Unified Workflow: A Toolbox and Framework for NWP Systems

Christina Holt

UW Team: Fredrick Gabelmann, Paul Madden, Emily Carpenter, Naureen Bharwani, Brian Weir

The Unified Workflow Project: The Team

An EPIC Agile Team welcoming contributors with various objectives to achieve a common end goal –
unification of workflows across UFS

The current team comprises staff from NOAA GSL with funding from the JTTI and SENA, and Raytheon/Element 84 Staff under the EPIC Contract

Powered By GSL

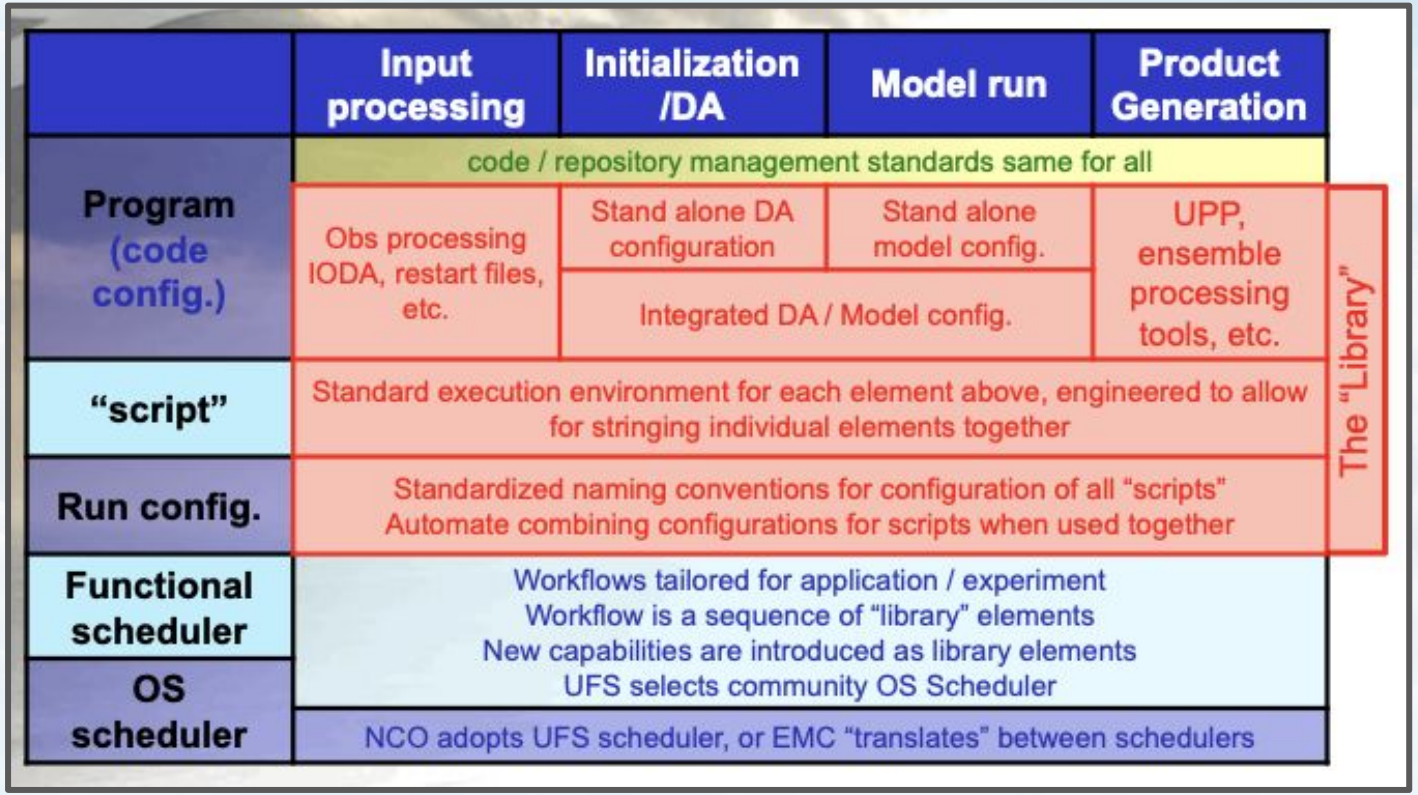
The Unified Workflow Project: The Goals

Unification:	Multiple apps using the ***same tools*** to perform the ***same tasks***
Ease of Use:	***Flattening the learning curve*** to get what you need out of a UFS App
Flexibility:	***Empowering scientists*** to realize experiments without being limited to what already exists
Facilitate R2O:	***Research and operations*** configuring and running the ***same components*** with the ***same languages and infrastructure***

Powered By GSL

NOAA

UFS Workflow Workshop Ideas



The Unified Workflow Project: The Software

A Python package that provides command line tools and a Python API to perform **common workflow tasks** and **drive UFS components** (forecast model, post processing, verification, etc.)

We publish each release to the ufs-community Anaconda channel for easy installation

We're planning to integrate the tools into multiple UFS Applications
Currently working with
Short Range Weather (a limited-area, static domain system)
Land DA (stand-alone JEDI-based land state cycling)

Powered By GSL

NOAA

uwtools v2.0.1

Install it with conda:

```
$ conda install -c ufs-community uwtools=2.0*
```

Use the command line tools:

```
$ uw [mode] [action] [-h]
```

Use the API:

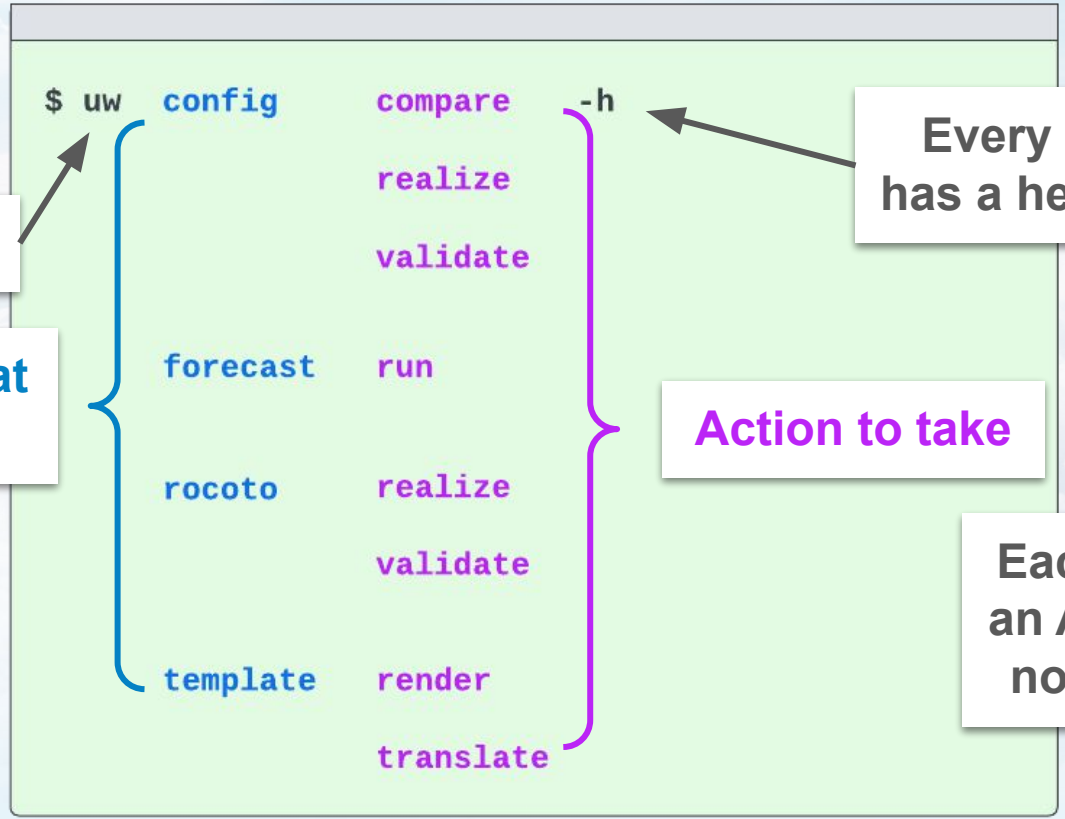
```
import uwtools.api as uwtools
```

On GitHub:

On Read the Docs:

Powered By GSL

Intro to uwtools v2.0.1 CLI



Powered By GSL

Generic Tools: Config

compare, transform, modify, and validate
key/value configurations
YAML, **Fortran namelists**, **INI**, **bash**

Generic Tools: Config Compare

```
(uwtools) $ uw config compare --file-1-path input1.nml --file-2-path input2.nml
[2024-01-25T17:52:27]     INFO - input1.nml
[2024-01-25T17:52:27]     INFO + input2.nml
[2024-01-25T17:52:27]     INFO ------------------------------------------------------------
[2024-01-25T17:52:27]     INFO atmos_model_nml:       blocksize:  - 32 + 40
[2024-01-25T17:52:27]     INFO atmos_model_nml:      ccpp_suite:  - FV3_GFS_v16 + FV3_HRRR
[2024-01-25T17:52:27]     INFO cires_ugwp_nml:    launch_level:  - 27 + 25
[2024-01-25T17:52:27]     INFO fv_core_nml:   agrid_vel_rst:  - False + None
[2024-01-25T17:52:27]     INFO fv_core_nml:        d2_bg_k2:  - 0.0 + 0.04
...
```

Generic Tools: Config Realize

input file

```
platform:
  NCORES_PER_NODE: 40

task_run_fcst:
  RUN_FCST_TN: "run_fcst"
  PE_MEMBER01: 221
  NNODES_RUN_FCST: !int '{{ (PE_MEMBER01 + PPN_RUN_FCST - 1) // PPN_RUN_FCST }}'
  PPN_RUN_FCST: !int '{{ platform.NCORES_PER_NODE // OMP_NUM_THREADS_RUN_FCST }}'
  WTIME_RUN_FCST: 04:30:00
  MAXTRIES_RUN_FCST: 1
  OMP_NUM_THREADS_RUN_FCST: 2
```

apply supplemental file

```
task_run_fcst:
  OMP_NUM_THREADS_RUN_FCST: 4
```

output file

```
platform:
  NCORES_PER_NODE: 40

task_run_fcst:
  RUN_FCST_TN: "run_fcst"
  NNODES_RUN_FCST: 12
  PPN_RUN_FCST: 20
  WTIME_RUN_FCST: 04:30:00
  MAXTRIES_RUN_FCST: 1
  OMP_NUM_THREADS_RUN_FCST: 4
```

Powered By GSL

NOAA

Generic Tools: Template

Leverage **Jinja** for a Turing-complete templating language

https://jinja.palletsprojects.com/en/3.1.x/

Generic Tools: Template Render

Tools are designed to leverage Linux pipes and env vars

```
(uwtools) $ export cycle=2023092112
(uwtools) $ echo '{{ cycle[0:4] }}' | uw template render
2023
(uwtools) $ echo '{{ cycle[8:] }}' | uw template render
12
(uwtools) $ echo '{{ cycle[0:4] }}' | uw template render --values-needed
[2024-03-01T12:25:05]     INFO Value(s) needed to render this template are:
[2024-03-01T12:25:05]     INFO cycle
```

`--values-needed` flag to introspect templates and provide a list of required variables

Generic Tools: Template Render

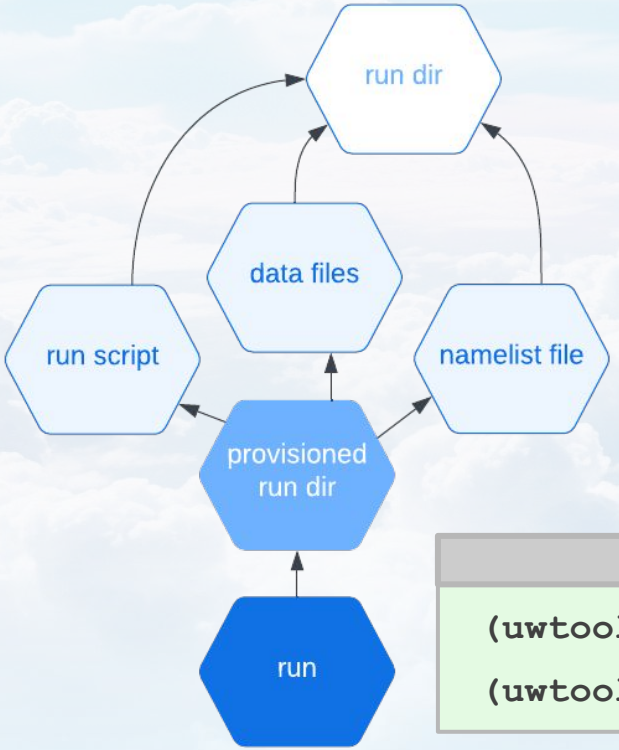
Render templates from environment variables or values files

```
start_year:             {{ cycle[0:4] }}
start_month:            {{ cycle[4:6] }}
start_day:              {{ cycle[6:8] }}
start_hour:             {{ cycle[8:] }}
start_minute:           0
start_second:           0
nhours_fcst:            {% if cycle[8:] == 12 %}48{%else%}12{%endif%}
dt_atmos:               {{DTATMOS}}
...
```

Powered By GSL

Graph-oriented Drivers

Graph-oriented Driver: Basics



Structured
YAML Config

Each task is represented by
a node in a graph

Dependencies are represented
by edges

The UW Driver can request any
node in the graph

```
(uwtools) $ uw some_component namelist_file -q
(uwtools) $
```

Powered By GSL

NOAA

Graph-oriented Driver: How?

iotaa python package: **It's One Thing After Another**

A workflow engine that expresses workflows using Python code to define tasks that are responsible for readying their assets, and the relationships between them.

```
@task
def namelist_file(rundir):
     fn = "input.nml"
     yield "Namelist file"
     path = rundir / fn
     yield asset(path, path.is_file)
     yield path.mkdir(parents=True)
     create_user_updated_config(...)
```

Asset name

The asset(s)

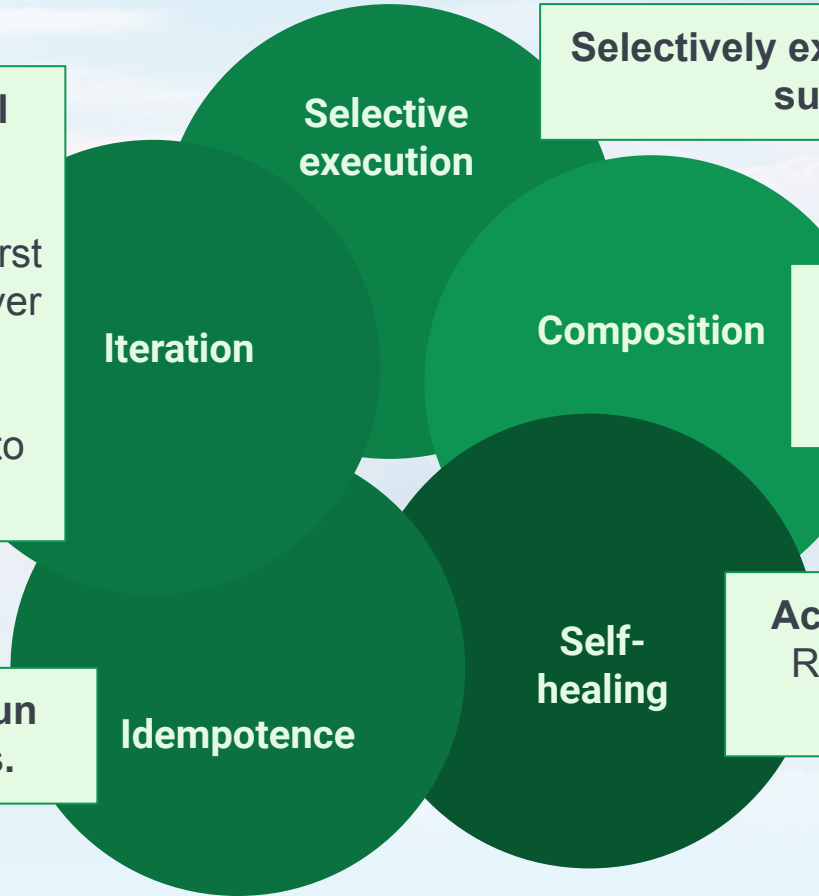
Requirements for readying the asset

How to complete the task

Powered By GSL

iotaa is developed and maintained by Paul Madden at https://github.com/maddenp/iotaa

Graph-oriented Driver: Benefits



Iterate with a driver until all tasks are complete.

Some data isn't ready on the first run? Wait for it and run the driver again.
Find an error in an asset? Remove it and run the driver to correct it.

Selectively execute any graph subtask.

Break complex tasks down into reusable pieces.

Accidentally delete a file? Run the driver to create it again.

Completed tasks won't run again, saving resources.

Powered By GSL

NOAA

Summary

uwtools is a tangible Python package that you should install today to help you with workflow tasks.

The tools are designed to help scientists be productive, and to help applications increase the automation of common tasks.

Graph-oriented drivers are more flexible than many existing solutions that are often hard-coded, procedural, and/or brittle.

Helps with hierarchical testing for research, and reliability and recovery in real time use cases

Upcoming Plans

- Release new tools/drivers regularly
- Integrate uwtools into existing applications
- Build drivers for more UFS components
  - FV3 global and coupled; regional is available with limited flexibility now
  - JEDI
  - MPAS
  - Preprocessing
  - UPP
- Build more generic tools
  - Moving files
  - Interfacing with ecFlow

```
(uwtools) $ uw jedi stage_obs
(uwtools) $ uw post run
(uwtools) $ uw forecast create_namelist
```

Acknowledgements

This research was supported in part by

- NOAA cooperative agreement NA22OAR4320151
- An FY22 WPO JTTI award: *Implementation of a Unified Workflow for the Unified Forecast System (UFS) Short Range Weather (SRW) Application*
- The OCIO Software Engineering for Novel Architectures program award to NOAA GSL
- The EPIC Contract
- The NOAA R2O Project
- In-kind contributions from NOAA EMC, NOAA GSL, and George Mason University

Powered By GSL