Turning Lessons Learned Into Guiding Principles: Ideas for Reinventing the Earth System Modeling Enterprise

Todd Ringler
Theoretical Division
Los Alamos National Laboratory

The ESM enterprise is ripe for a transformation — a transformation in culture, norms, and best practices.

The ESM enterprise is ripe for a transformation — a transformation in culture, norms, and best practices.

Creating the conceptual model of the earth system is a very different activity than producing performant code.

earth scientists — computational scientists

The gap between these two domains of expertise has been widening and will continue to widen.

The era of a typical scientist having sufficient expertise in both domains is long past, but our culture, norms, and best practices have not changed accordingly.

ESM continue to grow in complexity and risk becoming increasingly fragile in ways that impede our productivity and limit the impact of our work.

Different phases of the ESM enterprise:

unit and test cases
Demonstrations that the software is doing what is described in the design document (verification) and appropriately represents some physical process (validation).

production software
A HPC performant version of the prototype algorithm.

simulation data
Output (usually in the form of netCDF files) produced by executing the software.

design document
Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

prototype algorithm
A realization of the design document as software.

At least in concept, most in the community agree that each of these elements are important. The ordering of the elements and emphasis of each element varies significantly across the community.

In 2010, we had a chance to start over with the Model for Prediction Across Scales project.

This is a joint LANL-NCAR model development project to build a suite of unstructured-grid earth system system components based on a shared software framework.

What did we do right?

What would we do differently?

MPAS
Model for Prediction Across Scales

MPAS Home

Overview

MPAS-Atmosphere
MPAS-Albany Land Ice
MPAS-Ocean
MPAS-Seaice
Data Assimilation
Publications
Presentations

Download

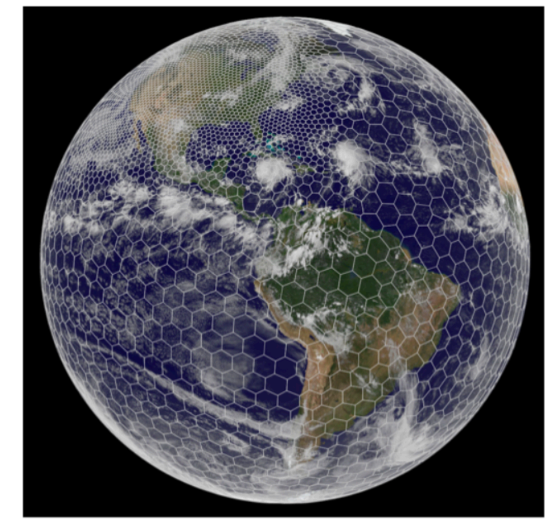
MPAS-Atmosphere download
MPAS-Albany Land Ice download
MPAS-Ocean download
MPAS-Seaice download

Resources

License Information
Wiki
Bug Tracker
Mailing Lists
MPAS Developers Guide
MPAS Mesh Specification Document

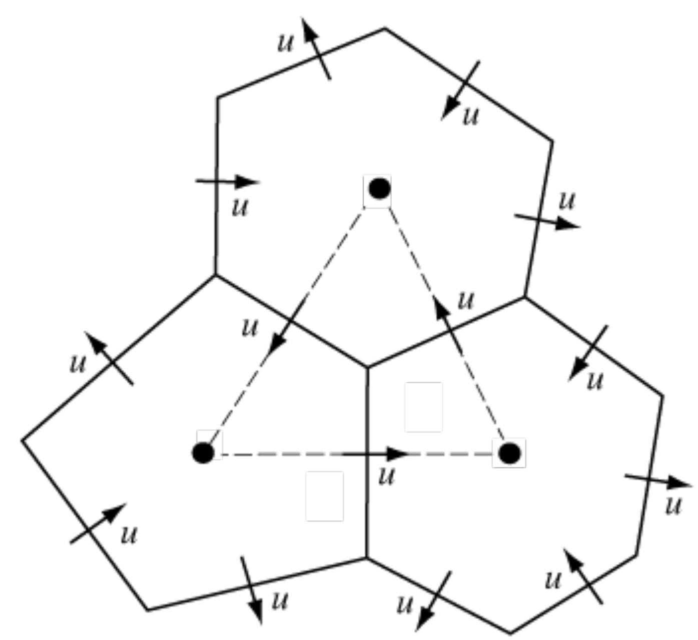
MPAS Overview

The Model for Prediction Across Scales (MPAS) is a collaborative project for developing atmosphere, ocean and other earth-system simulation components for use in climate, regional climate and weather studies. The primary development partners are the climate modeling group at Los Alamos National Laboratory (COSIM) and the National Center for Atmospheric Research. Both primary partners are responsible for the MPAS framework, operators and tools common to the applications; LANL has primary responsibility for the ocean and land ice models, and NCAR has primary responsibility for the atmospheric model.



The defining features of MPAS are the unstructured Voronoi meshes and C-grid discretization used as the basis for many of the model components. The unstructured Voronoi meshes, formally Spherical Centriodal Voronoi Tesselations (SCVTs), allow for both quasi-uniform discretization of the sphere and local refinement. The C-grid discretization, where the normal component of velocity on cell edges is prognosed, is especially well-suited for higher-resolution, mesoscale atmosphere and ocean simulations. The land ice model takes advantage of the SCVT-dual mesh, which is a triangular Delaunay tessellation appropriate for use with Finite-Element-based discretizations.

The current MPAS release is version 6.1. Please refer to each core for changes, and the github repository for source.



What did we see as problems in 2010?

<u>design document</u>

Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

<u>prototype algorithm</u>

A realization of the design document as software.

<u>unit and test cases</u>

Demonstrations that the software doing what is described in the design document (verification) and appropriately represents some physical process (validation).

<u>production software</u>

An HPC performant version of the prototype algorithm.

<u>simulation data</u>

Output (usually in the form of netCDF files) produced by executing the software.

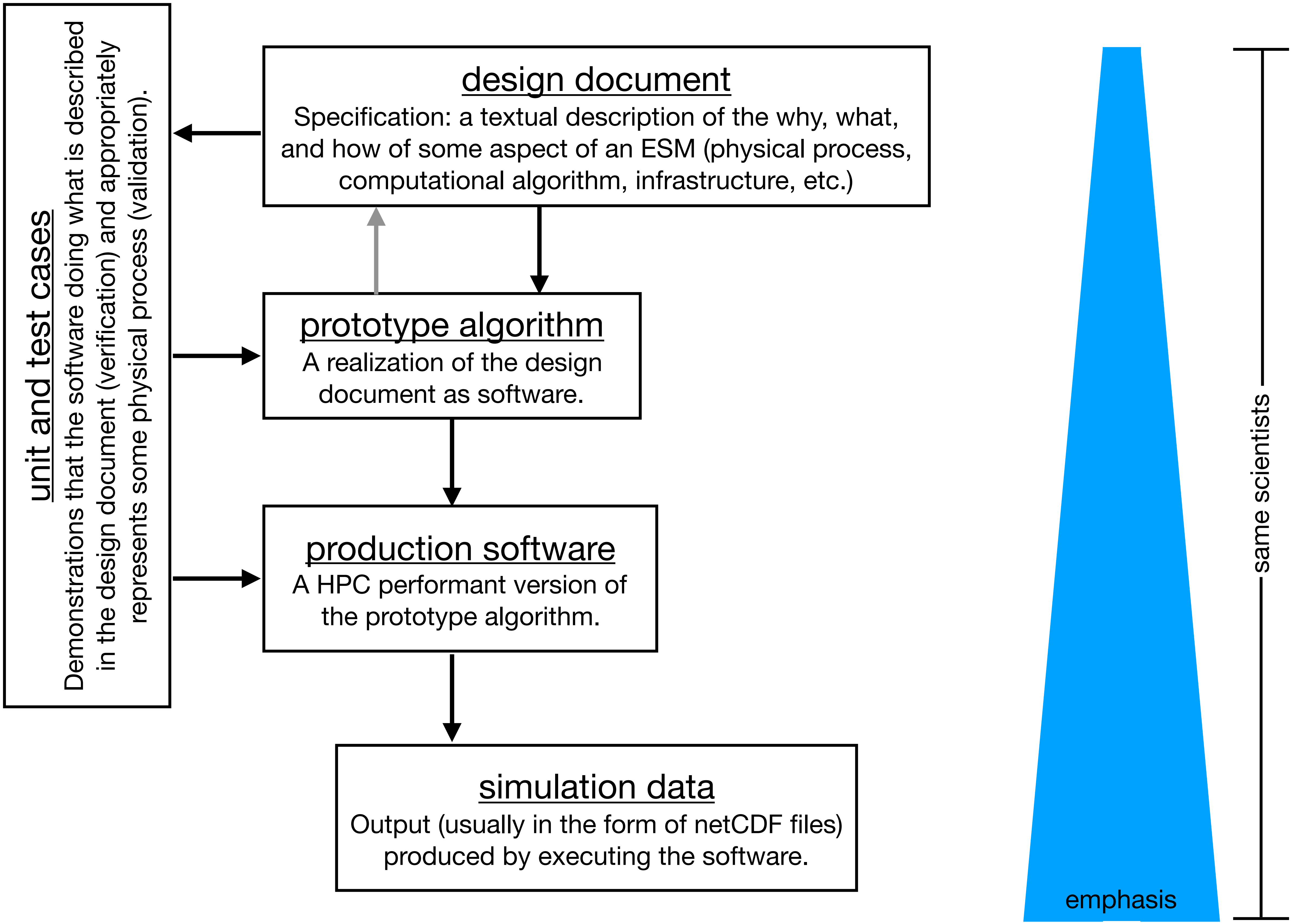
What did we do right?

auto-generate nearly all Fortran code from XML, Python, and C
git, Github, and adoption of git feature branch workflow
transparency — model development discussion on Github forum

a project-wide and uniform workflow for model development:

- design documents
- software and test case coherency
- proof of correctness — V&V
- continuous testing harness under revision control
- provenance — reproducibility of simulation data
- project review and fulfilling pull requests

We made dramatic changes in how we develop models ….



Design documents: Our biggest success. (And our biggest missed opportunity.)

Chapter 1

Summary

The purpose of this section is to summarize what capability is to be added to the MPAS system through this design process. It should be clear what new code will do that the current code does not. Summarizing the primary challenges with respect to software design and implementation is also appropriate for this section. Finally, this statement should contain general statement with regard to what is "success."

Chapter 2

Requirements

2.1 Requirement: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

Each requirement is to be listed under a "section" heading, as there will be a one-to-one correspondence between requirements, design, proposed implementation and testing.

Requirements should not discuss technical software issues, but rather focus on model capability. To the extent possible, requirements should be relatively independent of each other, thus allowing a clean design solution, implementation and testing plan.

Chapter 3

Algorithmic Formulations

3.1 Design Solution: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

For each requirement, there is a design solution that is intended to meet that requirement. Design solutions can include detailed technical discussions of PDEs, algorithms, solvers and similar, as well as technical discussion of performance issues. In general, this section should steer away from a detailed discussion of low-level software issues such as variable declarations, interfaces and sequencing.

Chapter 4

Design and Implementation

4.1 Implementation: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

This section should detail the plan for implementing the design solution for requirement XXX. In general, this section is software-centric with a focus on software implementation. Pseudo code is appropriate in this section. Links to actual source code are appropriate. Project management items, such as svn branches, timelines and staffing are also appropriate.

How do we typeset pseudo code?

```
verbatim?
```

Chapter 5

Testing

5.1 Testing and Validation: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

How will XXX be tested? i.e. how will be we know when we have met requirement XXX. Will these unit tests be included in the ongoing going forward?

Design documents: Our biggest success. (And our biggest missed opportunity.)

Chapter 1

Summary

The purpose of this section is to summarize what capability is to be added to the MPAS system through this design process. It should be clear what new code will do that the current code does not. Summarizing the primary challenges with respect to software design and implementation is also appropriate for this section. Finally, this statement should contain general statement with regard to what is "success."

Chapter 2

Requirements

2.1 Requirement: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

Each requirement is to be listed under a "section" heading, as there will be a one-to-one correspondence between requirements, design, proposed implementation and testing.

Requirements should not discuss technical software issues, but rather focus on model capability. To the extent possible, requirements should be relatively independent of each other, thus allowing a clean design solution, implementation and testing plan.

Chapter 3

Algorithmic Formulations

3.1 Design Solution: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

For each requirement, there is a design solution that is intended to meet that requirement. Design solutions can include detailed technical discussions of PDEs, algorithms, solvers and similar, as well as technical discussion of performance issues. In general, this section should steer away from a detailed discussion of low-level software issues such as variable declarations, interfaces and sequencing.

Chapter 4

Design and Implementation

4.1 Implementation: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

This section should detail the plan for implementing the design solution for requirement XXX. In general, this section is software-centric with a focus on software implementation. Pseudo code is appropriate in this section. Links to actual source code are appropriate. Project management items, such as svn branches, timelines and staffing are also appropriate.

How do we typeset pseudo code?

```
verbatim?
```

Chapter 5

Testing

5.1 Testing and Validation: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

How will XXX be tested? i.e. how will be we know when we have met requirement XXX. Will these unit tests be included in the ongoing going forward?

Pull requests should include a design document.
On average, the project produced about 10 design documents per year.

Why were design documents such a success?

- Think - Write - Discuss - Code - Test.
- Lots of people will read a text document.
- Good ideas are easy to incorporate.
- Author(s) not yet tied to code.
- Provides an anchor for when things go wrong.
- Turns a cost (documentation) into an investment.
- We now spend very little time “searching for bugs.”
- Empowering.

Chapter 1

Summary

The purpose of this section is to summarize what capability is to be added to the MPAS system through this design process. It should be clear what new code will do that the current code does not. Summarizing the primary challenges with respect to software design and implementation is also appropriate for this section. Finally, this statement should contain general statement with regard to what is “success.”

Chapter 2

Requirements

2.1 Requirement: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

Each requirement is to be listed under a ”section” heading, as there will be a one-to-one correspondence between requirements, design, proposed implementation and testing.

Requirements should not discuss technical software issues, but rather focus on model capability. To the extent possible, requirements should be relatively independent of each other, thus allowing a clean design solution, implementation and testing plan.

Chapter 3

Algorithmic Formulations

3.1 Design Solution: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

For each requirement, there is a design solution that is intended to meet that requirement. Design solutions can include detailed technical discussions of PDEs, algorithms, solvers and similar, as well as technical discussion of performance issues. In general, this section should steer away from a detailed discussion of low-level software issues such as variable declarations, interfaces and sequencing.

Chapter 4

Design and Implementation

4.1 Implementation: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

This section should detail the plan for implementing the design solution for requirement XXX. In general, this section is software-centric with a focus on software implementation. Pseudo code is appropriate in this section. Links to actual source code are appropriate. Project management items, such as svn branches, timelines and staffing are also appropriate.

How do we typeset pseudo code?

```
verbatim?
```

Chapter 5

Testing

5.1 Testing and Validation: XXX

Date last modified: 2011/01/05
Contributors: (add your name to this list if it does not appear)

How will XXX be tested? i.e. how will be we know when we have met requirement XXX. Will these unit tests be included in the ongoing going forward?

Our second most successful decision:
forcing coherency between model and test cases at every commit

unit and test cases

Demonstrations that the software doing what is described in the design document (verification) and appropriately represents some physical process (validation).

production software

An HPC performant version of the prototype algorithm.

```
drwxr-xr-x@ 19 TRingler  staff  646 May  3 12:32 ./
drwxr-xr-x@ 17 TRingler  staff  578 May  3 12:29 ../
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:30 baroclinic_channel/
drwxr-xr-x@ 16 TRingler  staff  544 May  3 12:33 global_ocean/
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:29 internal_waves/
drwxr-xr-x@  6 TRingler  staff  204 May  3 12:31 isomip/
drwxr-xr-x@ 12 TRingler  staff  408 May  3 12:33 isomip_plus/
drwxr-xr-x@  4 TRingler  staff  136 May  3 12:33 iterative_ssh_landIcePressure_scripts/
drwxr-xr-x@  4 TRingler  staff  136 May  3 12:29 lock_exchange/
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:28 overflow/
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:29 periodic_planar/
drwxr-xr-x@  4 TRingler  staff  136 May  3 12:32 regression_suites/
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:31 scripts/
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:28 sea_mount/
drwxr-xr-x@  4 TRingler  staff  136 May  3 12:31 single_column_model/
drwxr-xr-x@  9 TRingler  staff  306 May  3 12:31 soma/
drwxr-xr-x@  3 TRingler  staff  102 May  3 12:29 sub_ice_shelf_2D/
drwxr-xr-x@  6 TRingler  staff  204 May  3 12:31 templates/
drwxr-xr-x@  9 TRingler  staff  306 May  3 12:31 ziso/
```

test cases auto-configured from XML.

```
drwxr-xr-x@  9 TRingler  staff   306 May  3 12:31 ./
drwxr-xr-x@ 19 TRingler  staff   646 May  3 12:32 ../
drwxr-xr-x@  5 TRingler  staff   170 May  3 12:31 16km/
drwxr-xr-x@  5 TRingler  staff   170 May  3 12:31 32km/
drwxr-xr-x@  5 TRingler  staff   170 May  3 12:31 4km/
drwxr-xr-x@  5 TRingler  staff   170 May  3 12:31 8km/
-rw-r--r--@  1 TRingler  staff  2213 Apr 25  2017 soma_analysis_members.xml
-rw-r--r--@  1 TRingler  staff  3444 Apr 25  2017 soma_run.xml
-rw-r--r--@  1 TRingler  staff  1063 Apr 25  2017 soma_template.xml
```

Where did we come up short?

<u>unit and test cases</u>
Demonstrations that the software doing what is described in the design document (verification) and appropriately represents some physical process (validation).

<u>production software</u>
An HPC performant version of the prototype algorithm.

<u>design document</u>
Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

Having the design documents in a separate repo made is hard to maintain coherency between design and software.

Where did we come up short?

unit and test cases
Demonstrations that the software doing what is described in the design document (verification) and appropriately represents some physical process (validation).

production software
An HPC performant version of the prototype algorithm.

design document
Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

prototype algorithm
A realization of the design document as software.

Having the design documents in a separate repo made is hard to maintain coherency between design and software.

We do not track prototype algorithms.

Risks in front of us.

Disruption in the computing ecosystem ….

- Cori / KNL
- Summit / GPU
- Cloud computing / Xeon
- Aurora / World’s fast machine in 2021 / ??
- Los Alamos / ARM?

How do we get ~1M lines of code to be performant on such a wide diversity of computing architectures?

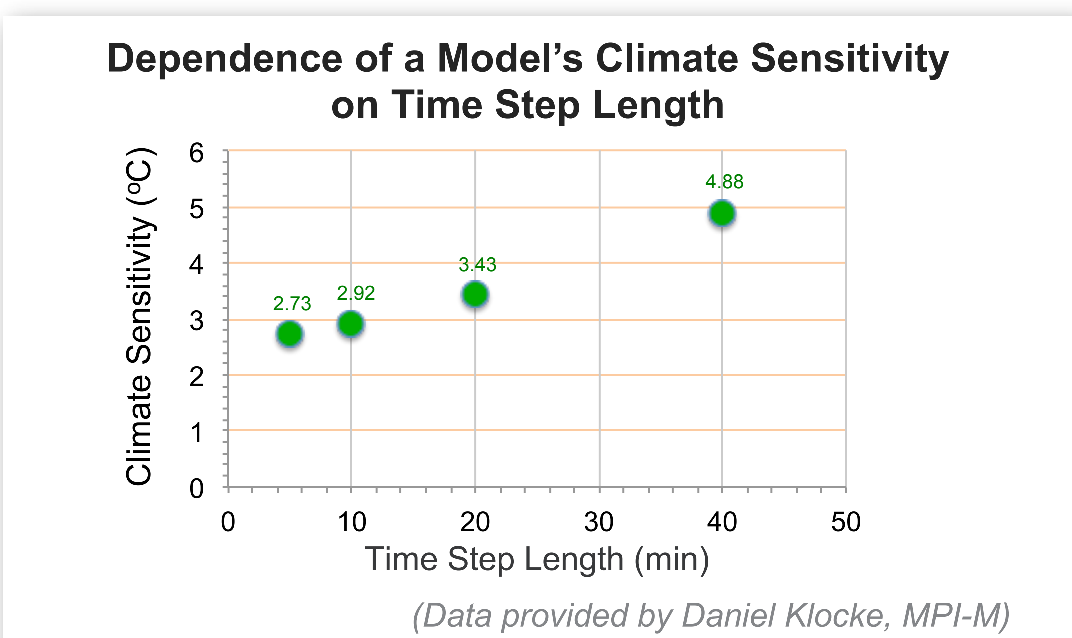
Risks in front of us.

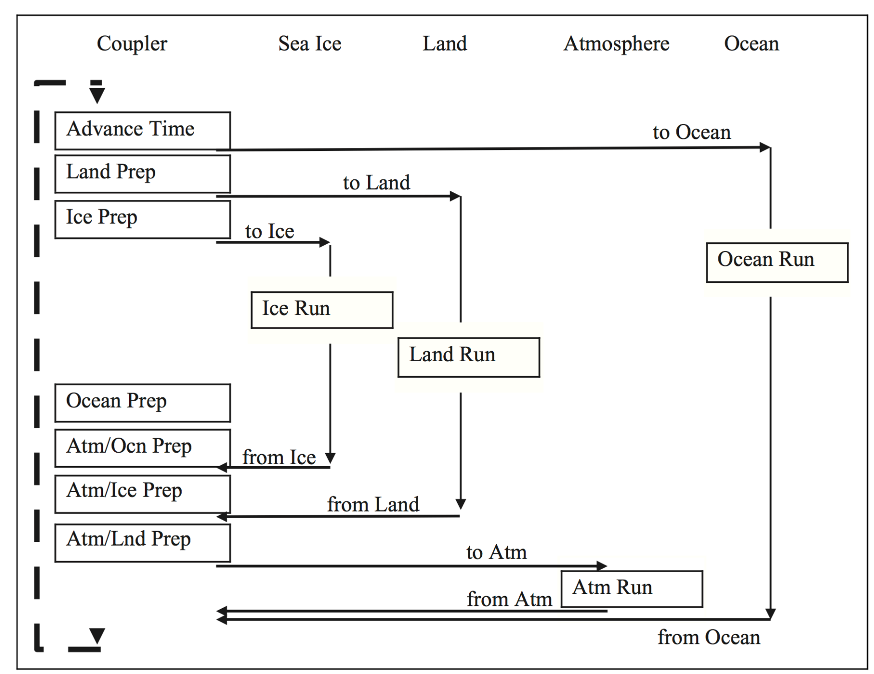
Disruption in the computing ecosystem ….

- Cori / KNL
- Summit / GPU
- Cloud computing / Xeon
- Aurora / World's fast machine in 2021 / ??
- Los Alamos / ARM?

How do we get ~1M lines of code to be performant on such a wide diversity of computing architectures?

Fragility due to being overwhelmed by model complexity ….





If I had it to do over again ….

earth scientists

An immersive, self-contained, and complete description of the ESM under revision control.

design document
Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

prototype algorithm
A realization of the design document as software.

unit and test cases
Demonstrations that the software doing what is described in the design document (verification) and appropriately represents some physical process (validation).

auto-generate

computational scientists

production software
An HPC performant version of the prototype algorithm.

simulation data
Output (usually in the form of netCDF files) produced by executing the software.

If I had it to do over again ….

Open development from day 1. (“Head” is visible to world.)
Clearly define and maintain a very high bar for the consideration of pull requests.
Entrain the best ideas (and developers) from around the world.
Work with journal editors to connect manuscripts to a hash (DOI).
(Imagine a world where all ESMs were open development …..)

Elements of a pull request

<u>design document</u>
Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

<u>prototype algorithm</u>
A realization of the design document as software.

<u>unit and test cases</u>
Demonstrations that the software is doing what is described in the design document (verification) and appropriately represents some physical process (validation).

Adopting more professional software practices will improve our productivity and impact.

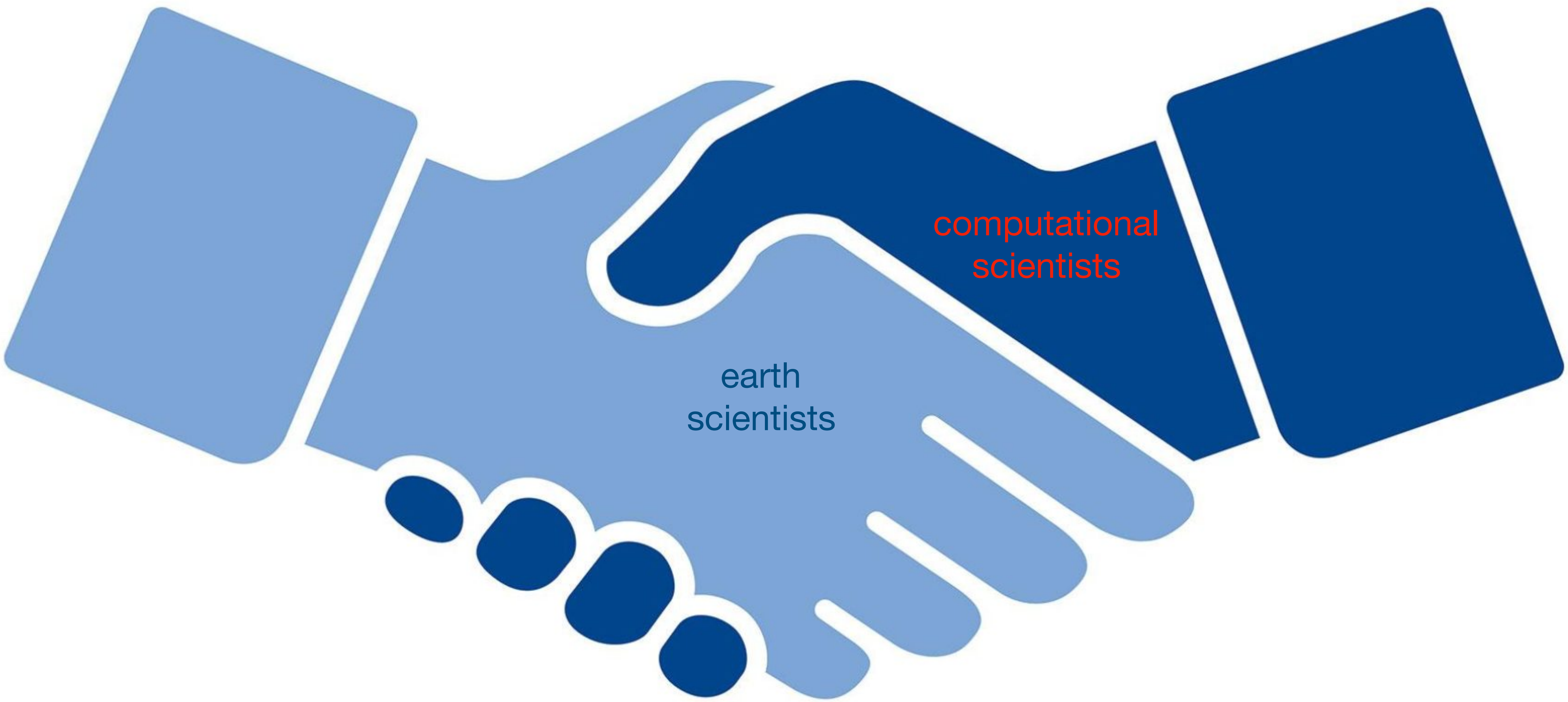
Changing “how we do business” has the potential to reinvent the ESM enterprise.

If I had it to do over again ….

Embrace coupled data assimilation from the outset.

Initialization of High-Resolution Earth System Models
Rockville, Maryland
April 9-10, 2018

Jointly produced by DOE and NOAA.


computational
scientists
earth
scientists

design document

Specification: a textual description of the why, what, and how of some aspect of an ESM (physical process, computational algorithm, infrastructure, etc.)

prototype algorithm

A realization of the design document as software.

unit and test cases

Demonstrations that the software doing what is described in the design document (verification) and appropriately represents some physical process (validation).

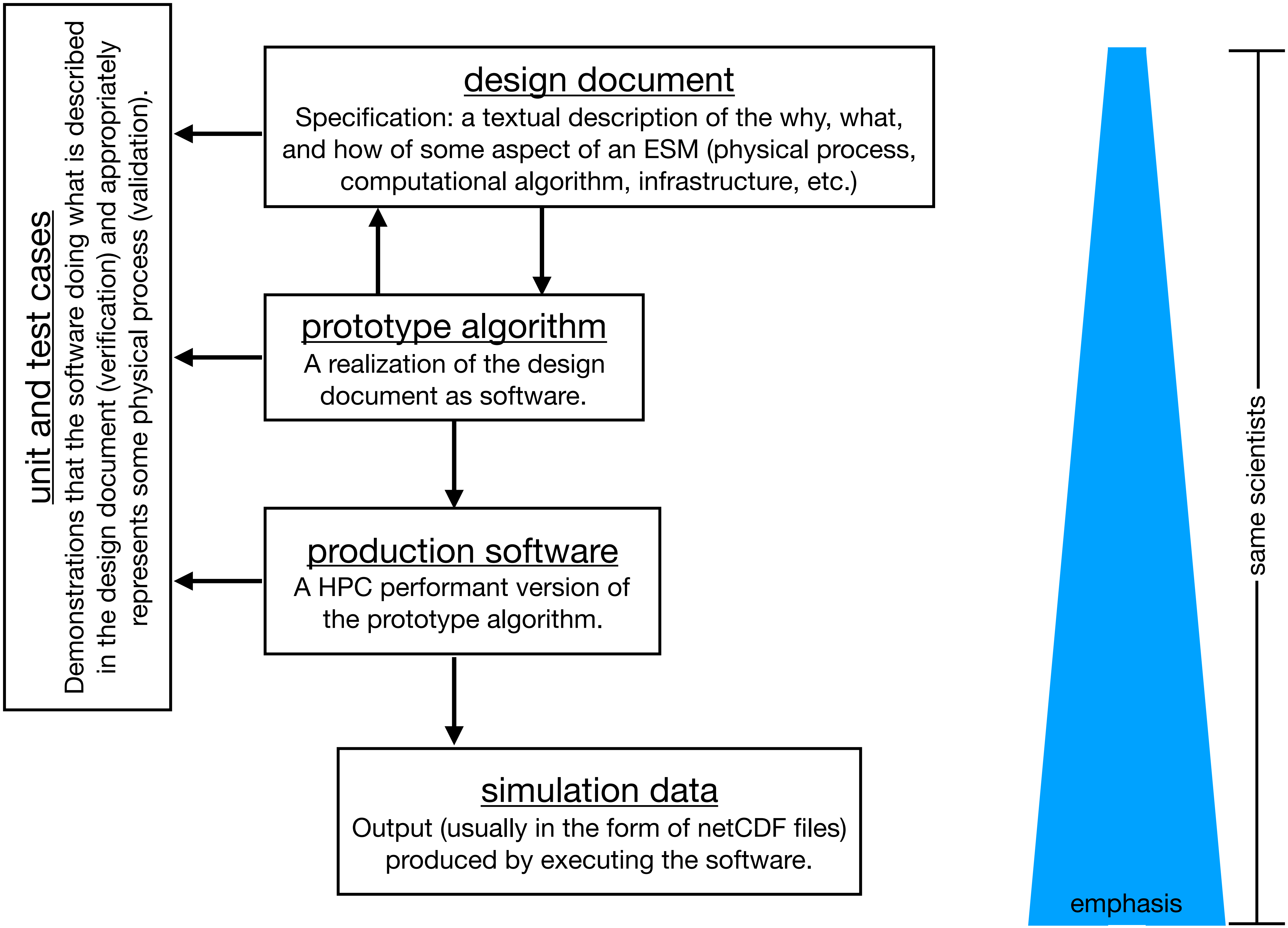
production software

An HPC performant version of the prototype algorithm.

simulation data

Output (usually in the form of netCDF files) produced by executing the software.

A better way to do business

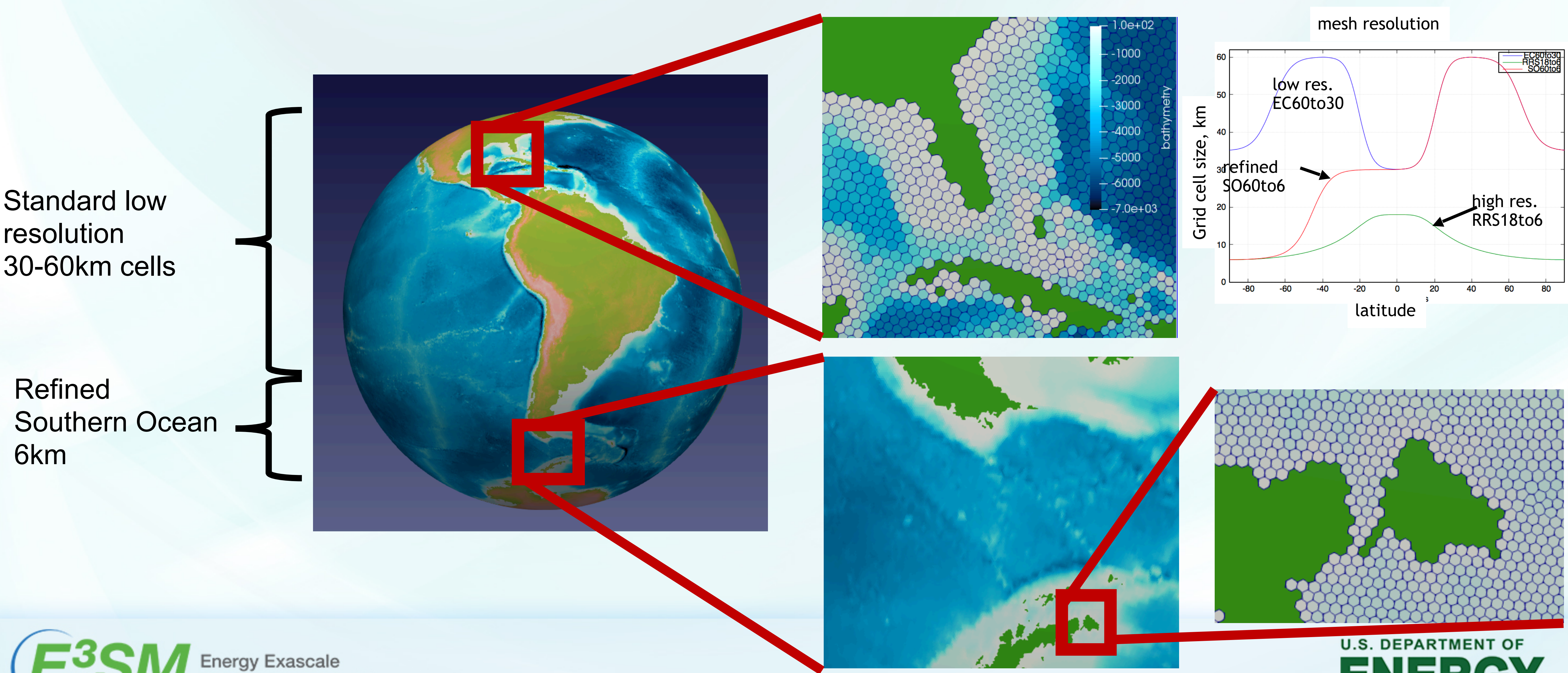


Southern Ocean refined 60 to 6km mesh

A variable-resolution mesh with 6 km grid cells in the Southern Ocean

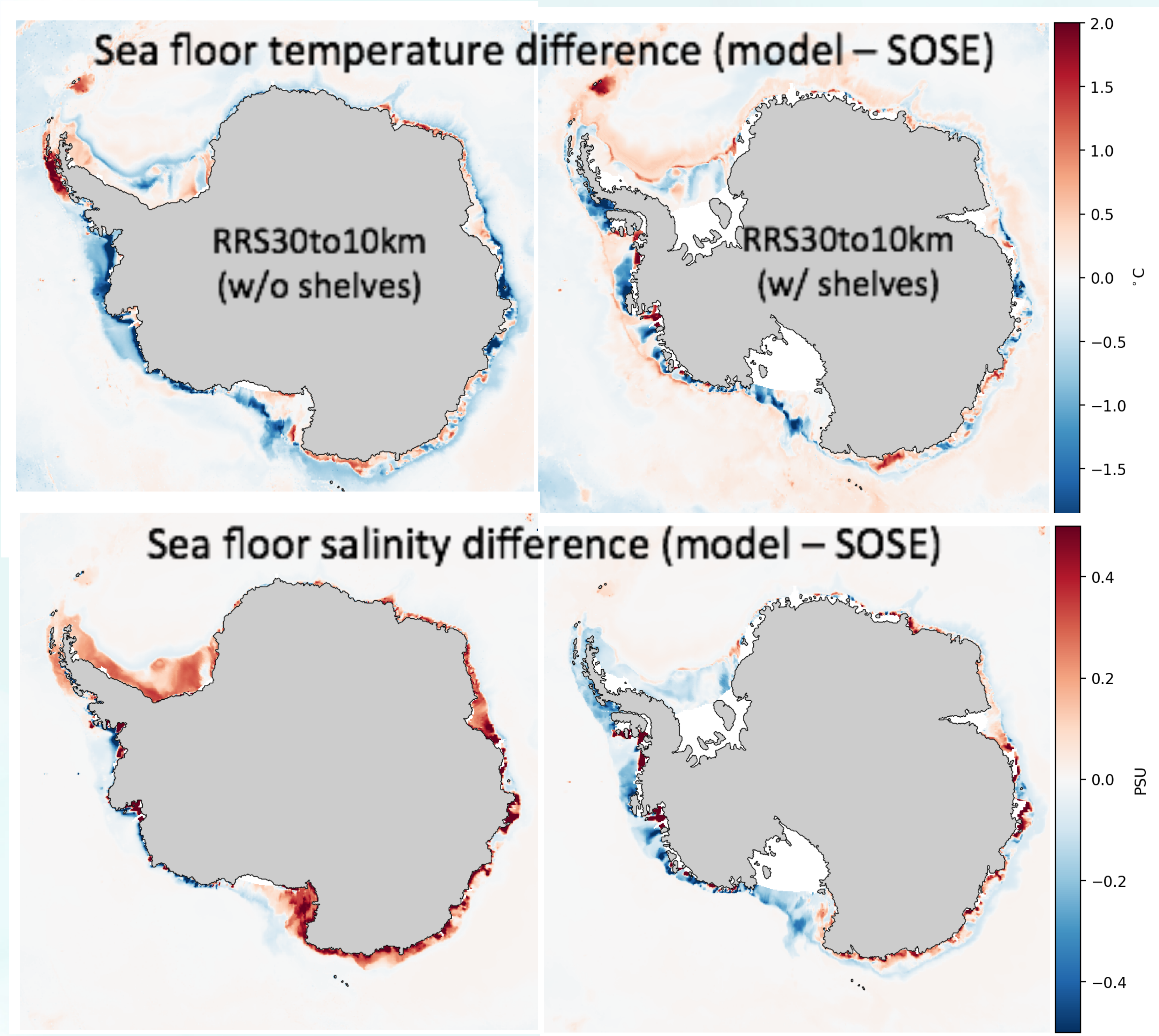
- Effects of eddies are well resolved in Southern Ocean
- Cost of ocean is ~3.5 times less than global high res.
- Domain includes Antarctic ice shelf cavities

Resolution	Horizontal grid cells
Low: EC60to30	0.2 million
Refined: SO60to6	1.1 million
High: RRS18to6	3.7 million



Ice cavities are included in our coupled simulations.

Note: color bar measures bias.



Overall, the simulation of T and S on ocean floor is improved by including ice shelf cavities.