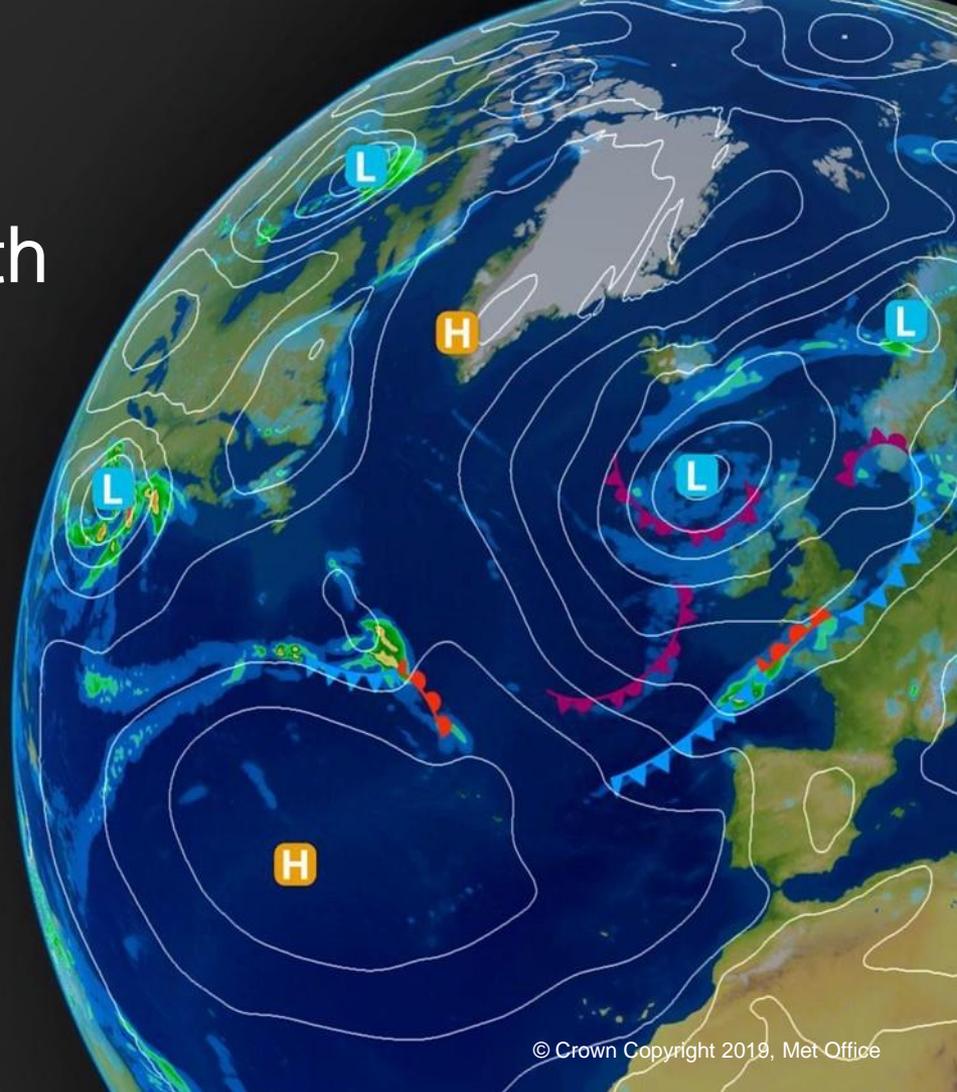


# Building robust and reproducible workflows with Cylc and Rose

Dr Stuart Whitehouse  
Data Workflows Team  
Met Office



# Contents

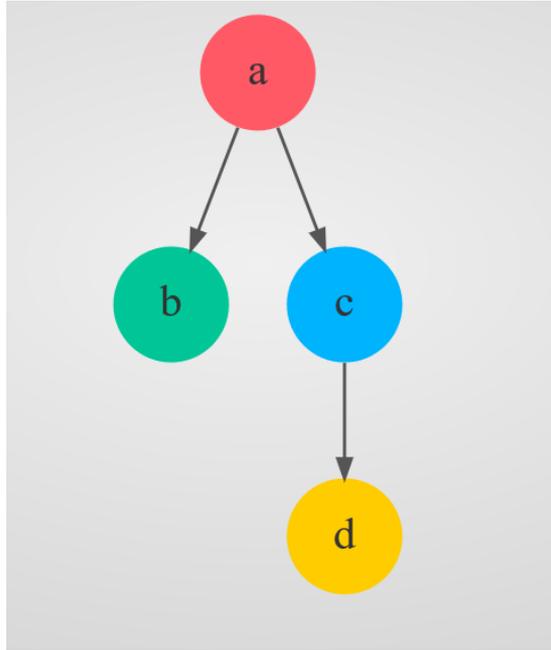
- Introduction to Cylc and Rose
- Robustness features
- Reproducibility
- Summary

# Introduction to Cylc and Rose

# What is Cylc?

- Cylc is a workflow engine for running suites of inter-dependent jobs
- Cylc can automatically:
  - Submit tasks across computer systems and resource managers.
  - Recover from failures.
  - Repeat workflows.
- Scales to very large, complex workflows
- Cycling suites:
  - Integer, real time, simulated time, externally driven

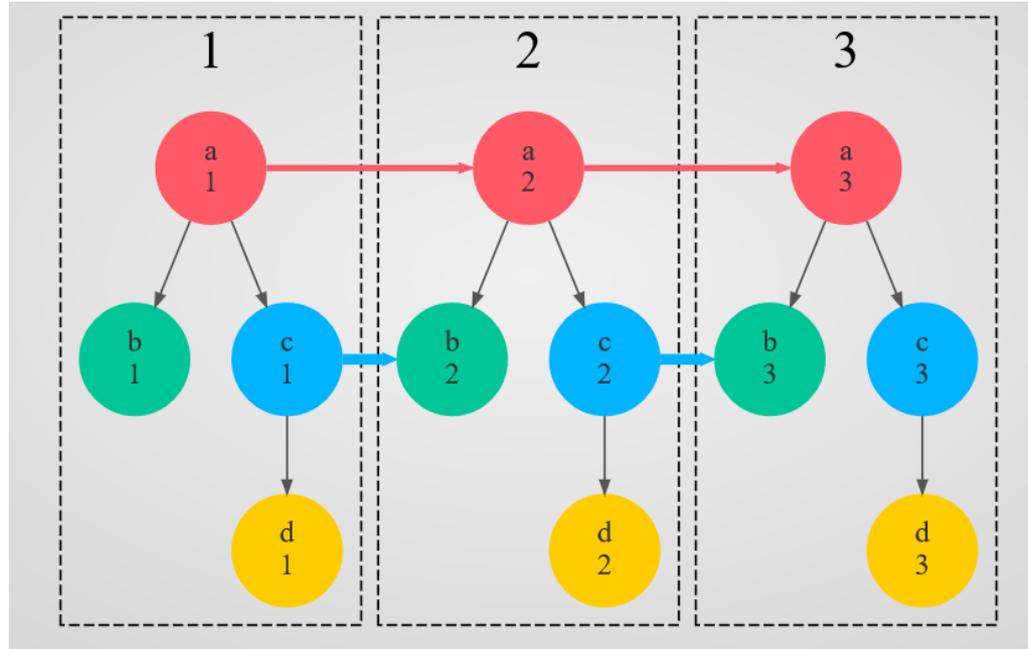
# A Simple Workflow



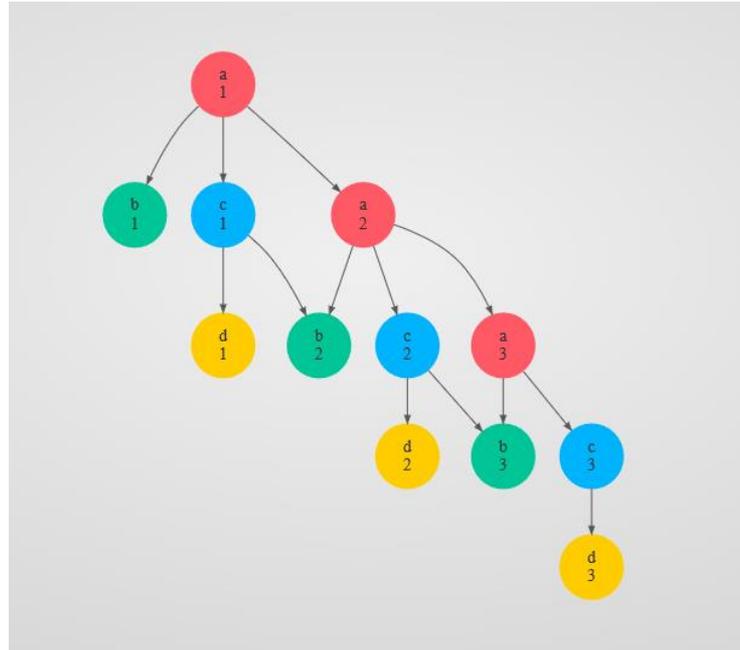
# Simple definition

```
[scheduling]
  [[dependencies]]
    graph = """
      a => c => d
      a => b
    """
```

# A Repeating Workflow



# A Cycling Workflow



# Cycling definition

```
[scheduling]
  [[dependencies]]
    [[[P1]]] # Repeat every cycle
    graph = """
      a => c => d
      a => b

      # inter-cycle dependencies
      a[-P1] => a
      c[-P1] => b
    """
```

# Date/time cycling

[scheduling]

```
initial cycle point = 20191014
```

```
final cycle point = 20191016
```

```
[[dependencies]]
```

```
[[[T00,T06,T12,T18]]]
```

```
graph = """
```

```
a => b & c
```

```
a[-PT6H] => a
```

```
"""
```

```
[[[T06, T18]]]
```

```
graph = """
```

```
c => x
```

```
"""
```

Initial and final dates

Times to run

*b* and *c* depend on *a*

*a* depends on *a* from the previous cycle

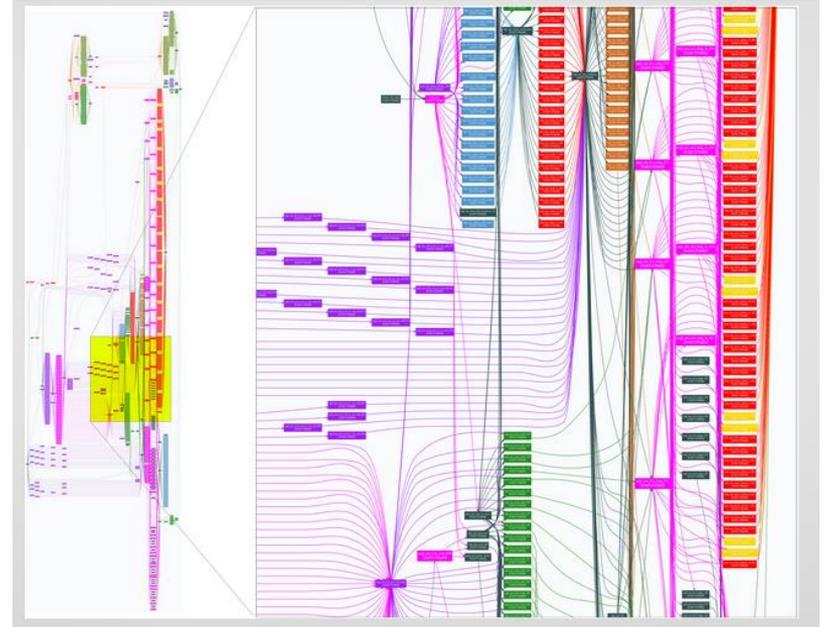
*x* depends on *c*, but only for T06 and T18

# Supports many batch systems

- at
- background
- loadleveler
- lsf
- moab
- pbs
- sge
- slurm

# Scaling

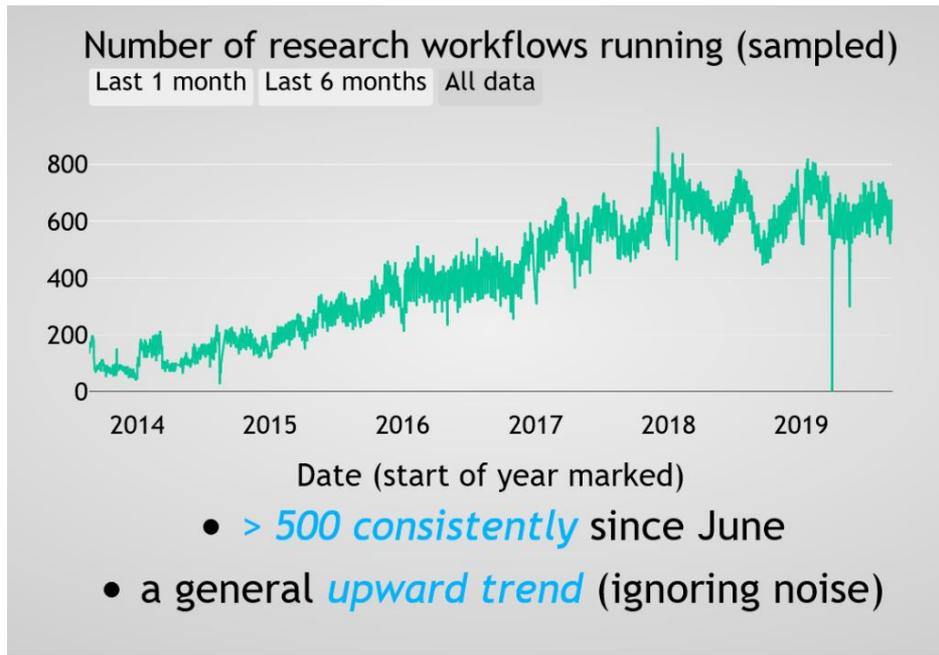
- Proven to work at the Met Office with:
  - Thousands of recurrences
  - Tens of thousands of tasks
  - Hundreds of thousands of dependencies
- Met Office workflows are often huge!



# Met Office Cylc GUI

The screenshot displays the Cylc GUI interface within a window titled "UM\_vn11.4\_trunk - exvcylc10.metoffice.gov.uk:43103". The window has a menu bar with "File", "View", "Control", "Suite", and "Help". Below the menu bar are icons for "View 1" and "View 2: None". The main area shows a workflow graph with several tasks represented by colored shapes: a green hexagon for "DESKTOP 1", a grey oval for "monitor\_xc40\_tasks 1", a green hexagon for "EXTRACT\_SOURCE 1", a green hexagon for "INSTALL 1", and several blue hexagons for tasks like "METO\_XC40\_COUPLED\_N96\_ORCAO25\_NCT\_TECHNICAL\_GC3 1", "METO\_XC40\_GNU\_HASWELL 1", "METO\_XC40\_INTEL\_HASWELL 1", "LINUX\_GA\_AMP\_EXP\_NAMING\_CRUN 1", "LINUX\_GA\_AMP\_EXP\_NAMING\_CRUN\_ARCH 1", and "ROSE\_ANA\_COMPARISON 1". A context menu is open over the "METO\_XC40\_GNU\_HASWELL 1" task, listing actions such as "Browse task URL", "Focus on 1", "Focus Reset", "Group", "UnGroup", "Recursive UnGroup", "Trigger (run now)", "Poll", "Kill", "Reset State", "Force spawn", "Hold", "Release", "Remove after spawning", and "Remove without spawning". The "Release" option is highlighted. At the bottom left, a status bar shows "running to stop at 1" with a color-coded indicator and "(filtered) live". The bottom right corner shows the time "12:03:49+01:00". The system tray at the very bottom shows the user "fruw@vid369 /var/tmp/persistent..." and the window title "UM\_vn11.4\_trunk - exvcylc10.me...".

# Met Office Usage



# Major Sites using Cylc



# What is Rose?

- Rose is a toolkit for writing, editing and running application configurations.
- Rose also contains other optional tools for:
  - Version control.
  - Suite discovery and management.
  - Validating and transforming Rose configurations.
  - Interfacing with Cylc.

# Robustness

# Robustness

- How do you ensure that your suite continues to run without manual intervention?
  - E.g. at the weekend, while you're at a conference, on holiday, etc.
- Job fail for a number of reasons
  - With larger and more complex systems there's more chance of a outage
    - E.g. HPC, archive, network problems, etc
    - There are also bugs in code, system errors, etc.
- These can cause errors either in job submission or job execution

# Retries

- Cylc can automatically resubmit jobs in the event of failure
- Failures in job submission and job execution can be handled separately
- You can customise the intervals and number of retries

```
[runtime]
  [[atmos]]
    [[[job]]]
      submission retry delays = 3*PT15M
      execution retry delays = 2*PT10M, PT1H
```

ISO8601 durations!



# Changing behaviour on retry

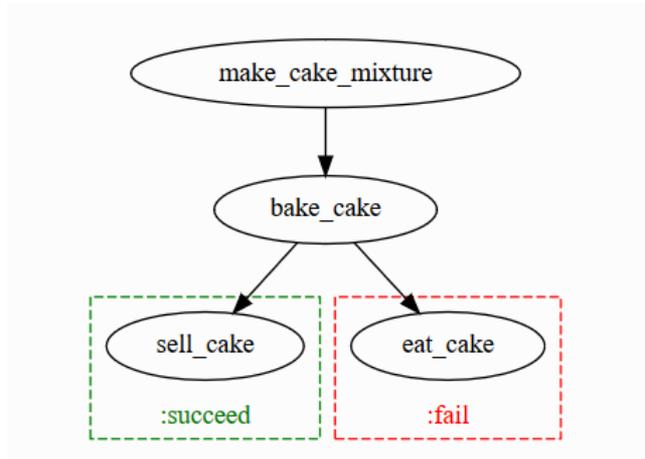
```
[runtime]
  [[atmos]]
    script = """
      test ${CYLC_TASK_TRY_NUMBER} -eq 3
      # do something different
    """

  [[[job]]]
    execution retry delays = PT0S, 2*PT10M
```

Use case:  
Changing the timestep when  
the model fails to converge

# Coping with Failure

- Modify the graph depending on whether a task passed or failed



```
graph = """
make_cake_mixture => bake_cake
bake_cake => sell_cake
bake_cake:fail => eat_cake
"""
```

# Changing workflow based on task output

```
[scheduling]
  [[dependencies]]
    graph = """ arch:archive_success => housekeep
                arch:archive_fail => other_archive"""

[runtime]
  [[arch]]
    script = """
      test ${CYLC_TASK_TRY_NUMBER} -eq 3
      # do something different
    """

  [[[job]]]
    execution retry delays = PT0S, PT10M

  [[[outputs]]]
    archive_success = Archiving succeeded
    archive_fail = Archiving failed
```

# Job Polling

- For supported batch systems, polling can query the batch system to determine the job state
- This can be triggered by the user through the GUI...
- ... or automatically when a job times out
  
- Timeouts terminate a job (etc) when Cylc has not heard from the job by an expected time:

```
[[[job]]]
    execution time limit = PT1H
```

# Events and e-mails

- Define a set of events you care about
- If these occur, the system will e-mail you
  
- Useful if you need to monitor lots of suites, for example:

```
[[[events]]]
```

```
mail events = submission failed, submission retry, failed, retry
```

# Metadata and Validators

- As well as the features in Cylc for runtime robustness, Rose includes validation of the settings in an application.
- For example, ensuring that you're using compatible options
- This is defined by metadata provided by the original developer and tested by the GUI
  - Settings which conflict with the metadata are highlighted
- More complex validation can be done using macros
  - (using Rose's Python API)

# Met Office Rose edit GUI

The screenshot displays the Met Office Rose edit GUI. The main window is titled 'um\_ukv1p5\_eg\_da - rose config-edit'. The left sidebar shows a tree view of configuration sections, with 'Ice processes' selected. The main panel shows the configuration for 'Ice processes' with various parameters and their values:

Parameter	Value
<input checked="" type="checkbox"/> L_psd	true
Use generic ice particle size distribution	
<input checked="" type="checkbox"/> L_shape_rime	true
Use ice particle shape-dependent riming rate	
<input type="radio"/> graupel_option	No graupel
Option for inclusion of graupel	
<input type="radio"/>	Original scheme
<input checked="" type="radio"/>	With snow-rain to graupel
<input type="radio"/>	Scheme based on new Field et al PSD
<input checked="" type="checkbox"/> ai	-1
Aggregate mass coefficient	
<input type="checkbox"/> bi	2.0000
Aggregate mass exponent	
<input checked="" type="checkbox"/> L_diff_icevt	true
Use two different ice fall speeds	
<input type="checkbox"/> c1c_input	1.024180e+3
Prefactor in fallspeed relation of small ice	
<input type="checkbox"/> d1c_input	1.000000
Exponent in fallspeed relation of small ice	
<input type="checkbox"/> c1l_input	1.426110e+1
Prefactor in fallspeed relation of large ice	
<input type="checkbox"/> d1l_input	4.164100e-1
Exponent in fallspeed relation of large ice	
<input type="checkbox"/> ar	1.00
Axial ratio for aggregates	
<input type="checkbox"/> a_ratio_exp	-0.2707
Exponent in the cross-sectional area ratio relation ice	

A dialog box titled 'Variable name list: run\_precip=ai' is open on the right. It shows the following information:

Modified since the last save, old value 2.3000e-2

**Data**

- name ai
- value -1
- comments
- error type: Value -1 is not in the range criteria: 0:
- flags
- ignored\_reason
- warning

**Metadata**

- compulsory true
- description Aggregate mass coefficient
- full\_ns /um\_ukv1p5\_eg\_da/namespace/UM Science Settings/section04/ice
- help This allows the explicit input of ice particle mass-diameter relationships. The mass-diameter of ice aggregates is defined as  $m(D) = ai D^{bi}$ . This variable is ai in this relationship and should take a positive value.
- id namespace/run\_precip=ai
- ns namespace/UM Science Settings/section04/ice
- range 0:
- sort-key mphys\_ice03
- type real

The dialog box has a 'Close' button at the bottom right.

# Reproducibility

# What do we mean by “Reproducible”?

- We run the same tasks in the same order?
  - On a different platform (portability)
- We get bit-comparable results?
- We get scientifically identical results?
- We get the same runtime every time (e.g. no variation in wallclock time)
- We get the same bug every time (e.g. detecting race conditions)

# Reproducing workflows

- Cylc's suite configuration file is a text file, as we've seen
  - No special tools needed so it's easy to edit
- Rose's application configuration file is also a text file
- Version control these files to reproduce the workflow later
  - The whole workflow and configurations inside can be reproduced later
- This doesn't protect from hardware/OS changes!

# Reproducing results

- Getting bit-comparable results is very difficult
  - Eventually you'll be forced to upgrade platforms/compilers which typically change results
  - Often a trade-off between performance/optimisation and getting the same results
  
- Scientifically identical is more achievable
  - But depends on a scientific judgement as to what is appropriate
  - How good is “good enough”?
  - And this differs depending on the quantity and the application

# Conclusion

# Summary

- Cylc and Rose provide a way of creating and reproducing a workflow
- They include lots of functionality to assist in robustness, including:
  - Retries
  - Polling, manual and automatic
  - Failure triggers
  - Events handling
  - Metadata validation

# Where to find out more

- Cylc and Rose are Open Source software
  - Cylc: <https://github.com/cylc/cylc-flow>
  - Rose: <https://github.com/metomi/rose>
  
- Support is available

is-enes  
INFRASTRUCTURE FOR THE EUROPEAN NETWORK  
FOR EARTH SYSTEM MODELLING



esiwace  
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER  
AND CLIMATE IN EUROPE

# Questions?

For more information please contact



[www.metoffice.gov.uk](http://www.metoffice.gov.uk)



[stuart.whitehouse@metoffice.gov.uk](mailto:stuart.whitehouse@metoffice.gov.uk)



+44 (0) 330 135 2491