Workflow Requirements

Workflow workshop -- June 2021

Based on discussions with and contributions from:
Arun Chawla, Ben Cash, Chris Harrop, Christina Holt, Kate Friedman, Walter Kolczynski, Jian Kuang, Bin Liu, Yannick Tremolet, Claude Gibert, Sam Trahan, Rahul Mahajan

The Issue

- **The PROBLEM**: Myriad of workflows supporting the different UFS applications -- Global workflow (for coupled / atmosphere only system), CIME (for global forecast only systems), Regional workflow, hurricane workflow, NG-GODAS/JEDI-SOCA, JEDI-EWOK, Observation processing, etc.

- **The Vision**: Develop a wide array of tools (in Object Oriented Python) that can be used to configure, execute and process a series of jobs / tasks as defined by an application suite
  - Application suite can range from model based (cycled, forecast only, coupled, ensembles, DA etc), pre-processing of model/data, reanalysis products, verification & validation, plotting & web hosting etc.

- **The Question**: How do we get from the current state to a state that can support operations, R2O as well as engage with the community ?

Requirements for an Ideal System

- Modular and Configurable
  - Broken down to sub tasks that allows efficient use of resources (minimize makespan and maximize allocated cpu usage)
  - Easily allows a task to be reconfigured, switched on or off
  - Common repeatable parts separated out as generalized functions/methods

- Documentation
  - Detailed guides on how to use and port system
  - Every function/script should have inline documentation on -- a) purpose; b) input/output; c) use case

- Portability
  - Workflow system should be easily portable on multiple platforms (HPC, non-HPC, Cloud)
  - Should use language easily available on all platforms (Python 3 for example)

Requirements for an Ideal System (contd.)

- Portability (contd)
    - System specific information should be through easily accessible configuration files and not buried in scripts
    - All paths through configuration files (no paths in any script unless relative)
    - Should contain no source code within the workflow framework
    - Utility/aux codes needed should be easily built and ported (e.g. CMAKE + HPC-Stack)
    - Libraries / packages needed should be easily deployable (use third party libraries as needed but judiciously)

- Workflow Engine
    - Should be interfaced generically with workflow engines used by operations and community (ecflow/cylc/rocoto)
    - Should work without a scheduler (e.g. slurm) on a workstation (have the ability to mix scheduled and non-scheduled jobs)
    - Should be able to add new or other workflow engines e.g. airflow, Amazon SWF

Requirements for an Ideal System (contd.)

- Configurable for multiple applications
  - UFS based applications that cover range of use case conditions
    - Fully coupled global system ((d)atm+waves+ocean+ice+chemistry) with multiple options
    - A regional system with multiple options
    - DA(var+ens.)/forecast only or DA+forecast (Note: DA itself will have multiple options)
    - A hurricane modeling system
    - Deterministic/ensemble mode
  - Observation Ingest and Processing
    - Decoding, tanking (storing), dumping and filtering, etc
  - Post processing, product generation, validation and verification

- Configuration management system
  - A workflow system will be driven by a range of configuration yaml files + namelist options
  - A web based tool with an easy to use interface (web based/GUI/other?) to build these files based on choices made by the user
  - Once created, these files can be shared/changed to run the workflow systems (i.e. the management system is not needed to run the workflow)

Options

- OPTION 1: Build a single workflow that satisfies all requirements
    - One repository used by all teams
    - Works for a range of environments
    - Is it practical?

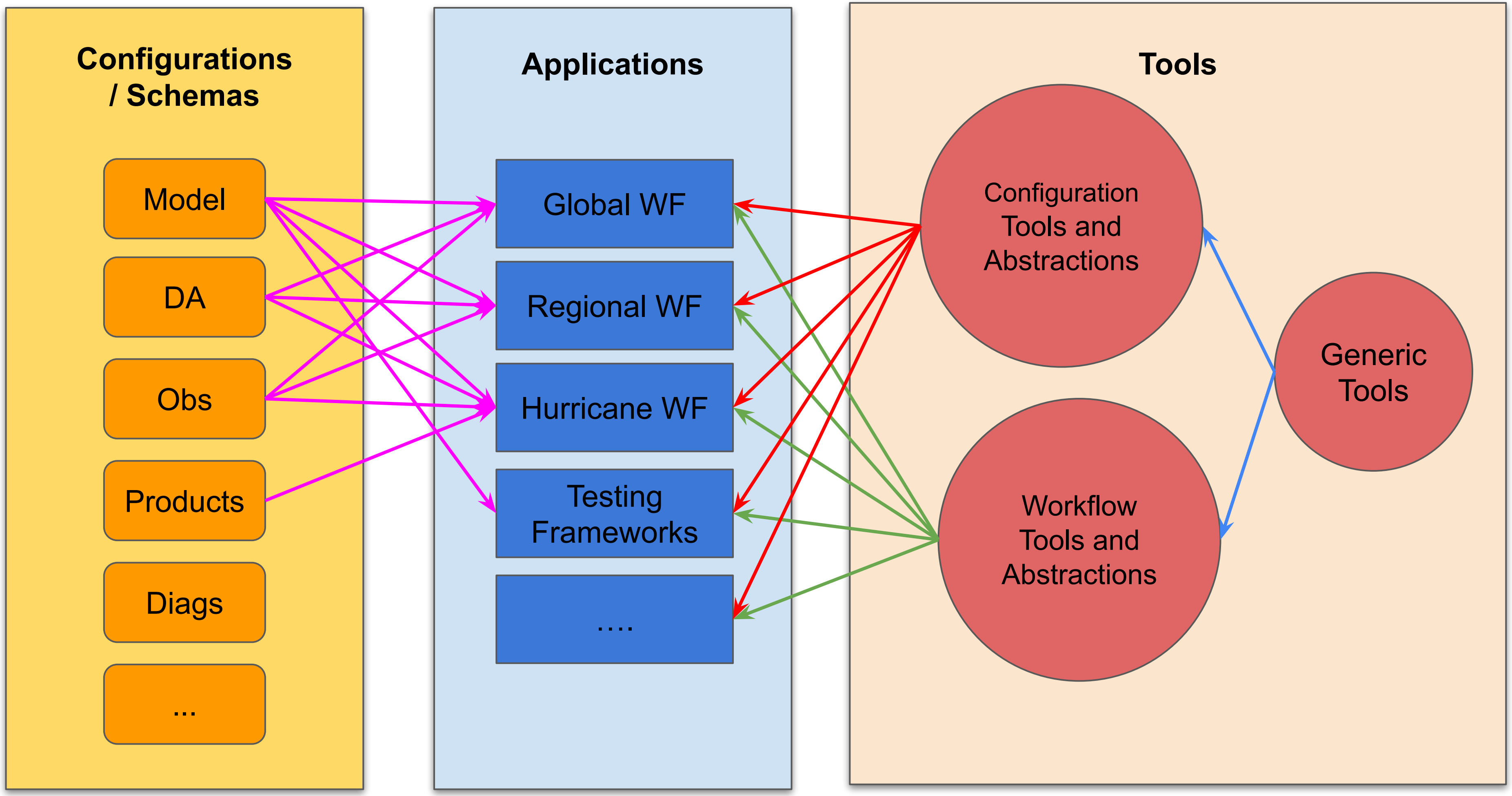
- OPTION 2: Build a series of tool sets / libraries and packages used by all
    - A finite number of workflows with different configurations.
    - Bulk of tasks done by common generic libraries / packages
    - A range of toolsets to support them and the UFS-weather-model
    - Only thing separating the workflows is the configuration layer

Our preference

- We are strongly leaning towards Option 2, as
  - It unifies workflows but provides a thin layer of separation for different teams to work without tripping over each other
  - Built in Python3 and use package managers to create and install workflow tools and packages
  - Maintains the separation of concerns
  - Keeps workflows simpler for porting to different platforms
  - Allows for piecewise implementation of new tool sets instead of a wholesale replacement (agile approach)
  - Provides a complete package (model+workflow per application) to operations
  - Allows for flexibility to create new workflows that have not yet been considered

How do we go about this?

- A Project plan with one PM
- All work in public repositories
- A lead software engineer (to direct solutions) and build generic toolsets
- Active team (with representations and contributions from application stakeholder?) to test in different applications for rapid/agile development
- A CI system from the beginning, coding standards, linters, testing, etc.
- Active documentation as the system develops
- Test on a range of platforms (MacOS/Linux/NOAA HPC/Cloud)
- Regular interactions / report outs to operations and community to ensure alignment


Configurations / Schemas
Model
DA
Obs
Products
Diags
...
Applications
Global WF
Regional WF
Hurricane WF
Testing Frameworks
....
Tools
Configuration Tools and Abstractions
Workflow Tools and Abstractions
Generic Tools