P3.1

SOFTWARE ARCHITECTURE OF THE NEXRAD OPEN SYSTEMS RADAR PRODUCT GENERATOR (ORPG)

Michael H. Jain

NOAA/ERL/National Severe Storms Laboratory, Norman, OK

1. INTRODUCTION

As a part of the modernization effort, the National Weather Service (NWS) is currently involved in an effort to evolve the existing NEXRAD WSR-88D system so that changing Tri-Agency operational requirements can continue to be met. This activity falls under the NWS project of NEXRAD Product Improvement (NPI) (Saffle, et al. 1997). The initial objective of the NPI project is to rehost the existing Radar Product Generator (RPG) functionality from the current, proprietary platform to one that is open systems compliant. The National Severe Storms Laboratory (NSSL), working with the WSR-88D Operational Support Facility (OSF), has been tasked with the responsibility of establishing the software architecture and developing the operational software for this project.

2. SOFTWARE ARCHITECTURAL GOALS

Several software architectural requirements have been identified for the ORPG:

- Portability. The ORPG software must be portable at the source code level to a variety of open system standards compliant hardware/operating system platforms. Software portability will protect the ORPG from technological obsolescence and provide the opportunity to utilize current, competitively priced technology.

- Expandability. Demands on the ORPG computing resources will continue to increase due to research advances and the development of new meteorological algorithms and products. The ORPG must have the capability to expand its processing and storage capacities through the addition of new hardware. The ORPG software should have the capability to accommodate these hardware changes without any or significant modification.

- Scalability. As operational requirements change and new meteorological algorithms are developed, the new ORPG must have the capability to easily incorporate new applications into the system. Additional applications should not require complicated integration into the ORPG. Rather, new applications should be independent functions that may be incorporated into the ORPG via system configuration files.

- Maintainability. Effective maintenance of a software system such as the ORPG will contribute to rapid defect detection and repair, timely software upgrades, and a higher quality product.

Corresponding author address: Michael Jain, National Severe Storms Laboratory, 1313 Halley Circle, Norman, OK 73069-8480; e-mail <mjain@nsslgate.nssl.noaa.gov>

- Availability. Being an operational system for the Tri-Agencies, the ORPG is required to maintain a high degree of availability. The software architecture should be highly fault tolerant.

3. THE ORPG SOFTWARE ARCHITECTURE

The ORPG can be described as a “loosely coupled” , distributed system that is composed of a collection of independent modules functioning in an interdependent fashion. Each ORPG process has a well defined and specific function that retrieves and consumes data from well known data storage locations as well as

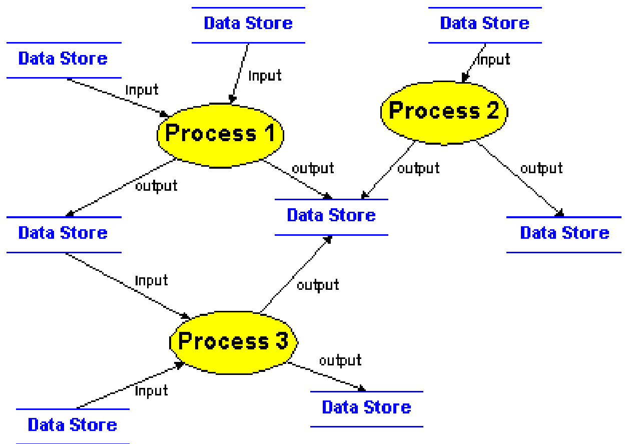

Figure 1. Example ORPG Data Flow Diagram

produces and stores data to well known data storage locations. Direct process-to-process communications are minimized in the ORPG. Instead, data or messages are written to an intermediate data store in the form of a disk file or shared memory (Figure 1). The ORPG processes neither know nor care what process(es) produce their input data or consume their output. This discrete modularity and encapsulation of software functions will make software maintenance and extensibility much easier than the current WSR-88D system. Further, this design concept provides considerable system configuration flexibility and supports a distributed processing environment.

Distributed processing is an extremely significant capability of the ORPG, supporting expandability and availability of the system. Additional processors may be added to the ORPG to satisfy increased computational requirements. System configuration files will establish the distribution of tasks and data stores across the ORPG with no alterations being required of the application software. Distributed processing will also provide fault tolerance to the ORPG via task load leveling and providing redundant processing capacity in the event of a processor failure.

A layered software architecture model, as depicted in Figure 2, is being applied in the development of the ORPG software. This model supports portability and maintenance of the ORPG software by isolating the application level programs from the underlying system infrastructure and operating system dependent layers.

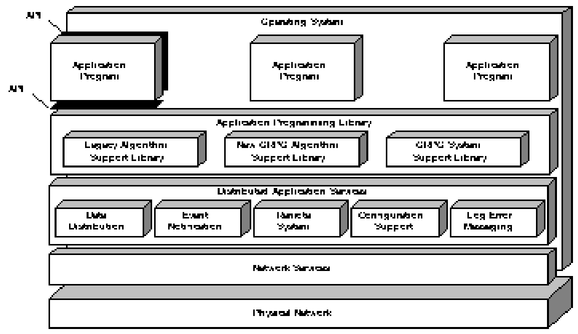

Figure 2. ORPG Layered Software Model

Each layer directly interacts with only the layers adjacent to it. This interaction between layers is achieved through a well documented Application Program Interface (API). This model assures application software are insulated from changes in the operating system or the underlying infrastructure as long as the API are maintained.

4. ORPG INFRASTRUCTURE SERVICES

These services are depicted in Figure 2 by the Distributed Application Services Layer. These services are not specific to RPG functionality, instead they represent services required of any distributed processing system. The current ORPG Distributed Application Services are:

- Data Distribution. The ORPG is considered to be a data driven system. These services deal with data storage and retrieval to single and/or multiple processes through the use of Linear Buffers (Jain, et al. 1997). Data and message buffering for real-time applications and handling of static data files to distributed applications are included.
- Event Notification. This service immediately notifies processes when particular events occur. This is important for time critical applications. Applications can register to receive events and/or can post events as required.
- Remote System. Allows ORPG system calls to be initiated from one hardware platform and executed on another hardware platform.
- Configuration Support. Designed to support dynamically reconfigurable, distributed applications whose processes, files, and other resources can be moved from one node to another without shutting down and restarting the entire application or system.
- Log/Error Messaging. This service is a formatting and distribution tool for log and error messages.

Additional information regarding these services can be found in Jain, et al. (1997).

5. METEOROLOGICAL ALGORITHMS AND PRODUCTS

The meteorological algorithms and product generators of the legacy RPG represent a substantial part of the existing RPG software. These applications are written in FORTRAN with Concurrent OS/32 operating system extensions and cannot be directly compiled and executed on a Unix Operating System platform. Rewriting this software would require a significant development effort as well as an intensive testing and verification process to determine the algorithms were recoded properly. Instead, these algorithms are being directly ported to the ORPG environment.

The porting goals guiding this process were:

- Modifications to existing code should be minimized.
- Ported software should run within the ORPG environment.
- Want to maintain only a single version of the RPG source code.
- Ported code can be compiled by any ANSI FORTRAN compiler and run on any POSIX compliant operating system with minimal modification.
- Ported code should run efficiently in terms of the hardware resources.

To achieve these goals a FORTRAN preprocessor was developed that converts the Concurrent FORTRAN to ANSI FORTRAN code at compile time. A Legacy Algorithm Support library was developed that maintains the legacy API and provides the links to the new ORPG software infrastructure. A more detailed description of the algorithm port is contained in Jing, et al. 1997.

6. ACKNOWLEDGMENTS

The author would like to acknowledge the Office of Systems Development of the National Weather Service for funding this project. I would also like to acknowledge the incredible work that is being performed by the NSSL/OSF Development Team in the development of the ORPG.

7. REFERENCES

Jain, M.H., Z. Jing, A. Zahrai, A. Dodson, H. Burcham, D. Priegnitz, and S. Smith, 1997: Software architecture of the NEXRAD open systems radar product generator (ORPG). Preprints 13th IIPS, Long Beach, CA, AMS, 238-241.

Jing, Z., S. Smith, M. Jain, and A. Zahrai, 1997. Migration of legacy WSR-88D algorithms and product generators to the open system RPG. Preprints 13th IIPS, Long Beach, CA, AMS, 245-248.

Saffle, R.E. and L.D. Johnson, 1997. NEXRAD product improvement overview. Preprints 28th Conf. On Radar Meteorology, Austin, TX, AMS, paper P3.2.