Component Level Testing in a Hierarchical Architecture

Tom Clune[1]

Natalie Patten[2], Ben Auer[1,3], Arlindo da Silva[1]

[1] NASA Goddard Space Flight Center
[3] Space Systeml, LLC.
[2] SSAI, Inc.

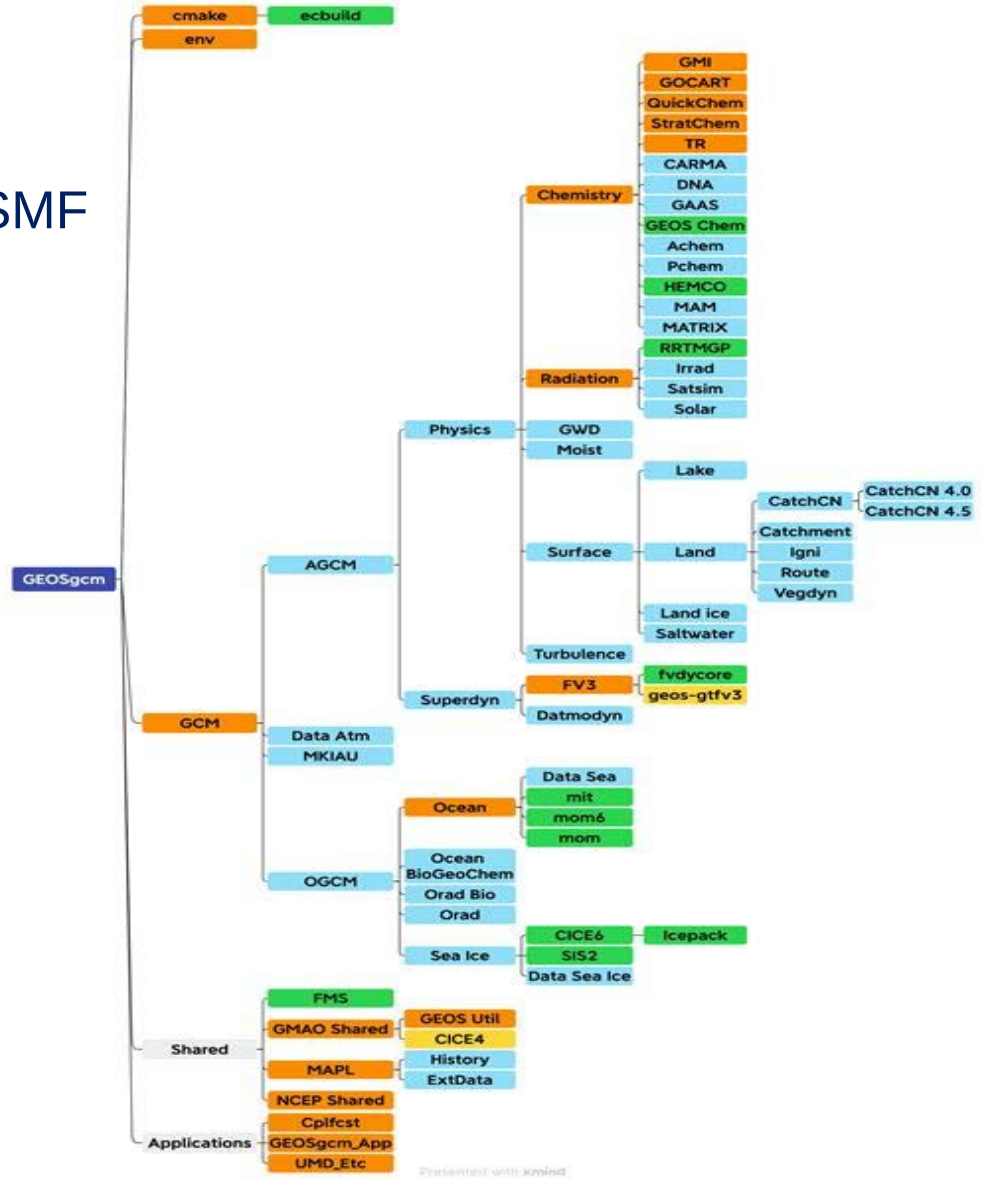
Background and Motivation: GEOS

GEOS is a highly configurable Earth System Model

- Hierarchical system of modular components built on top of ESMF
- Coupling based on standardized ESMF abstract data types
  - ESMF_State, ESMF_Field, ESMF_FieldBundle, ESMF_Geom, …
- Each component has an ***import*** and an ***export*** state
  - And optionally an ***internal*** state
- The MAPL infrastructure layer provides common services:
  - Allocation of fields and states
  - Connections between components
  - *Automatic checkpoint & restart*

I lead the software infrastructure (SI) team:

- Develop MAPL
- Port, profile, optimize, and **test** GEOS



Background and Motivation

Gratifying/rewarding

Unpleasant/discouraging

Writing new code (procedures, classes)

Modifying reasonably clean code

Writing unit tests

Debugging ~~my own~~ clean code

Debugging ~~other people's~~ legacy code

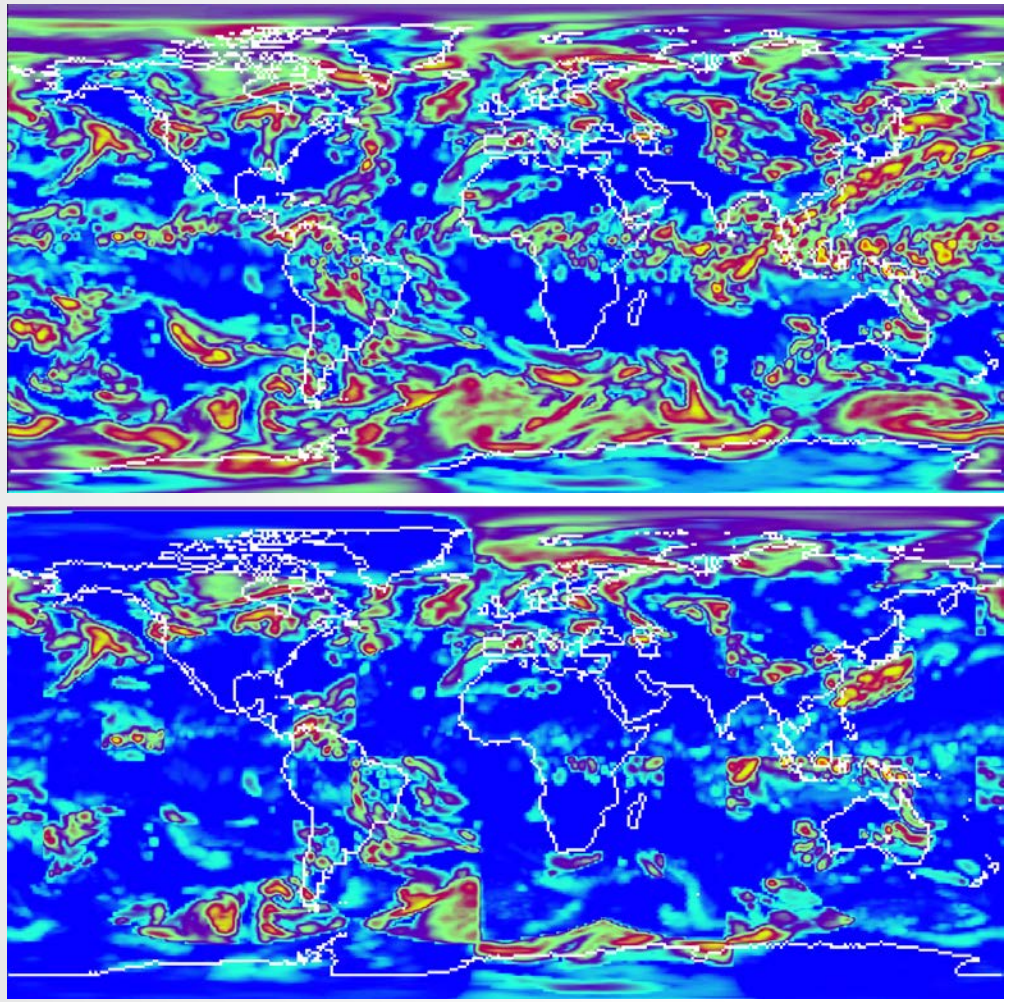
Modifying poorly written blobs of code

Finding workarounds for other software (esp. compilers)

> *To me, legacy code is simply code without tests.*
>
> *- Michael Feathers*

Story 1: Mysterious decomposition imprint

Ice Water Path



C90 resolution; 4x4 face decomposition

Defect: results show imprint of domain decomp

Details:

- Very weak signal – mostly visible in “obscure” qtys.
- Requires non-default moist physics option
  - Current GEOS CI is blind to this
- Signal disappears with -O0 (Compiler bug?)

Status: Unresolved

- Multiple people have spent weeks on this.
- One took 5-week vacation. *Coincidence?*

Story 2: Layout regression failure

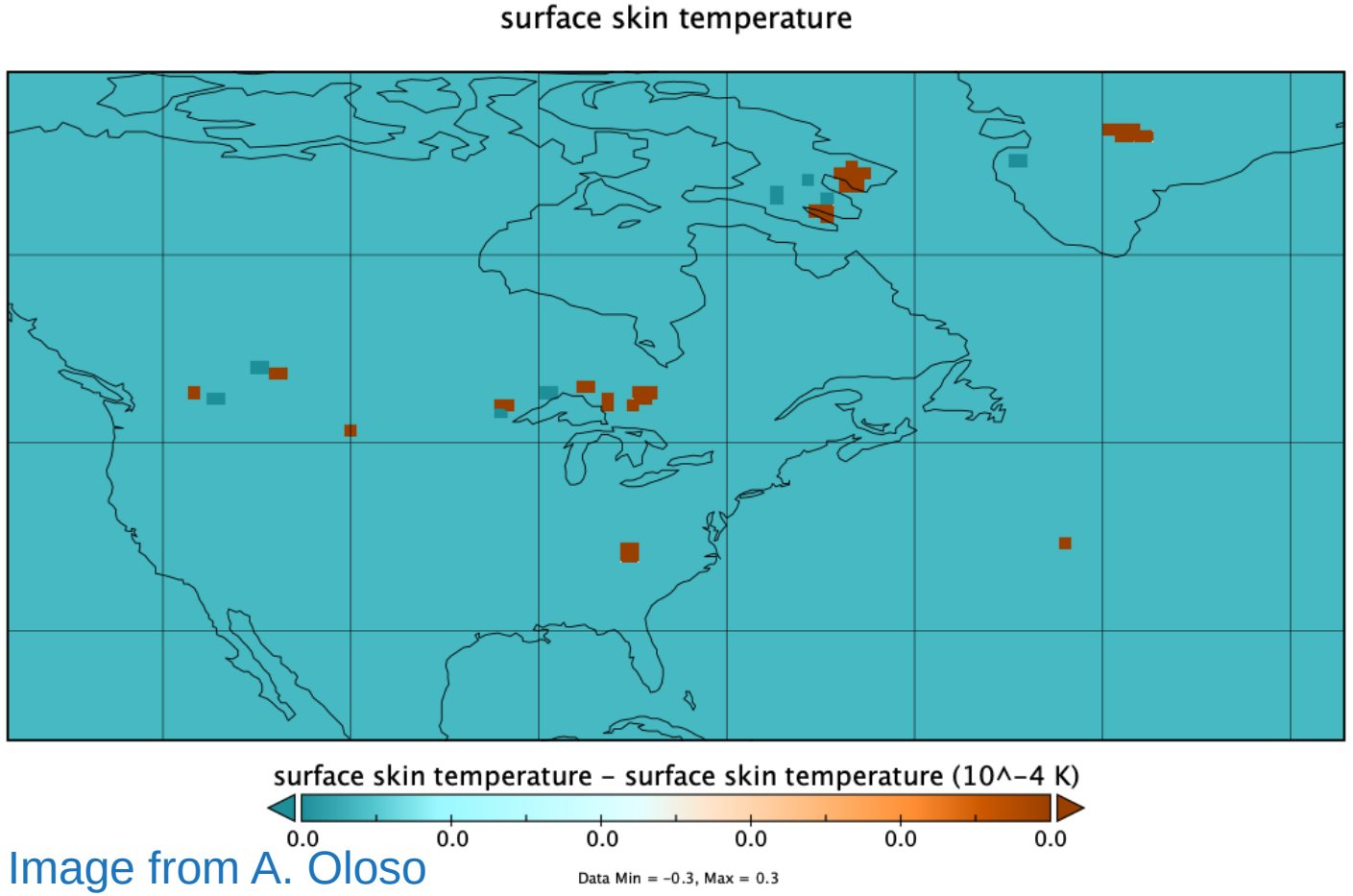


Image from A. Oloso

Defect: layout regression

Details:

- Does not fail for coarse resolutions
  - Requires $\Delta \leq \sim 50\ km$
- Unnoticed by CI for **months**! (200 km)
- Initial divergence is roundoff at isolated points in the domain.

Status: Unresolved

Existing GEOS testing: System Tests

Warner Bros Entertainment

Types of system tests

- Does it build? Does it run?
- Restart reproducibility (1 + 1 = 2)
- Layout reproducibility (3x4 = 6x2)
- Regression – no change to results
- ~~Scientifically equivalent results~~

What do failures tell us?

- Not much
- Maybe location for build/run failure

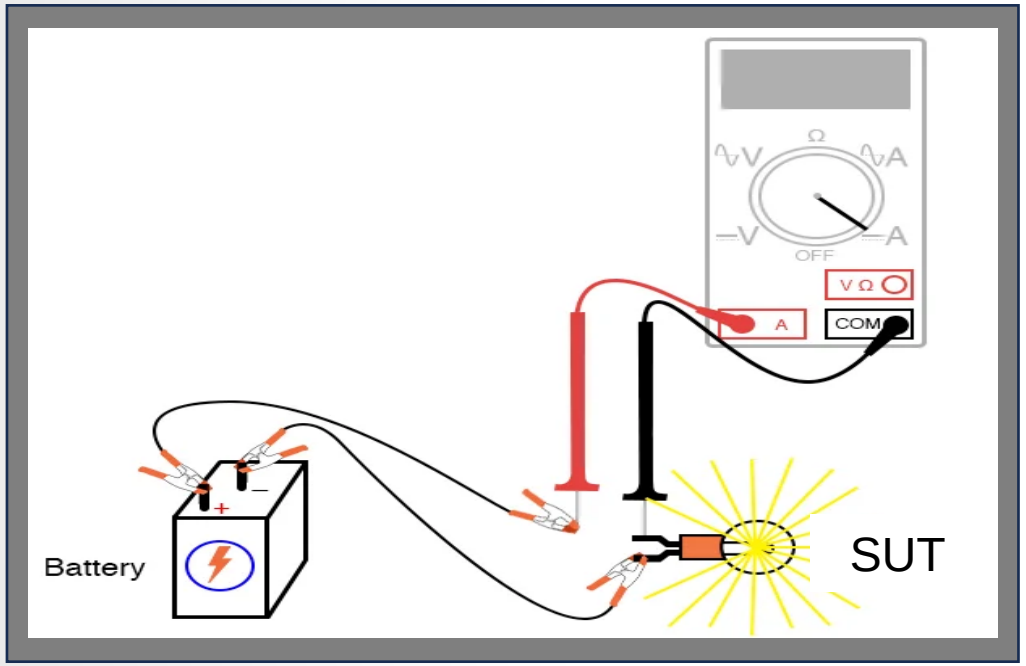
Pros

- Relatively good code coverage
- Easy to update as model evolves

Cons

- Weak isolation of defects
- Relatively resource intensive (as CI tests go):
  - Compute, memory, BC's, IC's

Existing GEOS tests: Unit tests



Unit test:

- Does a procedure (SUT) produce expected output for specific inputs

What do failures tell us?

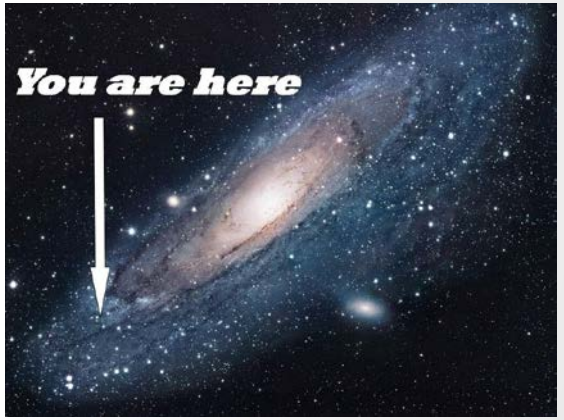
- Defect in the SUT covered by the test

Pros

- Excellent isolation of defects.
- Fast execution (1000's per second)

Cons

- Weak coverage (in practice)
- Labor intensive
  - Hard to write
  - Must be maintained



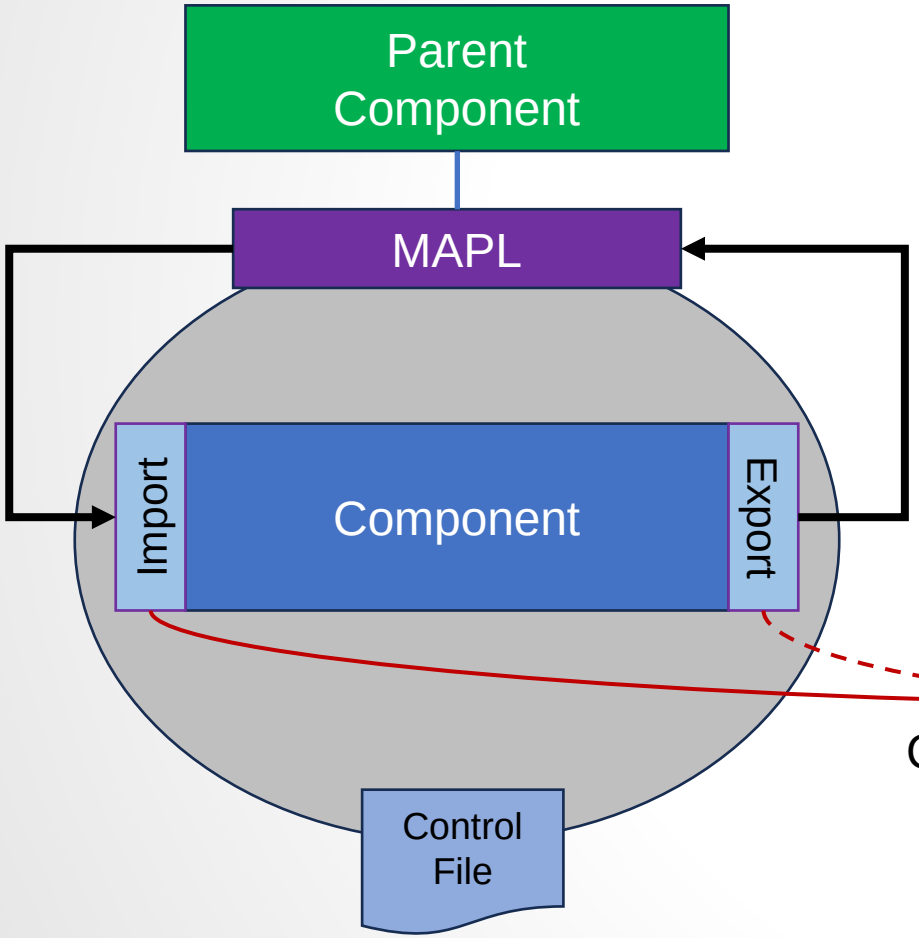
How about component-level tests?

- Collective coverage comparable to that of system tests
- Regression failures localized to specific components
- Resource efficient
  - Only test altered components
  - Fast execution, etc.
- **Sidesteps the weakest aspects of system tests and unit tests**

A generic test driver for components

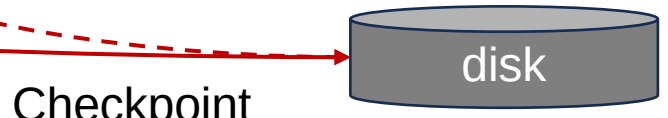
- All components have the *same* interface
- Should be straightforward to develop a generic testing app that:
  1. Expects an initial import state
  2. Runs component
  3. Compares resulting export state against expected import state
- How to obtain imports and expected exports?
  - Custom (per-component) data provided by component maintainer? (*Not holding my breath*.)
  - *Capture data from trusted execution of full system?*
    - Already have state-level checkpoint/restart layer that seems well suited.
    - Practical – can leave scientists out of it
    - Somewhat analogous to approach used by tools such as KGEN
- Have wanted to try this for many years, but … other priorities …
  - Along came an intern (spring 2023)

Leaf component: capturing functional characteristics



MAPL infrastructure layer

- Wraps execution of child run methods
- Has access to import and export states
  - Can even access internal state
- Provides component-level checkpoint/restart
  - *Minor extension: support export state*



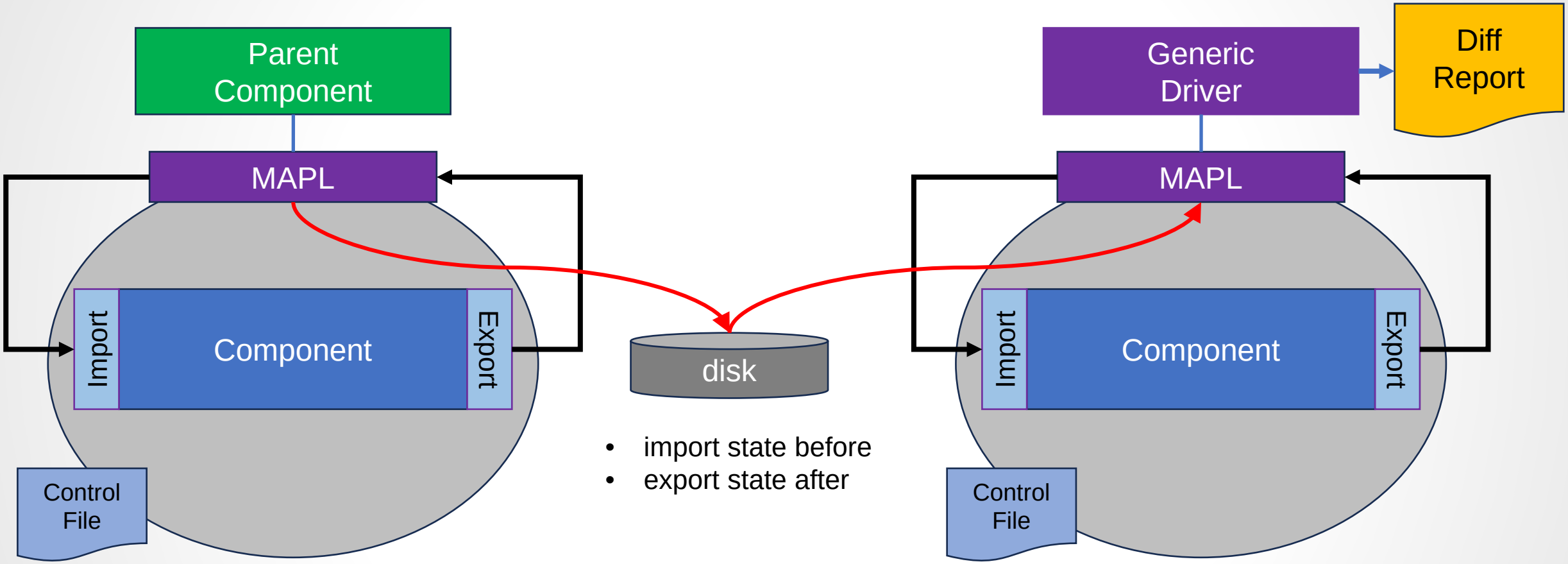
Checkpoint

Archived:
- import state before
- export state after

$$i \xrightarrow{F} e$$

Component as a function

Testing simple leaf component



- import state before
- export state after

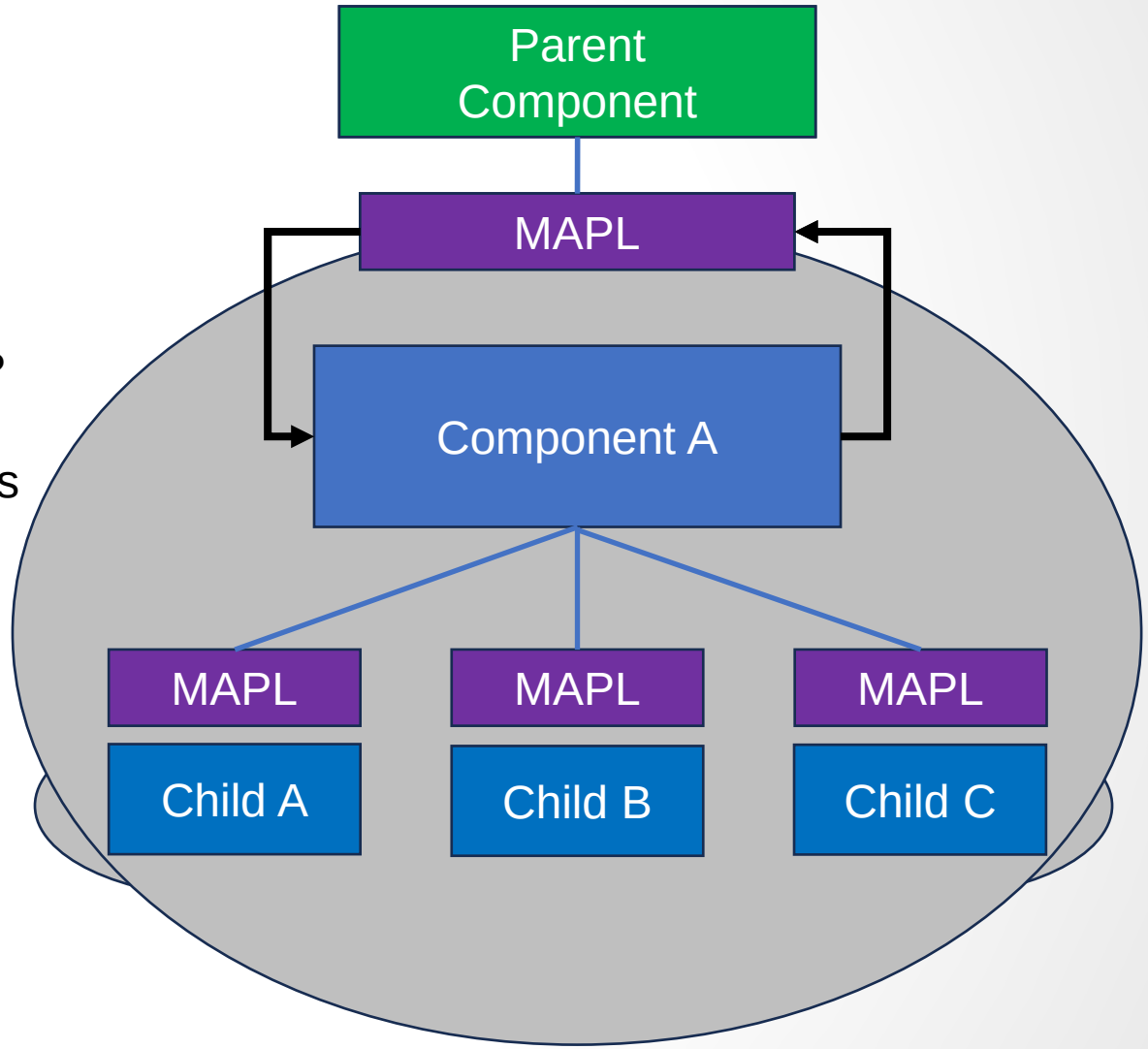
Capture Mode

Test Mode

Compare expected export state from disk against export state from test

Non-leaf components

- Treat children as part of component?
  - **Pro**: no extra machinery needed
  - **Con**: Does not completely isolate defects to changes in targeted component.
- Isolate component from children (dependencies)?
  - Capture children export states when run
    - Note: each might be called multiple times
  - Use *data mocks* during replay
  - **Pro**: proper isolation of defects
  - **Cons**: extra machinery needed
- Note: similar difficulties arise with callbacks



Details details …

Idealization: exports (e) are a function of imports (i)

$$i^c{}_{t=n} \xrightarrow{F^c} e^c{}_{t=n+1}$$

Most components have internal state (q)

$$(i^c; q^c)_{t=n} \xrightarrow{F^c} (e^c; q^c)_{t=n+1}$$

Some components modify imports (cheat!)

$$(i^c; q^c)_{t=n} \xrightarrow{F^c} (i^c; e^c; q^c)_{t=n+1}$$

Structure of states depends on coupling context: $c^* \neq c$

$$(i^{c^*}; q^{c^*})_{t=n} \xrightarrow{F^{c^*}} (i^{c^*}; e^{c^*}; q^{c^*})_{t=n+1}$$

Unfortunately, context matters

Structure of import and export state depends in part on coupling context within wider system:

- Some exports are “optional”. Only allocated & computed if connected to another component.
- Some exports are re-exports from children. Details not specified inside component.
- Some import bundles are elastic – servicing other components on request:
  - E.g., DYN component has an import field bundle for tracers. Contents determined by advection requests in other components.
- Parent component is allowed to modify child component’s control flags and parameters.

- **Difficulty: restart files are not used to define structure of component states**

Current status

Prototype developed by intern, Natalie Patten, in Spring 2023

- Added MAPL runtime option to capture component states before and after Nth time step
- Added required capabilities to MAPL checkpoint/restart layer
- Created small python layer to extract “thin” subset for column components
  - E.g., just keep a 4x4 set of columns out of a high-res run
- Created new standalone component test driver
  - Initialize and run component from DSO and archived ESMF states
  - Writes computed states back to disk
- nccmp used to compare expected and found results.

Works for simple leaf components

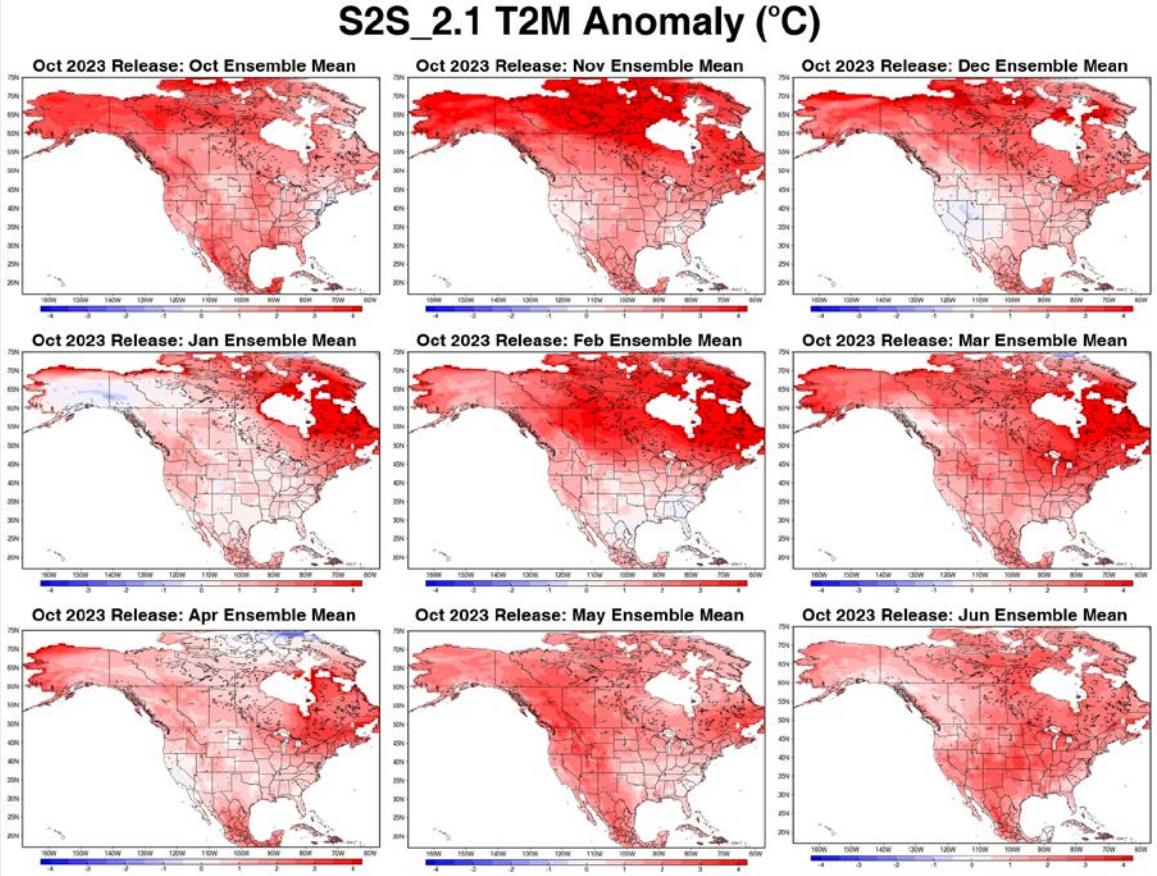
Summary

- Components based upon standard ESMF interfaces are conducive to generic CI testing
- Basic capability has been demonstrated with MAPL2
- Lots of difficulties encountered
  - Many remain
  - Some may require significant infrastructure changes
- MAPL3 will be a major redesign (motivated by somewhat unrelated concerns)
  - Expect (hope) to include necessary bells & whistles to robustly handle all GEOS components

Glossary

- ESMF Earth System Modeling Framework
- GMAO Global Modeling and Assimilation Office
- GEOS Goddard Earth Observing System model
- MAPL Modeling and Applications Layer

Horror Story 1: Atmosphere-Ocean General Circulation Model and Data Assimilation System Version 2 (GEOS-S2S-2)



S2S

- Production system ran for 180 simulated years without issue.
- Continuation runs with same configuration reliably failed
- Ultimate cause: negative salinity
- Difficulties
  - Routine checking implemented but required 2 switches. Only one was activated.
  - Debug flags that should help were not propagated to ocean (external component)
  - Fortunately resolved in late October.

Along comes an intern …

Have wanted this functionality for years, but priorities never aligned

- Add option to checkpoint procedure to save *export* state
- Instrument MAPL run_child() with new control logic
  - If capture mode and on targeted timestep do super-checkpoint
    - Import and Internal state before
    - Internal and Export state after
- Write MAPL driver program
  - Config file to specify
    - DSO of targeted component
    - Path to super-checkpoint
  - Initialize & Run component from saved super-checkpoint
  - Checkpoint “new” internal and export states
- Write python layer to diff saved vs new states

Phase I (support simple leaf components) was expected to be straightforward. Lots of surprises encountered.

Phase II was meant to be extend phase I to include support for child components and callbacks.