UFS System Architecture and Software Governance

Change History Table

Version #	Date	Changes
v1.0	August 2018	UFS Infrastructure System Integration Plan v2
v2.0	October 2024	Draft version of the combined UFS Application System Architecture & Infrastructure (PWS:4.4.1-ECC-1270) and UFS Application Software Governance (PWS:4.4.2 - ECC-1271)
v2.1	April 2025	Updates based off of stakeholder feedback from v2.0
v2.2	June 2025	Updates based off of stakeholder feedback from v2.1

Overview

This document describes the supported and planned application System Architecture and Infrastructure (SAI) for the Unified Forecast System (UFS) and the software governance needed both to support community development and to align closely with operational standards. This document highlights advances in the UFS as they pertain to the system architecture and infrastructure plan and provides recommendations for areas of further advancement. This plan combines the original UFS System Architecture document with EPIC-derived system architecture and software governance plans to form a cohesive document for users to better understand how the governance and system architecture is designed to support community innovation.

In the past few years, with the establishment of EPIC and technical advances in the field, new infrastructure components were added to enhance the UFS Application SAI — including Continuous Integration / Continuous Deployment (CI/CD) pipelines, containers, *spack-stack*, Unified Workflow (UW/UWF), and others — which required an update to the original plan.

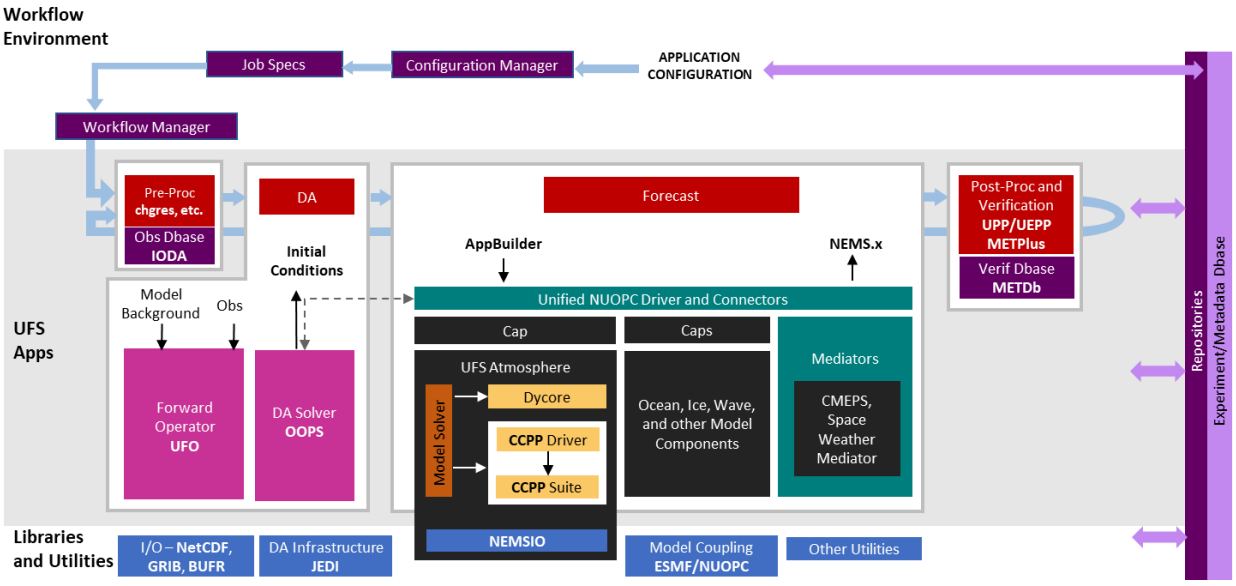

Figure 1: System Architecture (Provided by the UFS)

Background Information

Below are source documents that laid the background for the UFS system architecture and governance. This document has been updated to reflect changes in the source material and may be updated periodically with new changes.

- UFS Infrastructure: Repositories Sub-Group, A presentation to UFS Steering Committee - 08 June 2018: UFS Infrastructure: Repositories Sub-Group
- UFS Governance: Draft Document on Unified Forecast System Organization and Governance
- UFS Weather Model (WM) Coupling Infrastructure Capability
- UFS WM Wiki
- UFS Spack Configuration files collection, *spack-stack* Wiki
- UFS Global-Workflow (GW) Documentation
- UFS Weather Workflow Tools Python Library, *wxflow* Documentation
- UFS Code, Data, and Documentation Management for NOAA Environmental Modeling System (NEMS) Modeling Applications and Suites
- UFS Infrastructure v2
- EMC Code Management (CM) Repos and Gitflow

Figure 2: EPIC Overview

UFS Infrastructure Group Executive Overview

The UFS Infrastructure group has purview over three different project areas. Because these projects are unique, it was decided to disband the original Next Generation Global Prediction System (NGGPS) working group and create individual sub-groups dealing with Repositories, Data Portal, and Community Workflow.

Figure 3 shows the relationship among the Earth Prediction Innovation Center (EPIC), the Environmental Modeling Center (EMC), the National Centers for Environmental Prediction (NCEP) Central Operations (NCO), and the UFS Community. It also shows where governance will be in effect. Note that EPIC's focus is to refine potential innovations through the Research to Operations (R2O) funnel and to facilitate innovation in the codebases that are delivered to EMC for the transition to operations. While EPIC and the UFS community influence the funnel and contribute to the code that is ultimately delivered to EMC, they have no influence over decisions about what actually makes it into operations after that point.

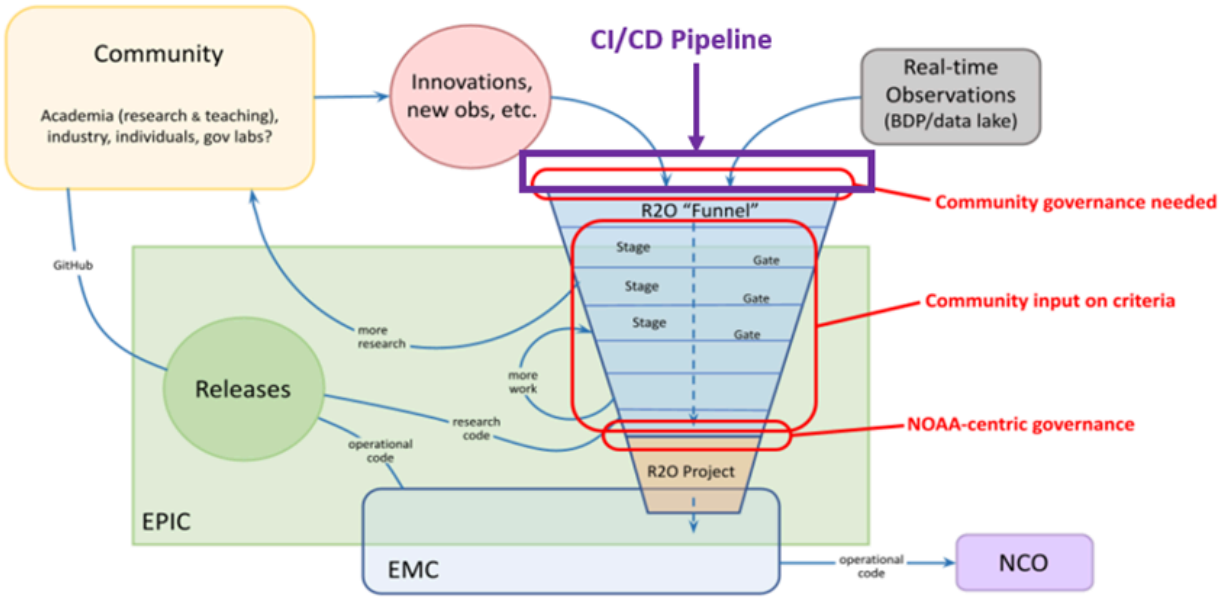

Figure 3: R2O Process

Although beyond the scope of this document, it is important to understand the relationship between NCO and community-available versions of operational UFS applications. Once a candidate for operations has been determined, NCO will acquire the complete source code for the specific application and place it in a single repository on a dedicated repository server behind

the firewall at NCEP/EMC.[1] While it is understood that NCO engineers will need to make changes as the transition progresses, this does not mean the operational version will diverge from that available to the community. It will be the responsibility of the team leading the transition to ensure operational sources maintained within the NCO internal repository remain synchronized with those accessible by the community (when possible as a stretch goal). The anticipated changes are to enhance performance, improve error handling, introduce/fix machine-specific constructs, and increase readability.

UFS System Architecture

UFS Applications

The UFS infrastructure and repository subgroup uses the UFS Steering Committee (UFS-SC) definition of an application. According to the UFS-SC, applications are "UFS configurations that support specific predictive targets (e.g. Medium-Range Weather, Subseasonal-to-Seasonal, Space Weather)" (DRAFT Unified Forecast System Organization and Governance). UFS Applications typically combine a numerical model, data assimilation, pre-processing, post-processing, and other elements into a workflow. The list of these applications will evolve as the UFS advances, and some applications may be retired, while others may be added. UFS applications include:

1. *Medium-Range Weather (MRW):* Atmospheric behavior out to about two weeks
2. *Subseasonal-to-Seasonal (S2S):* Atmospheric and ocean behavior from about two weeks to one year
3. *Hurricane:* Hurricane track, intensity, and related effects out to about one week
4. *Short-Range Weather (SRW)/Convection Allowing:* Atmospheric behavior from less than an hour to several days
5. *Space Weather:* Upper atmosphere and ionospheric behavior due to solar and geomagnetic activity and forcing from the lower atmosphere from real-time to about ten days
6. *Coastal:* Storm surge and other coastal phenomena out to about one week
7. *Air Quality:* Atmospheric aerosol and atmospheric chemical composition out to several days. Note: Air Quality may be folded into SRW/MRW and avoid being a separate application, these discussions are still ongoing, but are being tracked as an application currently.

[1] NCO will not work in open-development repositories. It is a requirement that NCO maintain an internally managed repository server to hold the source code for any and all applications running operationally.

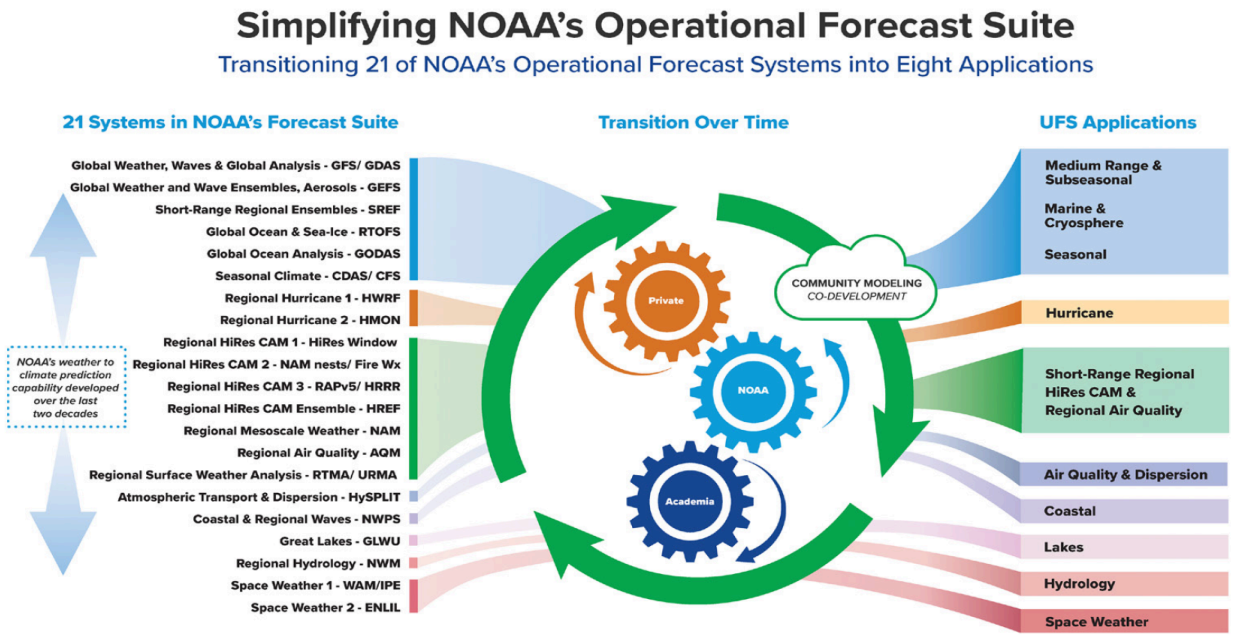

Figure 4: UFS Applications (Uccellini, et al.)

Current UFS Applications (including models and selected configurations)

Application Name	**Ownership**	**Managed by**	**Notes**
Weather Model	UFS Community	EPIC/EMC	Model
SRW	UFS Community	EPIC/GSL	Application
Land DA	UFS Community	EPIC	
Hurricane/HAFS	HAFS Community	HFIP	System
Global AR Configuration	UFS Community	EPIC	Configuration
Global Application	EMC	EMC/EPIC	Basis for MRW and S2S
RRFS v1	EMC	EMC	SRW configuration
Coastal	NOS	NOS	Application
GDASApp	EMC	EMC	UFS/JEDI enabler

UFS Components

The UFS ecosystem includes multiple repositories and subrepositories, many of which are used by multiple applications. The UFS WM itself contains 14 submodules, some of which have their

own submodules at least two layers deep. UFS applications typically have a top-level repository that handles the workflow and is hosted under *ufs-community* on GitHub. One exception to this arrangement is the *global-workflow*, which is being developed by EMC and is hosted under their GitHub organization (NOAA/EMC). In general, UFS applications include some version of the UFS WM itself (with subrepositories) and an assortment of other tools for pre- and post-processing of data.

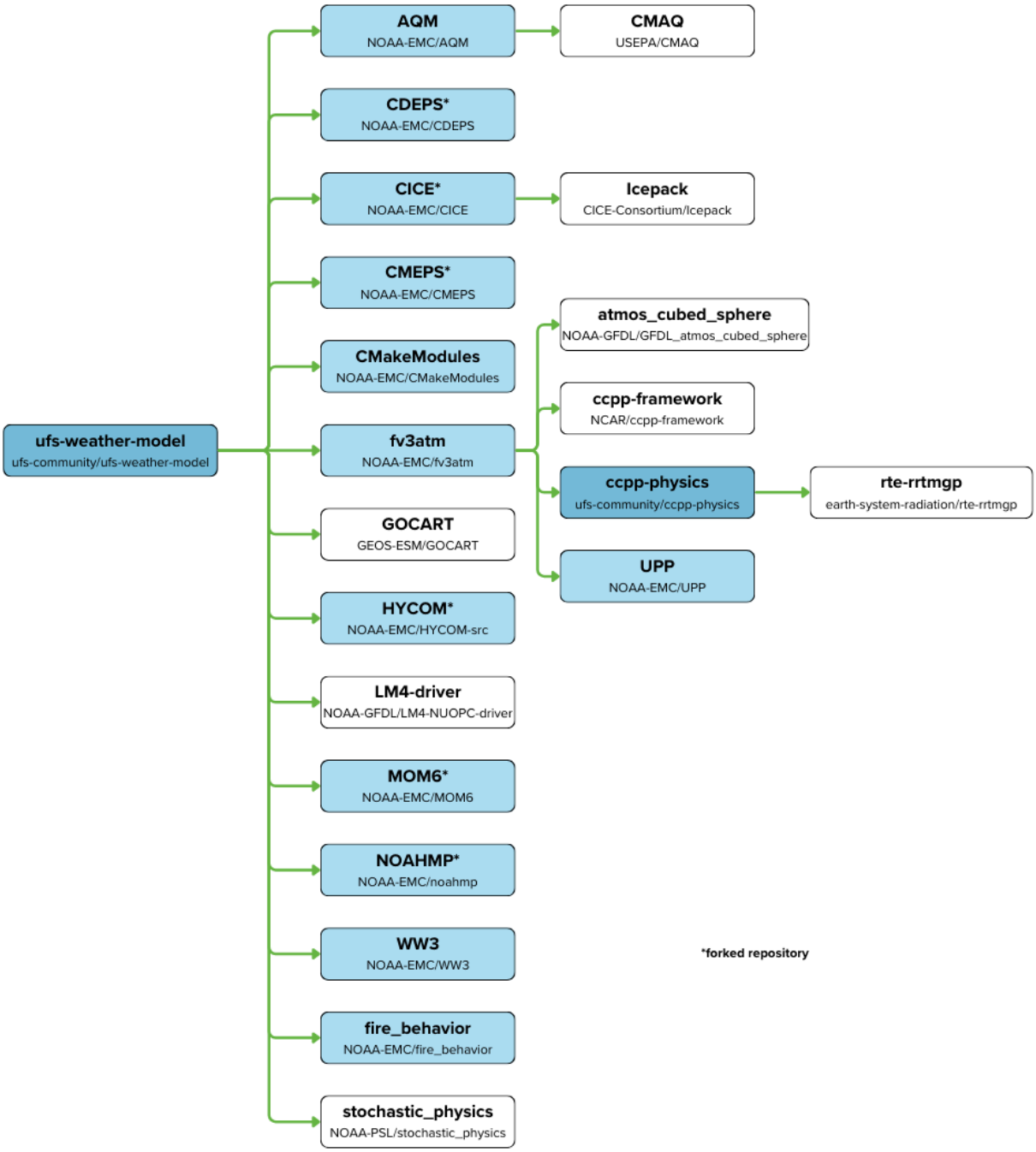

Figure 5: UFS Weather Model Hierarchical Repository Structure

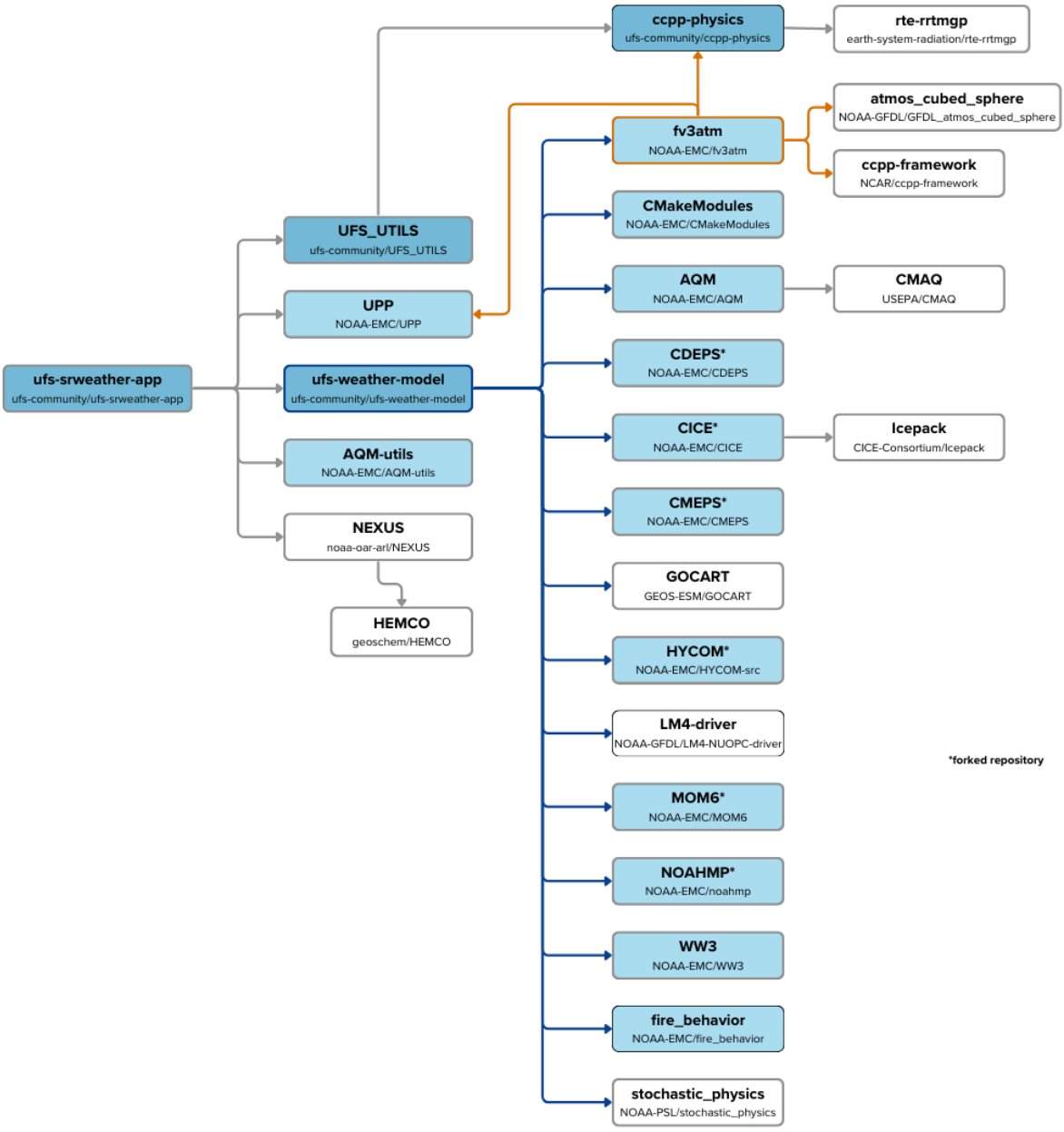

Figure 6: UFS SRW Hierarchical Repository Structure

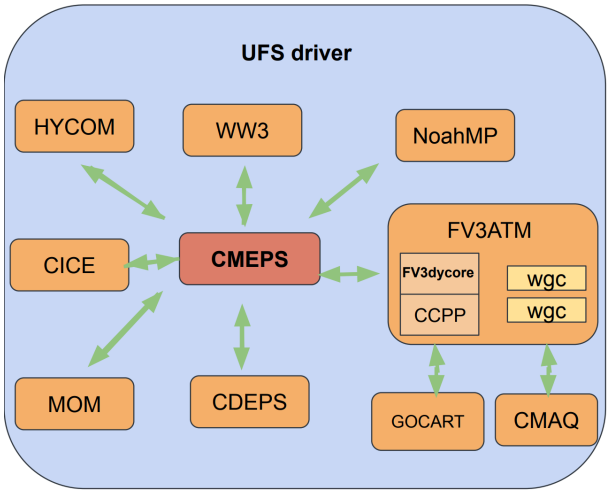

Figure 7: Model Infrastructure

UFS code bases: Weather Model sub-components and software infrastructure

Component Name	**Ownership**	**Managed by**	**Notes**
FV3atm	EMC	EMC	
FV3-cubed-sphere dynamics	GFDL	GFDL	Submodule of FV3atm
MPAS	UFS Community	GSL	In development
CCPP Physics	Physics Community/NCAR	DTC/EMC	Submodule of FV3atm
NoahMP	NCAR	EMC/GSL/NCAR	NOAHMP Interface
SeaIce/CICE	CICE Consortium	CICE Consortium	CICE Interface
Ocean/HYCOM	NRL	NRL/EMC	HYCOM Interface
Ocean/MOM6	Ocean Community	Ocean Community: GFDL/NCAR	MOM6 Interface
CMAQ	EPA	EPA	Submodule of AQM
Aerosol/AQM	EMC	EMC	
Wave/WW3	Wave Community/EMC	EMC	
CMEPS	ESCOMP	ESCOMP	CMEPS Interface

Component Name	Ownership	Managed by	Notes
CDEPS	ESCOMP	ESCOMP	CDEPS Interface
Stochastic Physics	PSL	PSL/EPIC	
Community Fire Behavior	NCAR	NCAR	
GOCART	NASA	NASA	
LM4	GFDL	GFDL	
UPP	EMC	EPIC/EMC	Infrastructure
UW tools	UFS Community	EPIC/GSL	Infrastructure
spack-stack	JCSDA	JCSDA/EPIC/EMC	Infrastructure
METplus	DTC	DTC	Infrastructure

Unified Workflow Tools to Facilitate R2O2R into UFS Applications

UW Tools, *uwtools*, facilitates Research to Operations to Research (R2O2R) for UFS applications. The National Oceanographic and Atmospheric Administration (NOAA) Global Systems Laboratory (GSL) developed real-time experiments for the Hurricane Forecast Improvement Program (HFIP) using the Model for Prediction Across Scales (MPAS) App. It is the first Numerical Weather Prediction (NWP) app built from the ground up with *uwtools*. This type of accomplishment shows that *uwtools* is ready to support application development. *uwtools* now needs to transition to supporting operational workflow developments. The use of new modular development techniques and the focus on testing and documentation makes it a software package that is easy to use and test.

- Repository: https://github.com/ufs-community/uwtools
- Documentation: Unified Workflow Tools – latest documentation

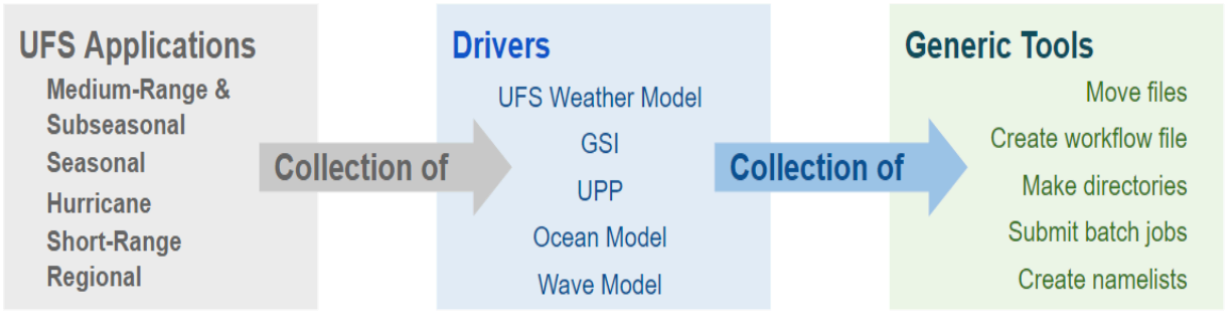

Figure 8: UW Tools Architecture

UFS Supported Platforms

A transparent dashboard was developed to view and understand current allocation, needs, and supported platform statuses (see EPIC Test Dashboard). A list of EPIC-supported Tier-1 platforms shown below.

Supported Platforms	**Supported Applications**	**Managed by**	**Notes**
AWS	WM, SRW, Land DA, Global-Workflow	EPIC	NOAA Cloud Resources
Azure	WM, SRW, Land DA, Global-Workflow	EPIC	NOAA Cloud Resources
GCP	WM, SRW, Land DA, Global-Workflow	EPIC	NOAA Cloud Resources
Derecho	WM, SRW	EPIC	NCAR CISL
GaeaC5	WM, SRW	EPIC	NOAA RDHPC
GaeaC6	WM, SRW	EPIC	NOAA RDHPC
Hera	WM, SRW, Land DA	EPIC	NOAA RDHPC
Hercules	WM, SRW, Land DA	EPIC	MSU
Orion	WM, SRW, Land DA	EPIC	MSU
Jet	WM, SRW	EPIC	Being deprecated (??)
Ursa	NA	EPIC	Coming soon (??)
Rhea	NA	EPIC	Coming soon

WCOSS2	All operational applications and systems	EMC	NOAA RDHPC

Research/Development Workflow/Test Infrastructure

EMC and NCO are reworking operational standards. Part of that discussion and the ongoing Short-Range Weather/Rapid Refresh Forecasting System (SRW/RRFS) Tiger team conversations align around how operations and research workflows can co-exist and converge (if possible). There is also an effort to highlight the components of a Hierarchical Testing Framework (HTF) that are needed to support both research and operations.

Integration with CI/CD Pipelines on Tier-1 Platforms

CI/CD pipelines exist on all Tier-1 systems across both supported and prototype applications. Since not all users have Common Access Cards (CACs), which are required for Jenkins access and results, EPIC developed an open-source dashboard (see EPIC Test Dashboard, CI/CD Artifacts tab) to allow all users to view results. The results displayed on the dashboard are all of the same artifacts that are available on Jenkins.

Enabling JEDI-Based DA via DA Proxy Apps

The global-workflow is an application that runs a global configuration of the UFS Weather Model in an end-to-end workflow for MRW forecasts. It contains many repositories, including the Global Data Assimilation System (GDAS) application. The setup of running GDAS through the GW can be thought of as a Data Assimilation (DA) Proxy application. The DA Proxy application may be configured to use the Joint Effort for Data assimilation Integration (JEDI) based DA, replacing the legacy Gridpoint Statistical Interpolation (GSI) DA process and allowing for testing with the new DA as it becomes more mature.

Portability Across On-Premises and Cloud Platforms

The use of containers creates a portable mechanism to allow a single software stack to be built and run in a portable manner across multiple platforms, both on-premises (on-prem) and in the cloud. EPIC maintains a repository of supported containers on Docker Hub. These container repositories have been utilized on Tier-1 platforms, NOAA Cloud platforms, MacOS, personal cloud platforms, and academic High-Performance Computing (HPC) platforms. Pre-built Singularity containers are also available via Amazon S3 buckets for public download.

Readiness of Containers to Facilitate Academic/Industrial Community Distribution and Engagement

EPIC works with academic partners and community users to support container installations of UFS applications, which allows UFS applications to have broad support over many HPC configurations, while minimizing the hours needed to support them.

A prime example of the portability of the UFS is a partnership between EPIC and George Mason University (GMU), which built a container stack of the SRW Application on GMU's academic platform, Frontera. Demonstrations of portable container installations on personally owned cloud resources and MacOS were held for the public, and it is desired to work with additional academic institutions over time.

Evaluation of Research and Operational Implementations

According to the UFS Organization and Governance v1.0 document, "UFS applications assure a focus on research and operational outcomes, not just on creating models and software." The UFS is set up so researchers can quickly test their scientific ideas in an easily configurable way but also allows operations to favor a stable and repeatable application release. Research and operations need to work together in the UFS so that there is a clearly defined path for R2O2R and that those outcomes are achieved.

UFS Repository Management

Introduction

The Unified Forecast System[2] is a community-based, coupled comprehensive Earth system modeling system to support NOAA's operational NWP system. The UFS is not a single application with support for hourly to seasonal timeframes, but instead is a collection of source systems used in building targeted applications for specific purposes. To be successful, the UFS must employ a common modeling architecture[3] and associated infrastructure. In this context, infrastructure consists of three major areas: repository management, workflow, and open access to a data portal. The goal of this document is not to define the suite of applications that will exist within the UFS, but to lay out a strategy for managing community development within an application.

Defining a comprehensive, community-friendly repository strategy for the UFS, which also satisfies operational constraints, is a complex problem. The approach here is to define the elements of the strategy — repository types, locations, and key interaction processes — and use this defined terminology to describe a set of use cases (including actors and events). The key principles are:

- Clearly define and communicate the UFS authoritative repository structure and practices
- Utilize open repositories to maintain transparency of code integration processes
- Facilitate collaboration between the UFS community and different agencies
- Be sufficiently flexible to support implementation of agency mission deliverables while allowing community contributors to focus on their goals
- Restrict development for each constituent component of the UFS to its own authoritative repository

Repository Management Strategy Overview

The proposed UFS repository management strategy places each UFS application (Seasonal Prediction, S2S, Weather Forecast, Regional, etc.) in a unique UFS umbrella repository. An umbrella repository contains no source code, but consists of configuration files. One configuration file will contain URL links and pointers to specific versions of model component code from external authoritative repositories. It is the combination of model component code that forms the application.

An authoritative repository is defined by the presence of a governance group and processes that indicate how changes are evaluated and incorporated, and when and how new reference versions

[2] https://ufs.epic.noaa.gov/
[3] https://docs.google.com/document/d/1Vyc-ZO4c7kqcf1EjAEHylJdTzQFee3SfYwk34oNiouU

are prepared for distribution. The use of authoritative repositories is central to the umbrella repository strategy and must satisfy a baseline set of criteria:

A. The governance group is willing to participate in community development
B. The code management policies and processes are well documented
C. The regression testing procedures are well defined

The component code and the umbrella repository will exist in authoritative repositories. The authoritative repositories are typically associated with the original development teams and are where code development and collaboration occur. Any specific code should lie in only one authoritative repository structure and can be accessed by multiple UFS applications. Governance of component repositories will be a combination of the conditions and procedures defined by the UFS and the native governance of the authoritative repository.

Each UFS application will have its own umbrella repository with a designated "code management gatekeeper." The gatekeeper is not responsible for the science but ensures that certain branches within the umbrella repository link to the appropriate versions of component authoritative repositories. A unique umbrella repository per application is necessary because different applications have different timelines for development and transition to operations. This also allows development groups working on different aspects of coupled modeling (weather scales, sub-seasonal, etc.) to work concurrently and independently (with some coordination).

Repository Governance

The National Earth System Prediction Capability Model Component Liaison committee is developing a set of guidelines for authoritative repositories.[4] The Infrastructure Repositories sub-group has adopted these guidelines and added additional rules for authoritative repositories hosting UFS components. The rules are broken down into different categories which are summarized below.

General repository practices:

- An authoritative central repository for the model components exists. The authoritative repository for model components will not reside on personal or local instances, it will be governed in the institutional repositories for consistency. Pushing and pulling often to make sure that all users are working off of the same repository structure and branches is critical.
- The repository uses a GitHub code versioning and management system.
- A governing or management body that sets and enforces policies for the repository exists.
- There are clear terms of use and there is a way for credentialed users to request access.

[4]

- Reference versions which incorporate selected code changes are delivered at semi-regular intervals (generally less than two years with an end goal of continuous releases).
    - Each reference version has a unique ID (e.g., tag, revision number).
    - Incremental changes made to the code between reference versions are documented.
    - Outdated and/or unsupported versions of the code are documented.
- The National Unified Operational Prediction Capability (NUOPC) cap for the component resides in the same authoritative repository as the model component code.

In addition, the following practices apply to "community" component models:

- Source code is either fully open or available through a registration process that takes less than a day.
- Policies are publicly documented, including:
    - A procedure for receiving, evaluating, reviewing, and incorporating code changes.
    - A process for creating new branches/forks for development or implementation.
    - A process for making policy changes.
- Documentation related to the code is public.
- A support contact or mechanism (e.g., forum) for the code with some backup is provided (i.e., not a personal email).
- An issue tracking mechanism is provided.
- Initial response times for support and issue tracking are generally less than a week, though resolution may be longer.

UFS repositories:

- There is a well-defined regression testing strategy, where applicable. The community needs to look at increasing unit testing capability where possible. The ability to run lightweight unit tests will allow applications to move more quickly from research into operations testing functional capability and conserving HPC resources.

The rules listed above can be used as a cookbook for fostering new projects which may emerge as the UFS application space expands.

Repository Types and Locations

The UFS repository strategy has two repository types:

- Umbrella repository
- Component repository

Each component repository contains the source code for a unique component of the UFS application and, where applicable, the NUOPC cap. The umbrella repository contains the

policies, documentation, and configurations required to link to the individual component repositories which, when brought together, define a given UFS application.

Umbrella Repository

An umbrella repository is essential to the UFS repository strategy and defines a unique UFS application. The umbrella repository must contain, at a minimum:

- Documentation for the application
- Configurations to obtain the required component repositories
- Policies and processes

Each umbrella repository will have a governance body to be established as UFS applications are identified and created. The goals of the governance body are to define the policies and procedures. The governance body will appoint gatekeepers to assist in the repository maintenance and enforcement of the policies. Included in the policies will be the branch structure and workflow. At a minimum, the branch structure should include:

- Main branch – Collection of approved changes from the community and operations
- Development branch(es) – Contain features in development or requested to be included in the main branch
- Operational branch(es) – Contain the current configuration and code used in operations
- Release branch(es) - Contain changes or updates from the operational or implementation branch that are merged into main
- Implementation branch(es) – Contain features currently in testing for future operational releases. Once approved, updates will be merged into an operational branch

The main and operational branch will reside within the authoritative umbrella repository. Development and implementation branches may reside in separate forks. The code management (CM) gatekeeper, following the repository policies, will work with the developers to incorporate the changes back into the authoritative umbrella repository. Some development and implementation branches may reside within the authoritative umbrella repository to help facilitate collaboration and testing. The EPIC team is looking to implement continuous releases in the future which will push the repositories to follow a different pattern, with a single authoritative branch and features, bugs, and hotfixes, will all merge onto a single branch.

Component Repository

The component repository is where component code (model, library, utilities, etc.) resides. It should also contain documentation, regression test (RT) procedures and the NUOPC cap, where applicable. Of note, the community is looking to increase unit test coverage to create more lightweight testing capabilities. The component repository must also have a governance body that decides and implements the repository policies. The component repository governance body

must also be willing to participate in community development, and work within the UFS repository policies and guidance.

While the branch structures of the component repository are defined by the component repository governance body, it is suggested the component repository use a similar branch scheme as the umbrella repository.

Prototyping the UFS Applications

The current UFS Applications are housed in ufs-community or NOAA institutional GitHub repositories. A presentation to the UFS Steering Committee resulted in a recommendation for the Repositories Sub-Group to prototype UFS applications — the UFS Regional Weather Forecast Application (now referred to as the SRW App) and the UFS Global Seasonal Prediction Application (now referred to by the community as MRW/S2S or “Global App”). Once the issues with the prototypes have been ironed out, the applications can be pushed to an open-development site, such as GitHub.

The prototyping process accounts only for the initial components comprising a specific UFS application. As the applications evolve and further components have been incorporated, they will be continuously added to the application umbrella repository (e.g., chemistry and aerosols, land, radiation, fire component, etc.)

UFS Regional and Global Weather Forecast Applications

The current UFS Weather Model and application GitHub repositories contain different levels of regression workflow testing systems. Subcomponent structures are coupled through the Community Mediator for Earth Prediction Systems (CMEPS) coupling component and referenced as GitHub submodules. The UFS Weather Model repository supports several UFS applications. It consists of the atmospheric Finite Volume Cubed Sphere (FV3) dynamical core; the Common Community Physics Package (CCPP); and ocean, sea ice, wave, land, aerosol, chemistry, and data model components with a central CMEPS mediator component to couple the components together. The coupling strategy among the components uses the common Earth System Prediction Suite (ESPS) architecture and an Earth System Modeling Framework (ESMF) and NUOPC-based coupling infrastructure framework. Each of these components has its own repository publicly accessible in GitHub to the broad community.

- Submodules: Air Quality Model(AQM), Community Data Models for Earth Prediction Systems (CDEPS), Community Ice CodE CICE), Common Make modulefiles (CMakeModules), CMEPS, Finite-Volume Cubed-Sphere (FV3) atmospheric component (fv3atm), Goddard Chemistry Aerosol Radiation and Transport (GOCART), Hybrid Coordinate Ocean Model (HYCOM), Modular Ocean Model version 6 (MOM6),

Noah-MP land surface model (NOAHMP), WAVEWATCH III (WW3), stochastic_physics

As the UFS source code repositories evolve, workflow is a critical system to meet the requirement of the regional and global data assimilation and forecast application processes. Authoritative workflow system repositories are expected to provide workflow management capabilities to efficiently set up, perform, and monitor interdependent jobs in a predefined order based on each UFS application. Broader UFS community support mandates workflow portability with enhanced documentation, flexibility, and user support. With the authoritative repositories in place for the model source code, application libraries, toolsets, and build system and workflow, the workflow umbrella repository can be structured to contain:

- Input to the build system for creating an executable for a specific compiler
- Information that tells the workflow to access a data portal and download Initial Conditions (ICs)
- Experiment configuration definition for use by the workflow
- Prototype workflow structure that may differ from the UFS Weather Model applications
- Key functionality of the workflow system: This key functionality builds on a combination of several components working together to prepare, analyze, produce, and post-process forecast data. The major components of the system are:
    - Workflow
    - Pre-processing
    - Analysis
    - Forecast
    - Post-processing
    - Verification

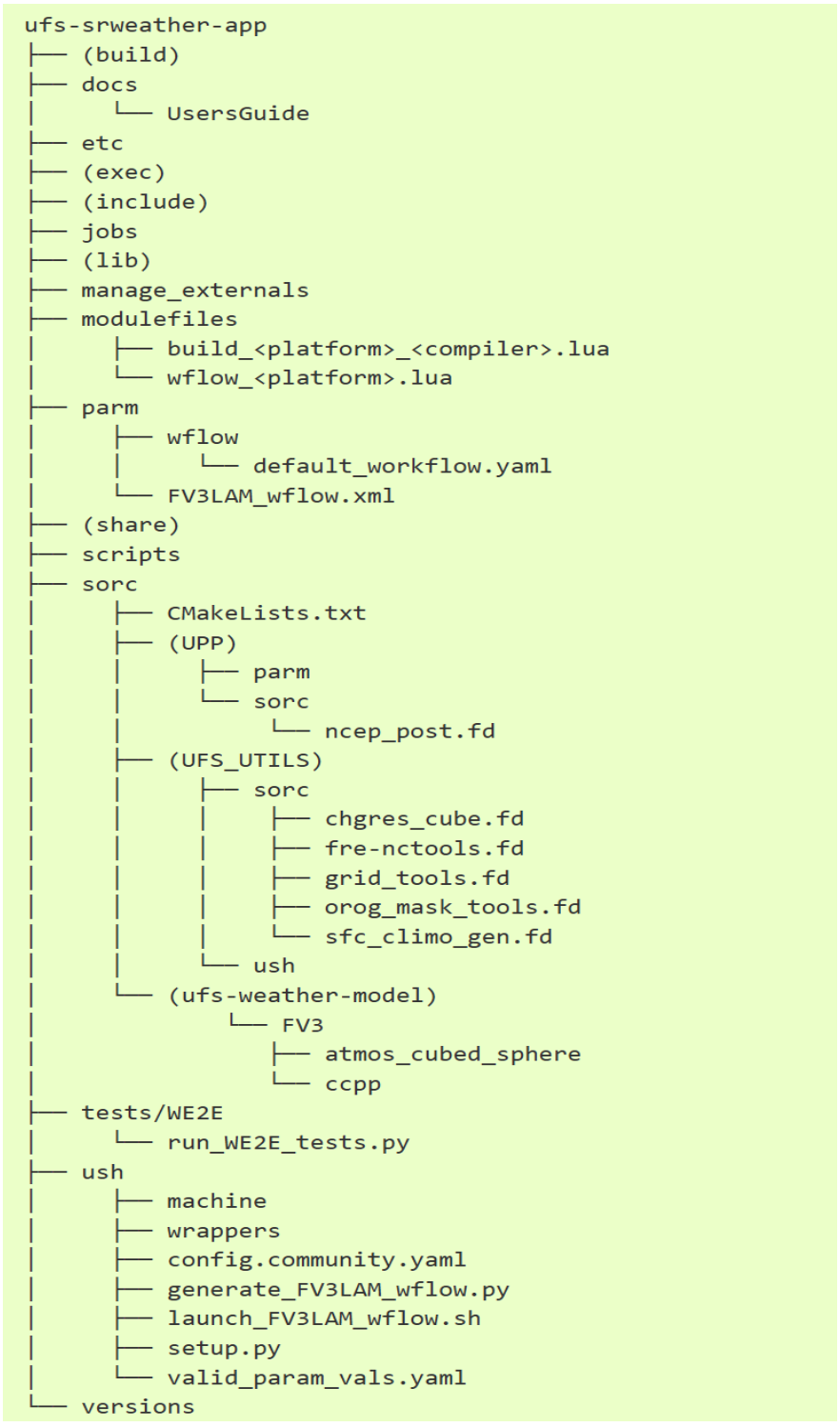

Figure 9: UFS SRW Application Directory Structure

UFS Application Software Governance

UFS Application Software Guidelines

Universal governance organization and system-wide development processes are defined in the document of the UFS Organization and Governance. Serving as the charter for the day-to-day CM operations, the document outlines the UFS software governance structure, based on key aspects of organizational structure and development processes: the Community Modeling Board (CMB), the UFS-SC, application and cross-cutting teams, component working groups, processes of code management, and code governance. For effective coordination of a large number of development teams, high-level code management and integration guidelines are set as follows.

- Do no harm: Considering the focus of the UFS on improving operational forecasting, any code update of the UFS that breaks operations is counterproductive. UFS code management therefore requires:
    - Continuous regression testing, not only on science output, but also on computational expenses and robustness/reliability.
    - Maintaining backward compatibility of an evolving code base (i.e., do not break what was already working).
    - Explicit concurrence from operations for any code change that breaks or has a clear negative impact on operations.
- Coding Standards: Modern, maintainable code requires coding standards. Whereas we do not intend to enforce UFS-wide coding standards, we expect all components to be defined as part of their governance. As part of this, and to be able to enforce “do no harm,” regression and unit testing are required for each UFS component.
- Ownership: Each contribution to the UFS or a part of its code base needs to have a designated entity responsible for its maintenance, with some costs incurred by NOAA, some are maintained by external organizations. For code used in NOAA operations, this often resides at or is funded by NOAA.

Application teams ensure the transition of research-to-operations (R2O) processes and community code releases. In the UFS R2O process, code convergence is recognized as a required vetting procedure to refactor the research-oriented code bases to an acceptable level of operational implementation standards. Ultimately, the goal will be to move toward a process flow that utilizes unit testing, linting, naging, regression testing, and other automated standards to ensure software is ready to move to operations. During the course of the UFS SRW workflow release activities, several challenging issues have been identified as root causes of code divergence.

- High-speed code integration of research code in authoritative repositories: new tools, different design patterns, etc.
- Different types of code integration processes are adopted in multi-site application repositories and branches.

- Lack of baseline reproducibility and regression testing validated across Research and Development HPC Systems (RDHPCS) and the Weather and Climate Operational Supercomputing System 2 (WCOSS2), for code integration.
- Knowledge management challenges and varying project timelines and priorities among distributed development organizations.
- Limited flexibility of the SRW workflow features to fully support operational requirements.

According to the CMB structure, the UFS-SC is responsible for providing overall UFS code compliance guidance based on the feedback from the application and cross-cutting teams and component working groups. From the UFS process perspectives, code convergence strategies and recommendations are summarized below to enhance the UFS R2O process.

Application Software Processes

Many groups and teams contribute to the development of the UFS WM and Applications. Code management and governance are largely distributed. In particular, distributed code ownership has been established in many of the component models. To further ensure code convergence and meet operational requirements, it is required to regularly review the UFS software management strategies and procedures.

UFS Code Management and Integration Process

In the UFS CM processes, software integrity and quality of incremental code changes are continuously tested to maintain the required baseline regression and workflow end-to-end (WE2E) tests. The UFS WM, SRW App, and Land DA GitHub repositories evolve with contributions from community developers who adopt the CM practices and CI/CD tools in the community-based development and operations (DevOps) processes. The main goal of the UFS Application test is to ensure a common code base infrastructure to facilitate and deliver new source code features, fixes, and updates in close alignment with the UFS operational and research and development (R&D) requirements. CM tasks include:

- Coordinating code integration and porting to supported platforms of the UFS WM and associated applications (SRW and Land DA) consistent with subcomponents; reviewing and tracking the UFS pull requests (PRs) daily, identifying issues and conflicts, and working with developers to ensure that PR problems are addressed.
- Communicating code management changes and status to subcomponent and application code managers in the UFS-related meetings, and coordinating the UFS WM and subcomponent code changes with workflow and downstream applications.
- Coordinating subcomponent repository updates and ensuring that subcomponents are regularly sync'd with authoritative repositories.

- Ensuring the UFS application testing framework reflects the latest development and code changes across the UFS applications every two weeks.
- Developing plans to improve the UFS testing frameworks to run tests efficiently and to ensure code quality; rebalancing the UFS test case workload and enhancing the UFS test systems with the hierarchical system development and test framework.
- Providing and archiving meeting minutes, and updating within 48 hours following every code management meeting.
- Supporting developers on sync'ing and merging Git repository branches, running the required UFS tests (i.e., WM regression and operational requirements tests, Land DA workflow WE2E test, and SRW comprehensive WE2E tests on the supported Tier-1 platforms.)

Deployment of Software Libraries

spack-stack provides a framework for installing software libraries to support the UFS WM and its applications. Deployment of software libraries involves:

- Evaluating new software and libraries through the UFS WM baseline test cases with developers and subject-matter experts and coordinating with the WCOSS2 library update and delivery schedule.
- Ensuring software beta version release tests, including system and library dependency tests, troubleshooting, and UFS subcomponent testing and validation.
- Consolidating software and library support requirements, including Python packages needed across workflow management and application repositories.

Enhancement of UFS Code Convergence

Avoiding code divergence requires applying operational implementation standards, maintaining consolidated baseline test cases, and unifying repository management strategies. Recommended action items for the SRW/RRFS workflow convergence are:

- Work as a team to define operational standards that can facilitate both operations and research.
    - Examples:
        - Develop script options to remove the workflow code bases that are not supportable in operational applications.
        - Build a directory structure to specify a landing space for research code, visualizations, and new features that are not yet ready to support operations.
- Work on a project plan to focus on a single workflow repository approach to support the UFS R2O process.
- Produce integration plan to merge SRW research workflow features to the operational RRFS repository.

- Establish code management team-level feedback to work through standard implementation and integration roadblocks.
- Ensure that the UFS R2O workflow re-alignment review process reflects the prioritization of innovative features for integration into operational applications.
- Rebalance and update UFS test systems and options in the hierarchical test framework, including RTs, functional tests, unit tests, linting, and code analysis.
- Add a platform deployment option for WCOSS2 Virtual Machine (VM) to mirror the production of applications in an operational workflow management environment.

UFS Application Software Summary

Leadership within NOAA and the UFS community are solidifying the UFS governance procedure. When the governance is complete and advertised to the community, updates will be made to reflect the strategic direction.

UFS Data Portal

Access to the data used for retrospective evaluation is a requirement for the community to contribute. While the amount of data needed is unknown at this time, it will be at least the previous 3 years of data (4 cycles per day) and include ICs for select major events. The data are not limited to initial conditions and must include the climatological and other forcings.

The Data Portal requires capital expenditures in hardware and manpower to achieve success. The best approach is for this to be addressed by the NOAA/National Center for Atmospheric Research (NCAR) Memorandum Of Understanding (MOU).

Data to Support the Development of UFS Apps on the Cloud

- SRW App data bucket: https://registry.opendata.aws/noaa-ufs-shortrangeweather/
- Land-DA System data bucket: https://registry.opendata.aws/noaa-ufs-land-da/
- GDAS Test Data: https://registry.opendata.aws/noaa-ufs-gdas-pds/
- UFS HTF Data: https://registry.opendata.aws/noaa-ufs-htf-pds/
- Hurricane Analysis and Forecast System (HAFS) Data: https://registry.opendata.aws/noaa-nws-hafs/
- UFS WM RT: https://registry.opendata.aws/noaa-ufs-regtests/
- Artificial Intelligence (AI) models:
    - UFS GEFSv13 replay 0.25 degree subsampled
    - UFS GEFSv13 replay 0.25 degree
    - UFS GEFSv13 replay 1.0 degree

Community Workflow

Ideally, the UFS workflow would have a number of key features. It would 1) satisfy operational requirements; 2) enable the research community to run and reconfigure the various UFS applications easily; and 3) share code as much as possible across applications. The Community Research and Operations Workflow (CROW) was initiated in Fiscal Year 2017 (FY17) to address these goals, but so far has focused on the operational aspects of the workflow. There is an effort in the Hurricane Supplemental to improve usability, portability, and hierarchical testing capabilities of the operational workflow by integrating elements of the Community Infrastructure for Modeling the Earth (CIME) tools used with NCAR and Department of Energy (DOE) models into the evolving CROW toolchain. The initial focus will be on the UFS Seasonal Prediction application HAFS that is currently being planned. First milestones include only the prognostic model; data assimilation, postprocessing, and other parts of the workflow will be addressed later.

Specific requirements are to ensure that "research-oriented aspects of the system should be usable in non-NCEP environments" and to ensure cross-platform portability[5]. Since HAFS and UFS-Seasonal will require active coupling of FV3GFS to separate ocean and wave components, the workflow needs to support system as well as unit testing and also include the ability to isolate feedbacks in the coupled system. Basic verification capabilities are needed as well to validate porting of HAFS forecast components between systems, both within NCEP and external to it. The tasks proposed are to integrate elements of CIME that address these needs into CROW.

SRW-RRFS Convergence Tiger Team Report

The SRW-RRF Tiger team recommends the following:
The Minimum Viable Product (MVP) project plan includes two Course of Actions (COAs):

a. COA1: Trial MVP that will merge in SRW features to the operational RRFS repository.
    i. Allow the team to work through roadblocks in standards and integration.
    ii. Allow the team to realize if this is a path that will work going forward.
    iii. This is similar to the approach HAFS/Hurricane Weather Research & Forecasting HWRF) model has taken, which is a proven process for community and operations.
b. COA2: Series of PRs to SRW that reduce the complexity of the run scripts to call uwtools and rely on the structured UW YAML configuration.
    i. Allow the team to work through roadblocks in standards and integration.
    ii. Allow the team to realize if this is a path that will work going forward.

[5] Unified Workflow Project Planning Document:
https://docs.google.com/document/d/1i8KzKvZULvXs9unVUZWwz-TPSd-FOTyrc30LyDRFWgM/edit#

The MVPs have been delayed until RRFSv1 hits a decision point on going to operations, and the supporting teams have the capacity to execute and post their results.

UWF Integration Plans

The Unified Workflow Tools' next integration needs to have an implementation on WCOSS2. The UWF team and the EPIC are putting together an implementation plan to get uw-tools deployed on WCOSS2 and Acorn. UW tools have prepared for easy installation of various versions by deploying them to conda (https://anaconda.org/ufs-community/uwtools).

Appendix A: Glossary

Repository – A specific location referenced via URL or other identifier that acts as an archive with tracking capability managed through some version control system (e.g. SVN or git). A software development repository may contain source code, an integrated regression test suite/system, documentation, etc.

Authoritative repository – A repository defined by the presence of a governance group and well-defined processes to manage development and periodic releases of reference versions. It exists as a definitive source for a given software development project and must contain a regression suite and documentation.

Component repository – A repository that contains a minimum source code for a (model) component.

Umbrella repository – A repository that contains references to external repositories, but no source code.

UFS application or app – model that falls under the definition put forth by the UFS Steering Committee. Examples include: Forecast App, Seasonal Prediction App, Regional App, Chemistry Transport App etc.

Umbrella Code – One or more configuration files that define a composite application through externals.

UFS Weather Forecast App – Global short-range, weather-prediction model currently known as *NEMSfv3gfs* or *FV3GFS* used to create forecasts.

UFS Seasonal Prediction App – Global coupled model, specific to the seasonal timeframes, composed of *FV3 dynamical core*, with appropriate physical parameterizations, coupled to *MOM6*, *CICE5*, and *WaveWatch III.*

UFS Regional App – Limited-area model based on the *FV3 dynamical core,* combined with suitable physical parameterizations, for undertaking detailed studies of a particular region or phenomena.

UFS Chemistry Transport App – Global model for Ozone Chemistry Aerosol Radiation and Transport for prediction of air quality and related phenomena.

Gatekeeper – Person responsible for managing releases in operational and implementation branches within a UFS application umbrella repository. This person is also responsible for enforcing documentation policies and tagging the updates to the trunk of the umbrella repository during a development cycle (e.g. new microphysics)

Appendix B: Acronym List

Acronym	Definition
AI	Artificial Intelligence
AQM	Air Quality Model
AR	Atmospheric Rivers
AWS	Amazon Web Services
CAC	Common Access Card
CCPP	Common Community Physics Package
CCT	Cross Cutting Team
CI/CD	Continuous Integration / Continuous Deployment
CICE	Community Ice CodE
CIME	Community Infrastructure for Modeling the Earth
CM	Code management
CMAQ	Community Multiscale Air Quality model
CMB	Community Modeling Board
CMEPS	Community Mediator for Earth Prediction Systems
CDEPS	Community Data Models for Earth Prediction Systems
COA	Course of Action
CPU	Central Processing Unit
CROW	Community Research and Operations Workflow
DA	Data Assimilation
DevOps	Development and Operations
DOE	Department of Energy

DTC	Developmental Testbed Center
EE2	Environmental Equivalence 2
EMC	Environmental Modeling Center
EPA	Environmental Protection Agency
EPIC	Earth Prediction Innovation Center
ESCOMP	Earth System Community Modeling Portal
ESMF	Earth System Modeling Framework
ESPS	Earth System Prediction Suite
FV3	Finite-Volume Cubed-Sphere
FV3atm	FV3 atmospheric component
GCP	Google Cloud Platform
GDAS	Global Data Assimilation System
GFDL	Geophysical Fluid Dynamics Laboratory
GFS	Global Forecast System
GMU	George Mason University
GOCART	Goddard Chemistry Aerosol Radiation and Transport
GPU	Graphics Processing Unit
GSI	Gridpoint Statistical Interpolation
GSL	Global Systems Laboratory
GW or GWF	Global Workflow
HAFS	Hurricane Advanced Forecast System
HFIP	Hurricane Forecast Improvement Program
HPC	High Performance Computing

HTF	Hierarchical Testing Framework
HWRF	Hurricane WRF
HYCOM	Hybrid Coordinate Ocean Model
IC	Initial Condition
IPD	Interoperable Physics Driver
JCSDA	Joint Center for Satellite Data Assimilation
JEDI	Joint Effort for Data assimilation Integration
Land DA	Land Data Assimilation
METplus	Model Evaluation Tools companion package
MOU	Memorandum of Understanding
MOM	Modular Ocean Model
MPAS	Model for Prediction Across Scales
MRW	Medium-Range Weather
MSU	Mississippi State University
MVP	Minimum Viable Product
NCAR	National Center for Atmospheric Research
NCEP	National Centers for Environmental Prediction
NCO	NCEP Central Operations
NEMS	NOAA Environmental Modeling System
NGGPS	Next Generation Global Prediction System
NOAA	National Oceanographic and Atmospheric Administration
NoahMP	Noah-MP land surface model
NODD	NOAA Open Data Dissemination

NOS	National Ocean Service
NRL	Naval Research Laboratory
NUOPC	National Unified Operational Prediction Capability
NWP	Numerical Weather Prediction
O2R	Operations to Research
PR	Pull Requests
PSL	Physical Sciences Laboratory
R&D	Research and Development
RDHPCS	Research and Development HPC System
R2O	Research to Operations
R2O2R	Research to Operations to Research
RRFS	Rapid Refresh Forecasting System
RT	Regression Test
S2S	Subseasonal-to-Seasonal
SAI	Systems Architecture and Infrastructure
SRW	Short-Range Weather
UFS	Unified Forecast System
UFS-SC	UFS Steering Committee
UPP	Unified Post Processor
UW or UWF	Unified Workflow
VM	Virtual Machine
WCOSS2	Weather and Climate Operational Supercomputing System 2
WE2E	Workflow End to End

WM	Weather Model
WRF	Weather Research & Forecasting Model
WW3	WaveWatch III

Appendix C: Actors

There are a number of different actors, or roles, within the repository strategy. They are divided into three main groups:

1. Non-contributors - People who want to download and run the code, or expect to make local changes only.
2. People who change the code:
    - Application developers - Developers who change science or technical code in order to improve it scientifically or computationally - no distinction is made at this time between EMC and other application developers, with the expectation that development processes may be the same for both
    - Integrators - People who change application code in order to transition an application into a full operational workflow, typically working on a restricted operational computer (e.g. WCOSS)
    - Testers - People who test the code
3. People who make decisions about code changes:
    - Governance bodies of authoritative repositories who set policies
    - Reviewers of code changes
    - UFS application leads
    - UFS code managers
    - UFS Working Groups
    - Field offices (they approve features coming into operations)

Appendix D: Use Cases

Case 1: Adding new physics capability to UFS Weather Forecast Application

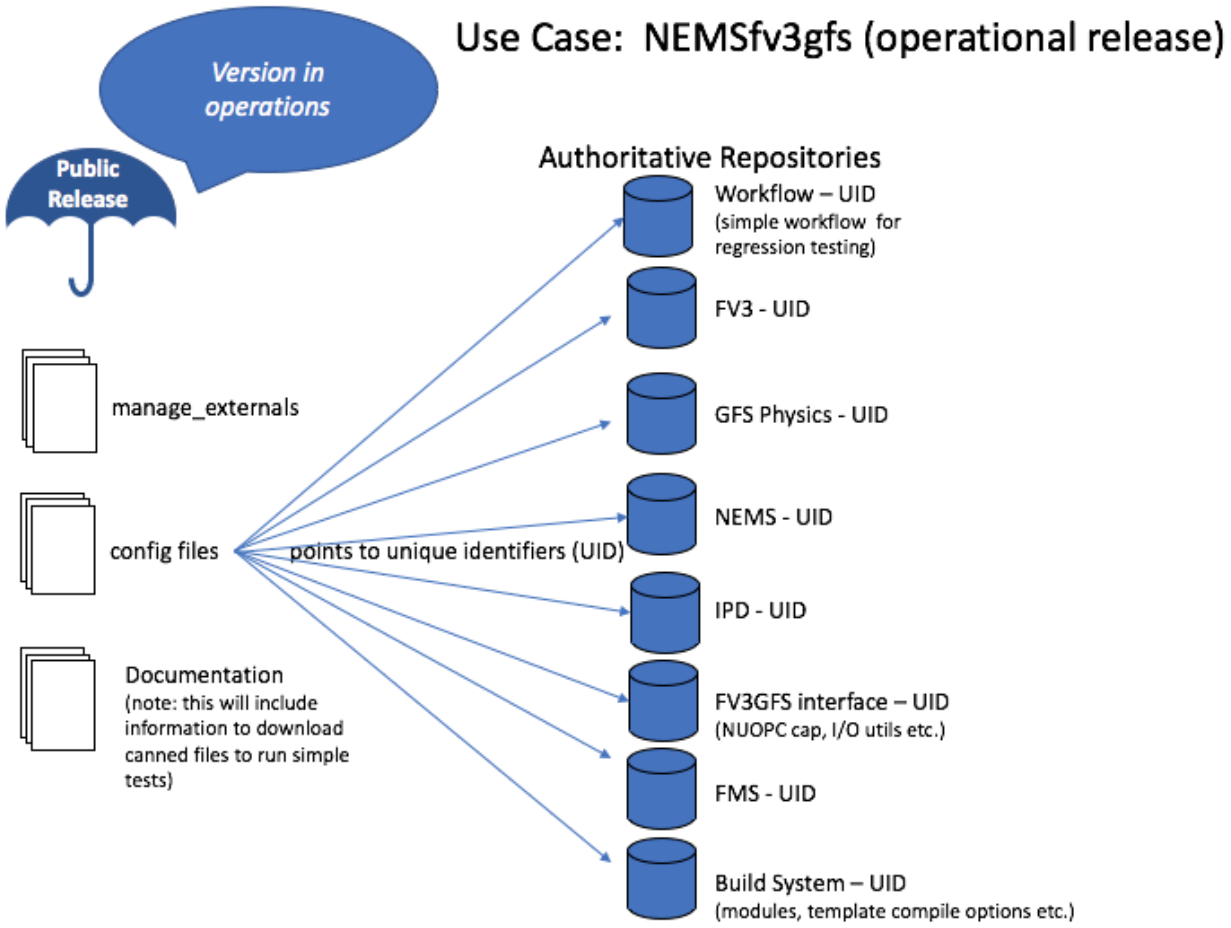

Figure 10: Umbrella repository for NEMSFV3GFS use case

Figure 10 provides a use case for the UFS Weather Forecast App. This represents the global atmospheric modeling system. Each of the cylinders in the figure represent an authoritative repository. For this application the umbrella repository connects to repositories for workflow, FV3 dycore, physics, physics driver etc. The umbrella repository here represents an operational system. The unique identifiers indicate a specific snapshot within an authoritative repository. Note, this is the representation of the operational system in the community UFS umbrella repository that is available to the public. As has been stated earlier, the operational repository that is placed inside the NCO managed server may have a different look. It is the responsibility of the development organization tasked with transition to ensure the source codes remain synchronized.

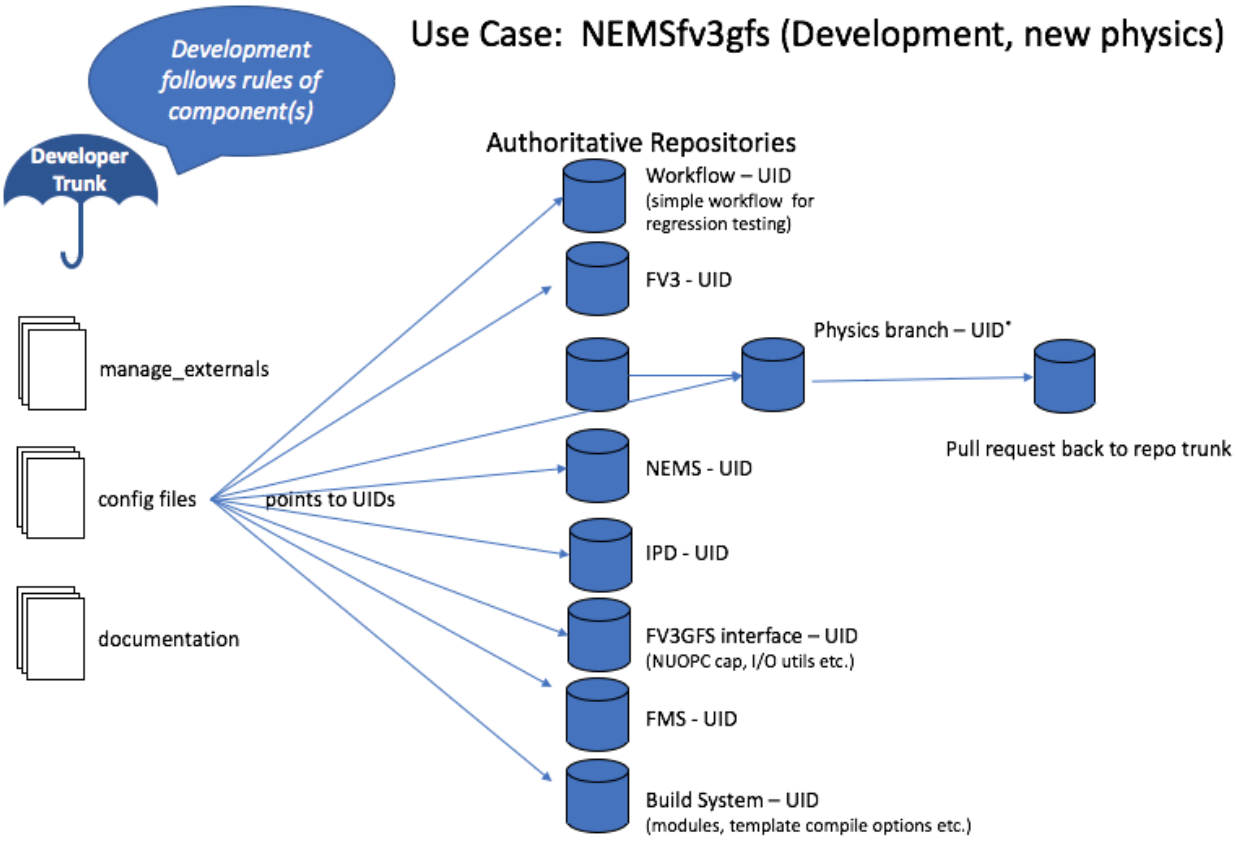

Figure 11: Development fork for UFS Weather Forecast Application

Figure 11 demonstrates a particular development flow for the UFS Weather Forecast Application. The starting point is the main development trunk. The developer(s) in this case want to develop a new physics package. They start from the main trunk and create a fork for their development. The umbrella fork will then be changed to point to a corresponding fork/branch that has been created in the Physics repository. Once testing is completed and changes are approved then a pull request is made to update the Physics repository. The main trunk is then updated to represent the updated tag in the umbrella repository.

Case 2: A Principal Investigator (PI) received an NGGPS award to implement a skin temperature parameterization scheme in a UFS application. This scheme was tested in CESM and showed improvements in the sub-seasonal forecast skill. The parameterization will be implemented in the coupled modeling system

- The starting point for this will be the UFS Seasonal Prediction Application. This application will look like the UFS Weather Forecast Application shown in Figures 5 and 6 but will include additional components for ocean (MOM6), ice (CICE5), waves (WaveWatch III) and land (NOAH-MP via LIS. Note: Land may still be coupled via Physics driver as opposed to an external component. That determination has not been completed yet).
- The developer(s) will provide their development plan to the governance body of the UFS seasonal application (who in turn will brief the UFS Steering Committee) as well as the code managers of the authoritative repositories to make them aware of their approach.

- They will begin with creating a development fork from the master of the appropriate application umbrella repository
- The developers will follow the rules of the relevant authoritative component repository(-ies) and create new workspaces where they will carry out the work for their project (in this case the developer would create forks of the Physics repository and possibly the ocean and wave repositories for impact on upper ocean mixing). They will then update the connections in their umbrella development fork to point to the appropriate branches/forks of the component repositories where they will be doing work.
- The seasonal prediction application will then be tested with their updates.
- The viability of the science will be shown to UFS Steering Committee and appropriate area-specific working group. If approved, a path will be identified to bring these changes into the master repositories of the model component repos.
- The developers will begin the process of coordinating their updates back into the authoritative repository. They will be responsible to ensure that
  a) their updates work with the top of the master
  b) that they do not break other applications reliant upon this component
  This is only possible with constant communication between the PI, code managers of all the repositories being touched, and the area-specific working groups responsible for evaluating the science.
- Once new features have been added into the master of the respective component repos, they will be readily available for implementation into operation. Use in operations will depend on satisfying implementation requirements which vary from application to application.

Case 3: Application developers move a UFS application to NCO for operational evaluation.

- Application developers and application leads define a version of a UFS application to go towards implementation.
- Implementation forks/branches are created in component repositories to isolate them from general development.
- An implementation fork is created in the umbrella repository which points to the individual unique identifiers for components created in the previous step.
- Integrators modify applications to run within an operational environment and carry out detailed evaluations with the forecasters in the field, who are using model guidance to issue forecasts and assess the impacts on downstream operational models.
- Approved implementations are tagged and transitioned to operations.
- NCO checks out the entire code and stores it in a separate operational repository (without externals).
- Once the application is in operations an operational release is created in the UFS umbrella repository. [NOTE: All components that make up this application will have a corresponding operational identifier as well.]
- NCO may need to make bug fixes during the life cycle of an operational application version. Corresponding changes will also need to be returned to the UFS operational fork that is visible to the community. That is why each operational implementation version should have its own so that we can track future minor bug upgrades.

- Updates made to individual component repos between beginning of implementation and the time the model goes into operations are brought back into the main repositories, thus ensuring a viable R2O and Operations to Research (O2R) feedback.

Case 4: NCO must rapidly address an error in a component model of a UFS application.

- NCO will only make updates to operational repository during the operational life cycle of that modeling system
- The Implementation team ensures that any emergency updates made to operational codes are reflected in the respective UFS repository structure (this can only be done manually as there is physical separation between development and operational repositories).

Case 5: A non-contributor obtains, runs, and verifies an operational version of an application.

- The non-contributor determines which version of the application to obtain and how to obtain it by looking at documentation on a UFS community website.
- The non-contributor goes to the operational branch of the application's umbrella repository and checks out the source code for the specific application version, a workflow including initial conditions, and diagnostics for verification.
- The non-contributor is able to run and verify correct operation of the operational version of the application.

Case 6: Cross Agency Interaction in an Authoritative Repository (the use case for Institutional forks)

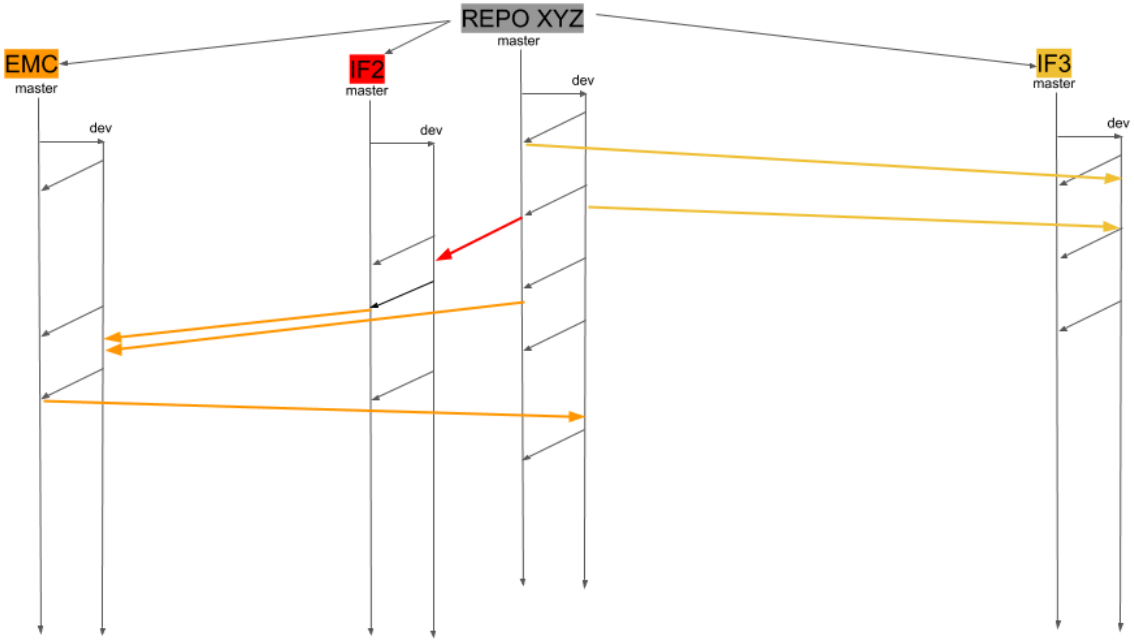

Figure 12: Repository representation

A significant source of worry is how future interactions work across agencies with different goals and who will ensure that work done in one place does not break or inhibit progress made in other places. Also, with a community repository, with multiple developers, how will best practice approaches be maintained across the board ?

The way to do this is to have institutional “blessed” forks for the authoritative repositories. Figure 12 illustrates this process. There will only be a single authoritative repository. This repository will have a master and development branch in accordance with git flow (see using git flow document for details) rules. All public releases of the codes are made from the authoritative repository alone. Apart from that there will be a finite number of trusted institutional repositories that are forked from the authoritative repository. The code managers at all these different repositories will be in constant communication with each other to ensure that development activities are kept in sync with each other. The institutional forks can sync with the development or master trunk of the authoritative repository. However, any updates that the institution makes will have to first go to the development branch of the authoritative repository, where it will be extensively tested before being moved to the master branch. This approach has a number of advantages. First, the institutions can isolate themselves to an extent, and prevent tripping over each other’s work without realizing it. And, second this makes repository management a lot easier with the institutional forks doing a significant amount of code testing and conflict management across developers. External developers who are not part of the specific institutes can choose to work with either the authoritative repository or with any of the institutional repositories.

Umbrella repositories that make up the UFS applications will connect to these repositories. For development they can connect to either the authoritative repository or the institutional fork, depending upon the development stage. It is ideal that any operational release be done through the authoritative repository, so developers will have to work with each other to ensure that this occurs. **This will ensure not only that we have a clear R2O but an O2R path as well**.

Appendix E: Application Architecture Diagrams

- SRW Tier -1: TBD
- Land-DA Tier-1: TBD
- GDAS: TBD
- HTF: TBD
- (HAFS) : TBD
- MRW: TBD
- UFS-WM Regression Test (RT): TBD
- Land DA: TBD
- SRW (including RRFS cap.): TBD
- Artificial Intelligence (AI) models: TBD

Appendix F: FAQ

Q. Will the UFS umbrella repositories sit in the open development environment?

A. *The UFS umbrella repositories must exist in an open development environment to foster community involvement.*

Q. Will the UFS umbrella repository allow anyone to mimic what is in operations?

A. *While every attempt will be made to provide operational versions to the community, there may be cases where this is not feasible. Examples include operational datasets and/or source code with restricted distribution rights.*

Q. Will all UFS applications be available via a single UFS repository?

A. *No, there will be one UFS umbrella repository per UFS application.*

Q. Will others be allowed to modify the configuration files within an umbrella project? If yes, will we enforce forking (assuming GitHub is the central repository service)?

A. *Yes. This is necessary for the community to share developments.*

Q. What would a UFS umbrella repository look like?

A. *Each UFS umbrella repository will contain a NCAR-developed repository management tool, a series of configuration files, and documentation. There will be no source code directly managed within the UFS umbrella repository with the repository management tool acting as a "broker" to pull in the necessary pieces specified within one of the configuration files. The result is a hub and spoke design with the spokes referencing authoritative repositories for individual components - be they model or infrastructure (community workflow, supporting application libraries, etc.). A baseline test case will be provided, with the remaining configuration files as inputs to the community workflow.*

Q. Is a UFS umbrella repository limited to containing only links to source code?

A. *No. All of the elements necessary for execution of a given UFS application, will be provided in some fashion - including supporting application libraries and access to initial conditions and other necessary datasets via a publicly-accessible data portal.*

Q. How does one define what applications warrant inclusion in a UFS umbrella repository?

A. *This is an important point and needs to be well-defined as otherwise the number of UFS applications can grow and become unmanageable. Since the UFS is being designed as the "operational" forecast guidance model for the weather service, only those modeling applications that are already in operations or have a path to operations will be considered for UFS umbrella repositories. It is not clear to us yet if this should be further limited to global systems.*

There is a separate document dedicated to defining the UFS is over here.

Q. How will operations be separated from development?

A. *The transition to operations will occur based on the science goals expressed by NCEP with inputs provided by the UMAC, the UFS-SC and the field offices. The state of the art available from the individual model components will be evaluated for inclusion based on the testing done by the PIs and evaluations of the UFS area-specific working groups. The work of integration and operational testing may occur in a private repository, but the resulting operational configuration will be made available to the community within the appropriate UFS umbrella repository.*

Q. What if new development degrades operational skill in a particular component?

A. *This will have to be addressed on a case by case basis. Our understanding is the governance group of individual component modeling systems that will make up the UFS are invested in making their modeling systems work in operations. In the case that new science development does not help operations the community must work together to find a solution.*

Q. How will new developments be integrated into operations?

A. *The candidates for transition to operations will be based on the science goals expressed by the NCEP with inputs provided by the UMAC, the UFS-SC, the individual UFS area-specific working groups and the operational folks in the field. Evidence-based decisions based on testing criteria will be used to determine which advancements will become candidates for T20. However, the final say for features going into operations will be determined by processes that are already in place at EMC for transition to operations. The final decision about anything going into operations is made by the NCEP Director, with input from NCEP, NCO and the Field Offices. See the EE2 document here for transition to operations.*

Q. What are the resources needed for maintaining umbrella repositories?

A. *This depends on the applications that make up the UFS and their complexity. As a rule of thumb, the more complex the development repository and the number of active developers, the larger the number of code managers needed. For example, the WAVEWATCH III repository has a single code manager, but the audience is getting large enough the governance committee is thinking of adding another code manager. For MOM6, the code management duties are split between 2 people. Back of the envelope calculations for the Weather Forecast app and the Seasonal Prediction app indicates a 6 - 10 Full Time Equivalent (FTE) effort. This does not include the integration team needed to test the end-to-end modeling systems. This is only sustainable if all the agencies involved make a commitment to share the support functionality*

Q. How will the UFS promote best practices for code management?

A. *There are multiple best practices guides out there and all the individual repositories have set up their own best practices that we can derive from. However, this is a good time to set up an overarching best practices guide. This does not have to start from scratch as smarter people have already solved a lot of these problems. For example, GITFLOW is a branch development technique that is very popular in development practices. There is a nice explanation here. And it comes with a set of tools for software management. This is not the only way to do things, but it is one way and a good way to do it. There is expectation of some requirements on the authoritative repositories, but the specifics have not yet been determined.*

Q. Who would be responsible for creating/maintaining the umbrella projects?

A. *TBD*

Q. Is there expected to be an authoritative repository for each model component? Where are model component issues expected to be submitted?

A. *Yes. Model component issues are expected to be resolved in the respective repository where the issue is being tracked. In the event a model component does not exist in an open repository but there is an expectation of active development by the community, an open repository should be created to hold periodic releases with approved community development manually merged into a given release. For utilities (application libraries, etc.) that are not involved in active development we will rely on public releases.*

Q. Do we need to consider the build process as part of the info in the various projects? If so, how would that work? If not, is that part of the community workflow?

A. *Build system will definitely be part of workflow but it will also be part of the corresponding model component. For example, WAVEWATCH III is a stand alone community model that is supported in multiple platforms and thus has the build options for different platforms that the community has put in over the years. But it also has a build system in the NEMS infrastructure to build the coupled system. The latter is derived from what is in the former.*

Q. What about input datasets, namelists, specification files, etc - would those also be included within the umbrella projects? What about sample test cases and regression testing needs?

A. *Although the individual component models will contain regression tests that must be run as part of development, it is imperative each application include access to a regression system for use by those interested in development. The regression system will not be static and developers should update as appropriate.*

Q. Will documentation concerning the UFS application be part of the UFS umbrella repository?

A. *Yes. Application-specific documentation is a key aspect of success and it is expected to be done in a "living" fashion. As developers share their work within a UFS umbrella repository, the code manager should ensure documentation is updated as appropriate.*