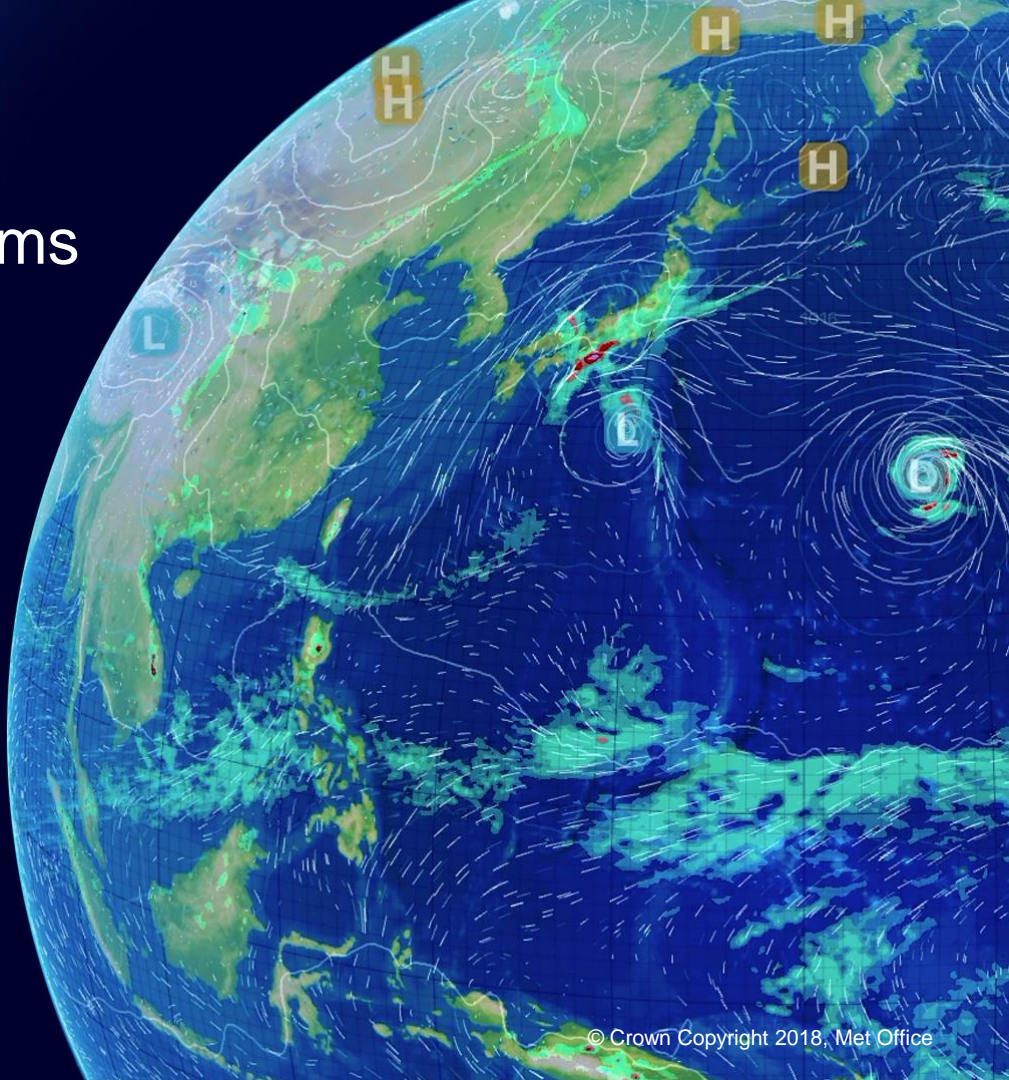


Coupled Model Control Systems Problems = opportunities!

Richard Hill

in discussion with Mike Hobson, Harry Shepherd,
Jean-Christophe Rioual, Matthew Hambley,
Marc Stringer, Mirosław Andrejczuk....
and many more.

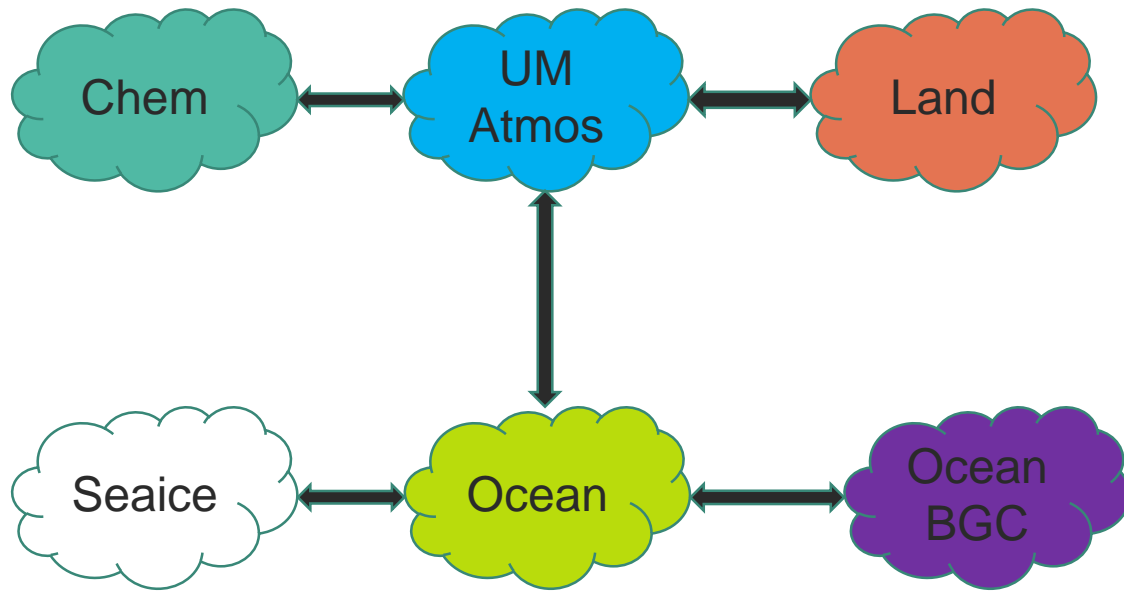
CW2020, Toulouse



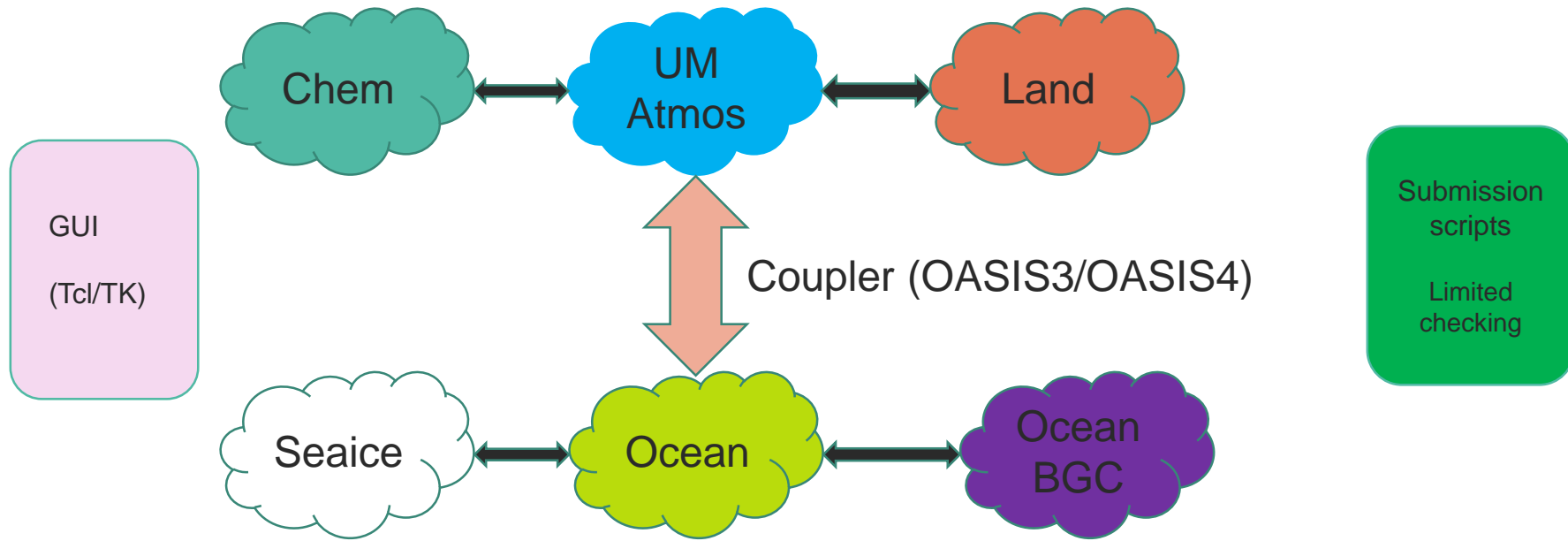
Our Coupled Model Control System - Outline

- How we got here
 - Current systems
- The difficulties
 - Who's responsible for what
- An opportunity for change
 - The way ahead
 - or at least ... **MY** thoughts on the way ahead

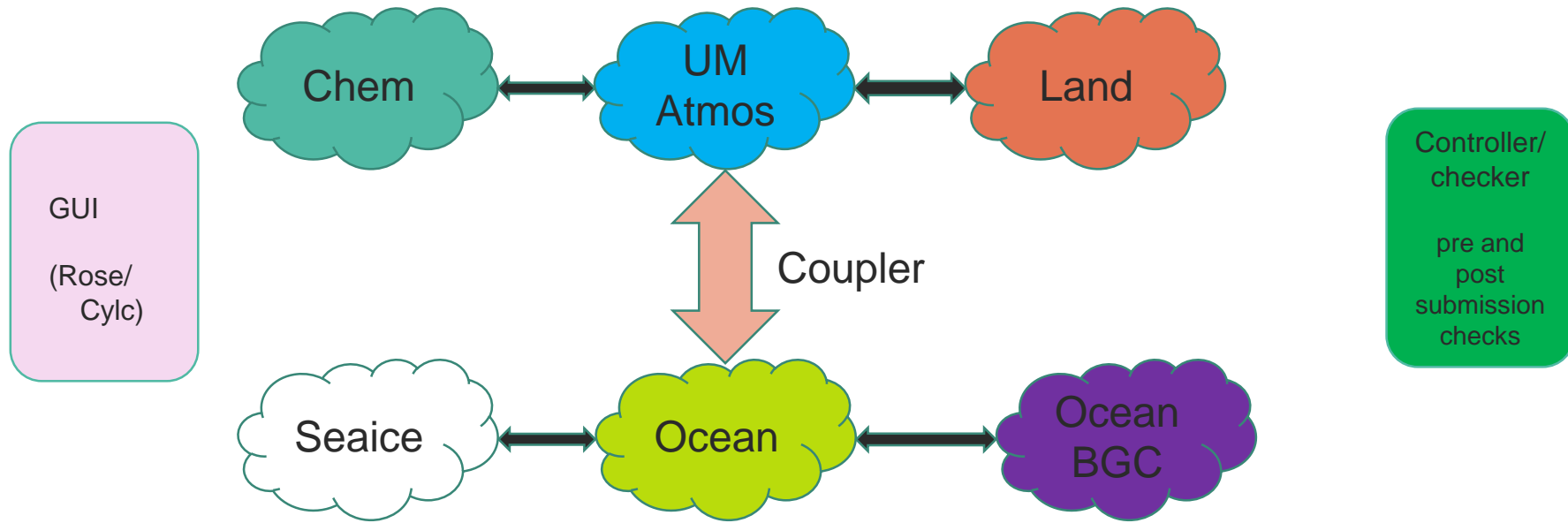
Evolution to Current System: From single exec with atmos-ocean (~early 1990's)



1st MPMD system (~early 2000's)



Current System (~early 2010's)



Coupled Model Control

- Configuration, build and deployment systems evolved from UM-specific solutions
 - Thus, generally UM-centric
 - At time of first MPMD coupling, GUI and control systems based on assumption that the UM was the only component model!
 - Components we don't own (e.g. NEMO, SI3, CICE, Wavewatch III) – initially no GUI and metadata... substantial effort put in to set up and maintain
 - Initially driven by the needs of coupled models.
 - Now adopted by all stand-alone configurations too!

Coupled Model Control

- Coupler and XIOS
 - Centrally managed standard builds
- We adapted to cope with OASIS couplers and non-UM components (NEMO, CICE, Wavewatch III, MEDUSA, XIOS etc.)
 - **Not redesigned from scratch** with these components in mind.
- It works!
 - **But.....**

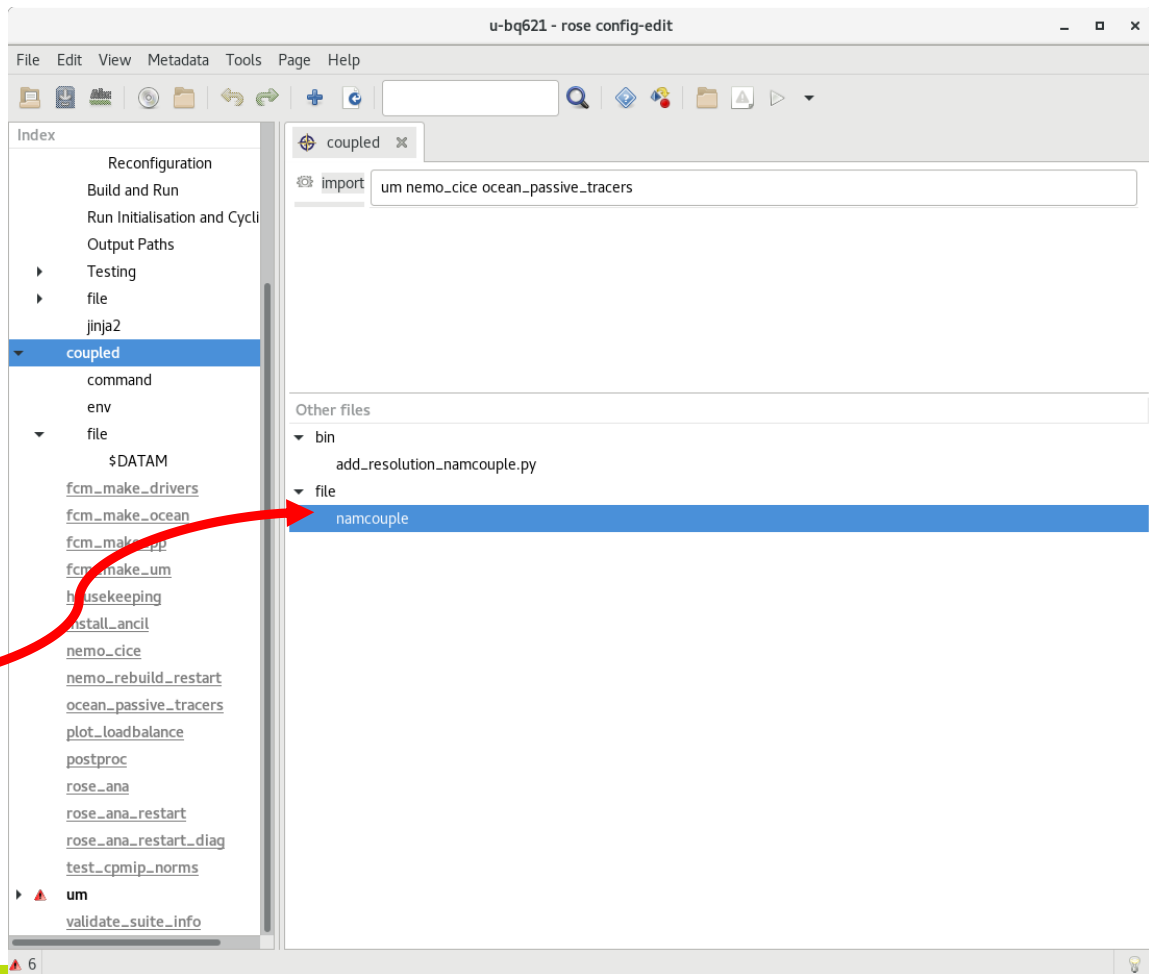
Coupled Model Control

- Data must be duplicated in multiple forms to facilitate coupling. E.g. In a simple atmosphere-ocean model:
 - Atmos must know what fields to couple, when, how those fields relate to internal model variables.
 - Ocean must know all the above – separately in its own input format
 - Coupler must know the above – separately in its own input format (namcouple files).
 - Lots of assumptions for **the user** here!

At the heart of the job definition is the critical control file....

The **pre-generated** coupler (OASIS3-MCT) control file.

Configuration dependent details.



The OASIS3-MCT “namcouple”

Note:

Field names. Component models must know these. How?

Coupling frequency.

(per field)

Component models must know these. How?

The user must assume!

```
namcouple - /net/home/h01/frhh/roses/u-bq621/app/coupled/file/
File Edit Search Preferences Shell Macro Windows Help
/net/home/h01/frhh/roses/u-bq621/app/coupled/file/namcouple byte 3927 of 30566 L: 102 C: 44
100 #####
101 # OCEAN --->> ATMOS
102 # -----]
103 #####
104 #
105 # Ocean SST
106 #
107 #####
108 # TRANSDEF: OCNT ATMT 1 25 #####
109 model01_O_SSTSST ocn sst 1 10800 1 atmos_restart.nc EXPORTED
110 362 332 96 72 tor1 atm3 SEQ=+1
111 P 2 P 0
112 #
113 MAPPING
114 #
115 rmp_tor1_to_atm3_CONSERV_FRACAREA_1st.nc
116 #
117 #####
118 # TRANSDEF: OCNT ATMT 2 41 #####
119 model01_OTepIce_cat01 oicet01 469 10800 1 atmos_restart.nc EXPORTED
120 362 332 96 72 tor1 atm3 SEQ=+1
121 P 2 P 0
122 #
123 MAPPING
124 #
125 rmp_tor1_to_atm3_CONSERV_FRACAREA_1st.nc
126 #
127 #####
128 # TRANSDEF: OCNT ATMT 3 42 #####
129 model01_OTepIce_cat02 oicet02 469 10800 1 atmos_restart.nc EXPORTED
130 362 332 96 72 tor1 atm3 SEQ=+1
131 P 2 P 0
132 #
133 MAPPING
134 #
135 rmp_tor1_to_atm3_CONSERV_FRACAREA_1st.nc
136 #
137 #####
138 # TRANSDEF: OCNT ATMT 4 43 #####
139 model01_OTepIce_cat03 oicet03 469 10800 1 atmos_restart.nc EXPORTED
140 362 332 96 72 tor1 atm3 SEQ=+1
141 P 2 P 0
142 #
143 MAPPING
144 #
145 rmp_tor1_to_atm3_CONSERV_FRACAREA_1st.nc
146 #
147 #####
148 # TRANSDEF: OCNT ATMT 5 44 #####
149 model01_OTepIce_cat04 oicet04 469 10800 1 atmos_restart.nc EXPORTED
```

Currently **the user** needs to deal with all the following

Build
configuration

Scientific
configuration

Cross check
coupling and
dump frequencies

What do I need
to rebuild?

Configure
science

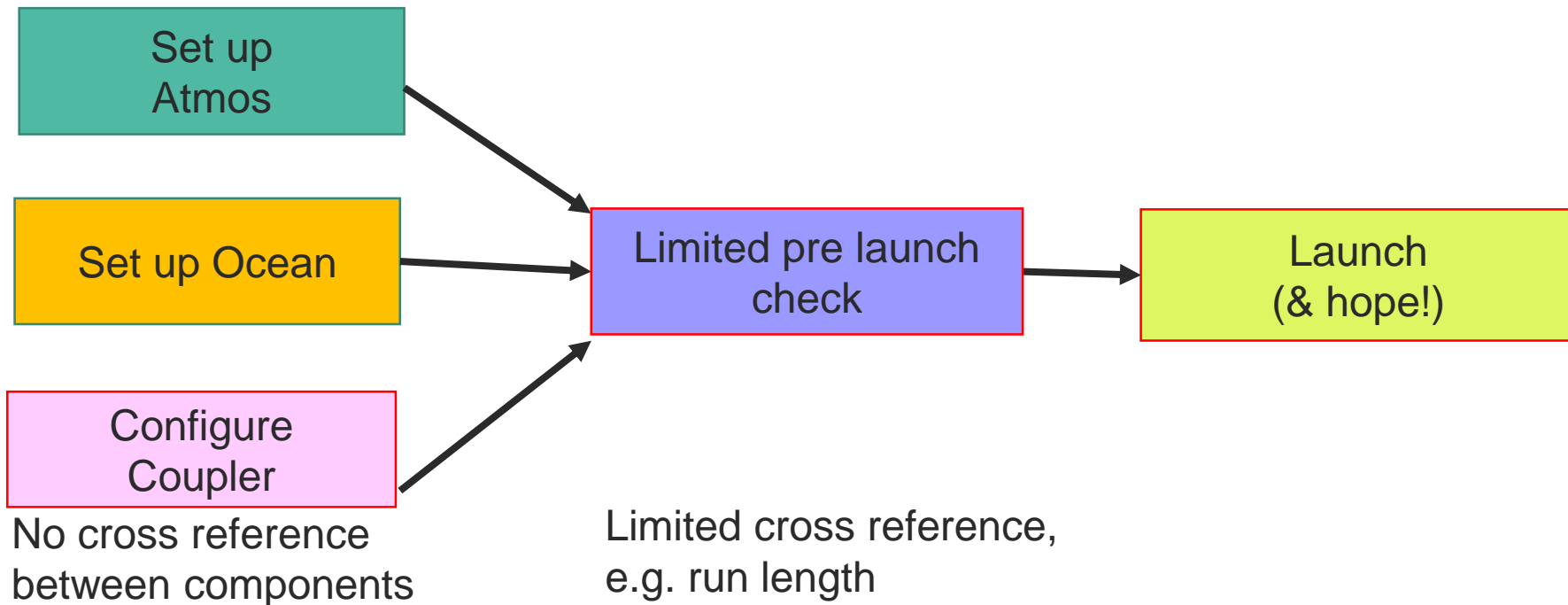
Model timestep and
coupling data
consistent across
components.

Compatible compiler
and library details
from central coupler
installation

Define coupling
exchanges

Generate
coupling controls

How does it currently fit together?



What does all this mean?

- Managing MPMD configurations is very difficult!
 - Models may be prone to error
 - **Users** have to acquire **a lot of incidental knowledge** in order to set up and run a viable system
- Can lead to wasted HPC, storage and time if things aren't right

Goal is to improve this, while catering for increasing flexibility... But how?

Once upon a time....

- All these worries led me to believe that the OASIS namcouple file should be treated as the starting point!
- i.e. treat the namcouple file as **THE SINGLE COMMON REFERENCE** for all components to obtain details of coupling data exchanges, remapping weights, maybe even libraries to be used in compilation.
- But normal namcouple files don't quite contain all the details needed for that!
 - So we added our own special "extra" data.
 - Allowing components to interrogate the namcouple to see what to do.
 - This works well enough but it's not without its problems
 - E.g. managing different namcouple files for a rapidly expanding zoo of model configurations
 - We're still ultimately dependent on a manually edited text file.

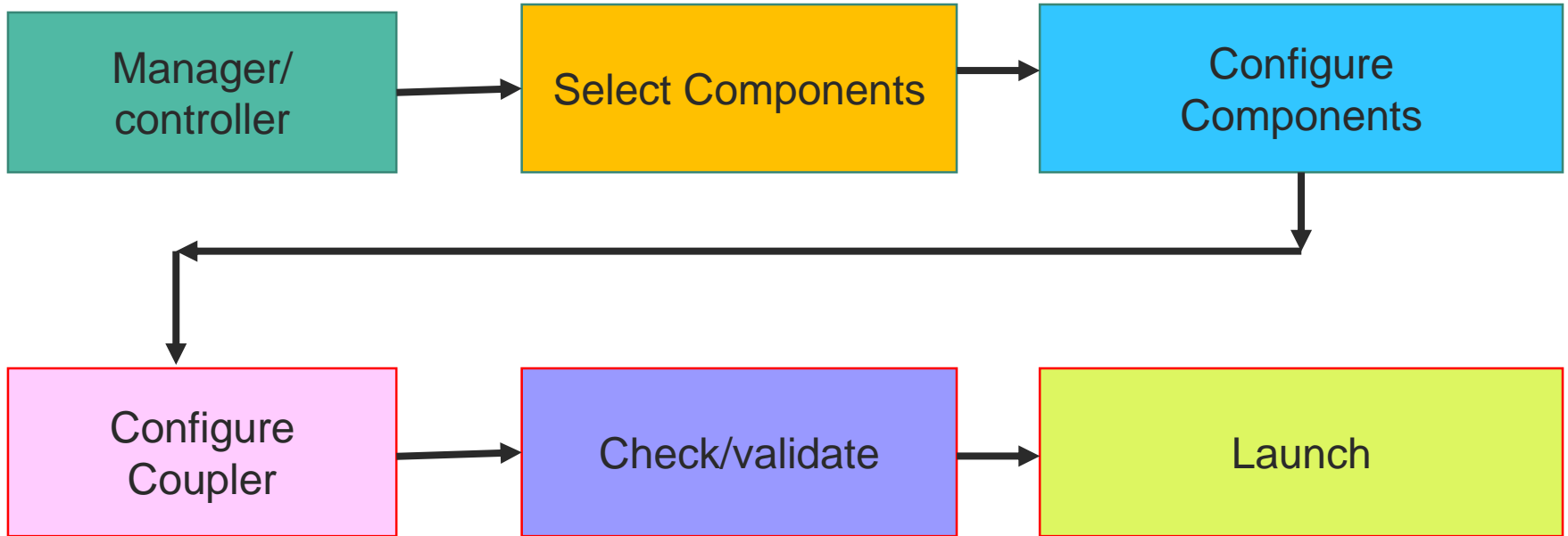
As if life wasn't complicated enough

- We now have to think about **Next Generation Modelling Systems**
- Models being created/redesigned for next generation architecture.
- In particular the UM is to be replaced with a brand new model, LFRic
 - Cubed sphere mesh, completely new dynamics code.

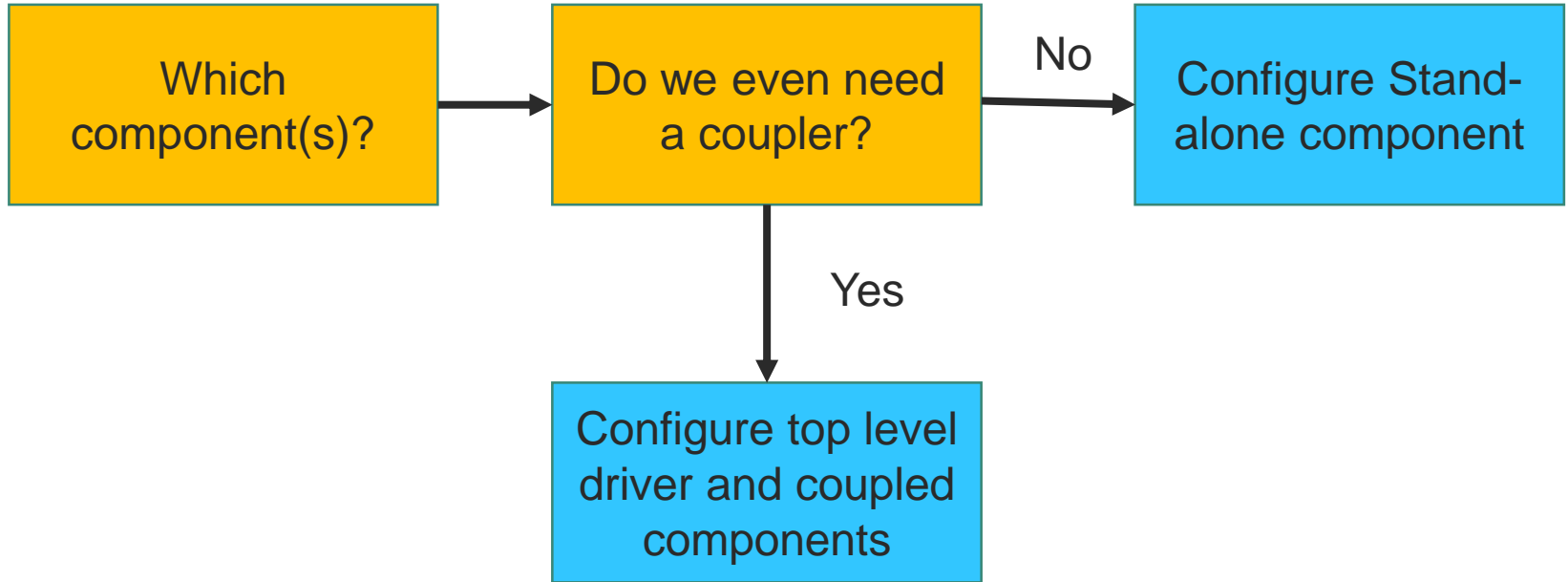
An opportunity for change?

- Next Generation Modelling Systems
 - Importantly, LFRic requires a **brand new control mechanism (which is still subject to development!)**
 - An opportunity!:
 - Move away from retro-fitted coupling crowbarred into a component-centric system
 - Design the control/driver system from the top down, with LFRic, and other components, as just that, optional components in an all-seeing model manager/controller.

What might this look/behave like?



Stage 1: Select components




Stage 2: Configure components

- Scientific settings
 - Aside from all the scientific settings (**which is the actual job of the scientist!**), these define what's viable in terms of coupling.
 - So **the user** can then go on to select coupling details
 - Which fields to couple
 - Which are incoming
 - Which are outgoing
 - Instantaneous/time mean coupling fields
 - Target component
 - Coupling frequency
 - Regridding mechanism

Stage 3: Configure Coupler

- The LAST thing to be done.
- Don't define the coupling control file at the START of the model definition process and worry about getting all components to conform to that.
 - Generate the namcouple file at the end of the process...automatically!
 - (i.e. make IT conform to the model configuration/details)
- **So I've changed my mind!**

I want to generate the coupler control file, etc. automatically.

- But this means **we** have to build in smart functionality to generate coupler control file (compatible with any supported coupler).
- Extra responsibility for someone! 
 - But that responsibility already exists – it currently mainly lies, as repeatedly indicated, with **the user**
 - The good news is we already do this for prototype models (e.g. 3D coupling in atmospheric chemistry which needs a huge level of flexibility)
 - The bad news – any change in requirements by the coupler inputs requires software changes
 - We may need to support multiple formats to cover different couplers or different versions of the same coupler or different component model versions.

Summary: I want to automate greyed-out items.... tough (for the developer!)

Build
configuration

What do I need to
rebuild? -
automated

Obtain compatible
compiler and lib
details from central
coupler installation

Scientific
configuration

Configure
science

Define coupling
exchanges

Cross check
coupling and
dump frequencies

Model timestep and
coupling data
consistent across
components.

Generate
coupling controls

But isn't that the point?

- Someone, somewhere has to deal with these issues!
- If we, the **software developers** don't take control, then it falls to **the user**.
 - In fact it falls to multiple users who may or may not have all the relevant knowledge and anyway are generally paid to do science not worry about technicalities of software demands!
- Isn't it better that this is dealt with by the software developer/owner rather than delegating responsibility to every separate user? **The needs of the many....!**
- I'd love to hear your stories, suggestions, recommendations and warnings!



Questions

- Answers (from, you!?)