Guiding the Next Phase of Radar Software Development and Data Access

Jennifer DeHart[1], Brenda Javornik[2], Michael Bell[1], Mike Dixon[2], Anna del Moral Mendez[2], Ting-Yu Cha[2], Wen-Chau Lee[2]

[1]Colorado State University, Fort Collins, CO

[2]National Science Foundation National Center for Atmospheric Research, Boulder, CO

Reproducible Science Requires Accessible and Open Software

Radars are complex instruments capable of producing copious amounts of data. Accessible software using vetted procedures frees scientists from redeveloping methods from scratch and improves reproducibility.

2012 Community Workshop on Radar Technologies
At the last workshop, attendees identified software priorities, including standardized packages, training, and an NCAR-maintained repository. The Lidar Radar Open Software Environment (LROSE), supported by NSF, was developed to meet these needs. Over the same period, the proliferation of Python and open science has led to a golden age of open-source radar software. This poster poses several questions to help guide future LROSE development.

Questions for Discussion
1. How do we ensure interoperability of radar software packages in different languages?
2. What is NCAR's role in radar software development?
3. What software intercomparisons are of value?
4. Can we use LLMs to improve user training?
5. How do we integrate cloud-optimized data with LROSE?

LROSE Overview

LROSE Development

LROSE contains core algorithms for processing steps that are well-understood and documented in the peer-reviewed literature. The tools include data conversion, data visualization, quality control, regridding, processing radar echoes, and single- and multi-Doppler wind analysis. The most recent LROSE version, Laguna, was released in August of 2025 for the AMS Radar Meteorology Conference. Laguna included increased Lidar and ancillary data support, additional functionality in RadxConvert, new applications for computing statistics on Ecco results and converting TITAN binary data to NetCDF, and bug fixes.

LROSE Science Gateway
The goal of the gateway is to improve LROSE accessibility through user education, by supporting educational tutorials for individual use, workshops, and classroom exercises. The gateway consists of requestable JupyterHub servers on Jetstream2 supported through an NSF ACCESS allocation. Jupyter notebooks take users through the steps to run applications, configure application parameter files, and interpret output. Current tutorials cover LROSE basics, radar mosaics, data quality control, quantitative precipitation estimation, hydrometeor classification, and multi-Doppler wind retrievals. Some tutorials contain all necessary parameters, while others provide a step by step guide for inspecting data so users can apply the procedures to their own datasets.

1. Software Interoperability

Open Radar Science Community is thriving, but packages span several programming languages

The Open Radar Science community is flourishing, with over 15 different packages across several institutions. Other packages by external developers are continuously being developed. These packages have different use cases and radar meteorologists will often use multiple packages in their workflows. Although Python is a frequently-used language for these packages (e.g., Py-ART), software packages are also written in C++ (e.g., LROSE, BALTRAD), possibly Fortran, Matlab (e.g., lrose-ecco) and Julia (e.g., Ronin, Daisho). ***How do we improve the interoperability of these packages?***

Software communication challenges

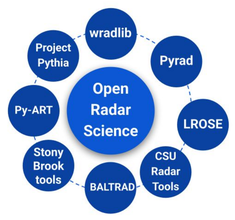
There are a few key communication issues: Fortran and C++ have different multi-dimensional array layouts (row order vs. column order). Objects and classes do not pass easily across languages. We need functional programming style interfaces; not object-oriented programming style. Functional interfaces can be composed, easily port to parallel and GPUs, easier to test, and validate for correctness, and easier to cross connect. With increased use of Functions as a Service (FaaS) platforms such as Amazon Web Services (AWS) Lambda functions, and Globus Compute, as well as the potential for quantum computing (QPUs), functions are the common denominator.

Common data model

There is no shared data model across languages. Currently, the only way to transfer data between languages is to write intermediate files to disk using standard data formats (CfRadial, WMO FM-301). For Python packages, Xradar is becoming the preferred common data model and is supported by the Open Radar Science organization.

Application Programming Interfaces (APIs) for many languages (provide a service) - Python bindings

Python bindings to each of the packages could make Python the common interstitial medium. This would be useful in Jupyter notebooks and then a common GUI would be the Jupyter notebooks and the many Python graphing packages.



2. Role of NCAR Software Development

One of the community requests from the 2012 NSF Community Workshop on Radar Technologies was an NCAR-maintained software repository with datasets for testing. The increase in other community software packages mean the development effort is more decentralized. NCAR instruments (e.g., S-Pol, HIAPER cloud radar) will still require continued software development. ***What is the community looking for from NCAR?***

How can we increase contributions to LROSE while retaining the vast C++ code base?

Please add your comments and suggestions below using a sticky note!

3. Software Intercomparison

- Many software packages have overlapping processing capabilities (e.g., multi-Doppler wind synthesis, hydrometeor identification, and quantitative precipitation estimation).
- A systematic intercomparison of packages across a range of meteorological phenomena and cases is needed to understand their respective strengths and weaknesses. ***How can we fund these studies?***
- What applications does the community want to compare? The SAMURAI and PyDDA teams are interested in an intercomparison study, but funding (and time) is a challenge.
- ***What types of cases or datasets is the community interested in?*** Cases from AMS and ERAD short courses could serve as benchmark datasets.

4. Improving Workflow Development using LLMs

- Complex applications and workflows require user configuration of numerous parameters, which often vary by case and phenomena.
- New users require substantial training and tutorials/training materials do not always cover the specifics of a user's case - how do we improve how users learn what parameters need to be adjusted?
- LLMs provide natural language interfaces that could be trained on documentation, tutorials, and other sources, which could accelerate user education.
- ***Could web-based platforms (e.g., Project Pythia, LROSE Science Gateway) be used to connect open source software, LLMs, data access, and compute resources to run analysis?***

5. Integrate software & cloud-optimized data

As radar datasets increase in size and reside on cloud servers, it is important to bring software to the data, read smaller portions of datasets, and label data.

"NSF NCAR's Geoscience Data Exchange, or GDEX, integrates NSF NCAR's community compute resources with our portfolio of rich data assets. GDEX was built to empower collaborative, open science", Everette Joseph, Director NSF NCAR.

Zarr is a cloud-optimized data format used by many packages. LROSE input and output are currently file-based and the core libraries need updates to accommodate vectors of data as well as files.

Improving accessibility to existing EOL datasets, adding flexibility to LROSE read and write functionality, and bringing data and compute together are important.

This material is based upon work supported by the NSF National Center for Atmospheric Research, which is a major facility sponsored by the U.S. National Science Foundation under Cooperative Agreement No. 1852977. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF.

Scan for more information:

LROSE Science Gateway

LROSE website