THE EMERGENCE OF OPEN-SOURCE SOFTWARE FOR THE WEATHER RADAR COMMUNITY

BY M. HEISTERMANN, S. COLLIS, M. J. DIXON, S. GIANGRANDE, J. J. HELMUS, B. KELLEY, J. KOISTINEN, D. B. MICHELSON, M. PEURA, T. PFAFF, AND D. B. WOLFF

Recent advances in community-based software development have demonstrated that open-source software can be a real benefit to the radar community.

Since the emergence of weather radar technology in the 1940s, research has sought to tap the full potential of weather radar observations. During the digital age, improvements in radar technology have been closely linked to advancements in computer science and software engineering. Making use of modern radars is not possible without software.

AFFILIATIONS: HEISTERMANN—Institute of Earth and Environmental Sciences, University of Potsdam, Germany; COLLIS AND HELMUS—Argonne National Laboratory, Argonne, Illinois; DIXON—National Center for Atmospheric Research, Boulder, Colorado; GIANGRANDE—Biological, Environmental and Climate Sciences Department, Brookhaven National Laboratory, Upton, New York; KELLEY AND WOLFF—Wallops Flight Facility, NASA GSFC, Wallops Island, Virginia; KOISTINEN AND PEURA—Finnish Meteorological Institute, Helsinki, Finland; MICHELSON—Swedish Meteorological and Hydrological Institute, Norrköping, Sweden; PFAFF—Institut für Wasser- und Umweltsystemmodellierung, University of Stuttgart, Stuttgart, Germany
CORRESPONDING AUTHOR: Maik Heistermann, University of Potsdam, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany
E-mail: maik.heistermann@uni-potsdam.de

The abstract for this article can be found in this issue, following the table of contents.
DOI:10.1175/BAMS-D-13-00240.1

In final form 12 May 2014
©2015 American Meteorological Society

Much progress has been, and continues to be, made in the acquisition, analysis, display, and use of weather radar data. A great deal of software exists for radar data processing—some of it old and some new, some proprietary and some freely available, some good and some not so good. What is clear to most people who work in this field is that the amount of time spent dealing with outdated or inadequate software significantly reduces the time available for making scientific progress. File formats are varied and nonstandard; software works on some platforms and not on others. Good documentation is scarce. Work is repeated by multiple organizations over and over again. Systems developed by different organizations do not interact well with each other.

In this paper, we argue that community-based open source software (OSS) development could provide the means to reduce these inefficiencies and to improve the standard and scope of weather radar software. We define OSS as software with its source code made available and licensed in a way that provides the rights to study, change and distribute the software to anyone and for any purpose (St. Laurent 2008, p. 8–11).

To explore the specific role of OSS in the field of radar data processing, we address the following topics: What is actually required from radar processing software (next section)? Why do we expect OSS to better meet these requirements

(section on "The potential benefits of open source software")? In the "Examples of individual open source software projects" section, we highlight five active OSS projects and present lessons learned from these projects in "Lessons learned." The "Combining individual projects into a true community effort" section discusses how the efforts of those projects might be combined such that the whole is greater than the sum of its parts. "Conclusions" finally presents our overall conclusion.

THE "RADAR PROCESSING CHAIN" AND ITS COMPONENTS. A prime feature of radar data processing is the large number and variety of steps that need to be taken along the different levels of a processing chain. Such a chain begins at the digital signal receiver and ends at one out of countless products required by different users. Basically, each of these specific user requirements implies the design of a specific processing chain. It would be beyond the scope of this paper to list all of the specific requirements—for example, in radar engineering, severe weather detection, atmospheric modeling, water resources management, agriculture, validation of remote sensing products, television broadcasting, or even the tracking of nonmeteorological echoes such as birds and insects.

Instead, we briefly outline typical components of a radar processing chain. The next section will then introduce how OSS can improve these individual components and gain more flexibility in tailoring processing chains to meet specific user requirements.

Receiver-level processing. Modern digital radar receivers provide access to the time series (pulse by pulse) data. By analyzing the time series and related Doppler spectra, it is possible to identify and possibly remove clutter, echoes from anomalous propagation, and other artifacts. Radar "moments" such as reflectivity, Doppler velocity and spectrum width, and dual-polarization fields may then be computed (Doviak and Zrnić 1993; Bringi and Chandrasekar 2001). On many systems this processing is complete by the time the moment's data are made available to the user, in which case time series data are not available.

Data handling and format conversion. In working with radar data, a frequent challenge is the decoding of a multitude of different file formats for data storage and exchange. Despite efforts toward common data models (see next section), different dialects still exist in addition to a large variety of legacy formats.

Translation to Cartesian coordinates, merging data from multiple radars. Raw radar observations exist in spherical coordinates, with the added complexity of Earth curvature and beam bending due to atmospheric refraction. For many applications, variables in polar coordinates must be projected to a Cartesian reference system—a precondition for their integration with other geodata. If multiple radars are operated in a network, it is likely that the data must be collated and merged onto a common grid.

Algorithms. A wide variety of algorithms may be applied in a typical weather radar processing chain. These algorithms include, for example, Doppler-velocity dealiasing, echo and storm motion tracking, as well as vertically integrated liquid estimation. Often, these fields serve as primary inputs to operational and research-grade severe storm products that require single- and multi-Doppler wind retrievals, mesocyclone or tornadic rotation identification, and echo classification such as convective/stratiform discrimination and hail designation. Subsequent algorithms introduce conversions from radar moment to product fields—most notably the conversion from reflectivity factor to precipitation intensity. Challenges in quantitative precipitation estimation are addressed in the following paragraph in more detail.

Quantitative precipitation estimation. For quantitative precipitation estimation, a multitude of potential error sources need to be accounted for. These are typically inhomogeneous in space and time (Germann et al. 2006, 2009). On the one hand, these errors are introduced through the fundamental limitations of the measurement approach—the instantaneous, volume-integrated measurement of a quantity aloft that is only indirectly related to precipitation. On the other hand, quantitative estimation is impaired by a wide range of specific errors and artifacts such as calibration, ground echoes, attenuation, bright-band echoes associated with the melting layer, uncertain reflectivity–rainfall (Z–R) relationships, and others (Villarini and Krajewski 2010). Addressing these errors requires a combination of advanced correction algorithms. It should be emphasized, though, that different users usually have different perceptions of "data quality" that may imply different priorities in the correction of quantitative errors. Most algorithms come with an intrinsic trade-off, as they potentially introduce new errors while removing others. For example, one user might favor an aggressive clutter elimination in order to assimilate the

radar product into a numerical weather prediction model (Fornasiero et al. 2006; Peura et al. 2006), while another user might prefer a more conservative product in order to detect small-scale convective features. A single product, even if created with the best methods currently available, will not be able to accommodate all these needs simultaneously.

Displays and visualization. One of the most important requirements for end users is the visualization of observed and derived quantities, ideally including different varieties of horizontal and vertical cross sections, animated loops, and integration with data in other coordinate projections.

THE POTENTIAL BENEFITS OF OPEN SOURCE SOFTWARE. OSS has grown significantly over the recent decades and has a strong presence in scientific computing. The motivations for using and developing OSS are mixed, ranging from the philosophical to purely practical (see, e.g., WGLS 2000).

Raymond (1999) highlighted the community aspects of OSS, while Casson and Ryan (2006) pointed out the benefits of affordability, flexibility, transparency, interoperability, and perpetuity (or longevity). In the following, we would like to point out specific implications of these benefits for weather radar software.

Affordability (of infrastructure). From a software engineering perspective, the most challenging part of a radar processing chain is the establishment of a common infrastructure that enables the implementation of advanced processing algorithms. Building infrastructure is expensive, but if done right it enables rapid implementation of more complex algorithms and other processing steps. Given good infrastructure, developers can focus on making actual progress instead of redundant programming efforts.

Flexibility (in tailoring processing chains). As we pointed out in the previous section, different users have different notions of what constitutes "good quality." From the perspective of application development, specific user priorities are typically addressed by a specific combination of algorithms in the processing chain. For that purpose, open algorithms can be used as building blocks to facilitate tailoring such custom production chains. More generally, OSS allows modifying the software in order to custom-tailor solutions—for example, for addressing specific needs toward the integration into specific infrastructures or large companies or agencies.

Transparency (and its implications for scientific progress). Making algorithms transparent, open, and well documented has implications for scientific and technological progress. Scientific publications usually do not provide sufficient detail to allow other researchers to exactly reproduce the presented results. A reference to transparent open software code could provide the required level of detail and thus allow for reproducibility and comparability. Combining the benefits of community-based development with a public review of open code could lead the way toward standardization of specific processing steps. In fact, the innovation process of OSS in general has been related to the process of knowledge production in science (von Krogh and Spaeth 2007). Furthermore, OSS could accelerate scientific and technological progress by providing mechanisms for incorporating code from other software products—for example, as libraries, modules, or code fragments. We can think of this as cross-fertilization, the efficiency of which is determined by the level of interoperability (see next paragraph) and the standard of documentation. The concept of open algorithms can be seen as a "best practice" in this regard. Open algorithms implement standardized documentation at a high abstraction level, independent from any specific programming language, in order to facilitate implementation on any platform. An exemplary collection can be found in the BALTRAD cookbook (http://git.baltrad.eu/trac/wiki/cookbook).

Interoperability (using common and open data models). Open data models offer transparency to both data producers and users. These models encourage international data exchange (Viglione et al. 2010) and help to bridge gaps between communities (e.g., academia and operational organizations). The adoption of open and common data models simplifies software by reducing the number of formats to be supported, which in turn reduces the software development costs and increases interoperability among different software platforms.

In the radar world, the universal format (Barnes 1980) was an early attempt at achieving this, and it is still supported through the National Aeronautics and Space Administration (NASA)'s Radar Software Library (RSL) and Radx (see next section). More recent data models are mainly based on netCDF and Hierarchical Data Format, version 5 (HDF5) formats. The Operational Programme for the Exchange of

Weather Radar Information (OPERA) Data Information Model (ODIM) has evolved in Europe (Michelson et al. 2014, 2003) and exists in HDF5 and BUFR representations. The ODIM for HDF5 (ODIM_H5) representation has spread widely, being embraced by the community to the extent that, as of November 2014, it is used by 23 of the 25 countries that currently provide their data to the European radar composite products being produced by OPERA's data center Odyssey. ODIM_H5 is also widely supported by industry. Similarly, the successful development and proliferation of climate and forecast (CF) conventions (Eaton et al. 2011), building on NetCDF, has led to the CfRadial data format for data in polar coordinates (Dixon et al. 2013).

Longevity (of community efforts). Individual (closed source) efforts often last only as long as their principal author remains actively involved, and they subsequently tend to languish through lack of software maintenance and support. Community projects can achieve longevity provided the community exceeds a critical size. The Linux Foundation is an example of how communities have even established their own institutions to achieve this aim.

EXAMPLES OF INDIVIDUAL OPEN SOURCE SOFTWARE PROJECTS. This section describes five community-based OSS projects, without claiming to be exhaustive. We outline the history, background, and intended users of these projects and show how they themselves have benefited from other open source software. In addition, we provide tabular details with respect to technical features (Table 1), supported data formats (Table 2), and available features for quality control and error correction (Table 3).

LROSE/TITAN. The initial goal of Thunderstorm Identification, Tracking, Analysis and Nowcasting (TITAN; www.rap.ucar.edu/projects/titan) was to evaluate the effectiveness of rainfall-enhancement experiments in South Africa in the early 1980s. Development then moved to the National Center for Atmospheric Research (NCAR) with the goal of thunderstorm tracking and nowcasting (Dixon and Wiener 1993). In 2002, TITAN was released as open source. Its permissive license "allows you [. . .] to use, copy, display, perform and redistribute the Software, with or without modification, for any legal purpose, free of charge." Over time TITAN has grown into a relatively large system, including a data infrastructure layer, a large suite of algorithms, multiple displays, and real-time process control.

The Lidar Radar Open Software Environment (LROSE) project was conceived as a follow-on to TITAN. Seed funding for the project was obtained from the U.S. National Science Foundation in 2012, with initial priority given to the implementation of a standardized open data model. To date, progress

Table 1. Technical features of the different software systems.

	BALTRAD	LROSE/ TITAN	Py-ART	RSL	wradlib
Supported platforms	Linux	Linux, Mac	Linux, Mac	Linux, Mac	Windows, Linux, Mac
Programming language	C/C++, Java, Python	C++, Java, Python	Python, C, FORTRAN	C, IDL	Python, C, FORTRAN
API	C, Java, Python	C++, Python	Python	C, IDL	Python
Stand-alone components	DEX (data exchange subsystem), BALTRAD toolbox (data processing framework)	Many apps are stand-alone	Set of example apps	—	—
Version control	Git	Git	Git	—	Mercurial
Public code hosting	Dedicated Git server	Available	GitHub	Available	Bitbucket
Issue tracking	Trac	Jira	GitHub	—	Bitbucket
Forums and mailing lists	BALTRAD cookbook, Google apps	Available soon	Py-ART users, Facebook, Twitter	—	wradlib-users, wradlib-dev
Documentation	Online, Doxygen	Online	Online, Sphinx	Online	Online, Sphinx

Table 2. Supported radar file formats.

FILE FORMAT	BALTRAD	LROSE/TITAN	Py-ART	RSL	wradlib
ARM NetCDF: Atmospheric Radiation Measurement Program netCDF		✓	✓		
CfRadial: CF-compliant netCDF		✓	✓		
DORADE: Doppler Radar Data Exchange Format		✓			
DWD-DX: Deutscher Wetterdienst DX					✓
DWD-RADOLAN: DWD Routineverfahren zur Online-Aneichung der Radarniederschlagsdaten mit Hilfe von automatischen Bodenniederschlagsstationen					✓
EDGE NetCDF: Enterprise Doppler Graphic Environment NetCDF					✓
GAMIC HDF5: Gesellschaft für Angewandte Mikrowellen- und Informationstechnologie und Consulting HDF5		✓			✓
Lassen			✓	✓	
McGill			✓	✓	
MDV: Meteorological Data Volume		✓	✓		
ODIM_H5: ODIM for HDF5	✓	✓			✓
ODIM_BUFR: ODIM Binary Universal Form for the Representation of Meteorological Data	✓				
RAINBOW v.5		✓			✓
Rapic		✓			
SIGMET		✓	✓	✓	
UF: Universal format		✓	✓	✓	
WSR-88D: Weather Surveillance Radar-1988 Doppler		✓	✓	✓	

Table 3. Algorithms available.

	BALTRAD	LROSE/ TITAN	Py-ART	RSL	wradlib
Echo classification and clutter	Peura (2002); Ośródka et al. (2014); Szturc et al. (2012); Gill et al. (2012); Michelson and Henja (2013)	Vivekanandan et al. (1999); Hubbert et al. (2009)			Gabella and Notarpietro (2002); Vulpiani et al. (2012)
Advanced *Z–R* transformation			Ryzhkov et al. (2014)		German Weather Service (2004)
Phase processing	Gill et al. (2012)	Wang and Chandrasekar (2009)	Giangrande et al. (2013)		Vulpiani et al. (2012); Wang and Chandrasekar (2009)
Attenuation correction	Ośródka et al. (2014); Gill et al. (2012)		Gu et al. (2011)		Hitschfeld and Bordan (1954); Krämer and Verworn (2009); Jacobi et al. (2012); Vulpiani et al. (2012)
VPR	Networked VPR correction				Average VPR
Advanced compositing	Peura (2010); Henja and Michelson (2012)		Barnes (1964)		Weighted composition based on quality
Gauge adjustment	Michelson (2006)				Additive, multiplicative, and mixed error model
Dealiased radial winds	Haase and Landelius (2004)	James and Houze (2001)	James and Houze (2001)		
Partial beam blocking	Szturc et al. (2012); Henja and Michelson (2012)				

has been made on the CfRadial data format and on Radx, an open source library and a basic set of applications for manipulating radar and lidar data in polar coordinates (www.eol.ucar.edu/software/radx). LROSE builds upon the open source code base provided by TITAN and other legacy software developed at NCAR, as well as on the open source netCDF and HDF5 libraries.

RSL. RSL was developed in the early 1990s to support the Tropical Rainfall Measuring Mission (TRMM) and the Global Precipitation Measurement (GPM) Ground Validation (GV) programs. The goal of RSL was to provide a set of library functions to ingest various formats into a well-defined radar "super structure" that could be used with data analysis modules in a transparent way. This super structure is composed of a header containing site information, along with a number of volume structures. Using this paradigm allows for writing format-independent modules and, thus, interoperability. RSL forms the backbone of NASA's Ground Validation System (GVS) which provides science products (3D Cartesian grids, rain maps, rain type, etc.) to NASA and the science community. RSL is also used by other groups from within and outside the United States, including government agencies, academia, and other research groups. As a library, RSL is intended for software developers. RSL has been successfully incorporated into the Python Atmospheric Radiation Measurement Program (ARM) Radar toolkit (Py-ART) software (see below), and a high-level implementation is available for Interactive Data Language (IDL).

BALTRAD. Fifteen partners in 12 countries are developing a system for weather radar networking and data processing using European structural funds. The BALTRAD project is operationally oriented, with the objective to provide real-time infrastructure to the Baltic Sea region in support of a multitude of applications, including the transport sector, hydrology, radiation and nuclear safety, and numerical weather prediction. The community-based nature of the partnership is a cornerstone that stems from the reality that many small organizations do not have sufficient weather radar expertise of their own and are therefore reliant on international cooperation to make progress (Michelson et al. 2010, 2012).

The primary goal in developing the BALTRAD software was to create and use modern data exchange methods. Data processing is optional, recognizing that some partners already have processing chains with which they are content. Therefore, the system is built modularly so that the major components are independent and communicate using open transmission control protocol (TCP)-based mechanisms. Instead of enabling the management of data in various formats, the approach has been to convert all data to the open ODIM_H5 standard (see previous section). The community has been remarkably successful in creating translation software that is applied in each country to convert from in-house or proprietary formats to ODIM_H5. The decentralized nature of the network implies that all partners use the same system to exchange polar radar data, and the system can then be used by each partner locally, using a common set of algorithms, to tailor the production chain to meet their purposes.

The BALTRAD toolbox forms the data processing framework (Henja et al. 2010). The real-time focus calls for a design that uses common functionality for file input/output (I/O) and the ability to chain well-defined processing algorithms in memory. Tools developed by different partners, some of which predate the project, are integrated by combining the partners' open algorithms with the toolbox file I/O functionality; examples are Ropo (Peura 2002), Rack (Peura 2012), and data quality algorithms made on 3D radar reflectivity volume data (RADVOL-QC) (Ośródka et al. 2014). Together with asynchronous parallel processing, this allows for scaling of the applications to the European continental level (Henja and Michelson 2012).

Py-ART. Py-ART is an architecture for geophysical retrievals from radial and gridded remote sensing data. Py-ART was designed to add value to the ARM (Ackerman and Stokes 2003) Climate Facility's scanning radars (Mather and Voyles 2013). In Py-ART, polar volumes and Cartesian grids are read into standard data objects (the radar and grid objects), which are fully self-describing. The underlying "radar" structure is based on the CfRadial information model (see previous section); however, Py-ART also supports a range of other formats (see Table 2).

The following is an example of an application chain that has been set up within ARM: 1) read in raw data from the ARM C-Band system in Oklahoma, 2) adjust for reflectivity offsets, 3) extract the propagation component from the differential phase (Giangrande et al. 2013), 4) infer specific attenuation based on Gu et al. (2011), 5) retrieve rainfall rates as a function of specific attenuation (Ryzhkov et al. 2014), and then 6) use a *k*-dimensional (*k*-d)- or ball-tree-based objective analysis technique (Barnes 1964) to map these rates to a Cartesian grid.

Py-ART is available on GitHub as a community code base and has already received contributions from users within and outside ARM. In addition to various open source scientific Python libraries (NumPy, SciPy, matplotlib), Py-ART incorporates radar-specific open source libraries such as RSL (see above) and the four-dimensional Doppler dealiasing scheme (James and Houze 2001).

wradlib. The development of the weather radar processing library (wradlib; Heistermann et al. 2013) was initiated in 2011 by the Universities of Potsdam and Stuttgart, Germany. wradlib aims to facilitate interactive analysis of radar data and offline processing in research environments. Operational applications might be possible, but are, so far, not the main development goal. Basic processing offers access to a wide range of national and international file formats, georeferencing, gridding, and compositing, as well as high-level visualization on the polar and Cartesian levels. Other modules address the detection and correction of major error sources as well as differential phase processing (Table 3). In this way, wradlib allows the user to customize workflows in order to meet various requirements. Among the users and the developers, there is a strong focus on hydrological applications. wradlib is designed to support Windows, Linux, and Macintosh Operating System (Mac OS) platforms. This is facilitated by using Python as the principal programming and interface language. wradlib also makes use of various open scientific Python libraries (e.g., NumPy, SciPy, matplotlib).

Another motivation for wradlib was to provide a community platform to develop and share code across institutional boundaries. Given that it has no dedicated funding, wradlib's development is actually driven by its (as yet small) user community. Therefore, extensive steps have been undertaken to facilitate collaboration and exchange between users and developers, including distributed version control and public code hosting, mailing lists for users and developers, and extensive online documentation (http://wradlib.bitbucket.org), with an infrastructure to keep code and documentation synchronized. This documentation not only provides a detailed application programming interface (API) reference, but also allows newcomers to get started with tutorials, recipes, and worked examples that also demonstrate how to combine the different functions in order to create complete workflows.

LESSONS LEARNED. In "The potential benefits of open source software," we pointed out the promises of the open source paradigm to the field of radar data processing. The recent emergence of several OSS platforms provides evidence that some of these promises might just come to fruition: Five active OSS community projects were reviewed in the previous section, and some distinct lessons can be learned from their comparison.

First, most projects are very specific with respect to their background and their intended users. Py-ART and wradlib are similar in their intention to facilitate convenient, interactive tool sets in typical research environments. This implies a focus on rapid prototyping of new algorithms and is achieved by using a high-level language such as Python and correspondingly high-level libraries for scientific computing and visualization. In contrast, BALTRAD's prime feature is the operational, real-time exchange and processing of data in large radar networks. LROSE is somewhere in between, as it aims to suit both operational system requirements and research environments. Finally, RSL focuses entirely on its role to provide a uniform interface for legacy formats.

Second, we have shown how some of the projects have benefited from the incorporation of code from other OSS projects (both specific and unspecific to radar), supporting our hypothesis that OSS speeds up scientific progress in the field of radar data processing. In the section on "The potential benefits of open source software," we established that interoperability enhances the potential for "cross-fertilization" effects among different platforms. The interaction between the BALTRAD exchange system and the BALTRAD toolbox as well as the incorporation of the RSL library in Py-ART are excellent examples. It is interesting to note, though, that strategies to ensure interoperability differ among platforms. BALTRAD entirely relies on the ODIM data model and leaves it to the community to decide how to convert local data to achieve ODIM compliance. LROSE, Py-ART, RSL, and wradlib also support the modern open data models; however, their strategy is to include as many legacy formats as possible to meet the requirements of their respective user communities.

Third, the role of "community" deserves some clarification. So far, we have mainly focused on the role of OSS, but the projects presented all have a substantial community involvement. This is a typical feature of OSS projects, but not a prerequisite. Furthermore, the notion of community in OSS is necessarily fuzzy. Traditionally, we distinguish between user and developer communities, but for OSS, these communities interact at multiple levels. According to Robles (2004), users should rather be

considered as codevelopers. This concept is very real in all of the aforementioned OSS projects where the developer communities actually form part of the user communities. Users contribute feedback in terms of bug reports and feature requests. Most of the projects undertook extensive measures to encourage user feedback (e.g., by issue tracking and mailing lists). As a side effect of mailing lists, active user communities are typically eager to provide support to fellow users.

The experiences with community involvement in BALTRAD deserve some special attention. BALTRAD is not only unique from a technical point of view but also with respect to the scale of funding, the size of the partnership, and the operational ambition. Following its public release, the software has been downloaded widely and deployed in some cases by organizations outside the partnership. The incubator nature of the EU structural funds has obviously succeeded in creating a critical mass and momentum that is set to continue. The funds have been helpful in establishing a large community, but it has been a challenge to efficiently implement the concepts of community-based development. In particular, it proved difficult to combine the two aims of 1) establishing a central, sustainable, and operationally viable architecture and 2) integrating the interests of the partners, some with significant legacy code bases. This conflict has the potential to impede the implementation of OSS concepts, particularly if issues of distrust among partners are not addressed, and if partners feel that their own interests do not align with community interests.

It might be obvious that such tensions are inevitable in large consortia. However, for scientists and OSS enthusiasts it might be a new and perhaps disillusioning experience that the open source sector is not immune to such issues. The lesson learned at least from BALTRAD is that working as a community with open source requires making an extra effort, which must not be underestimated, to understand how each partner can both contribute and benefit.

However, this is not the only area of conflict. Other issues emerged from the previous section that indicate trade-offs and the need for compromises:

i) Real-time operations versus research laboratory: Performance is a fundamental issue in any real-time operation that involves large radar networks. In settings where performance is a priority, we traditionally find dense software code written mostly in low-level languages, at the cost of clarity and transferability. In contrast, researchers often prefer high-level programming languages that allow for rapid prototyping and legible code, however, at the cost of performance. Good coding practices can minimize this trade-off but probably not resolve it. Likewise, good system design can reduce the effort required to migrate code from research to operations.

ii) Linux versus Microsoft: Many operational services traditionally rely on UNIX/Linux systems. This also holds true for many scientists from the meteorological community. Nonetheless, Microsoft Windows is widely used in research environments. Therefore, platform support (including Windows) is a substantial criterion for some users. Platform independence is easier to achieve via high-level programming languages, but again, somewhat at the cost of performance. So far, wradlib is the only package that explicitly supports both Linux and Microsoft Windows.

iii) Enterprise versus community support: There is no definite distinction between proprietary and open source products with respect to support and maintenance strategies. There may be substantial community support available for commercial products. Then again, developers or third parties often provide commercial support and maintenance packages for community-based OSS products. The same fuzziness applies to long-term support: users generally expect long-term stability and support from commercial vendors; however, this expectation falters if a company leaves the weather radar business or if key developers leave the company. Community-based OSS products usually come without guarantees; however, long-term viability can be achieved if the size of the community reaches a critical mass. While this critical mass is certainly exceeded for BALTRAD, LROSE, and RSL, wradlib and Py-ART are currently in the process of expanding their community bases.

iv) Turnkey solutions versus flexibility: A common feature of the OSS tools presented in the previous section is that successful application requires advanced computational skills for system configuration or application development. In contrast, proprietary software often comes as a complete "turnkey" solution that closely integrates radar hardware, control software, as well as data processing, visualization, and dissemination. Often, users without the expertise necessary to use such OSS solutions do not have a choice but to start with the proprietary product. Only after a while, users might recognize that the solution does not meet their specific needs. It should be possible,

though, to develop more "user friendly" and complete OSS solutions. Yet, insufficient support of transparent and open data models can be considered as a main barrier. Furthermore, the tools presented in the previous section are, with the exception of BALTRAD, mainly rooted in academic environments. It might just be a matter of time until these tools will be streamlined for broader user communities.

COMBINING INDIVIDUAL PROJECTS INTO A TRUE COMMUNITY EFFORT. In the previous sections, we have discussed five OSS projects that have been successful, each in their own way, in providing software tools to the weather radar community.

Each project on its own, however, falls short of the goal of a true community-wide effort. Funding is limited, development teams are small, and the problem domain is large. No single project has the resources to meet the diverse needs of the whole community. What is needed is a way of combining the efforts such that the whole is greater than the sum of the parts. To do so requires interoperability between the systems so that development in one project can save time and effort in another. As the following points show, we have good reason to be optimistic.

Enhanced communication between the project teams. Team members from each of these projects are included in the author list of this paper. That in itself has raised the awareness of what the other projects are doing, and will lead to improved communication, collaboration, and decision making in the near future. As a first direct result, members of BALTRAD, Py-ART, and wradlib jointly organized a Short Course on Open Source Radar Software within the Eighth European Conference on Radar in Meteorology and Hydrology (ERAD) on 31 August 2014 (www.pa.op.dlr.de/erad2014), including a demonstration of software interoperability.

Advances in common data formats. Table 2 lists 17 actively used radar data formats, and this list is not exhaustive. Dealing with so many variants is inefficient and costly. However, two modern formats (ODIM_H5 and CfRadial) are emerging as having wide support and are becoming de facto standards. ODIM_H5 is based on the HDF5 framework, while CfRadial is based on NetCDF. Both are therefore self-describing and readily accessible via any language that has HDF5 and netCDF support.

Improvements in language tools. The growth of Python as a scientific computing platform has resulted in a proliferation of tools on which to build radar processing packages (Lin 2012). Many legacy modules are, however, in C and C++. Fortunately, Python tools can easily be layered on top of C and C++ modules, thereby permitting reuse of legacy code that has been thoroughly tested and debugged. So Py-ART can, as previously mentioned, use code from RSL and LROSE. And BALTRAD could, in principle, import modules from TITAN.

Improvements in collaborative tools. While we would generally expect an increase in the number of developers to speed up the rate of scientific and technological progress, a large developer community also poses challenges in achieving efficient collaboration. Py-ART, wradlib, and BALTRAD have all chosen a uniform approach, and LROSE will soon follow suit. They are maintained by a limited number of lead developers and make use of distributed version control systems (such as Git). Collaborative community contributions are managed via the so-called Fork and Pull model (see http://help.github.com/articles/using-pull-requests): Any user is allowed to fork from the main branch, any user can propose changes, and any user can review these changes, but the final decision about whether a requested change is actually included into the main branch is made by the lead developers. This approach has been successfully applied to many OSS efforts, including the very large but collaborative Linux kernel project for which Git was originally developed.

CONCLUSIONS. This article is only a snapshot of developments that are ongoing. We discussed the specific benefits of the open source paradigm for the weather radar community and presented the activities of five disparate OSS projects in the field of radar data processing. We found that these projects have been remarkably successful in addressing the needs of specific user groups.

There may be a natural desire to consolidate these efforts into a single uniform community platform. Based on our findings, though, we conclude that a single solution will not be able to accommodate the diverse needs of the entire community. Furthermore, there are priorities (national, institutional, personal) that will most likely prevent such a consolidation. Instead, we expect different projects to coexist and to interact in a dynamic collaboration. The guiding principle of such a cross-project collaboration will be interoperability, allowing not only for the

platforms to exchange data, but also to exchange code (e.g., as shared libraries, code fragments, and open algorithms) and thus speed up technological and scientific progress through cross-fertilization. In particular, this development will facilitate the transfer of mature algorithms from the research domain to operational applications. But while the standardization of algorithms is an important concern of the radar community, we should also be aware that diversity of approach is an important aspect of scientific endeavor.

To foster active communication within the community, we have created a community resource to present and discuss radar-related OSS tools: http://theradarcommunity.wikidot.com. We invite the *BAMS* audience—as a subset of both the developer and user community—to use this site as a resource to help decide which software to use for their specific requirements or which software effort to support through development collaboration.

ACKNOWLEDGMENTS. The development of TITAN was funded by the U.S. Federal Aviation Administration. The development of LROSE is funded by the U.S. National Science Foundation. BALTRAD software has been developed as part of the BALTRAD and BALTRAD+ projects that have been partly financed by the European Union (European Regional Development Fund and European Neighbourhood and Partnership Instrument). Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, under Contract DE-AC02-06CH11357. This work has been supported by the Office of Biological and Environmental Research (OBER) of the U.S. Department of Energy (DOE) as part of ARM. The development of wradlib was partly funded by the German Federal Ministry for Research and Education within the PROGRESS project. The development of RSL and RSL-in-IDL were supported by NASA's Precipitation Measurement Missions program. The authors thank Jonathan J. Gourley, Norman Donaldson, and Marco Borga who reviewed this paper and who substantially contributed to its improvement.

REFERENCES

Ackerman, T. P., and G. M. Stokes, 2003: The Atmospheric Radiation Measurement program. *Phys. Today,* **56,** 38–44, doi:10.1063/1.1554135.

Barnes, S. L., 1964: A technique for maximizing details in numerical weather map analysis. *J. Appl. Meteor.,* **3,** 396–409, doi:10.1175/1520-0450(1964)0032.0.CO;2.

——, 1980: Report on a meeting to establish a common Doppler radar data exchange format. *Bull. Amer. Meteor. Soc.,* **61,** 1401–1404.

Bringi, V. N., and V. Chandrasekar, 2001: *Polarimetric Doppler Weather Radar.* Cambridge University Press, 636 pp.

Casson, T., and P. S. Ryan, 2006: Open standards, open source adoption in the public sector, and their relationship to Microsoft's market dominance. *Standards Edge: Unifier or Divider?,* S. Bolin, Ed., Sheridan Books, 87–99. [Available online at http://ssrn.com/abstract=1656616.]

Dixon, M., and G. Wiener, 1993: TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A radar-based methodology. *J. Atmos. Oceanic Technol.,* **10,** 785–797, doi:10.1175/1520-0426(1993)0102.0.CO;2.

——, W.-C. Lee, B. Rilling, and C. Burghart, 2013: CfRadial data file format: Proposed CF-compliant netCDF format for moments data for RADAR and LIDAR in radial coordinates. NCAR, 66 pp. [Available online at www.eol.ucar.edu/system/files/CfRadialDoc.v1.3.20130701.pdf.]

Doviak, R. J., and D. S. Zrnić, 1993: *Doppler Radar and Weather Observations.* 2nd ed. Dover Publications, 562 pp.

Eaton, B., and Coauthors, 2011: NetCDF Climate and Forecast (CF) Metadata Conventions version 1.6. Tech. Doc., 124 pp. [Available online at http://cfconventions.org/documents.html.]

Fornasiero, A., J. Bech, and P. P. Alberoni, 2006: Enhanced radar precipitation estimates using a combined clutter and beam blockage correction technique. *Nat. Hazards Earth Syst. Sci.,* **6,** 697–710, doi:10.5194/nhess-6-697-2006.

Gabella, M., and R. Notarpietro, 2002: Ground clutter characterization and elimination in mountainous terrain. *Proc. Second European Conf. on Radar Meteorology,* Delft, Netherlands, Delft University of Technology, 305–311. [Available online at www.copernicus.org/erad/online/erad-305.pdf.]

German Weather Service, 2004: Projekt RADOLAN: Routineverfahren zur online-aneichung der radarniederschlagsdaten mit hilfe von automatischen bodenniederschlagsstationen. DWD Final Rep., 111 pp. [Available online at www.dwd.de/RADOLAN.]

Germann, U., G. Galli, M. Boscacci, and M. Bolliger, 2006: Radar precipitation measurement in a mountainous region. *Quart. J. Roy. Meteor. Soc.,* **132,** 1669–1692, doi:10.1256/qj.05.190.

——, M. Berenguer, D. Sempere-Torres, and M. Zappa, 2009: REAL—Ensemble radar precipitation

estimation for hydrology in a mountainous region. *Quart. J. Roy. Meteor. Soc.,* **135,** 445–456, doi:10.1002/qj.375.

Giangrande, S. E., R. McGraw, and L. Lei, 2013: An application of linear programming to polarimetric radar differential phase processing. *J. Atmos. Oceanic Technol.,* **30,** 1716–1729, doi:10.1175/JTECH-D-12-00147.1.

Gill, R. S., M. B. Soerensen, T. Boevith, J. Koistinen, M. Peura, D. Michelson, and R. Cremonini, 2012: BALTRAD dual polarization hydrometeor classifier. *Proc. Seventh European Conf. on Radar in Meteorology and Hydrology,* Toulouse, France, Meteo France, 2.6. [Available online at www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/MIC_110_ext_abs.pdf.]

Gu, J.-Y., A. Ryzhkov, P. Zhang, P. Neilley, M. Knight, B. Wolf, and D.-I. Lee, 2011: Polarimetric attenuation correction in heavy rain at C band. *J. Appl. Meteor. Climatol.,* **50,** 39–58, doi:10.1175/2010JAMC2258.1.

Haase, G., and T. Landelius, 2004: Dealiasing of Doppler radar velocities using a torus mapping. *J. Atmos. Oceanic Technol.,* **21,** 1566–1573, doi:10.1175/1520-0426(2004)0212.0.CO;2.

Heistermann, M., S. Jacobi, and T. Pfaff, 2013: Technical Note: An open source library for processing weather radar data (*wradlib*). *Hydrol. Earth Syst. Sci.,* **17,** 863–871, doi:10.5194/hess-17-863-2013.

Henja, A., and D. Michelson, 2012: Improving the quality of European weather radar composites with the BALTRAD toolbox. *Proc. Seventh European Conf. on Radar in Meteorology and Hydrology,* Toulouse, France, Meteo France. [Available online at www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/MIC_110_ext_abs.pdf.]

——, M. Szewczykowski, S. Ernes, and D. Michelson, 2010: The BALTRAD technical platform. *Proc. Sixth European Conf. on Radar in Meteorology and Hydrology,* Sibiu, Romania, Meteo-Romania. [Available online at www.erad2010.org/pdf/oral/wednesday/dataex/02_ERAD2010_0232.pdf.]

Hitschfeld, W., and J. Bordan, 1954: Errors inherent in the radar measurement of rainfall at attenuating wavelengths. *J. Atmos. Sci.,* **11,** 58–67, doi:10.1175/1520-0469(1954)011<0058:EIITRM>2.0.CO;2.

Hubbert, J. C., M. Dixon, and S. M. Ellis, 2009: Weather radar ground clutter. Part II: Real-time identification and filtering. *J. Atmos. Oceanic Technol.,* **26,** 1181–1197, doi:10.1175/2009JTECHA1160.1.

Jacobi, S., M. Heistermann, and T. Pfaff, 2012: Evaluation and improvement of C-band radar attenuation correction for operational flash flood forecasting. *IAHS Publ.,* **351,** 33–38.

James, C. N., and R. A. Houze, 2001: A real-time four-dimensional Doppler dealiasing scheme. *J. Atmos. Oceanic Technol.,* **18,** 1674–1683, doi:10.1175/1520-0426(2001)0182.0.CO;2.

Krämer, S., and H.-R. Verworn, 2009: Improved radar data processing algorithms for quantitative rainfall estimation in real time. *Water Sci. Technol.,* **60,** 175–184, doi:10.2166/wst.2009.282.

Lin, J. W.-B., 2012: Why Python is the next wave in Earth sciences computing. *Bull. Amer. Meteor. Soc.,* **93,** 1823–1824, doi:10.1175/BAMS-D-12-00148.1.

Mather, J. H., and J. W. Voyles, 2013: The ARM Climate Research Facility: A review of structure and capabilities. *Bull. Amer. Meteor. Soc.,* **94,** 377–392, doi:10.1175/BAMS-D-11-00218.1.

Michelson, D., 2006: The Swedish weather radar production chain. *Proc. Fourth European Conf. on Radar in Meteorology and Hydrology,* Barcelona, Spain, CRAHI. [Available online at www.crahi.upc.edu/ERAD2006/proceedingsMask/00101.pdf.]

——, and A. Henja, 2013: Implementation of hit-accumulation clutter filter in BALTRAD toolbox. EUMETNET OPERA Working Doc. WD_2012_02p, 8 pp.

——, I. Holleman, H. Hohti, and M. Salomonsen, 2003: HDF5 information model and implementation specification for weather radar data. COST 717 Working Doc. WDF_02_200204_1, 27 pp. [Available online at www.smhi.se/cost717/doc/WDF_02_200204_1.pdf.]

——, R. S. Gill, M. Peura, and J. Szturc, 2010: Community-based weather radar networking with BALTRAD. *Proc. Sixth European Conf. on Radar in Meteorology and Hydrology,* Sibiu, Romania, Meteo-Romania. [Available online at www.erad2010.org/pdf/oral/wednesday/dataex/01_ERAD2010_0170.pdf.]

——, J. Koistinen, T. Peltonen, J. Szturc, and M. R. Rasmussen, 2012: Advanced weather radar networking with BALTRAD+. *Seventh European Conf. on Radar in Meteorology and Hydrology,* Toulouse, France, Meteo France. [Available online at www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/NET_073_ext_abs.pdf.]

——, R. Lewandowski, M. Szewczykowski, H. Beekhuis, and G. Haase, 2014: EUMETNET OPERA weather radar information model for implementation with the HDF5 file format, version 2.2. EUMETNET OPERA Deliverable, 38 pp. [Available online at www.eumetnet.eu/sites/default/files/OPERA2014_O4_ODIM_H5-v2.2.pdf.]

Ośródka, K., J. Szturc, and A. Jurczyk, 2014: Chain of data quality algorithms for 3-D single-polarization radar reflectivity (RADVOL-QC system). *Meteor. Appl.,* **21,** 256–270, doi:10.1002/met.1323.

Peura, M., 2002: Computer vision methods for anomaly removal. *Proc. Second European Conf. on Radar Meteorology,* Delft, Netherlands, Delft University of Technology, 312–317. [Available online at http://copernicus.org/erad/online/erad-312.pdf.]

——, 2010: The living composite. *Proc. Sixth European Conf. on Radar in Meteorology and Hydrology,* Sibiu, Romania, Meteo-Romania. [Available online at www.erad2010.org/pdf/oral/wednesday/dataex/03_ERAD2010_0275.pdf.]

——, 2012: Rack—A program for anomaly detection, product generation, and compositing. *Seventh European Conf. on Radar in Meteorology and Hydrology,* Toulouse, France, Meteo France. [Available online at www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/DQ_304_ext_abs.pdf.]

——, J. Koistinen, and H. Hohti, 2006: Quality information in processing weather radar data for varying user needs. *Proceedings of the Fourth European Conference on Radar in Meteorology and Hydrology,* CRAHI, 563–566.

Raymond, E. S., 1999: *The Cathedral and the Bazaar.* O'Reilly Media, 241 pp.

Robles, G., 2004: A software engineering approach to libre software. *Open Source Jahrbuch 2004,* R. A. Gehring and B. Lutterbeck, Eds., Technical University of Berlin, 193–208. [Available online at www.opensourcejahrbuch.de/download/jb2004/chapter_03/III-3-Robles.pdf.]

Ryzhkov, A., M. Diederich, P. Zhang, and C. Simmer, 2014: Potential utilization of specific attenuation for rainfall estimation, mitigation of partial beam blockage, and radar networking. *J. Atmos. Oceanic Technol.,* **31,** 599–619, doi:10.1175/JTECH-D-13-00038.1.

St. Laurent, A. M., 2008: *Understanding Open Source and Free Software Licensing.* O'Reilly Media, 208 pp.

Szturc, J., and Coauthors, 2012: Data quality in the BALTRAD+ project. *Seventh European Conf. on Radar in Meteorology and Hydrology,* Toulouse, France, Meteo France. [Available online at www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/DQ_374_ext_abs.pdf.]

Viglione, A., M. Borga, P. Balabanis, and G. Bloeschl, 2010: Barriers to the exchange of hydrometeorological data across Europe: Results from a survey and implications for data policy. *J. Hydrol.,* **394,** 63–77, doi:10.1016/j.jhydrol.2010.03.023.

Villarini, G., and W. F. Krajewski, 2010: Review of the different sources of uncertainty in single polarization radar-based estimates of rainfall. *Surv. Geophys.,* **31,** 107–129, doi:10.1007/s10712-009-9079-x.

Vivekanandan, J., D. S. Zrnić, S. M. Ellis, R. Oye, A. V. Ryzhkov, and J. Straka, 1999: Cloud microphysics retrievel using S-band dual-polarization radar measurements. *Bull. Amer. Meteor. Soc.,* **80,** 381–388, doi:10.1175/1520-0477(1999)0802.0.CO;2.

von Krogh, G., and S. Spaeth, 2007: The open source software phenomenon: Characteristics that promote research. *J. Strategic Inf. Syst.,* **16,** 236–253, doi:10.1016/j.jsis.2007.06.001.

Vulpiani, G., M. Montopoli, L. D. Passeri, A. G. Gioia, P. Giordano, and F. S. Marzano, 2012: On the use of dual-polarized C-band radar for operational rainfall retrieval in mountainous areas. *J. Appl. Meteor. Climatol.,* **51,** 405–425, doi:10.1175/JAMC-D-10-05024.1.

Wang, Y., and V. Chandrasekar, 2009: Algorithm for estimation of the specific differential phase. *J. Atmos. Oceanic Technol.,* **26,** 2565–2578, doi:10.1175/2009JTECHA1358.1.

WGLS, 2000: Free software/open source: Information society opportunities for Europe? Working Group on Libre Software Rep., Version 1.2, 42 pp. [Available online http://eu.conecta.it/paper.pdf.]