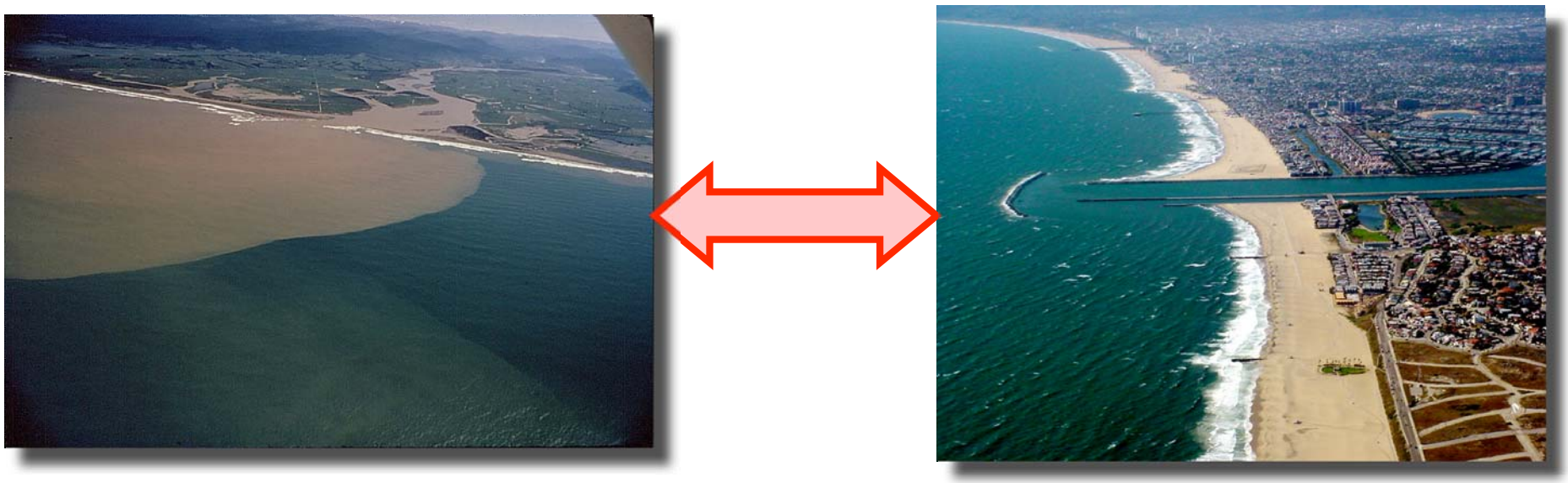
Creating and Linking Components in ESMF and CCA

Eric Hutton

Cyberinformatics and numerics working group

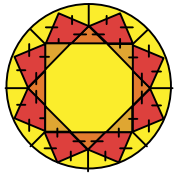
February 2008



My experiences creating and linking components from existing code

Create component:
sedflux

Create an ESMF application



Create component:
sedflux

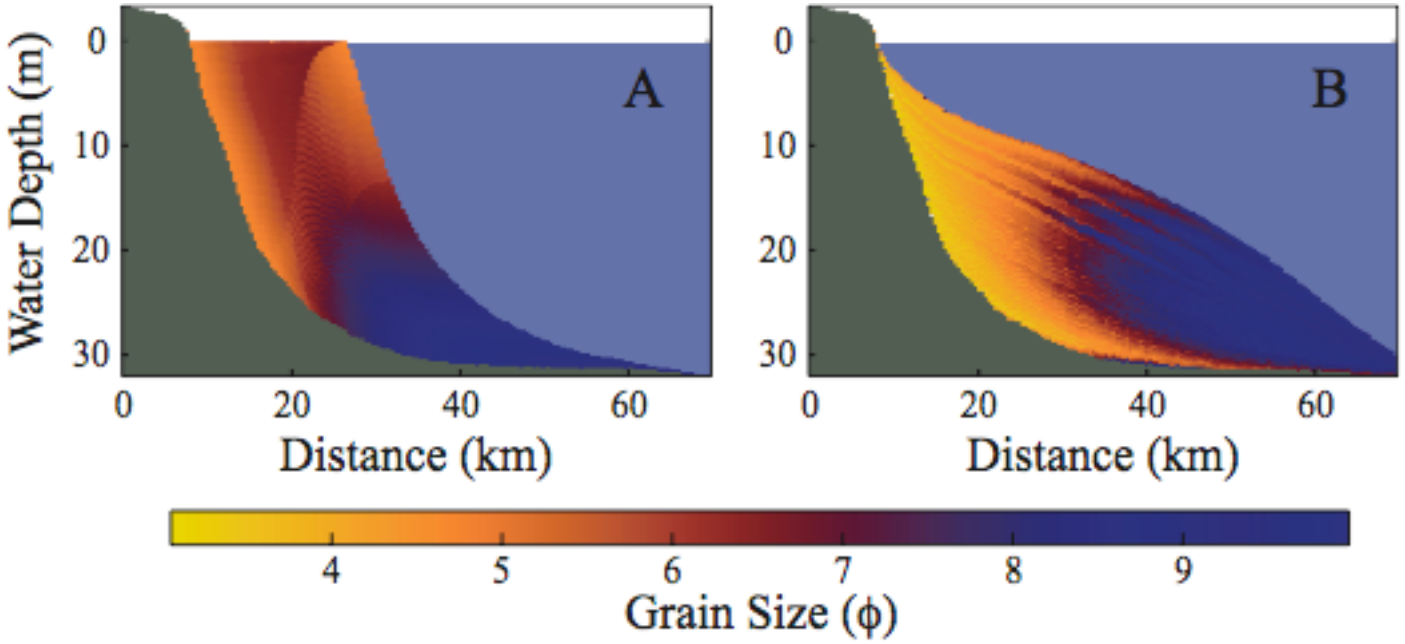
River
Plume
Subsidence

Link components to create an application

A component is an encapsulated “object” defined by its public interfaces

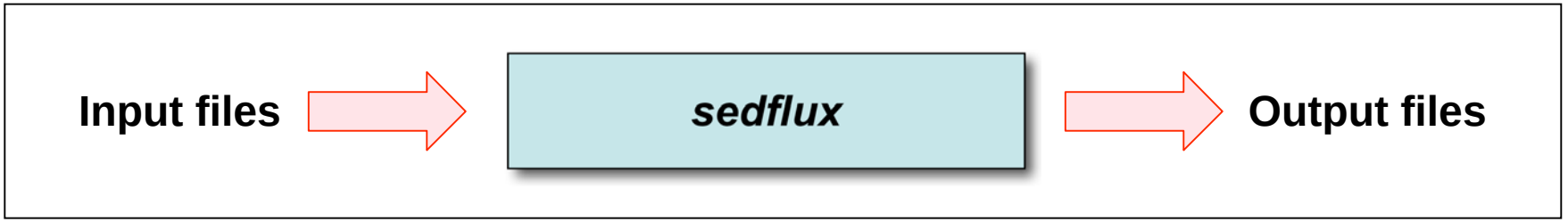
Interface Component Interface

sedflux links process models to build stratigraphy on continental margins



To become an ESMF component, the *sedflux* programming interface was refactored

Like many models, *sedflux* is called from the command line,



Create a *sedflux* library that contains a programming interface:

`sedflux_init()`:	**anything done before time stepping (allocate resources, open files, etc.)**
`sedflux_run()`:	**advance the model one time step**
`sedflux_destroy()`:	**anything done after time stepping**

The first difficulty was that sedflux is written in c and ESMF in FORTRAN

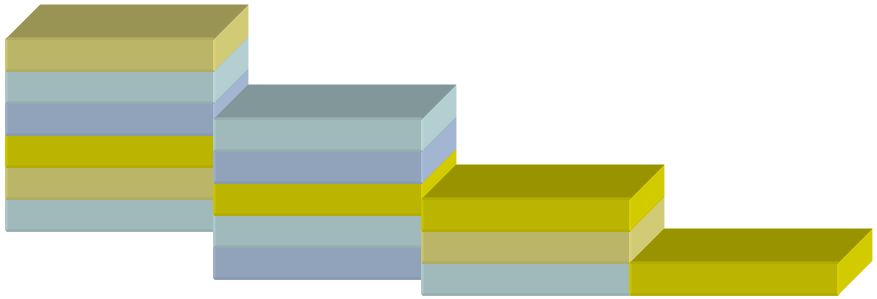
Details in communicating between c and FORTRAN can be both platform and compiler specific.

Difficulties calling c from FORTRAN (or vice versa) include:

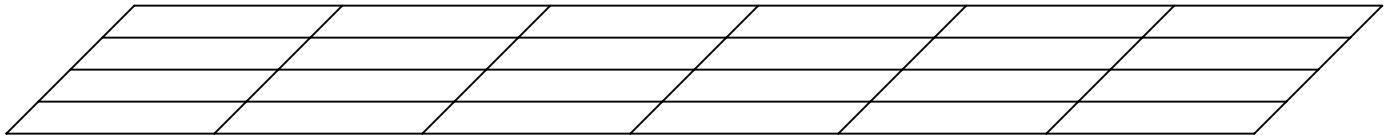
Name mangling:	`Foo()` **becomes** `foo_`**, or** `F00`**, or** `F00_`**, or …**
Arrays:	**FORTRAN arrays are not simply pointers**
Unsupported features:	**Complex numbers, pointers, structs**

The second difficulty was that *sedflux* is not grid based in the same way the ESMF is

sedflux **thinks of the world as cubes of sediment stacked on top of one another to form columns.**

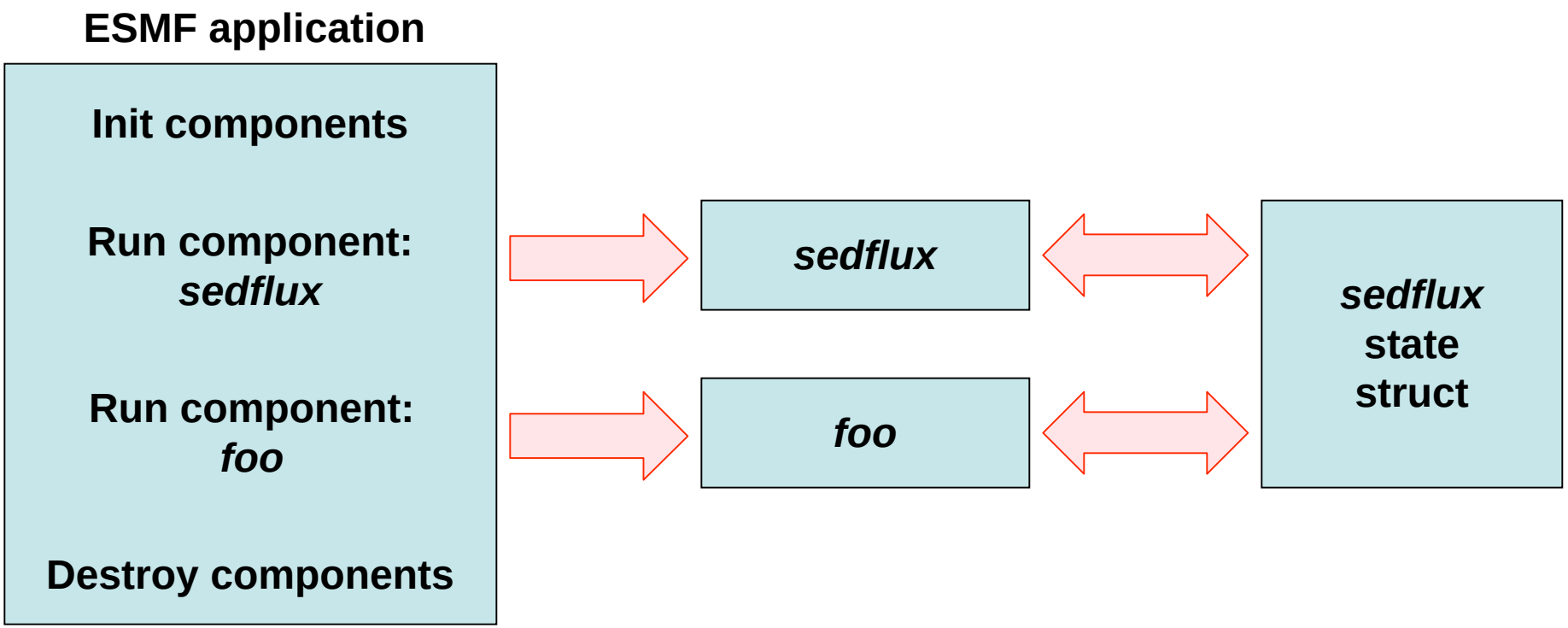


ESMF likes uniform grids of variables.



To get around this, sedflux kept track of its own state through global variables.

To get around this, *sedflux* kept track of its own state through global variables



For another component to interact with *sedflux* it needs to know about this state variable. Imposes *sedflux*'s implementation.

Not Good

In CCA, interfaces (or ports) define a component

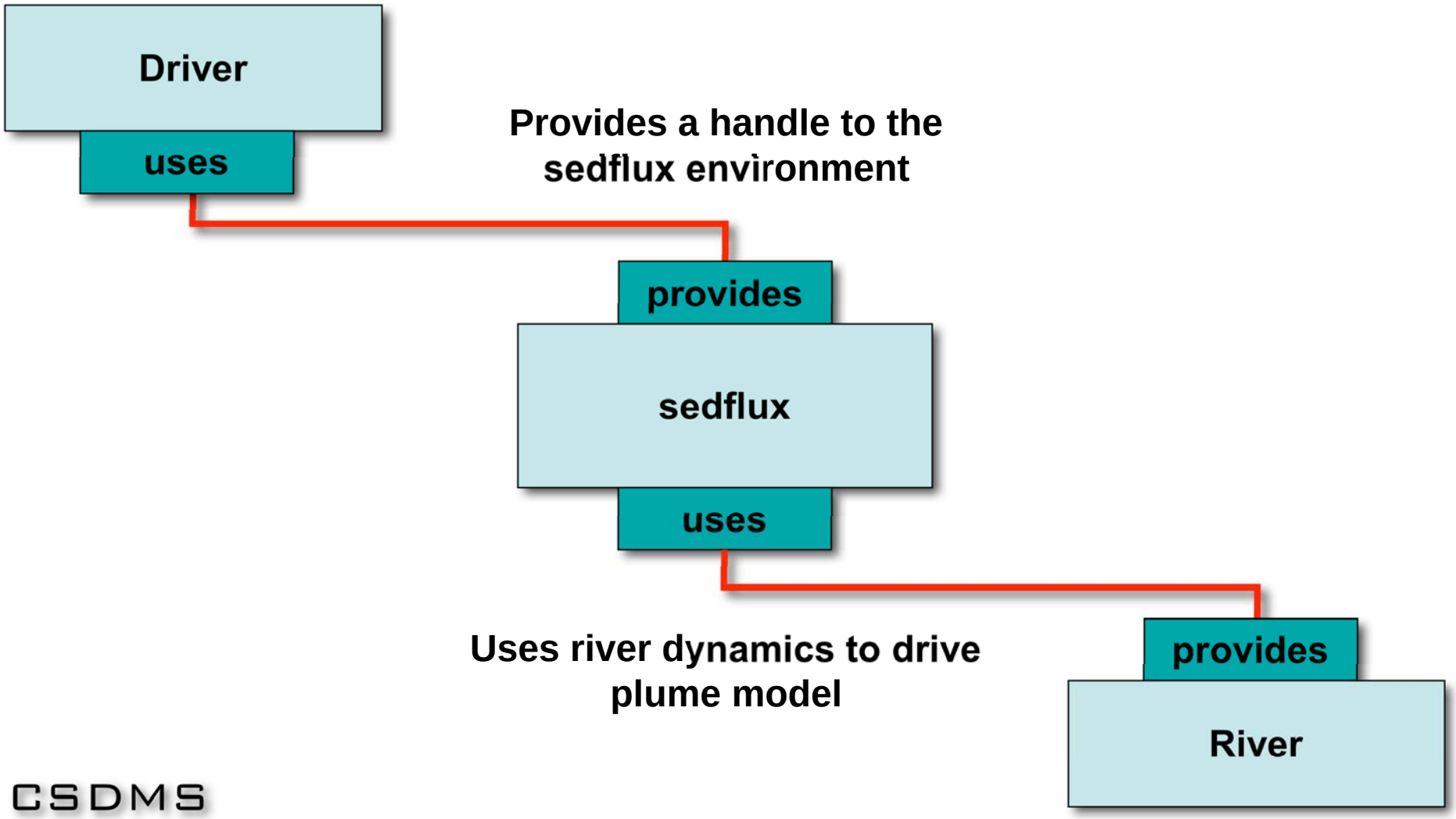
A port is not an implementation only a description. The description is written in either SIDL (Scientific Interface Definition Language) or XML.

A subsidence component could have a `deflectionPort`

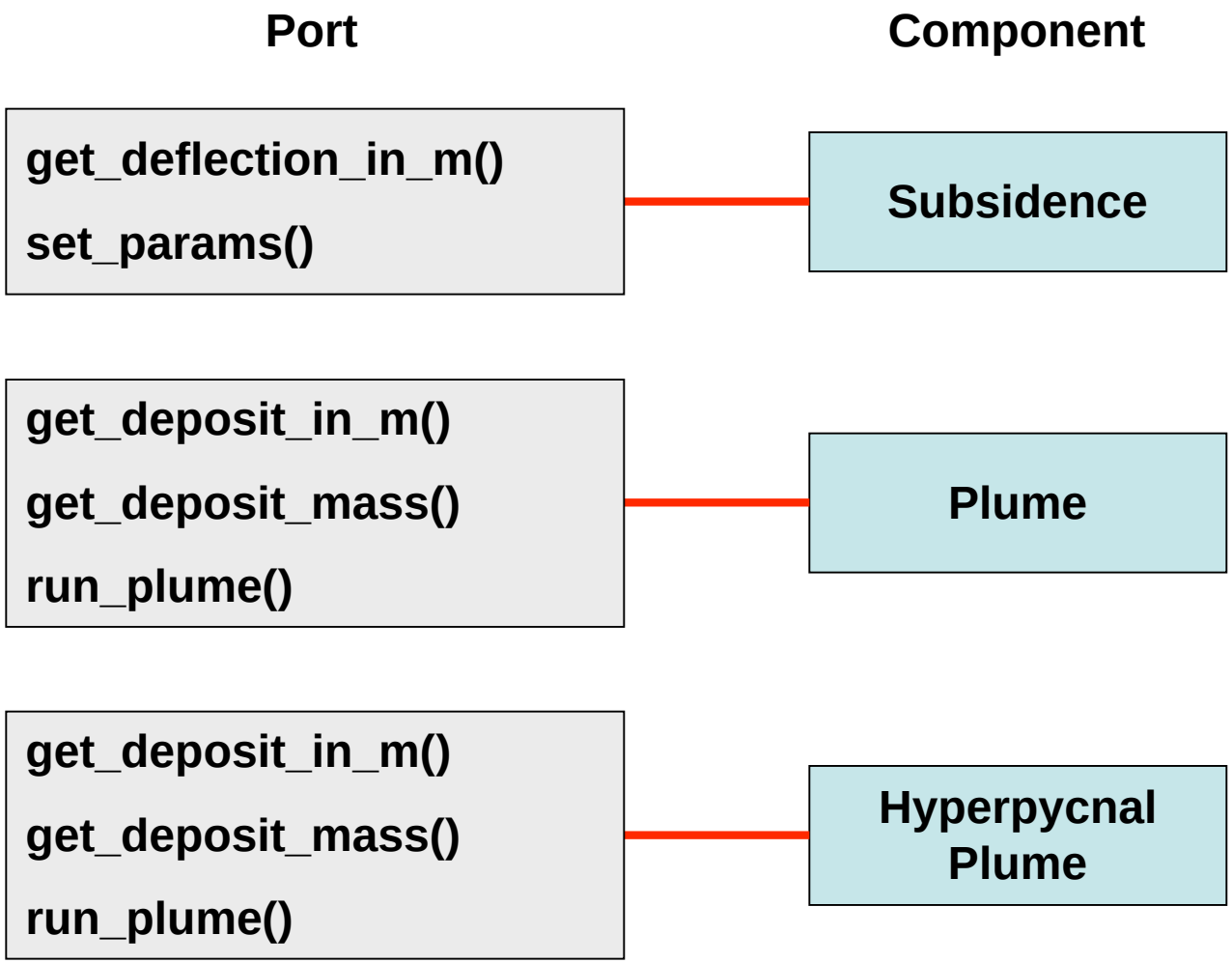
```
interface deflectionPort extends gov.cca.Port
{
   // Set constants from a file.
   int init( in string file );

   // Get the deflection (in meters) due to the applied loads.
   int get_deflection_in_m( inout rarray<double,1> dz(len)   ,
                            in    rarray<double,1> x(len)    ,
                            in    rarray<double,1> load(len) ,
                            in    int              len );
}
```

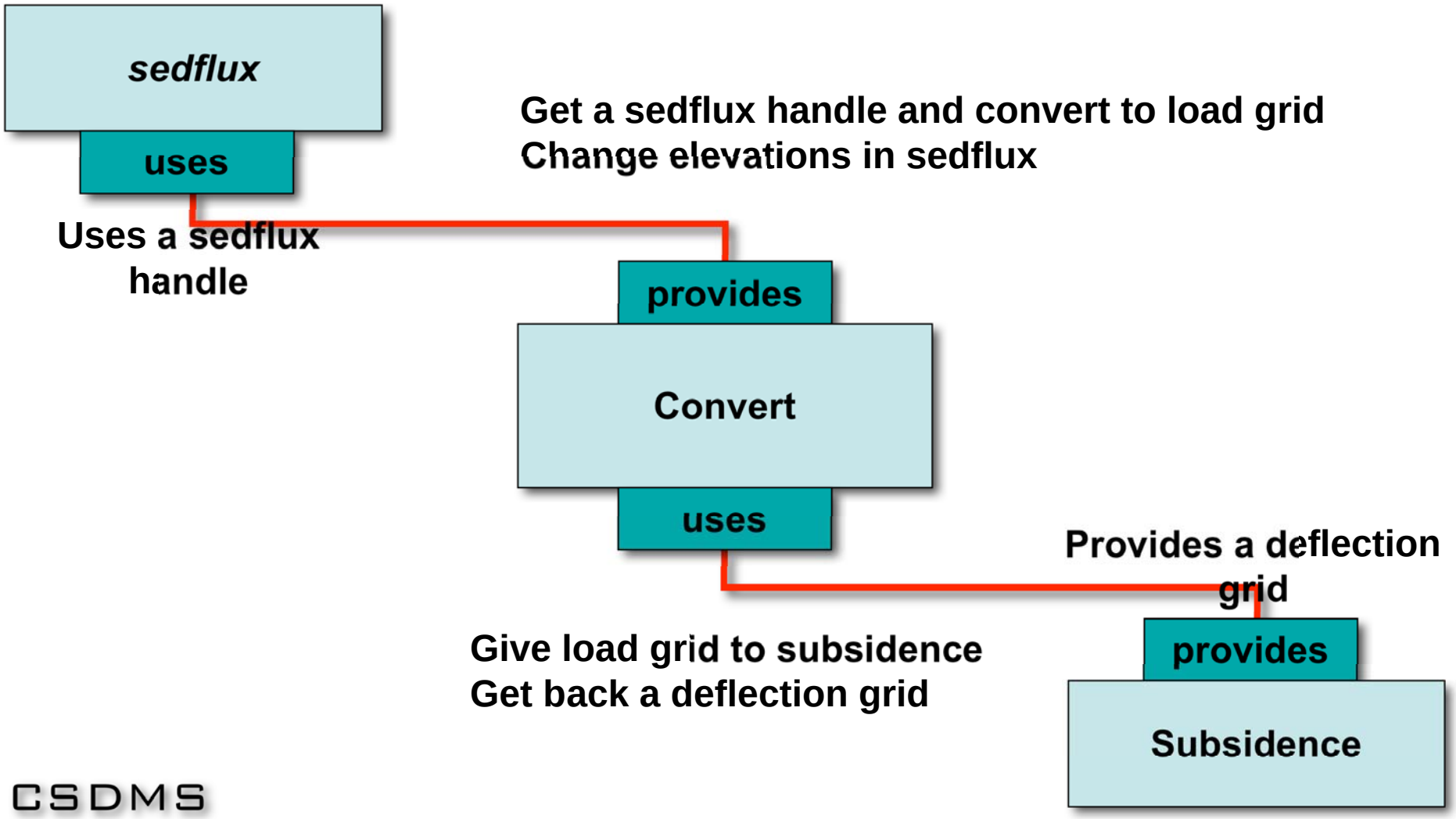
A CCA component can both use and provide data through a port



Smaller models provide another level of granularity

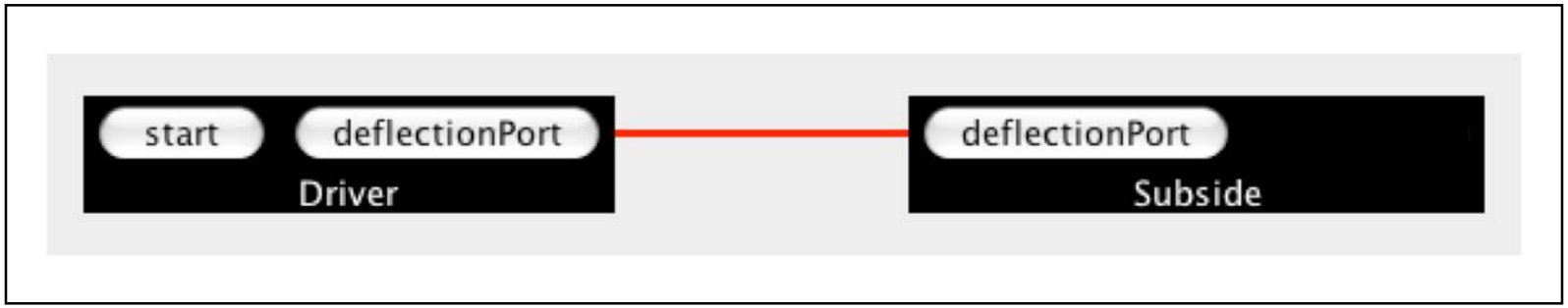


Adding a subsidence model to *sedflux* requires a converter component



In CCA, components are connected using ccafe

ccafe can be run in gui mode (needs some work):



or on the command line:

```
> connect Driver deflectionPort Subside deflectionPort
```

In conclusion, CCA offers more flexibility while ESMF more infrastructure

CCA is more flexible

- **Language neutral**
- **Does not impose a framework**

ESMF comes with a large amount of infrastructure

- **Timers**
- **States**
- **Grid manipulation**
- **Decomposition elements**

But ESMF could be incorporated into CCA as a toolkit

Questions?