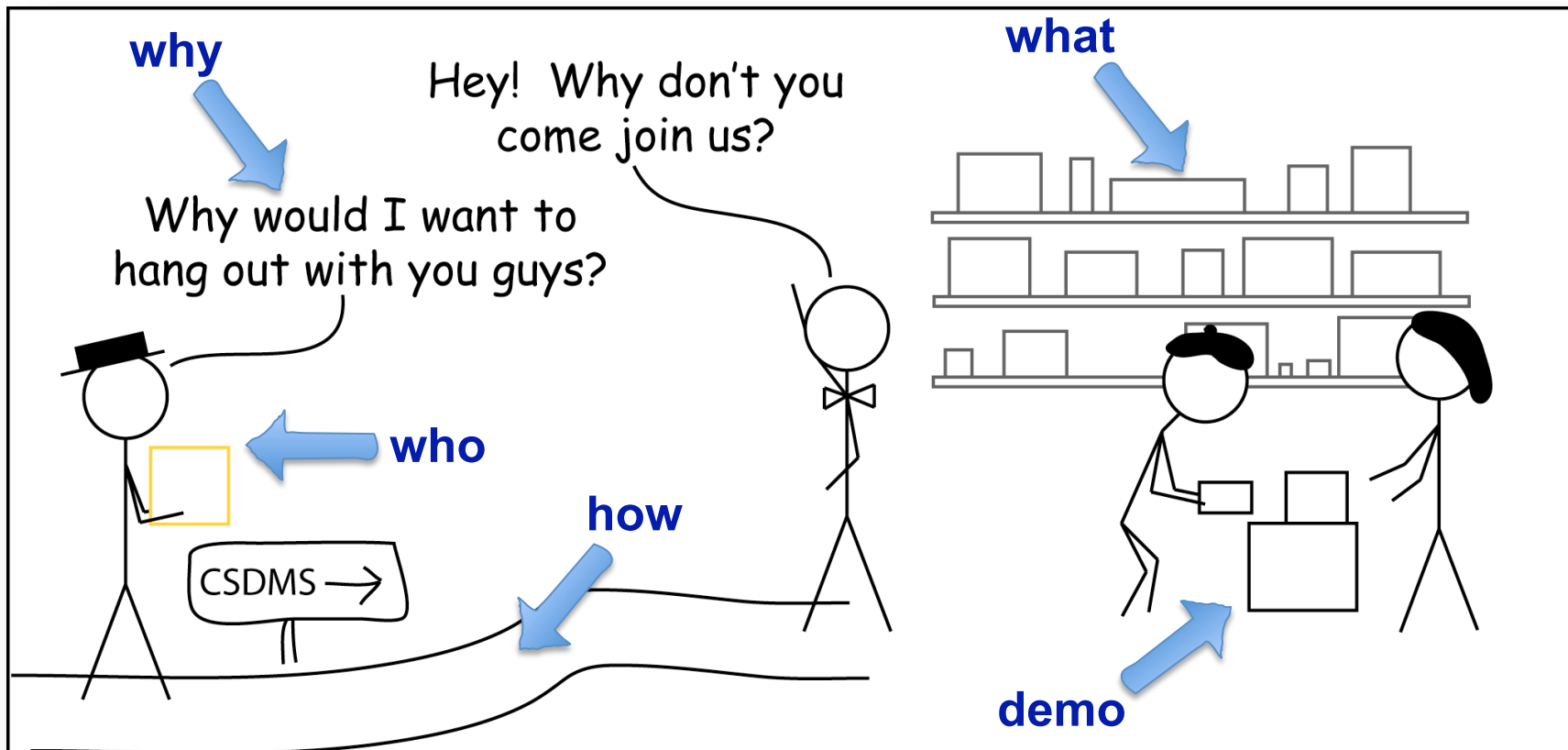
CSDMS - Community Surface Dynamics Modeling System

(pronounced 'sistəms)



Image by Flickr user Dezz

I am going to describe the steps a model takes to become a component, and finish with a demo



CSDMS uses two powerful tools for component-based modeling



CCA provides:

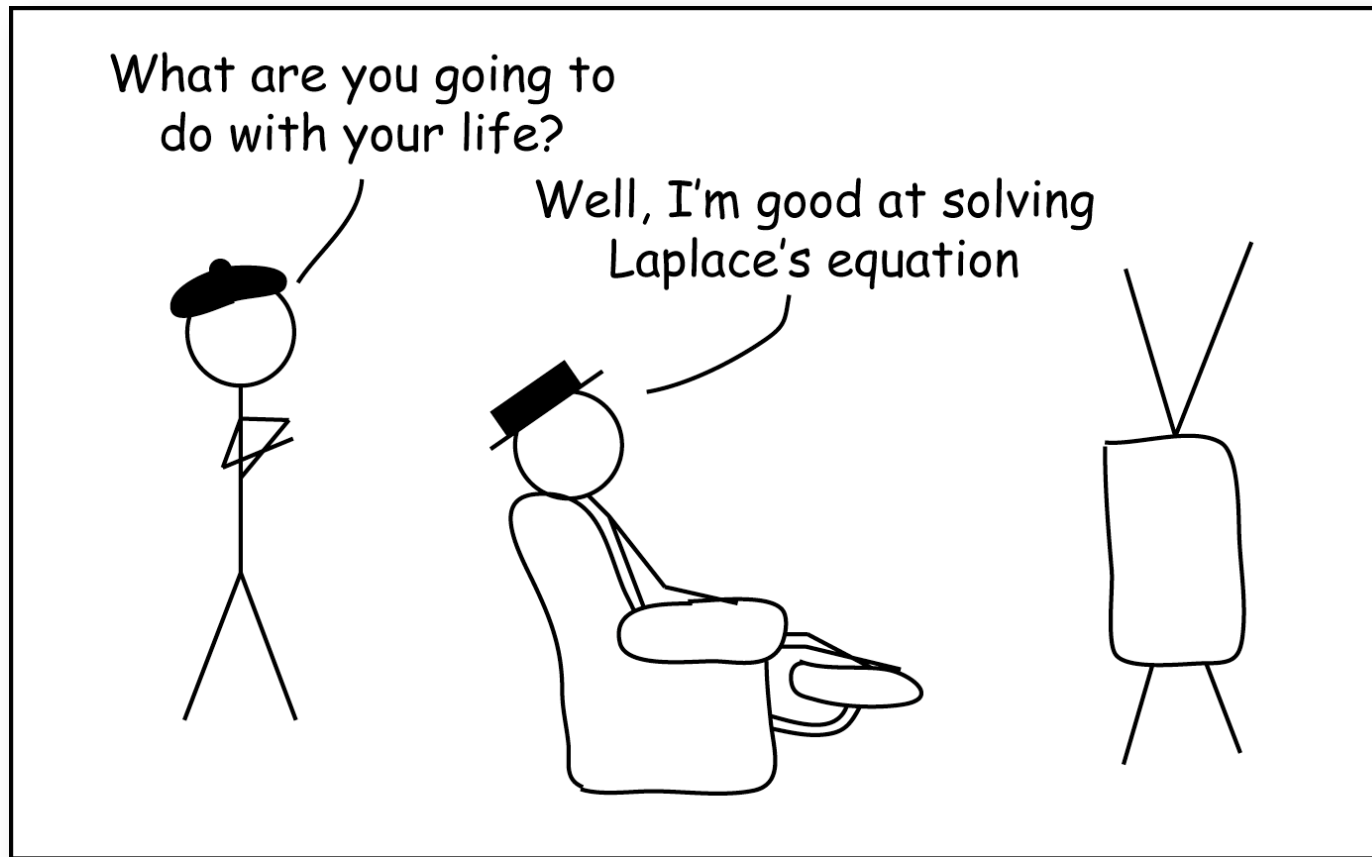
- Language interoperability
- RPC support
- Project management tool
- GUI to connect components



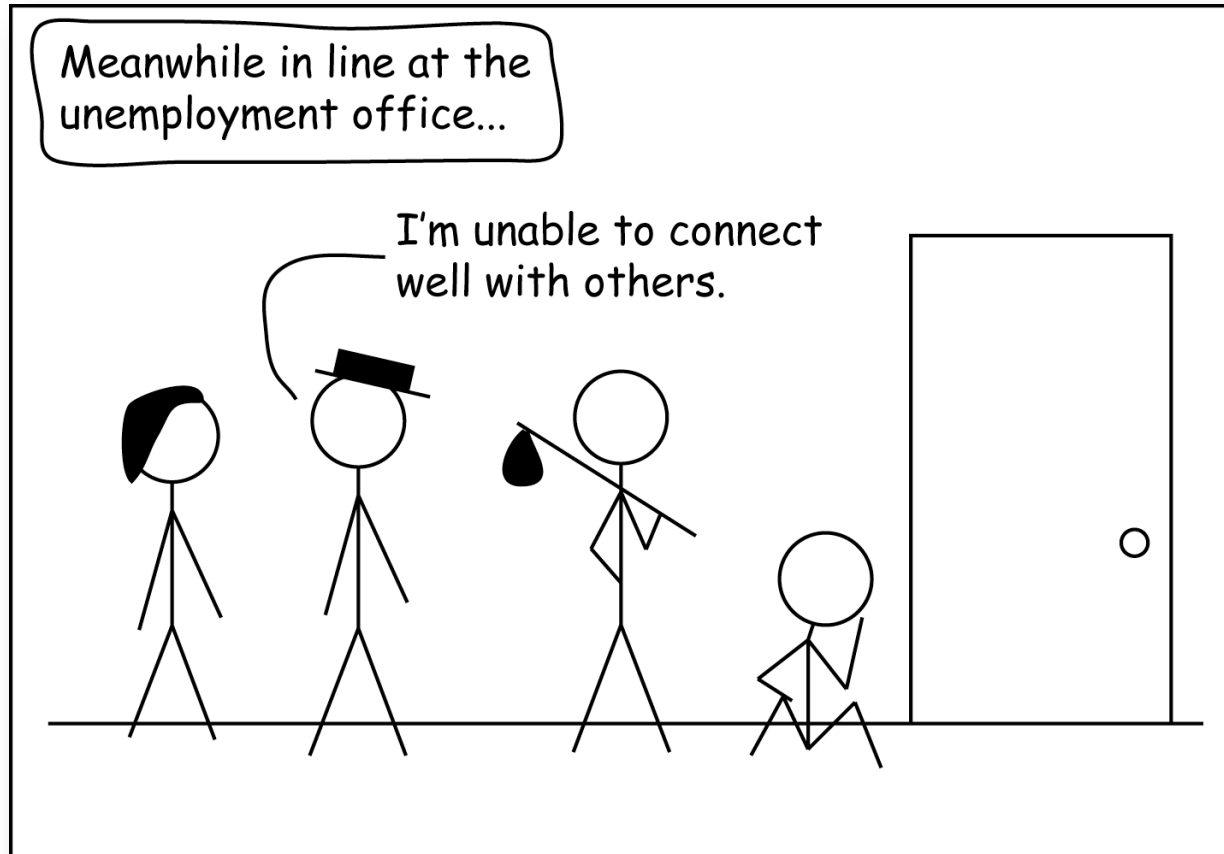
OpenMI provides:

- An interface standard
- Data-passing tools

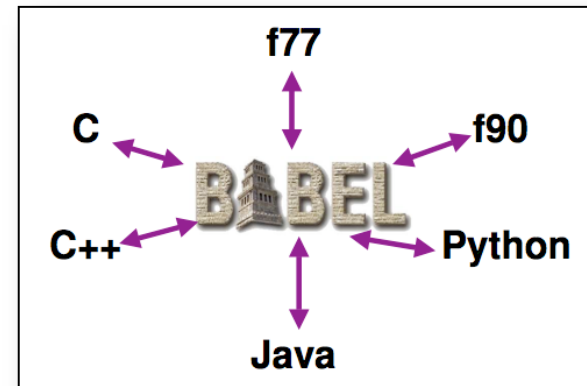
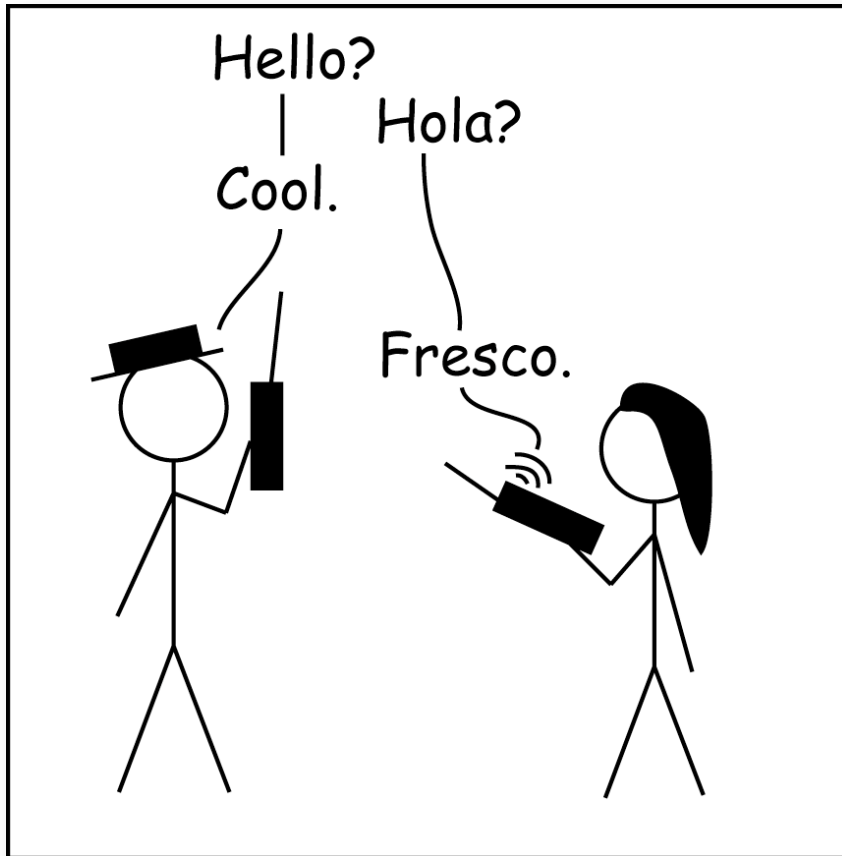
From an underemployed model to CSDMS component



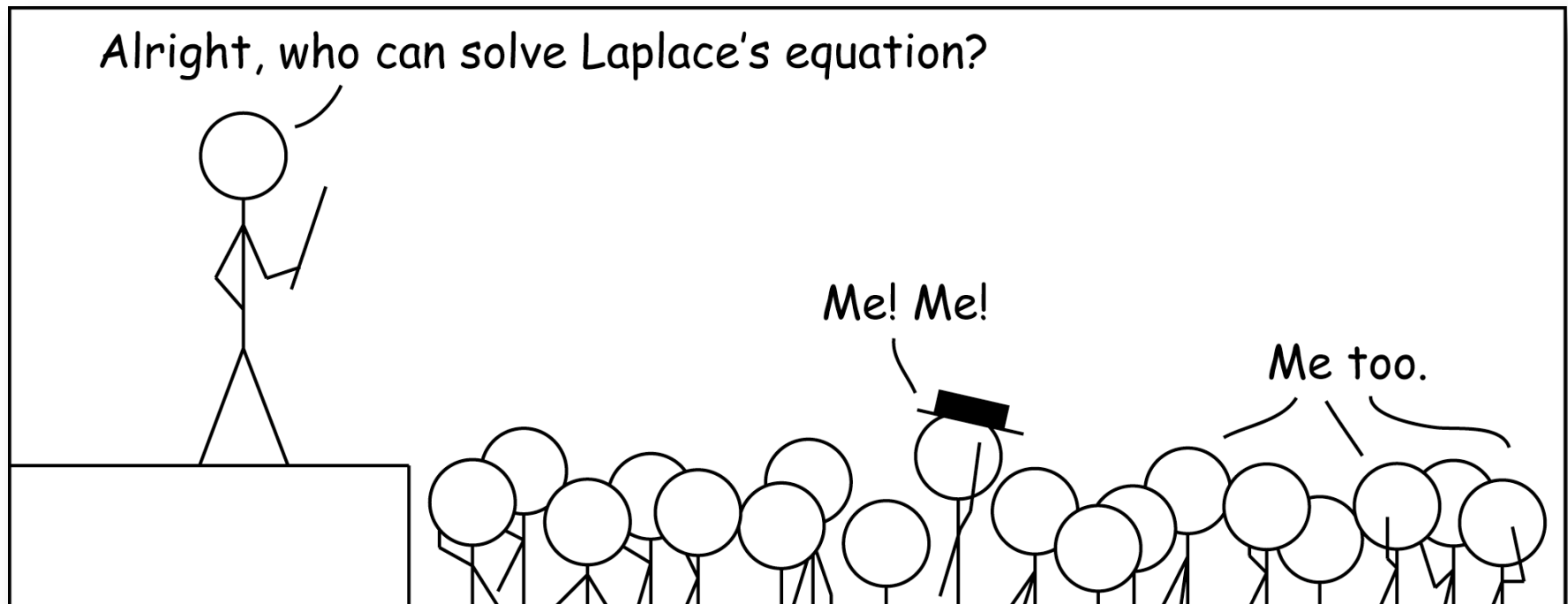
Models are able to better communicate with one another if they have an *interface*



The babel compiler allows models of different languages to communicate with each other



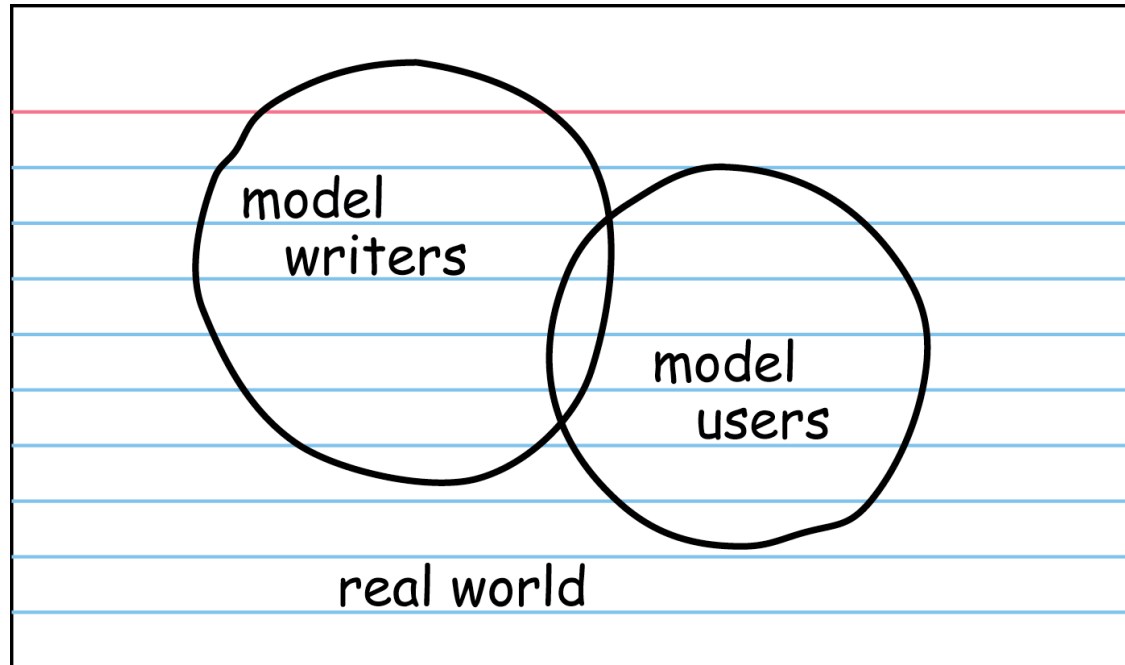
**There are lots of models that do the same thing.
Try them all and choose the best for your task.**



Components can be assembled to perform unique tasks



Our community can be divided into three main groups



How you use CSDMS depends on who you are...

Model developers can use the CCA tools, or not

Model developers

Use CCA tools to build new components

Use model libraries in a framework-less environment

CSDMS provides model users with a place to download, run, and connect models

Model users

Download preexisting models from our repository

**Connect components with GUI to build new models
(remotely or locally)**

CSDMS will gladly accept and host your model no matter what it looks like

If you would like your model to be part of the CSDMS model library:

Become a member of CSDMS

Choose an *open source* license

Package it up and send it to one of us at CSDMS

The CSDMS model library provides a single location to find models

After you submit your model to the CSDMS library, we will:

List it alongside all of the other CSDMS models

Create a wiki page for your model

Make your model available via FTP

Create a subversion repository for your model

There are a number of requirements if you want your model to become a component

In addition to the previous requirements we ask that your code:

- 1. Is written in a Babel-supported language**
- 2. Compiles with a CSDMS-supported compiler**
- 3. Has an IRF interface**
- 4. Has annotations of all input and output items**
- 5. Uses standard file formats for i/o**

We also ask that your model package contains:

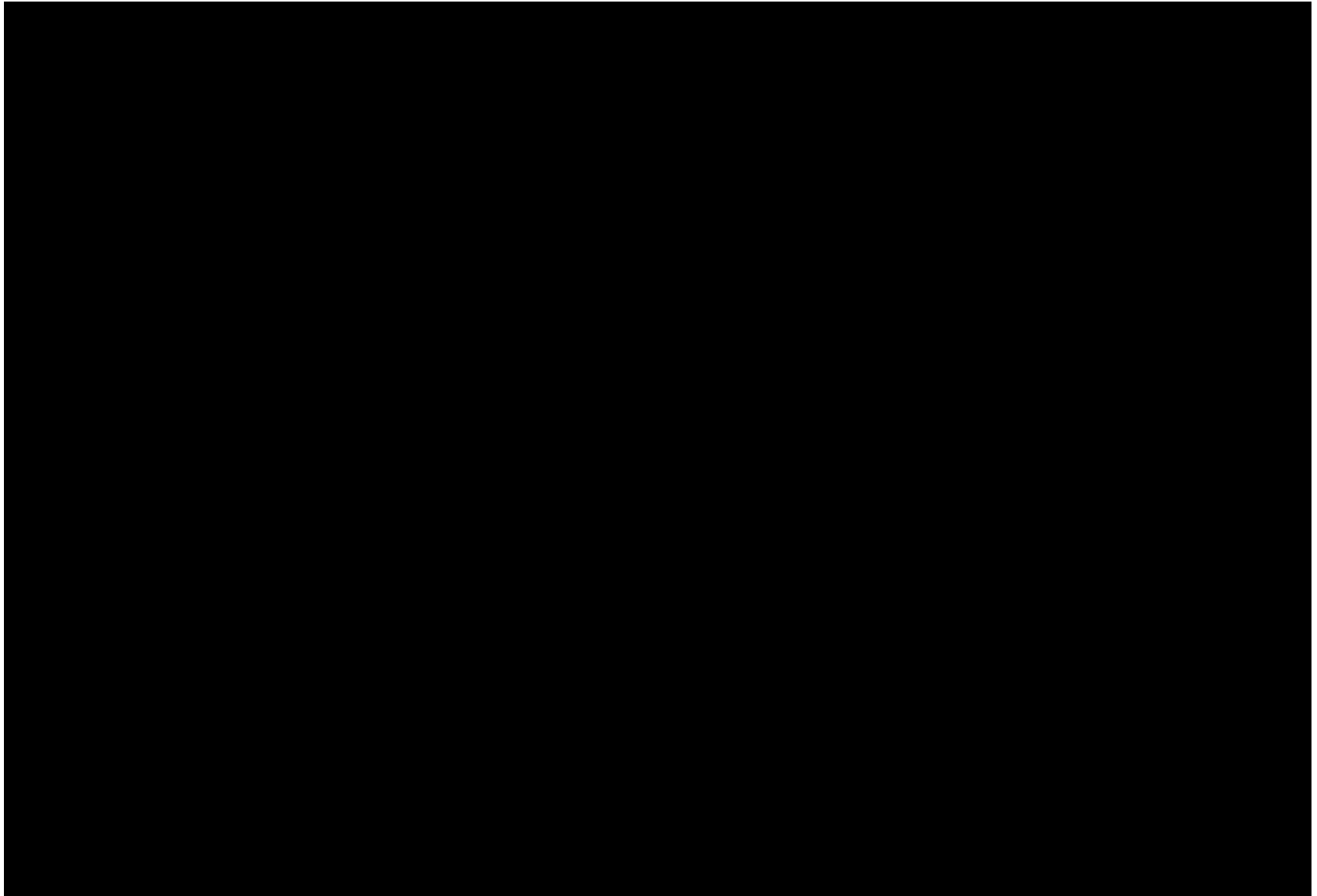
- 1. Testing procedures and data**
- 2. Basic documentation or a user's guide**

You do not need to be an expert in CCA or OpenMI to contribute code

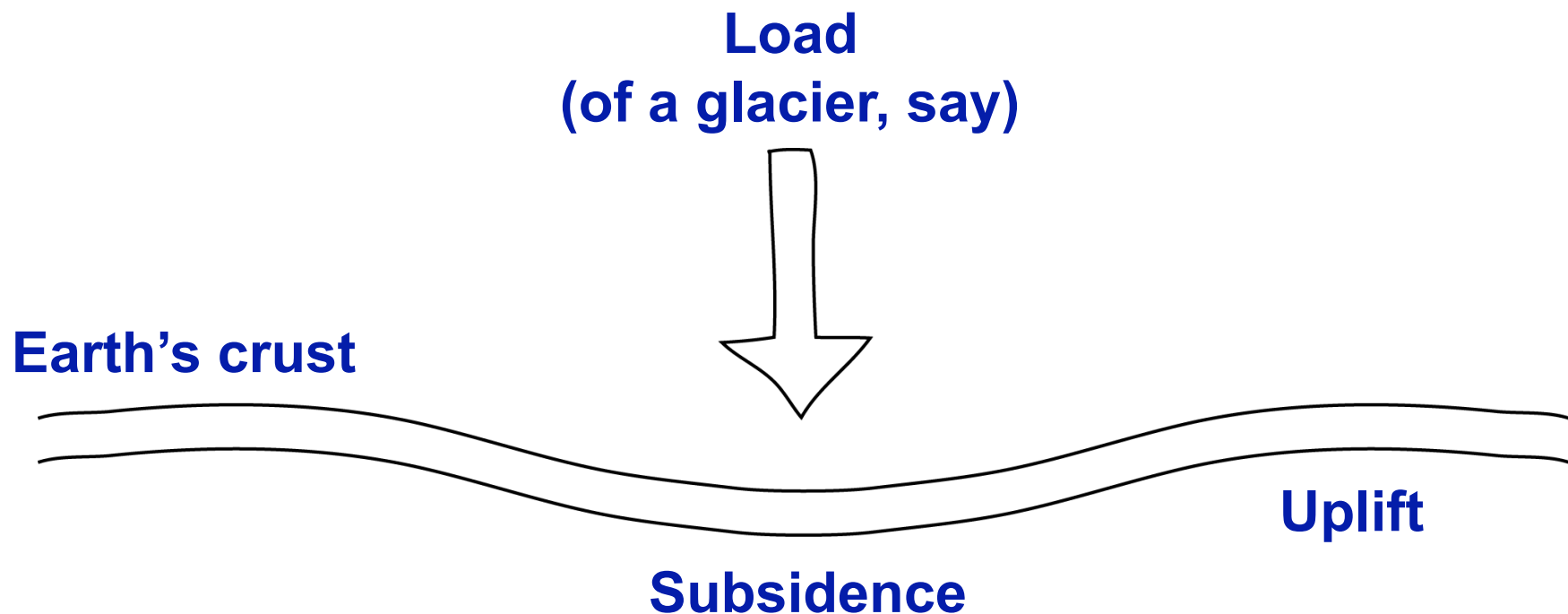
We have some non-requirements for code contributors:

- 1. You don't need to know how to use CCA's babel tool**
- 2. You don't need to know how to use CCA's bocca tool**
- 3. You don't need to know the details of the OpenMI interface**

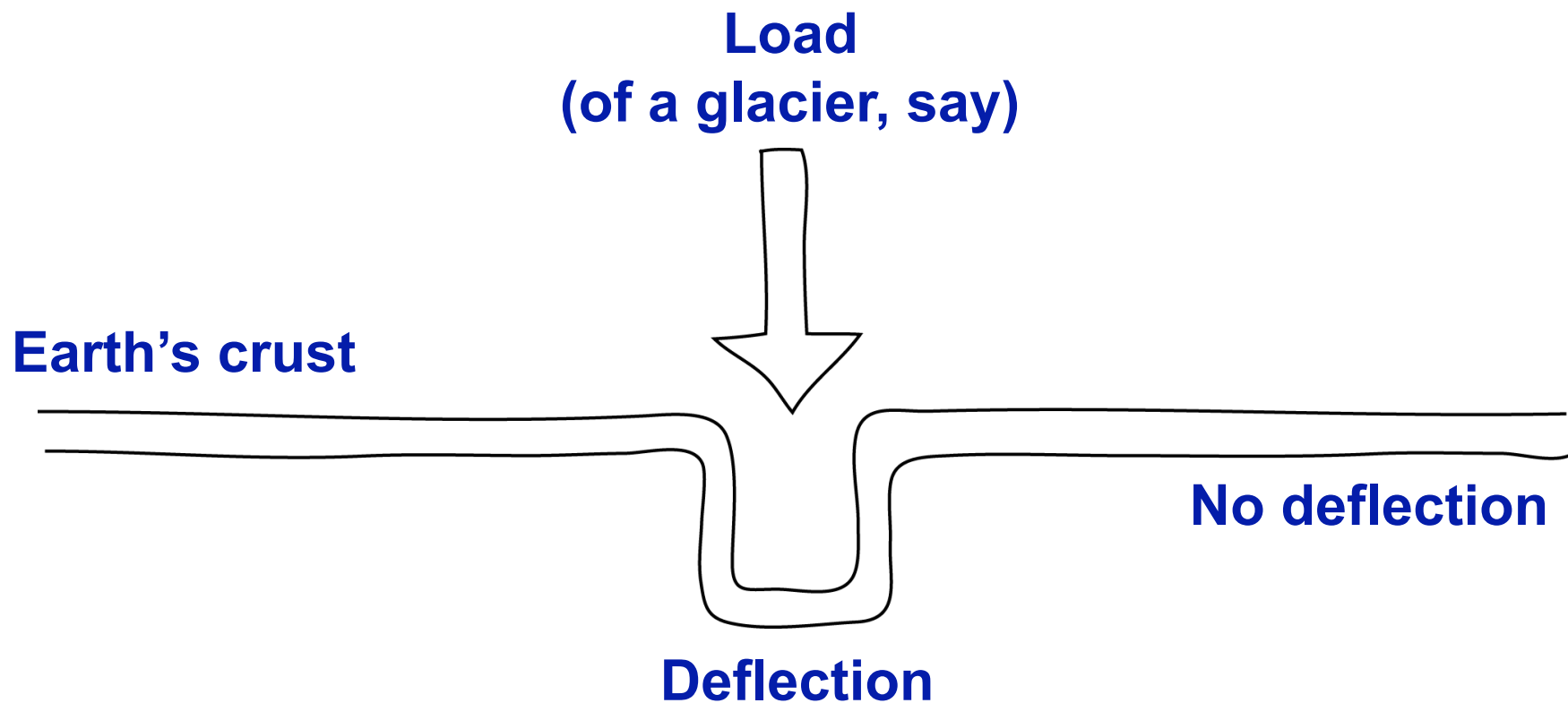
GC2D: grows and shrinks glaciers



FLEX2D: Flexure assumes that Earth's crust is rigid and so deflections are non-local



SUBSIDE: Airy subsidence deflects Earth's crust directly below the applied load

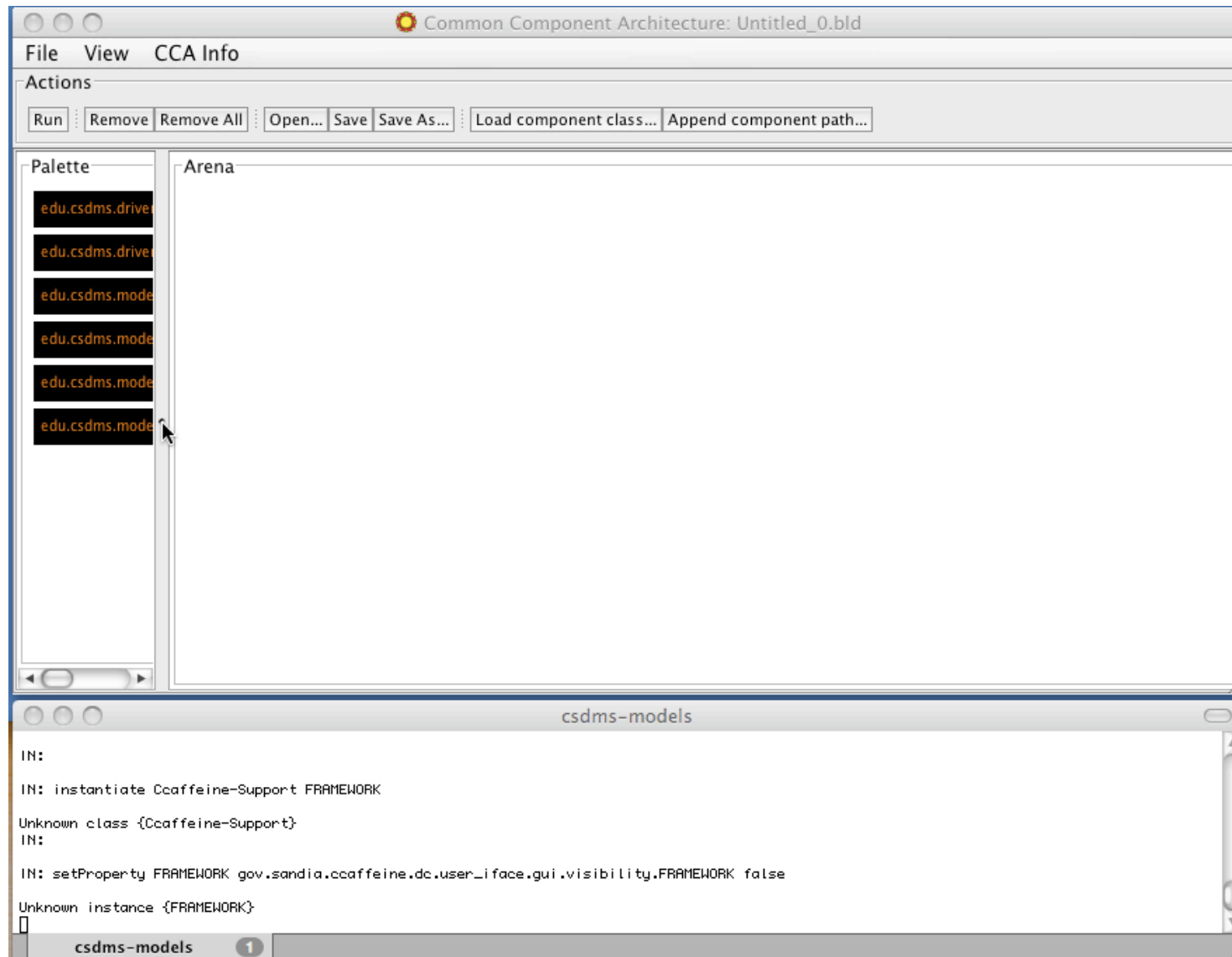


The driver is the new model that orchestrates the component models

The driver is unusual because it can be written without its component models

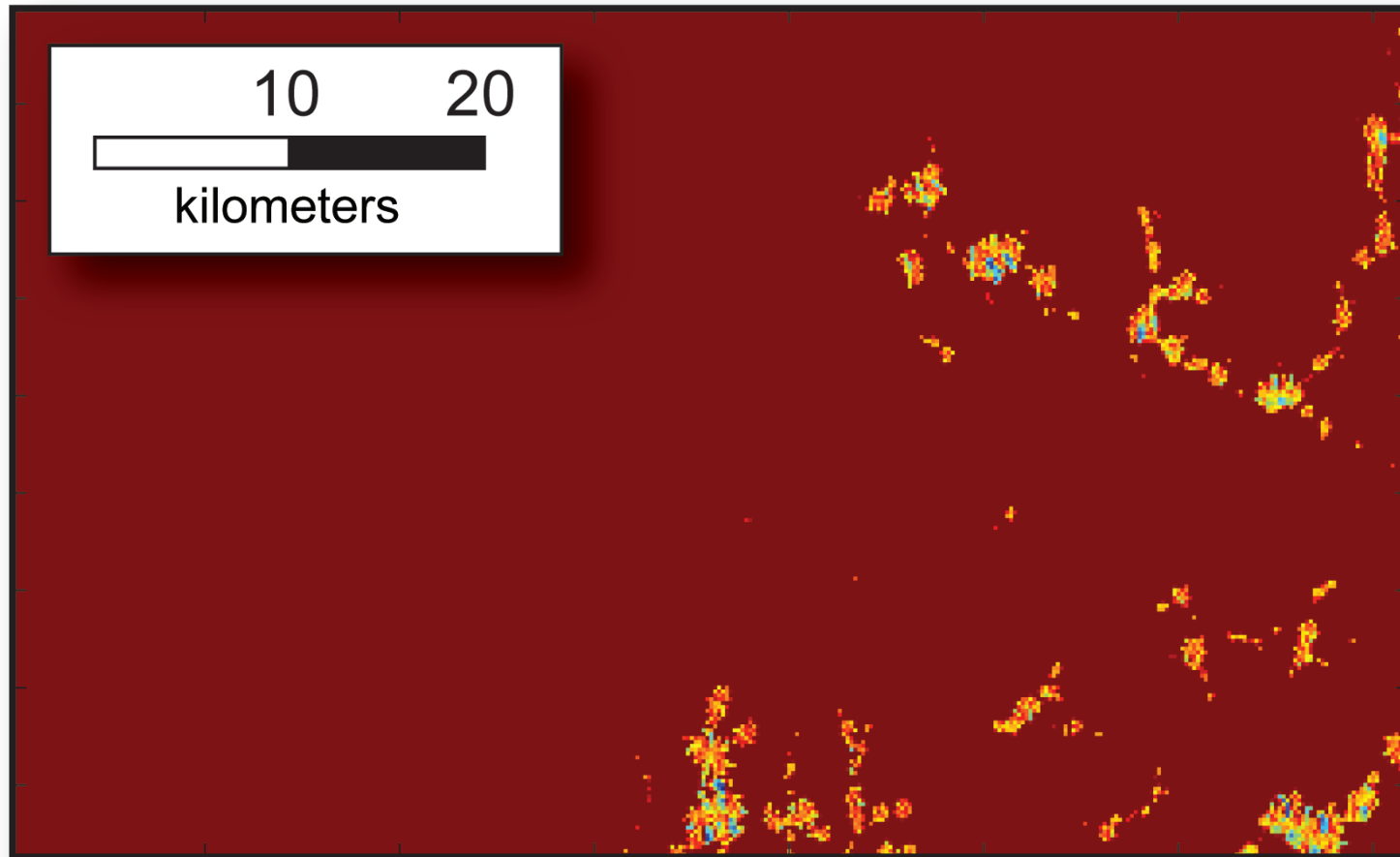
- **Sure, it won't function; it can be written though**
- **The writer of the driver model doesn't need to know about the component models**
- **This demonstrates the Inversion of Control design pattern**

Linking a glacial model with a subsidence model



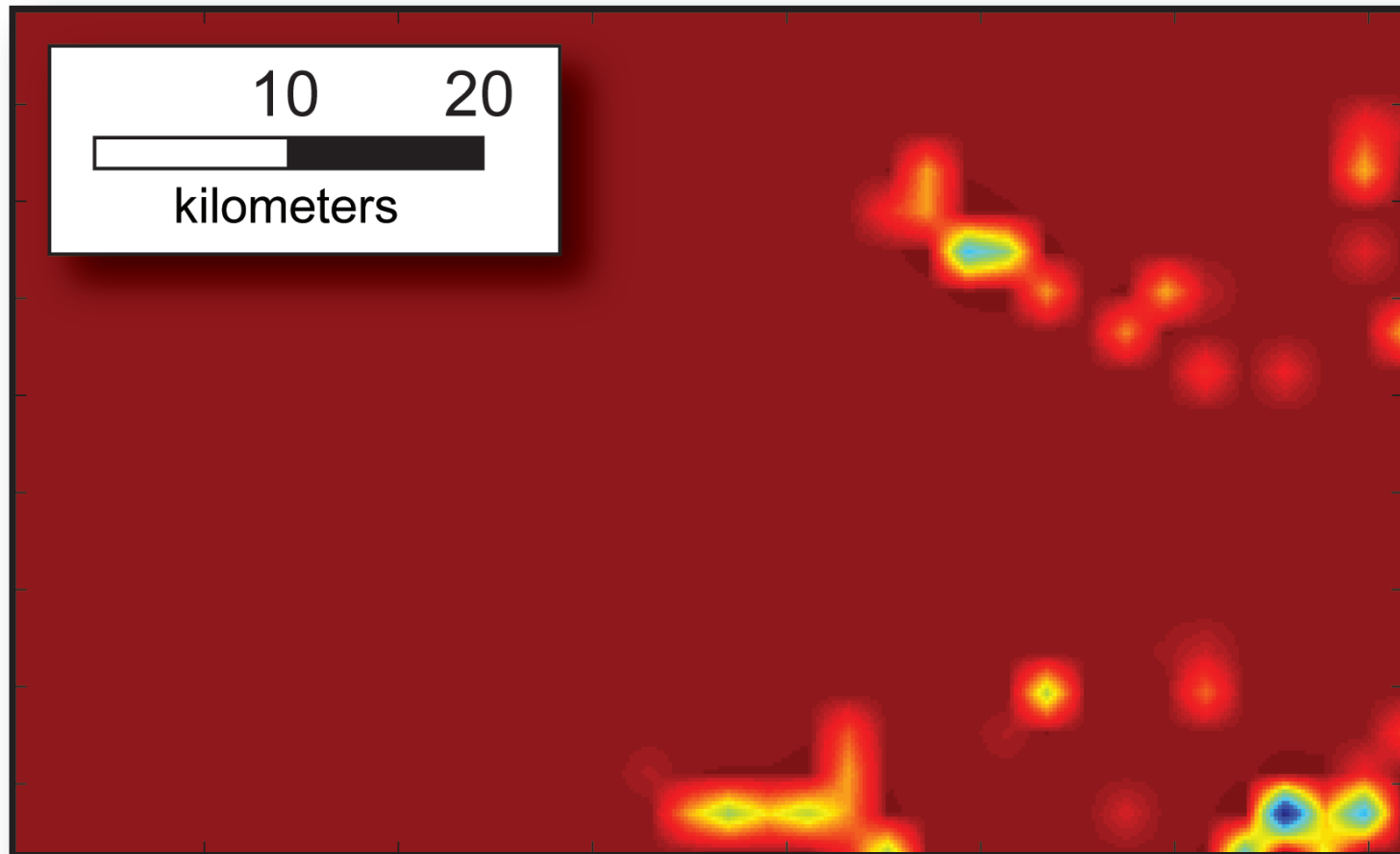
Airy subsidence shows deflections directly beneath load

Maximum Deflection = 37 m



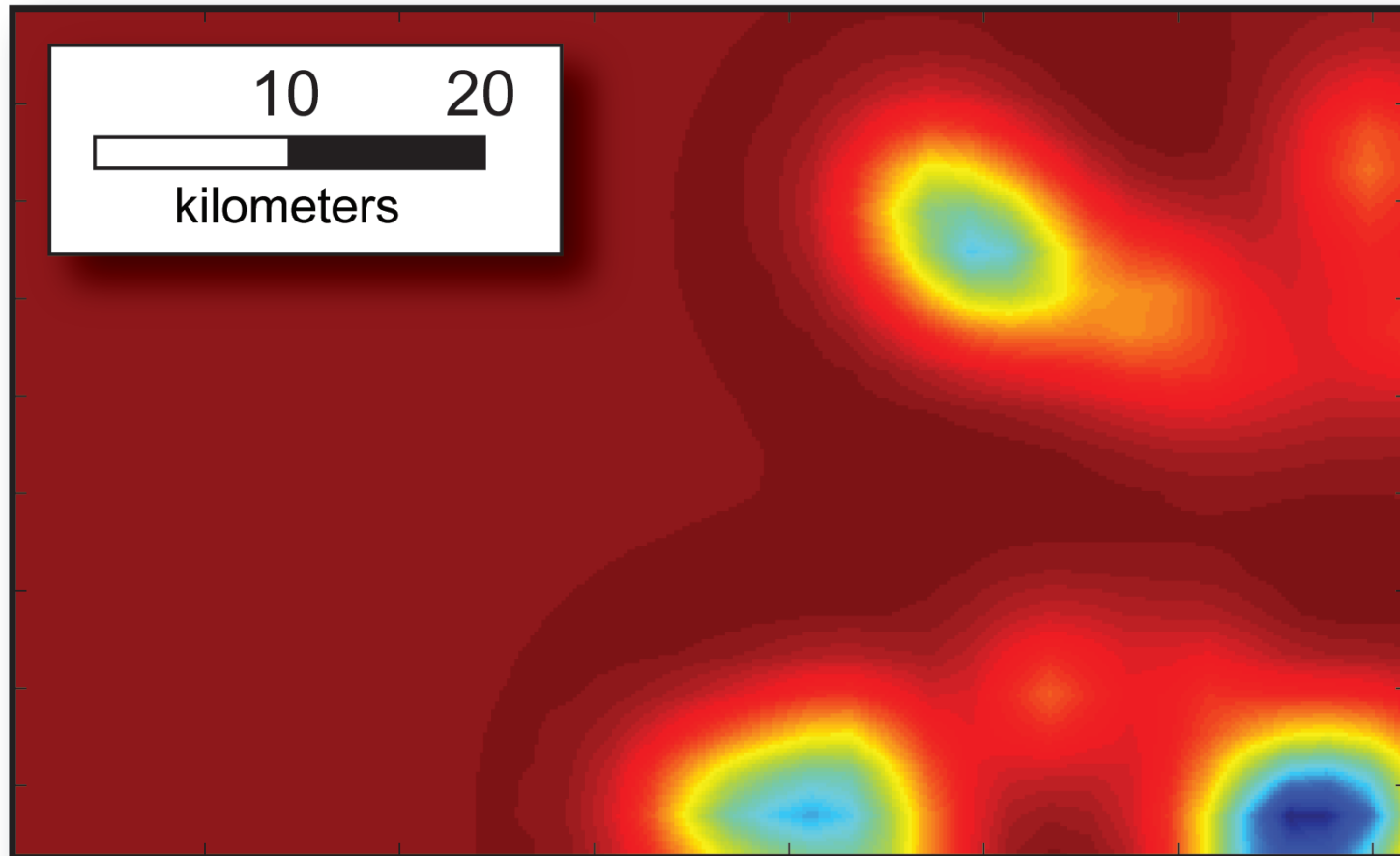
As the flexure parameter is increased, subsidence is more distributed

Maximum Deflection = 9 m



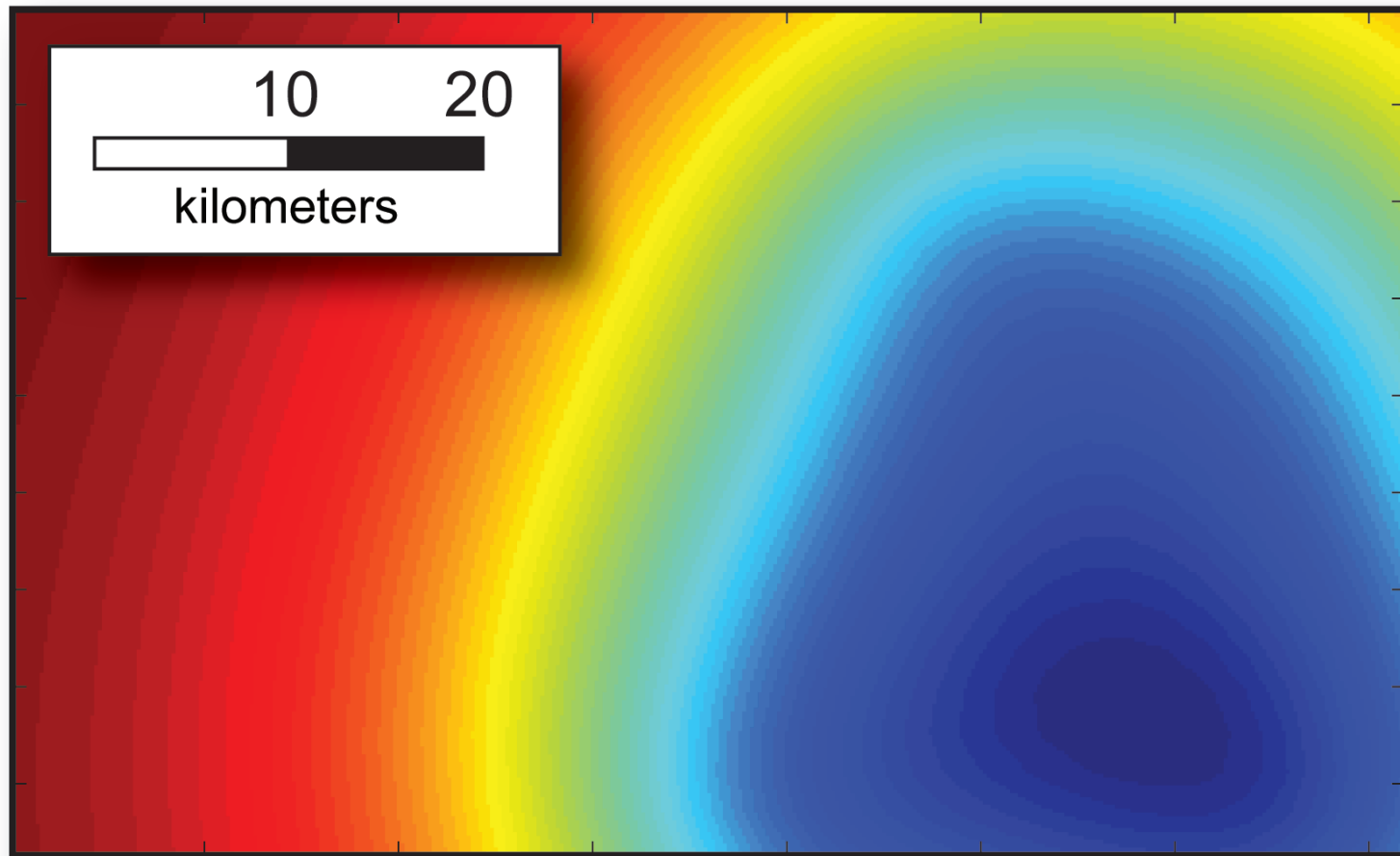
As the flexure parameter is increased, subsidence is more distributed

Maximum Deflection = .6 m



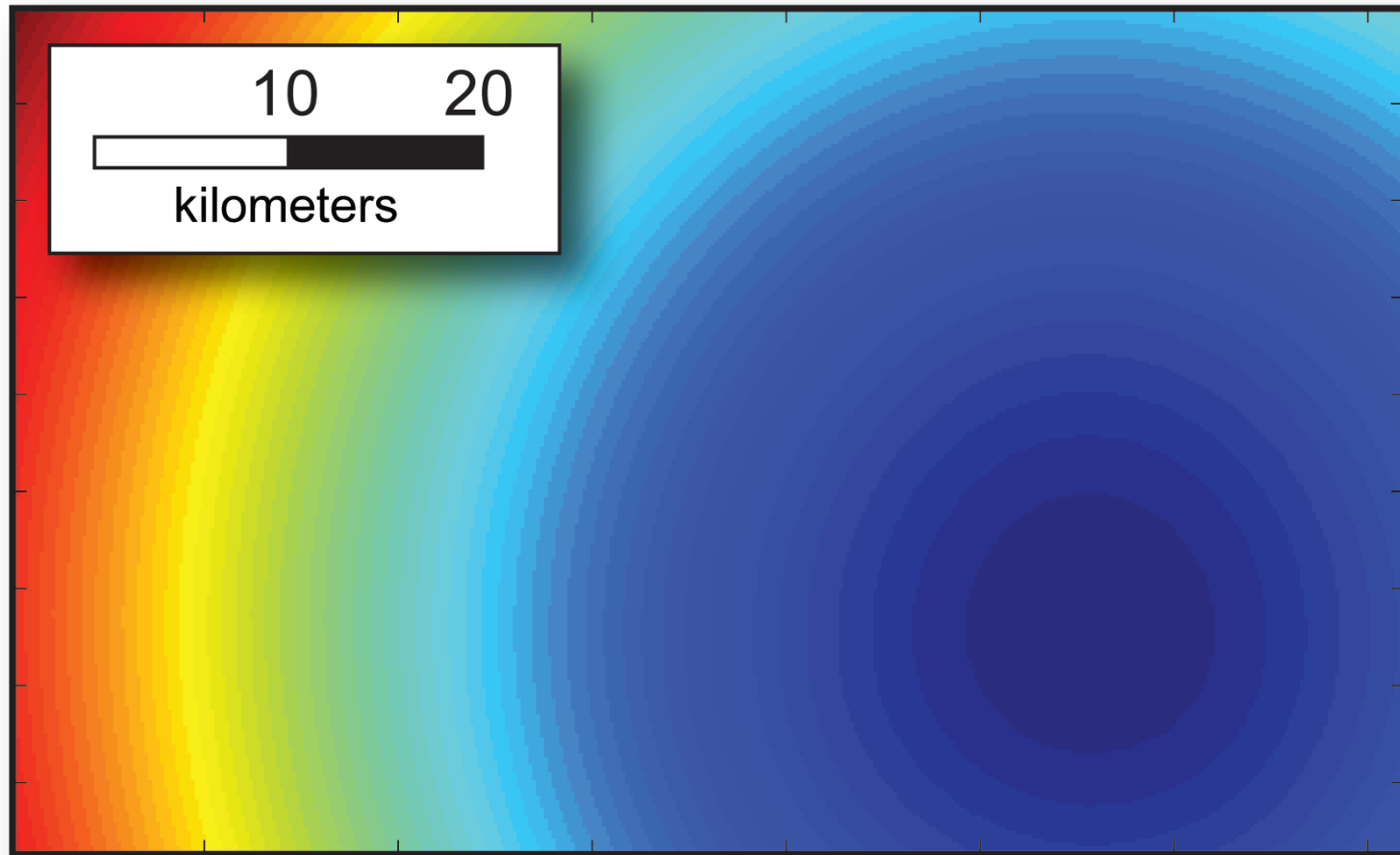
As the flexure parameter is increased, subsidence is more distributed

Maximum Deflection = .05 m

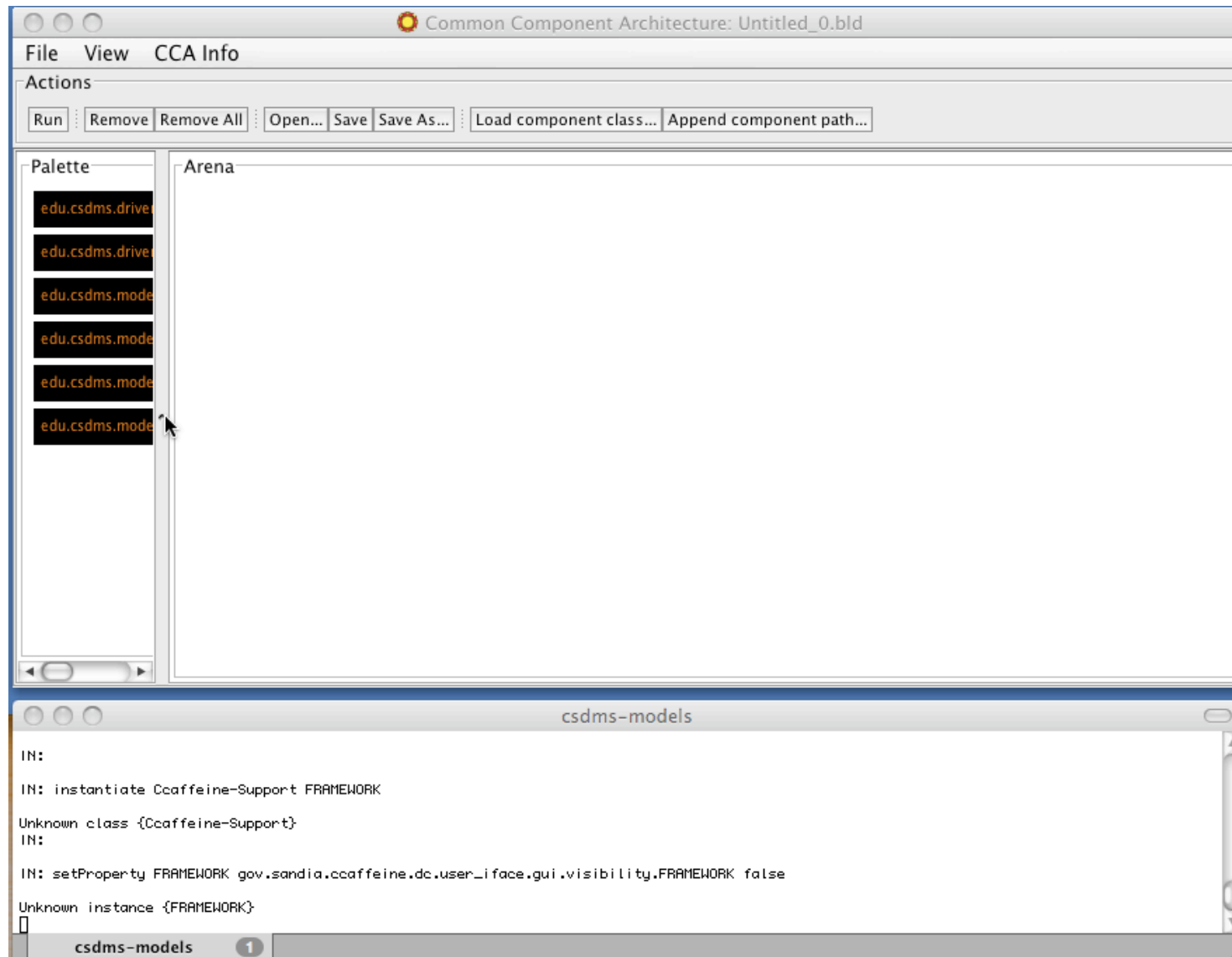


For realistic flexure parameters, the glaciers appear as a point load

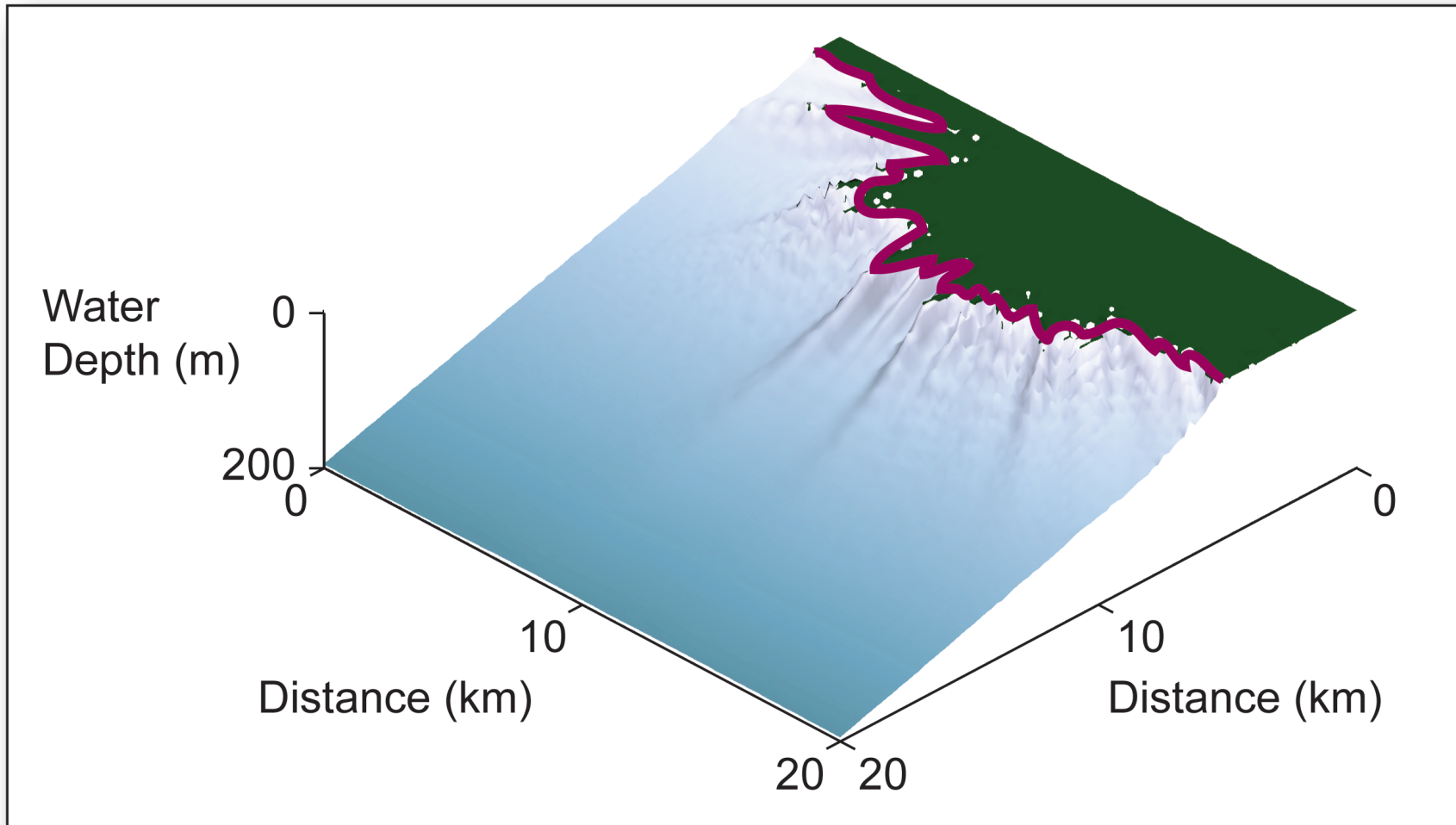
Maximum Deflection = .003 m



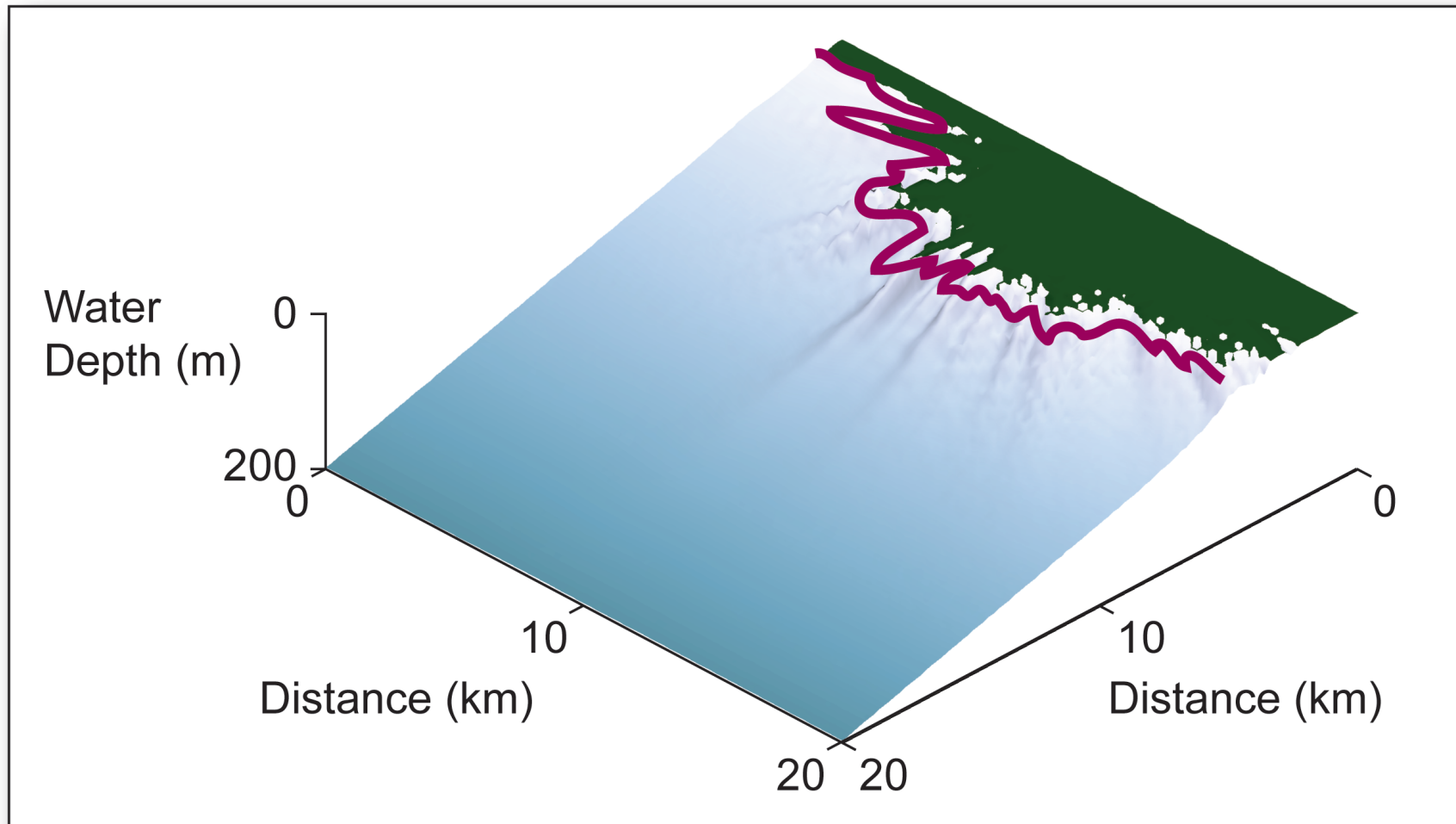
Linking a delta model with a subsidence model



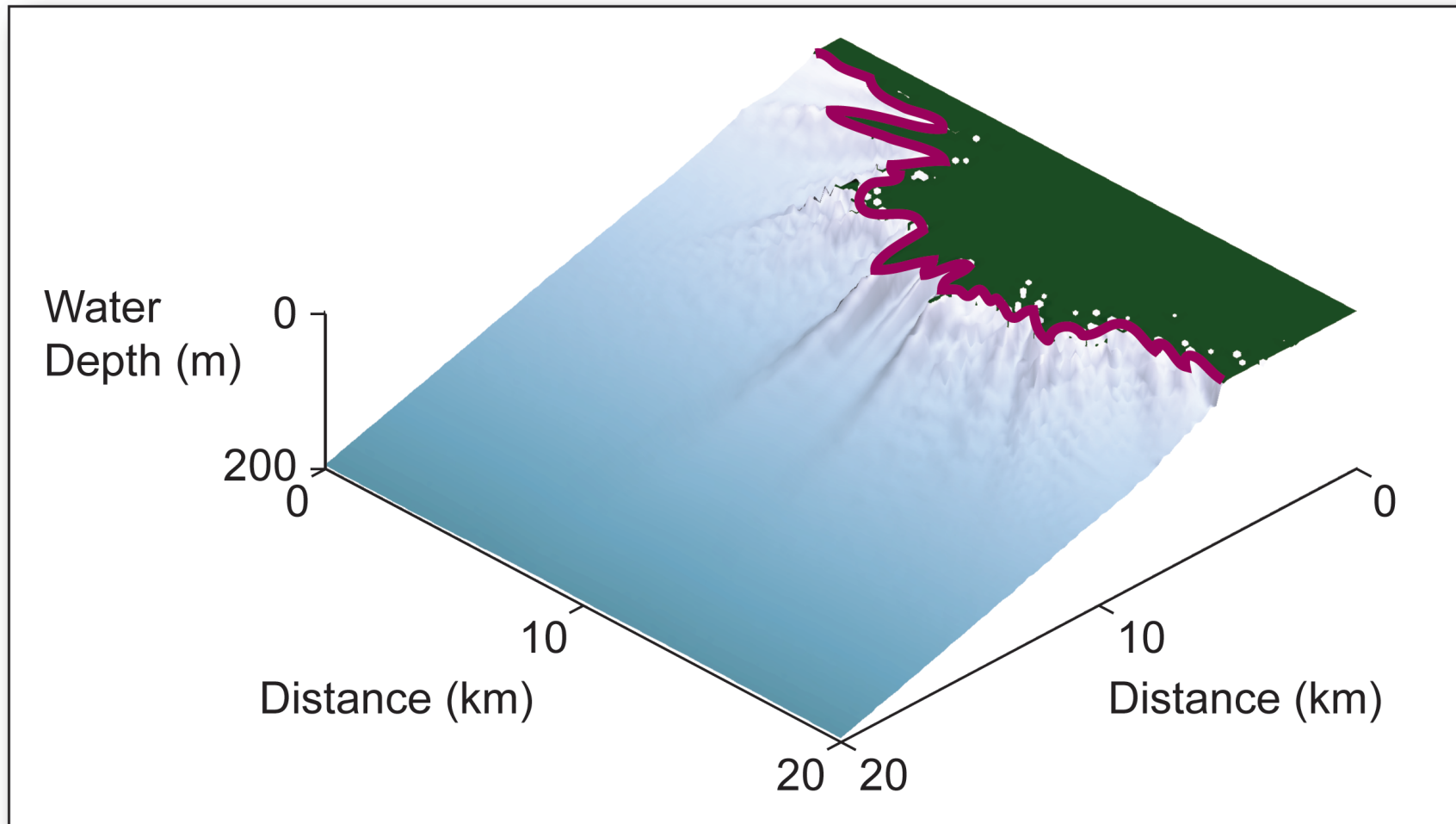
Without subsidence, the delta progrades about 7km



**With Airy subsidence, the delta progrades less.
About 5km**



With flexural subsidence, the delta progrades as if there were no subsidence at all



It is perfectly ok to couple models without a framework at all.

This is what I did when I coupled HydroTrend and Andrew's coastal model (deltas):

Both models are written in C

Both models have an easy-to-use interface

I maybe would not have done it this way if:

Models were written in another language

I was a Python programmer

I wanted to use CCA functionality (RPC, modularity)

The main loop of the DeltaTrend model is simple.

For i over each day of the simulation:

```
hydro_run_until(ht, i);
```

```
qb = hydro_get_bedload_flux(ht);
```

```
deltas_set_sediment_flux(cem, qb);
```

```
deltas_run_until(cem, i);
```

This is pretty much what the code would look like if I had used the CCA tools.

In conclusion,

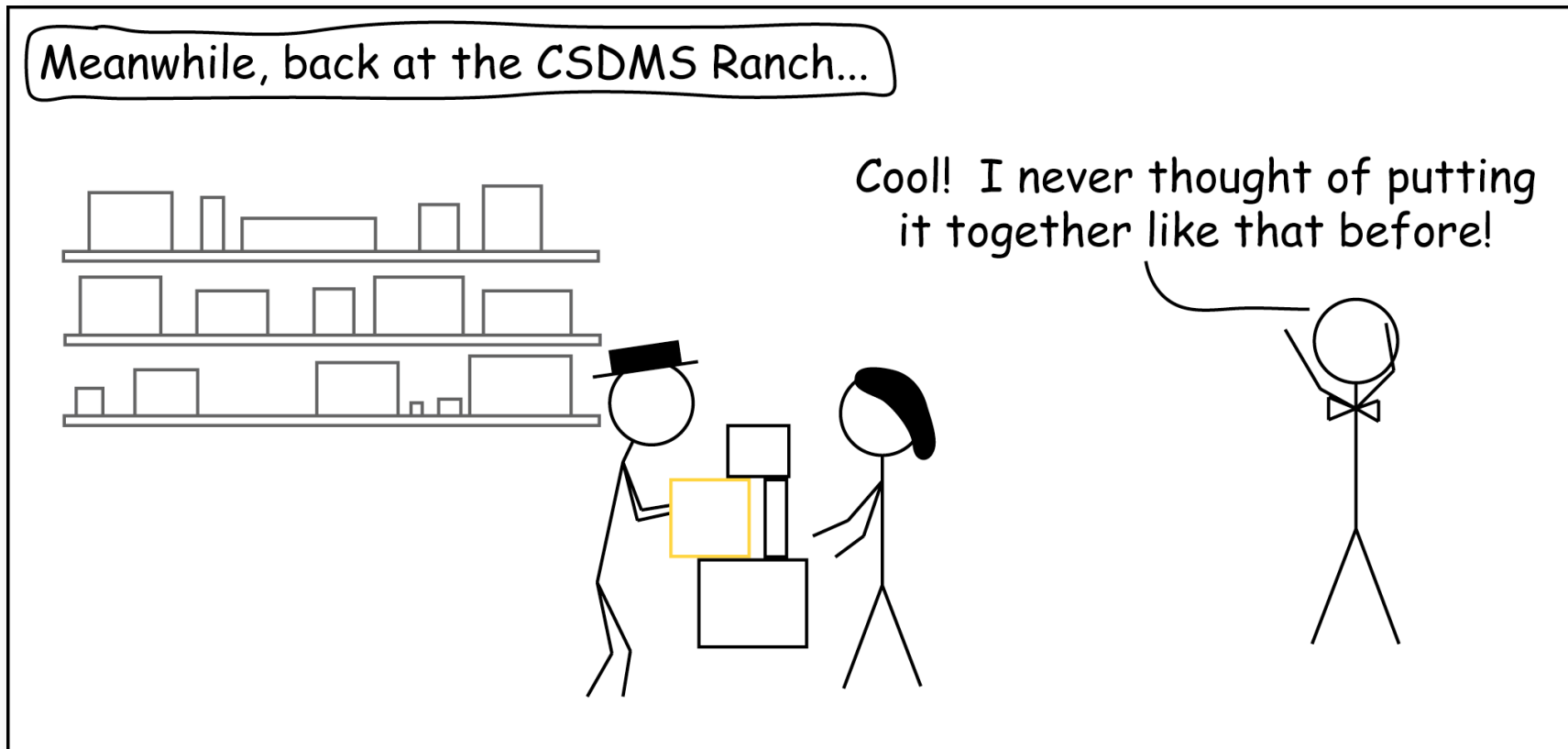
CSDMS employs a component-based modeling approach

**Authors contribute models with an IRF interface and in a
babel-supported language**

**CSDMS staff assists with converting models to
components**

Users assemble components to create new models

In conclusion, our hero was able to connect his model with others to create a fantastic new model



Questions?