# A Workflow Management System for Automating Weather and Climate Simulations

Christopher W Harrop[1,2], Ligia Bernardet[1,4], Mark Govett[1], Jeff S Smith[1,3], Stephen Weygandt[1]

[1]NOAA Earth System Research Laboratory, Global Systems Division
[2]Cooperative Institute for Research in Environmental Sciences
[3]Cooperative Institute for Research in the Atmosphere
[4]Systems Research Group

## Introduction

One of the challenges of modeling and simulating weather and climate phenomena is the laborious process of running the various codes that are required to carry out a simulation. For example, to perform a single end-to-end run of a weather simulation, a scientist must often run several data preprocessing and assimilation programs, the model code, several postprocessing programs, codes to produce visualizations, and at least one forecast verification program. These codes often have complex interdependencies and if the simulation is run in real time, such as in an operational setting, the various programs must run at specific times of the day and must finish before the resulting forecasts become irrelevant. Another complication is that many of these codes must run on High Performance Computing (HPC) platforms that are not always reliable and are notoriously difficult for scientists to use. The onerous task of properly managing the execution of all the various simulation codes is exacerbated even further by the fact that hundreds to thousands of end-to-end runs of the simulation are required to complete many experiments. The net effect of all these obstacles is that scientists often become overwhelmed by the process of running their simulations and have little time left to pursue the scientific questions they set out to investigate.

In collaboration with the Development Testbed Center (DTC), the Global Systems Division (GSD) of the NOAA Earth System Research Laboratory (ESRL) has developed a scientific Workflow Management System (WFMS) to address the above challenges. A scientific WFMS is software that helps scientists construct scientific workflows and automate their execution. A scientific workflow is simply a collection of tasks and a description of their runtime requirements. In the following we describe the requirements and design of our system and provide some real world examples to illustrate how it has enabled scientists to focus more on science and less on babysitting their simulations.

## Design

The concept of workflow management has existed for a long time in the business world but has only recently emerged as an important tool for computational science. In the last few years, computational scientists have developed a number of scientific WFMSs (Yu and Buyya, 2005) in response to the challenges of growing complexity and scale of simulation experiments in a wide variety of scientific disciplines. Most of these efforts are focused on enabling the creation and execution of workflows that use the Grid to access data and computational resources at remote locations (Taylor et al. (eds.) 2007). Another common theme is the use of web services to implement loosely

coupled workflow tasks that execute on remote resources. And some WFMSs, such as the one used in the Linked Environments for Atmospheric Discovery (LEAD) (Plale et al. 2005), are themselves components of much larger problem solving environments.

Although many existing general purpose WFMSs have impressive capabilities, they do not meet our needs. One major impediment is that local computing policies, mostly related to security, prevent us from using Grid and web services technologies. Local policies also dictate that the WFMS must make use of our existing batch queuing software for scheduling and monitoring all workflow tasks. As (Eide et al. 2006) have pointed out, a WFMS will not be successful if it violates local computing policies or significantly perturbs the scientists' experimental methods. For these reasons we are developing a simple WFMS with the following fundamental design goals. First, we have limited the scope of our system to the weather and climate modeling domain so that our design can take advantage of any common properties of weather and climate simulations. Second, new capabilities are developed incrementally so that, as our system evolves, changes to the way scientists construct and run their experiments is minimized. Third, we never add capabilities that are not directly motivated by a scientist's real world experience. Finally, as with all WFMSs, reliability and fault tolerance are very important. Our WFMS must be reliable enough to be used in an operational setting. It must withstand and recover from system failures and must automatically rerun tasks that fail.

Many existing WFMSs have sophisticated graphical user interfaces (GUIs) for creating and running workflows. However, in adhering to our design philosophies, our system does not currently have a GUI because our users have not yet experienced or expressed a need for one. Instead, users use an editor and a simple custom XML-based workflow language to define their workflows in the form of human readable text. We chose to devise our own workflow language because the existing workflow language standards are far too complex to be useful for our purposes and are difficult for scientists to understand and use. In contrast, the workflow language we created is very simple and allows weather and climate researchers to define workflows in terms that are meaningful to them.

The workflow document that scientists create describes a set of tasks and their runtime requirements and dependencies. It also specifies a set of "cycles" that usually correspond to simulation initialization times. The entire workflow is run for each cycle specified and, subject to the declared dependencies and real time constraints (if applicable), multiple cycles will be run concurrently. Scientists can compactly specify an arbitrarily large series of runs of a workflow that will automatically be executed as quickly as the underlying resources and specified dependencies allow.

Our WFMS is implemented in Ruby and has a layered, object-oriented architecture (see Figure 1). The Workflow layer at the top parses the XML workflow document and orchestrates the overall execution of the specified workflow. It communicates with the Task layer to create, run, and monitor the status of workflow tasks. The Task layer runs and tracks the progress of the tasks for each cycle specified in the workflow. It starts tasks when their dependencies are satisfied and retries tasks that fail. The actual work for each task is done by batch system jobs that run under the control of the local batch queuing

system. The Task layer sends requests to the Batch Job layer to create those batch jobs and query their status. Finally, the Batch System layer at the bottom accepts job submission and status requests from the Batch Job layer. The Batch System layer's function is to translate generic requests from the Batch Job layer into the actual commands used by the underlying batch system, execute those commands, and return their results. The advantage of this architecture is that we can extend our system to support other batch systems simply by adding a new batch system class to the Batch System layer. For example, it could, in principle, be extended to support a metascheduler for running workflows on Grid-based resources. Our system currently supports the Sun Grid Engine (SGE), Load Sharing Facility (LSF), and LoadLeveler batch systems.

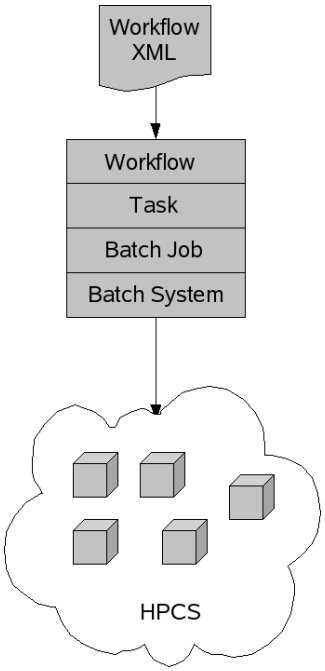


*Figure 1: Workflow Management System Architecutre*

## User Experience

It would be difficult, if not utterly impractical, to measure and evaluate the utility of our WFMS using formal methods. However, a substantial amount of anecdotal evidence suggests it has become an essential tool for the scientists who routinely use it to configure and run real world experiments. In the following we briefly describe some of those experiments and how the WFMS contributed to their success.

The WFMS has been used extensively by scientists at the Developmental Testbed Center (DTC), an institution that tests and evaluates new developments in the Weather Research and Forecasting (WRF) System with the intention of accelerating the transfer of new numerical weather prediction techniques from developers to operational forecasting centers.

Since 2003, the WRF DTC has performed two retrospective tests (the WRF Test Plan and WRF Rapid Refresh Core Test) and two realtime forecasting experiments (DTC Winter Forecasting Experiment and NMM5-CONUS, where NMM stands for the Non-hydrostatic Mesoscale Model dynamic core of WRF). Additionally, the WRF DTC participated in the Terrain-induced Rotor Experiment (T-REX), which had a realtime and a retrospective component. All experiments made use of the WRF DTC end-to-end system, which is comprised of input data collection and pre-processing, WRF forecasting, post-processing, dissemination of forecasts, forecast verification, and archiving. In all but one of the projects cited above, concurrent simulations of multiple configurations of the WRF model were processed, with the goal of determining the sensitivity of the forecasts to variations in dynamical core, physics suite, or initial conditions.

The processing of multiple WRF configurations for multiple forecast initialization times was greatly simplified by using the WFMS to schedule all tasks.

The WFMS was able to successfully detect the presence of data for initial conditions and to trigger the pre-processing and all following tasks in a timely manner. In case of failure of any task, the WFMS would automatically re-start the task. With the WFMS, the runs required minimal or no manual monitoring, which allowed greater productivity by making processing possible even at times when no staff was present, such evenings and weekends. Additionally, by not spending time monitoring runs, the scientific staff could focus on the analysis of the results.

The WFMS has also been an extremely valuable tool to scientists within the Analysis and Modeling Branch (AMB) of the Earth System Research Laboratory (ESRL). Within AMB, current research efforts are focused on developing a new hourly updated mesoscale analysis and forecast system for possible operational implementation at NCEP in 2009. Known as the Rapid Refresh (RR), this system is comprised of several software packages that must be run in a highly coordinated manner with multiple contingencies based on data availability and other factors.

The WFMS has proven to be an ideal tool for providing automated management of the complex task sequencing needed to run the RR prediction system. These tasks include: input data collection and format conversion (both background grids and observation datasets), GSI analysis pre-processing, GSI analysis execution, WRF model pre-processing and execution, model post-processing and graphics generation. This particular application requires a very complicated task sequence decision tree based on various data or task failures. Because the WFMS provides features such as file, task, and time dependency handling, it excels for this application. In addition to providing complete automated control of the analysis and forecast system, The WFMS is relatively easy to configure and use. After just a brief learning period we were able to provide the WFMS to a colleague at a different institution and have him successfully using it and adapting it the same day.

In summary, The WFMS has provided an easy to use, extremely robust, automated system for managing the complex task of running an hourly cycled analysis and forecast system and in the process greatly reduced scripting and run monitoring difficulties.

## A Component of WRF Portal

In addition to providing workflow management capabilities directly to scientists, the WFMS is also a core component of WRF Portal. WRF Portal is a Java application that simplifies model testing and evaluation by providing modelers with an intuitive graphical user interface for composing, running, and monitoring workflows. It also helps scientists track large numbers of model configurations, runs, and results.

Users interact with WRF Portal's GUI to create, run, and monitor WRF modeling experiments. When a user instructs WRF Portal to run a workflow, it automatically generates the XML description of the workflow, sends it to the supercomputer, and invokes the WFMS. WRF Portal retrieves information about the status of workflow tasks from the WFMS log and reports that to the user. Tasks are automatically rerun by the WFMS without explicit handling by WRF Portal.

## Future Work

Although the WFMS has played a key role in the success of several recent experiments, there are a number of areas

where improvement is needed. For example, scientists often need to define a large number of post processing tasks that differ only in the forecast time and the name of the file that they process. This results in very long, repetitive, workflow documents that are difficult to maintain. Another problem is that production real-time runs require the ability to specify quality of service (QoS) constraints, such as start and end deadlines, for each task of a workflow. The current workflow language does not provide that capability. A third issue with the workflow language is that it currently requires users to specify task runtime properties using batch system-specific syntax. This means workflow documents are not portable across batch systems. One other lesson we have learned is that it is often necessary to rerun arbitrary portions of a workflow that have completed successfully in order to correct non-fatal mistakes or make adjustments to code or simulation parameters. Although the WFMS can easily rerun tasks that crash, there is currently no convenient way to roll back a workflow to rerun tasks that have completed successfully.

The first issue can be addressed, in part, by providing a software tool to help scientists create and maintain workflow documents. However, we believe new workflow language elements that allow scientists to compactly define groups of similar tasks are also necessary. Simple augmentations to the workflow language are also planned to address the lack of QoS constraints and the non-generic specification of task runtime properties. Unfortunately, the implementation of workflow roll back presents many technical problems that we have not yet determined how to solve.

## Conclusions

We have developed a simple scientific WFMS that has successfully automated the orchestration of several recent complex, large-scale, weather simulation experiments. Many of its users now consider it to be a crucial tool for conducting their experiments and are even encouraging their colleagues to give it a try. No scientific software system will be successful if scientists do not buy in to it. We believe much of the success of our WFMS is due to a design driven entirely by user experience and by limiting its scope to weather and climate modeling applications.

## References


Eide, E., L. Stoller, T. Stack, J. Freire, and J. Lepreau, 2006: Integrated Scientific Workflow Management for the Emulab Network Testbed. *Proc. 2006 USENIX Annual Technical Conf.*, Boston, MA. May-Jun. 2006

Plale, B., D. Gannon, Y. Huang, G. Kandaswamy, S. Pallickara, and A. Slominski, 2005: Cooperating Services for Data-Driven Computaional Experimentation. *Computing in Science and Engineering*, vol 7, no 5, Sept/Oct, 34-43

Taylor, I.J., Deelman, E., Gannon, D.B., and Shields, M. (Eds.), 2007: *Workflows for e-Science Scientific Workflows for Grids*. Springer-Verlag, 530 pp.

Yu, J. and R. Buyya, 2005: A taxonomy of workflow management systems for Grid computing. Technical Report GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, Univ. of Melbourne, 33 pp.