Scripts I

HWRF Python Scripts Training

Miami, FL

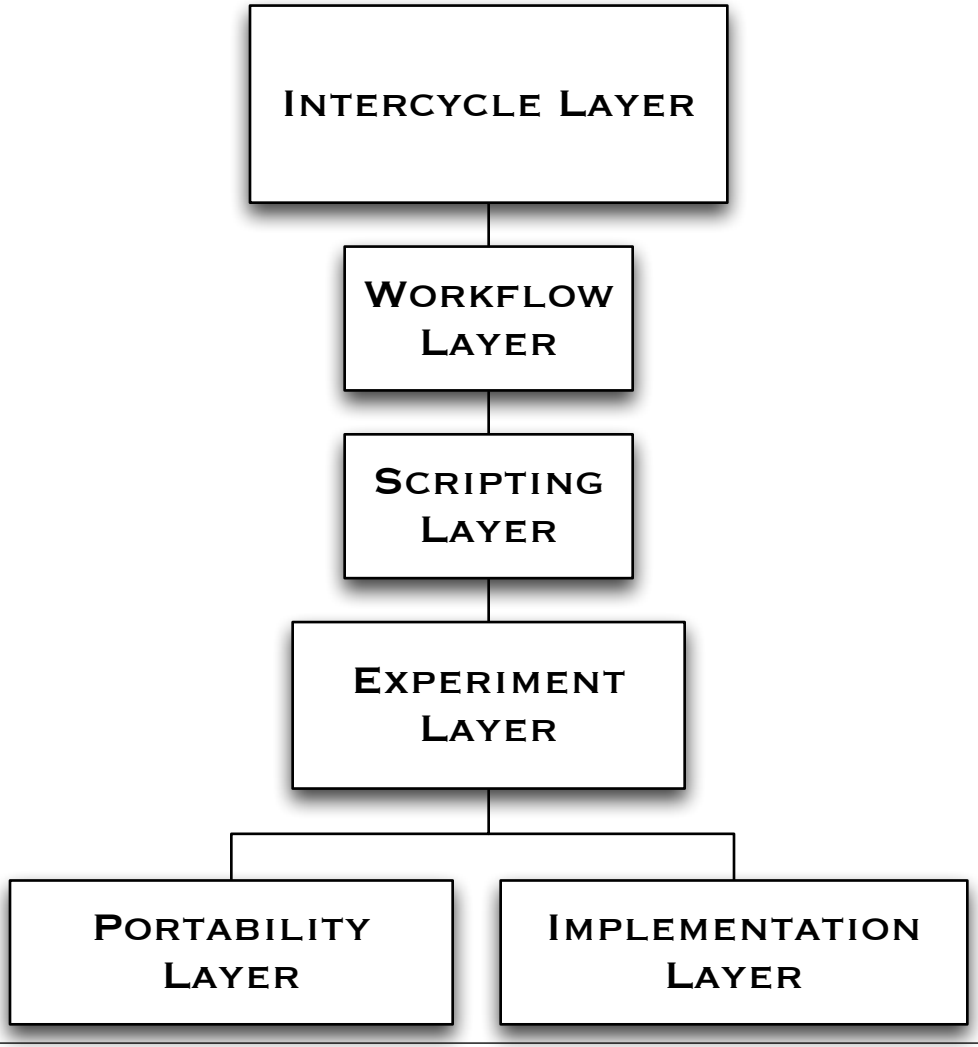
November 19, 2015

Adapted from the HWRF scripting system documentation available with each checkout and online at http://www.emc.ncep.noaa.gov/HWRF/scripts/index.html

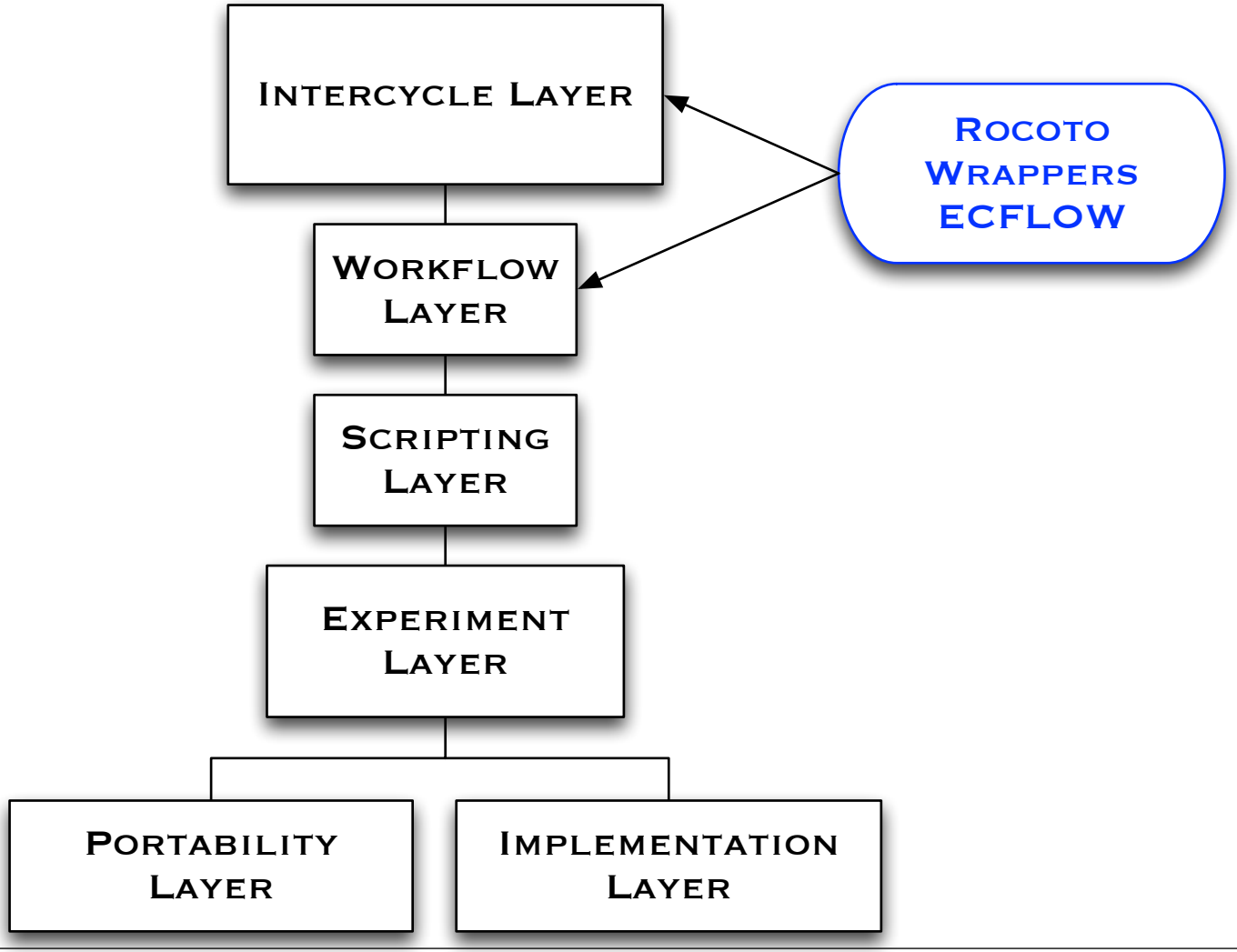
Motivation

- Wasted manpower in parallel script maintenance
- Difficulty in porting to new batch systems, workflow management systems, and operating systems, and running new configurations
- PyHWRF made it possible to unify the scripts among many organizations, minimizing the work needed to modify it

HWRF System Overview



HWRF System Overview



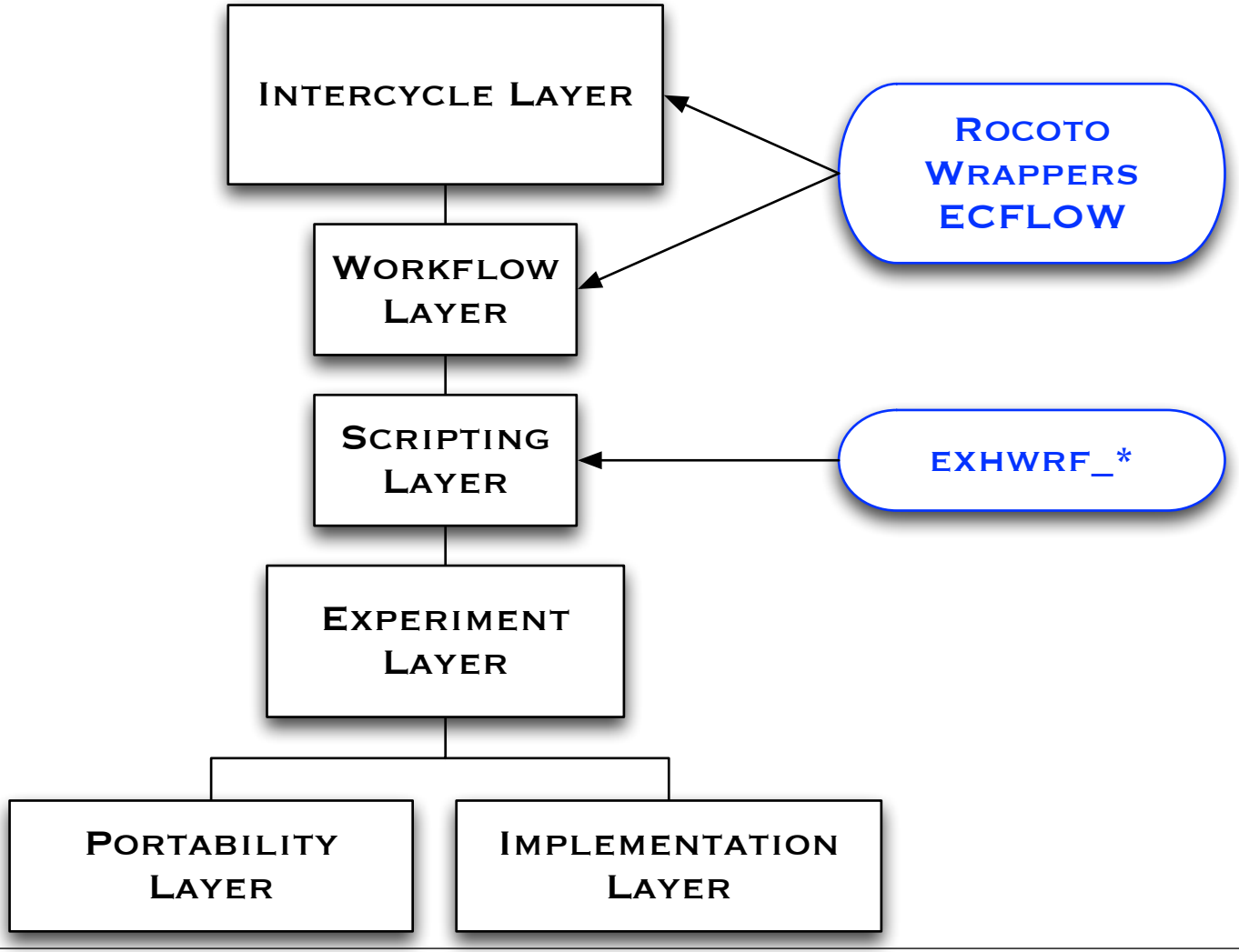
Intercycle Layer

- Handles interactions between several cycles
  - Complex dependencies
  - Files passed between them
  - Archiving
  - Scrubbing
- Rocoto automation takes care of these items
- Not needed for a case study
- Critical for a large retrospective study, and for real-time automation

Workflow Layer

- Splits work into multiple batch jobs
- Handles dependencies, submission, failures, and resubmission of jobs

HWRF System Overview



Scripting Layer

- Loads programs and libraries into computing environment
- Ensures connection to filesystem on compute node
- Pass file and executable locations to the next lower layer
- Layer is optional – can be done manually by user
- Standard template
  - Initialize produtil.setup and hwrf_expt Python modules
  - Log a message to jlog saying that the script is starting
  - Import 1+ hwrf.hwrftask.HWRFTask objects and call run() methods
  - Log a message to jlog of success/failure

Exercise

Open exhwrf_init.py

Discuss complexity of a script

Scripts Overview

- Launcher
- Data Pulling/Pushing
- Initialization
- Forecasting & Post-processing
- Data Assimilation Ensemble

Launcher

- Creates the directory structure
- Creates database files and other critical initial files needed by the HWRF system
- Run sanity checks to see if the system will be able to run the requested configuration

Data Pulling/Pushing

- **exhwrf_input** — pulls data for input to HWRF
- **exhwrf_bufrprep** — turns data tanks into bufr files for GSIs consumption
- **exhwrf_para_archive** — pushes data to tape after HWRF has finished
- **exhwrf_wrfout_archive** — special extra archiving job for native wrfout files
  - Compresses wrfout's before archiving

Initialization

- **exhwrf_ocean_init** — generates ocean initial and boundary conditions.
- **exhwrf_init** — spectral processing, interpolation and creation of wrfanl, ghost, wrfinput, wrfbdy and other input files
- **exhwrf_relocate** — takes input from the exhwrf_init job, and relocates the vortex, resizing it and changing its intensity if needed
- **exhwrf_gsi** — runs the GSI data assimilation system on the relocated vortex
- **exhwrf_merge** — merges the relocated vortex and GSI output to create the final input to WRF

Forecast/Post-processing

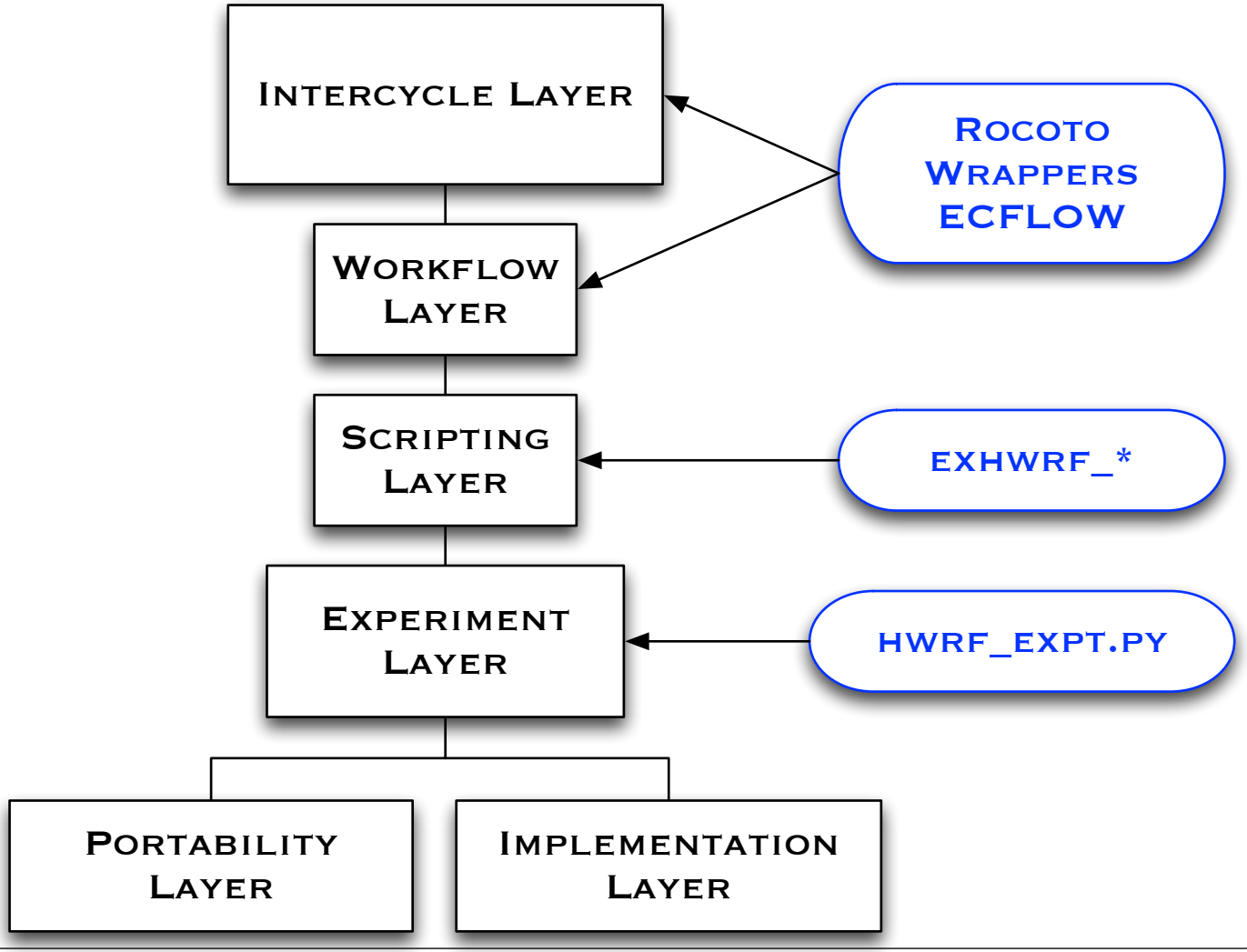
- **exhwrf_gsi_post** — post-processes the inputs and outputs of GSI to create lat-lon GRIB2 files suitable for study.
  - This is for examining the effect of data assimilation on the input conditions to the forecast.
- **exhwrf_forecast** — runs the full-length forecast, either with or without ocean coupling.
  - Takes inputs from the exhwrf_ocean_init, exhwrf_init, exhwrf_relocate and exhwrf_merge jobs.
- **exhwrf_unpost** — deletes the output of the exhwrf_post, exhwrf_products and some of exhwrf_output, allowing the post-processing to be redone.
- **exhwrf_post** — runs the Unified Post Processor on the output of the exhwrf_forecast to create native grid GRIB files.
- **exhwrf_products** — runs GRIB regridding utilities on the output of the exhwrf_post to create lat-lon GRIB2 output files suitable for use by forecasters.
  - Runs the GFDL vortex tracker to create a hurricane track file
- **exhwrf_output** — delivers the output of the exhwrf_products and exhwrf_forecast to their destination

Data Assimilation Ensemble

- **Set of scripts that handle the 6 hour HWRF ensemble forecast based on the GFS EnKF**
- **Output of the ensemble is used by GSI in the next cycle for the computation of the forecast error covariance**

exhwrf_ensda_pre — determines if the ensemble should be run
exhwrf_ensda — runs one member of the forecast ensemble (hwrf.ensda)
exhwrf_ensda_output — checks to see if the exhwrf_ensda scripts all completed

HWRF System Overview



Experiment Layer

- Describes the HWRF workflow
- Creates the object structure that connects all the pieces
  - i.e. GSI should use input from the GDAS relocation output
  - Each object has a run() function to perform the actual task
- Instantiates the hwrf_expt module

hwrf_expt module

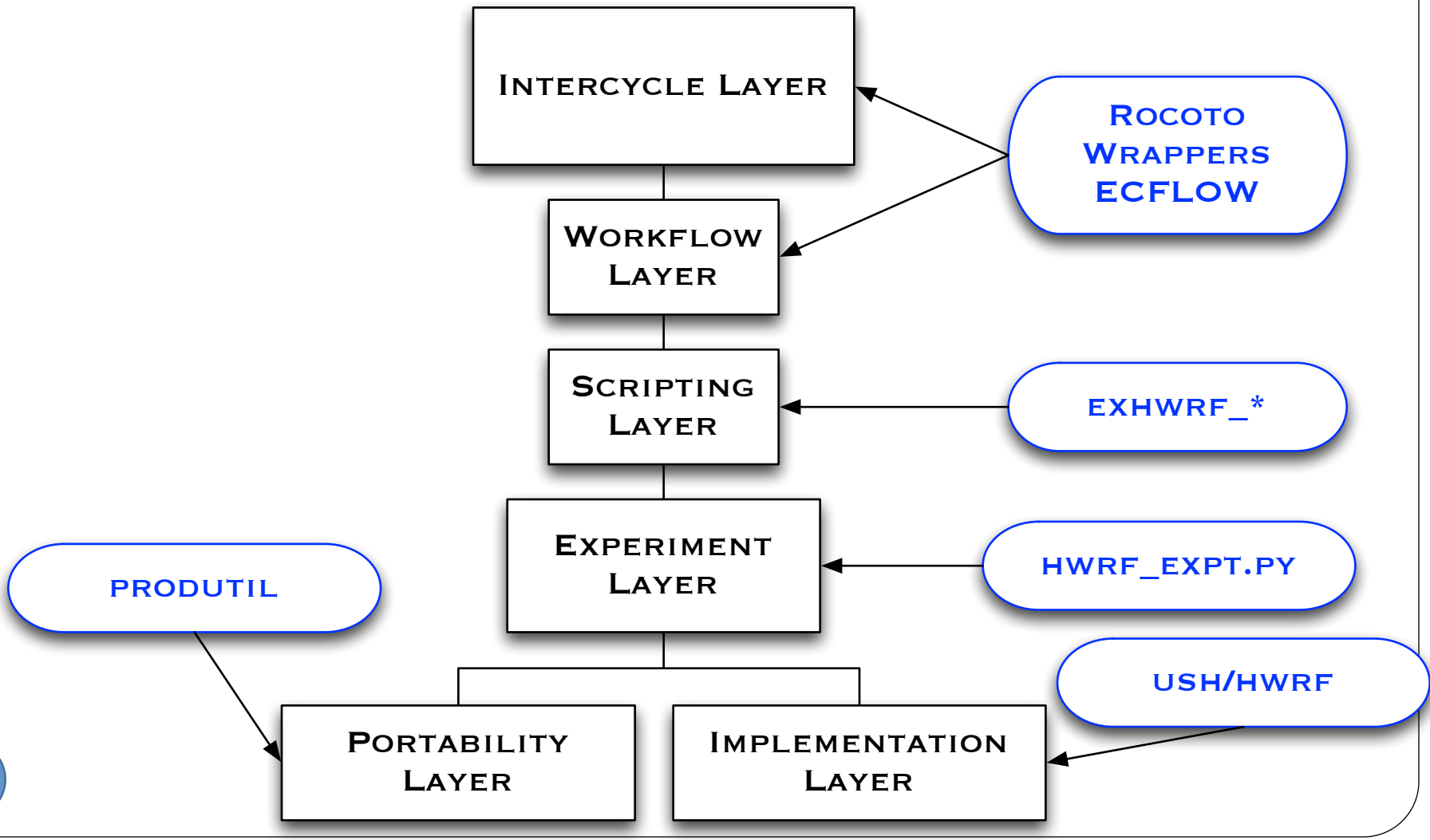
- Functions
  - prelaunch
    - makes per-cycle modifcations to configuration file, storm1.conf
  - sanity_check
    - runs a sanity check on the modules content
    - to be called after init_module
  - inputiter
    - iterates over all inputs required by a particular configuration
  - init_module
    - initializes the HWRF object structure

Exercise

Open hwrf_expt.py

Overview the init_module

HWRF System Overview



Implementation Layer

- A set of Python classes and functions used by the Experiment layer to run HWRF
- Each component has its own class and set of functions
- Some classes perform utilities to support the system, such as predicting filenames and performing time/date arithmetic
- Two packages
  - pom – Princeton Ocean Model initialization
  - hwrf – Implementation of most of the HWRF system

hwrf Python High-Level Packages

- Mostly HWRFTask subclasses
- Scripts (or batch jobs) call the run() functions of these subclasses
- Later tasks obtain input data by calling the products() iterator of earlier tasks

hwrf.launcher	creates the initial HWRF working directories and important files such as the database, configuration, holdvars, and storm information
hwrf.input	obtains input data from disk, FTP, SSH or tape to meet the input data requirements given by each tasks' inputiter() iterator.
hwrf.wps	runs the WRF Pre-Processing System (WPS) on parent model data to produce inputs to the real_nmm and wrf programs.
hwrf.prep	runs the prep_hybrid program on parent model spectral data to produce inputs to the real_nmm
hwrf.relocate	relocates, resizes and modifies the intensity of the tropical cyclone vortex.

hwrf Python High-Level Packages

hwrf.bufrprep	converts data dumps to bufr files for **input to GSI**
hwrf.gsi	**runs the GSI** data assimilation system
hwrf.fcsttask	runs the atmosphere-only **WRF and real_nmm** (short simulations for generating wrfanl files, six hour analysis cycle simulations & 126hr fcst job)
hwrf.mpipomtc	interfaces with the pom package to run the **POM ocean model initialization** and run the **POM-coupled WRF forecast**
hwrf.post	runs the HWRF post to **convert model output to GRIB files**
hwrf.gsipost	a wrapper around hwrf.post that **handles inputs and outputs to GSI**, assisting in GSI diagnostics.
hwrf.gribtask	runs copygb, wgrib, grbindex and similar programs to **manipulate GRIB files** and copy them to their final destination
hwrf.copywrf	copies WRF input and output **data to some destination**
hwrf.nhc_products	**NHC-specific product** creation and delivery
hwrf.tracker	runs the GFDL **vortex tracker**
hwrf.ensda	data assimilation **ensemble**
hwrf.rocoto	utilities to interface between HWRF and the **Rocoto workflow automation** system

hwrf Packages that Describe HWRF

- Describe the workflow and how it's to be executed
- Allow for complex querying and modification of the work before the work is actually started
- Provides details on the inputs and outputs to all other tasks in the workflow

hwrf Packages that Describe HWRF

hwrf.wrf, hwrf.wrfbase	describe a **WRF simulation**, generates the WRF namelist from the HWRF configuration files, predicts input and output filenames based on namelist settings
hwrf.regrib	describes **regribbing operations** and has most of the implementation of those operations. This is used by hwrf.gribtask to do the actual regribbing.
hwrf.ensda	contains classes to describe a two-dimensional ensemble-vs-time array of tasks that represent the **steps of an ENKF or hybrid ENKF** data assimilation system. Also has wrappers around many of the high-level modules to create the GFS ENKF-based HWRF DA ensemble, hwrf.ensda.FromGFSENKF.
hwrf.hwrfsystem	a wrapper around many of the high-level modules that simplifies the definition of the **HWRF post-processing and data delivery**
hwrf.init	a wrapper around many of the high-level modules that combines objects in complex ways to create the HWRF **initialization system**.

Low-level Logic Modules

hwrf.config, hwrf.hwrftask	allows for **configuration of the HWRF system using UNIX Conf file**s. Implements some functionality common to all, or nearly all, HWRF tasks. This includes product listing, directory and executable specification, scrubbing settings, input data requirements and others.
hwrf.exceptions	exception classes thrown by the HWRF module. All exceptions defined in the hwrf package that can leave an hwrf module are defined here to avoid cyclic dependencies in the import statements. This allows one to just do "from hwrf.exceptions import *" to get all **HWRF-specific exceptions**.
hwrf.constants	**constant values** used in the HWRF system
hwrf.numerics	**time and date manipulation** and other numerical routines used throughout the HWRF system.
hwrf.prelaunch	utilities for changing the HWRF configuration before the hwrf.launcher completes. (**per-cycle configuration changes**)
hwrf.storminfo	parsing of ATCF, message and tcvitals files, which **specify storm information.**
hwrf.revital	complex **manipulations of tcvitals data**

Portability Layer

- Implements cross-platform methods of performing common tasks
  - MPI implementation
  - OpenMP
  - Serial programs
  - File operations
  - Batch system interaction
  - Manipulate resource limitations
  - Interact with database file