4.5: WDSS-II: An Extensible, Multi-source Meteorological Algorithm Development Interface

V Lakshmanan[*]
National Severe Storms Laboratory & University of Oklahoma

Abstract

We describe the requirements – scientific, technical and economic – of scientists and engineers who are developing new meteorological algorithms and products. We argue that these requirements call for quick and ready access to multiple, disparate sources of data (including the outputs of other algorithms) in a uniform manner. We also argue that developer convenience dictates that it be possible to develop and test algorithms using data from an operational system without impacting that system. For efficiency, algorithm developers should be able to run multiple instances of any algorithm and to compare the performances of the different instances. It should be possible to easily visualize the intermediate and final outputs of the algorithms alongside the original data and the outputs of other algorithms. Application Programming Interfaces (APIs) geared towards algorithm development should provide all of these features to create a robust community of meteorological algorithm developers.

In this paper, we describe an API that meets these developer needs. Other papers in this session describe new algorithms that have been developed using this API, called the Warning Decision Support System – Integrated Information (WDSS-II). These algorithms illustrate different facets, both of the requirements of new algorithms and of the capabilities of the API.

[*]Corresponding author address: lakshman@nssl.noaa.gov

1. Future algorithms' requirements

Algorithms in development in the meteorological community range from radar-only algorithms to algorithms that use data from multiple radars to algorithms that integrate information from multiple sensors and sources. At the same time, there are several sources of any one data stream. For example, radar data can be obtained through the CRAFT project and Unidata's Local Data Manager (LDM), through NSSL's RIDDS system, through the ORPG BDDS or through AWIPS. In addition, radar data may already be present in the user's environment on disk or obtained through archives from NCDC.

Who develops algorithms? Algorithm developers range from applied research scientists at various research laboratories and graduate students at research universities to meteorologists at weather forecast offices.

The final and intermediate outputs of the algorithms developed by these developers range from gridded products to vector products to tabular data.

What kinds of requirements are posed on an algorithm development environment by such a varied and disparate group and by such an assortment of algorithm inputs and outputs?

The Warning Decision Support System has been redesigned to be an excellent algorithm-development platform. We considered the needs of an algorithm developer and designed the system around that. Some of the requirements we identified, and the way we met those requirements follow.

a. *Inexpensive*

The development platform must be inexpensive and easy to set up and get started. Consequently, the WDSS-II system is available for Linux (Solaris versions are also available). We use free software tools for compilers, GUI toolkits, XML parsers, source code configuration, etc.

The only hardware requirement for WDSS-II is a personal computer (PC) with a graphics card. Users have bought algorithm development machines for as little as $1000.

b. Open, Extensible and Standard Data Formats

The data formats used should be open, extensible and standard. Open formats allow the data to be ported to different platforms and architectures. Extensible formats allow the easy creation of new products, a must in an algorithm development environment. Standard formats allow the examination and verification of input, intermediate and output format with external tools.

The native format for the WDSS-II system is XML Bray et al. (2000) for small chunks of data (such as tabular data) and Unidata's NetCDF Jenter and Signell (1992) for array-based data. Use of these open, extensible and standard data formats has made it possible to easily accomodate new algorithm outputs. Thus, WDSS-II algorithms have ingested and output disparate data sources, such as multi-sensor SCIT, supercell identification, TDWR data, and advected fields with *no source code modification* to the underlying codebase. There are libraries freely available to read and write these formats. There are tools available (besides WDSS-II) to examine the output files. In addition to these native data formats, the WDSS-II system has been enhanced by specialized ingest classes for ORPG, RIDDS and some experimental formats. These libraries may be configured out of a deployed system.

c. Connect to Real-time system

Some of the proposed algorithm development environments require the algorithm to be present on the same machine as the operational (or quasi-operational) system that provides it data, with the operational system controlling the algorithm. While this is a desirable situation in an operational setting, it is very inconvenient for a developer. It should be possible for an algorithm developer to connect to a real-time system without having to be responsible for maintaining that real-time system.

WDSS-II algorithms typically receive data from remote locations. Thus, it is easy for an algorithm developer to get going – they don't need to somehow maintain a data feed. Through CRAFT and LDM, we do provide the capability for the user to have a clone of the entire system on the desktop.

d. Archive and Real-time

Algorithms should not be developed exclusively on archive cases and then expected to work in real-time. To ensure that algorithm developers test their algorithms on a real-time stream, it should be possible to switch between archive sets and real-time data streams quickly and easily.

The algorithm developer should not be required to run a data feeding algorithm to get data to her algorithm – the algorithm should be able to run off data on remote machines.

WDSS-II algorithms can switch from archive to real-time easily. They usually achieve this via a command line option to the algorithm ("-r"). Except for multisensor simulations, where a synchronizing data feeding process is required, WDSS-II algorithms can be directly run on an archived case on disk

e. Simulate Multi-Sensor Streams

It should be possible to easily simulate multi-sensor streams from archive. WDSS-II provides a tool that does this simulation, so that the algorithm processes the data as if it were real-time. The tool, like all WDSS-II algorithms and tools, handles data sources of different types and on different platforms, so that the user does not need to physically colocate all the archive sets.

To save bandwidth, the transmitted data is usually compressed, and uncompressed on the fly by the data ingest API.

f. Part of System, if required

While algorithms are developed and initially tested in isolation, at some point, the developer wants the algorithm to be tested in a real-time system. Thus, it should be possible to incorporate developed algorithms into a real-time system of algorithms, or to run them independently.

WDSS-II algorithms can currently be packaged into a real-time system using base data obtained using LDM –

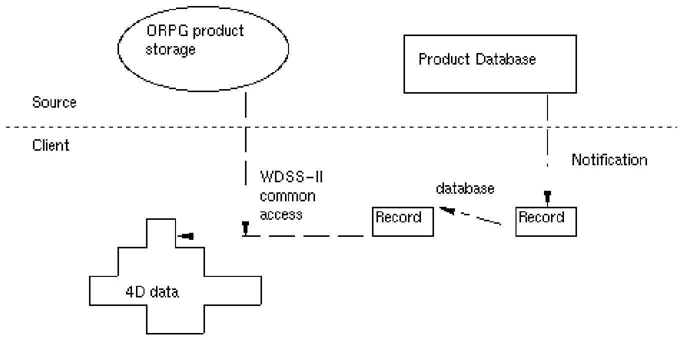

Figure 1: Architecture of a WDSS-II algorithm inside the ORPG. Compare with Figure 2 – what changes is how the ingest is done.

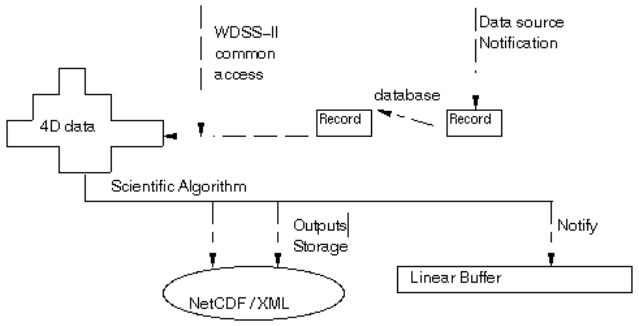

Figure 2: Architecture of a WDSS-II algorithm. The algorithm processes the abstracted four-dimensional data that it obtains by connecting to and listening to an Index. It writes out intermediate and final outputs as NetCDF/XML files and adds information about the products to the Index of products.

this is what we are doing at NSSL, and in the Norman and Jackson Forecast Offices.

But what about the ORPG? WDSS-II was able to provide base data and products from an ORPG to algorithms. We achieved this capability by having the data ingest API connect and listen to the ORPG product database and provide data and products to algorithms. Thus, there is a path for algorithms developed in WDSS-II to work in a real-time system – see also Figure 1.

g. Multi-sensor and multi-source

The algorithm development environment should support multi-sensor and multi-source ingest for algorithms. A lot of existing algorithms can be greatly improved by incorporating inputs such as a nearby radar, or local environment conditions from a model.

WDSS-II integrates multi-source and multi-sensor information extremely well since it was designed with this capability in mind.

2. Easy Development and Testing

The development environment should live up to its description, supporting easy development and testing of new algorithms. In the next section, we will demonstrate some of the capabilities of the system that really help an algorithm developer.

Instead of showing possibilities, we will show case studies of how people have actually used our system.

Shown in Figure 2 is the architecture of a new algorithm in WDSS-II. The algorithm processes an abstract four-dimensional data class (such as RadialSet, CartesianGrid, LatLonGrid, DataTable, etc.). These data are obtained by the algorithm from an Index. The Index varies by data source. For example, the Index in an ORPG is a class that bridges the Index API to the Product Database. The Index in a CRAFT/LDM feed is created by a process ldm2netcdf that decodes the LDM files. The algorithm writes out intermediate and final outputs as NetCDF/XML files. Finally, for every product written out, the algorithm should add indexing information about that product to an Index of products. In a real-time system, the Index of products is the same as the Index of inputs, but in an algorithm development environment they are separate, so that the algorithm can be rerun as many times as required on the same set of inputs.

a. Portability

The WDSS-II system was designed to ingest data from many different data sources, and therefore abstracts out many of the source-dependent details into base classes that provide common APIs that the algorithms depend on.

Different data sources typically have either a data feeding process that provides input data as NetCDF/XML or has a Factory that creates the appropriate Index, builders and formatters for that data source. This entire Factory and associated factories are located in a shared libary. In an algorithm development environment where the only input data is going to be ORPG data, shared libraries corresponding to RIDDS or satellite will never be loaded or used. While building the system, such unused libraries should be turned off and used packages turned on.

Thus, it is possible to build a real-time system of algorithms with minimal size. There is a large set of common classes and functionality that is always built into the system and can be relied on by the algorithms. These classes are either part of WDSS-II (for example, a DataConverter class that converts a RadialSet into a CartesianGrid, a Location class to do earth-navigation) or are part of the C++ Standard Template Library (sort, random shuffle, nth element, etc.).

Comparing the algorithms in Figure 1 and 2, we see that an algorithm developed on one data source can easily be migrated to another, since it relies only the common functionality. One Index type and its associated builders and formatters can be substituted for another simply from the command line.

The algorithm run on the ORPG would be launched as: `myApp -i orpg:/export/data/orpg/productdb -o /data/output` while the same algorithm running on LDM data would be launced as: `myApp -i xmllb:/home/ldm/data/KTLX/index.lb -o /data/output` Of course, the shared libraries available on the two systems are different, but the algorithm source code is unmodified.

b. Flexibility of the API

Not only the API easy to use and get started on, it is also very flexible. Other papers in this section will talk about rapid algorithm development (LLSD development (Smith 2002)), about using multiple radar inputs (multisensor algorithms (Stumpf et al. 2002)), about algorithms that depend on base data as well as on other algorithms' outputs (Supercell (Lynn 2002)), on switching between satellite and radar inputs (KMeans (Lakshmanan et al. 2000)) and on new concepts (virtual volumes (Lynn and Lakshmanan 2002) and time-based mergers (Lakshmanan 2002)).

The final paper in this session will provide an overview of the WDSS-II components and future plans for the WDSS-II API.

3. Conclusion

The WDSS-II algorithm developer API allows an algorithm developer to quickly access both real-time and archived data in a uniform manner. It allows the developer to run the algorithm without modifying operational systems in any way. The developer is able to quickly visualize the outputs of the algorithm. We showed the flexibility of the API, to handle multiple radars, multiple colloborative algorithms, to switch between radar and satellite inputs and even to represent and provide access to radically new ways of looking at radar data. Finally, the API and associated tools are available for use by a community of algorithm developers with little up-front costs.

Acknowledgement The work detailed in this paper was supported with funds from the National Science Foundation, NEXRAD Product Improvement group at the National Oceanic and Atmospheric Administration, the Federal Aviation Authority and the National Severe Storms Laboratory (NSSL).

The authors would like to thank Don Bailor, Karen Cooper, Charles Kerr, John Krause, Jason Lynn, Lulin Song, Tad Thurston, Thomas Vaughan and Lingyan Xin of the University of Oklahoma and Kurt Hondl of NSSL for many contributions to the design and implementation of the WDSS-II system.

References

Bray, T., J. Paoli, C. M. Sperberg-McQueen, and E. Maler: 2000, Extensible markup language (XML). Technical report, World Wide Web Consortium, available at http://www.w3.org/TR/REC-xml.

Jenter, H. L. and R. P. Signell: 1992, NetCDF: A freely-available software-solution to data-access problems for numerical modelers. *Proceedings of the American Society of Civil Engineers Conference on Estuarine and Coastal Modeling*, Tampa, Florida.

Lakshmanan, V.: 2002, Real-time merging of multisource data. *21st Conference on Severe Local Storms*, Amer. Meteo. Soc., San Antonio, TX.

Lakshmanan, V., R. Rabin, and V. DeBrunner: 2000, Identifying and tracking storms in satellite images. *Second Artificial Intelligence Conference*, American Meteorological Society, Long Beach, CA, 90–95.

Lynn, R.: 2002, The WDSS-II supercell identification and assessment algorithm. *21st Conference on Severe Local Storms*, Amer. Meteo. Soc., San Antonio, TX.

Lynn, R. and V. Lakshmanan: 2002, Virtual radar volumes: Creation, algorithm access and visualization. *21st Conference on Severe Local Storms*, Amer. Meteo. Soc., San Antonio, TX.

Smith, T.: 2002, A two-dimensional, local, linear, least-squares method of derivative estimates from doppler radial velocity. *21st Conference on Severe Local Storms*, Amer. Meteo. Soc., San Antonio, TX.

Stumpf, G., T. Smith, and A. Gerard: 2002, The multiple-radar severe storm analysis program for WDSS-II. *21st Conference on Severe Local Storms*, Amer. Meteo. Soc., San Antonio, TX.