

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

M. Heistermann¹, S. Collis², M. J. Dixon³, S. Giangrande⁴, J. J. Helmus²,
B. Kelley⁵, J. Koistinen⁶, D. B. Michelson⁷, M. Peura⁶, T. Pfaff⁸, and D. B. Wolff⁵

- [1] {University of Potsdam, Institute of Earth and Environmental Sciences, Potsdam, Germany}
[2] {Argonne National Laboratory, Argonne, Illinois, USA}
[3] {National Center for Atmospheric Research (NCAR), Boulder, Colorado, USA}
[4] {Brookhaven National Laboratory, Environmental Sciences Department, Upton, New York, USA}
[5] {NASA GSFC/Wallops Flight Facility, Wallops Island, Virginia, USA}
[6] {Finnish Meteorological Institute, Helsinki, Finland}
[7] {Swedish Meteorological and Hydrological Institute, Norrköping, Sweden}
[8] {University of Stuttgart, Institut für Wasser- und Umweltsystemmodellierung, Stuttgart, Germany}

Correspondence to: Maik Heistermann, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany; e-Mail: maik.heistermann@uni-potsdam.de; phone: +49 331 977 2671

May 2014

For publication in
Bull. Am. Meteorol. Soc.
(e-view, doi:10.1175/BAMS-D-13-00240.1, 2014).

Environmental & Climate Sciences Dept.

Brookhaven National Laboratory
P.O. Box 5000
Upton, NY 11973-5000
www.bnl.gov

Notice: This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

This preprint is intended for publication in a journal or proceedings. Since changes may be made before publication, it may not be cited or reproduced without the author's permission.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Capsule

To date, progress in the use of weather radar observations has been impeded by a lack of community-based software development. Recently, though, several Open Source products have demonstrated that OSS can be a real benefit to the radar community.

Abstract

Weather radar analysis has become increasingly sophisticated over the past 50 years, and efforts to keep software up to date have generally lagged behind the needs of the users. We argue that progress has been impeded by the fact that software has not been developed and shared as a community.

Recently, the situation has been changing. In this paper, the developers of a number of Open Source Software (OSS) projects highlight the potential of OSS to advance radar-related research. We argue that the community-based development of OSS holds the potential to reduce duplication of efforts, and to create transparency in implemented algorithms while improving the quality and scope of the software. We also conclude that there is sufficiently mature technology to support collaboration across different software projects. This could allow for consolidation towards a set of interoperable software platforms, each designed to accommodate very specific user requirements.

1 Introduction

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.

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 non-standard; 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).

In order 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 (Section 2)? Why do we expect OSS to better meet these requirements (Section 3)? In Section 4, we highlight five active OSS projects, and present lessons learned from these projects in Section 5. Section 6 discusses

how the efforts of those projects might be combined such that the whole is greater than the sum of its parts. **Section 7** finally presents our overall conclusion.

2 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, e. g. in radar engineering, severe weather detection, atmospheric modeling, water resources management, agriculture, validation of remote sensing products, TV broadcasting, or even the tracking of non-meteorological echoes such as birds and insects.

Instead, we briefly outline typical components of a radar processing chain. **Section 3** will then introduce how OSS can improve these individual components, and to 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 Zrnic 1993, Bringi and Chandrasekar 2001**). On many systems this processing is complete by the time the moments 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 towards common data models (see [Section 3](#)), 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 geo-data. 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 e.g. 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 measurements of a quantity aloft which 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 Z(R) relationships, and others (Villarini et al. 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.

3 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 [Section 2](#), 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, e. g. for addressing specific needs towards 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 towards 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, e.g. 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 *RSL* and *Radx* (see Section 4). More recent data models are mainly based on NetCDF and HDF5 formats. The OPERA Data Information Model (ODIM) has evolved in Europe (Michelson et al. 2014, Michelson et al. 2003), and exists in HDF5 and BUFR representations. The ODIM_H5 representation has spread widely, being embraced by the community to the extent that, as of February 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 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.

4 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).

Insert Tables 1-3 within Section 4 of the manuscript

(see tables at the end of the manuscript)

LROSE / TITAN

The initial goal of *TITAN* (<http://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 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 US National Science Foundation in 2012, with initial priority given to the implementation of a standardized open data model. To date, progress 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

(<http://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

NASA's Radar Software Library (*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 (3-d 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 in the *Py-ART* software (see below), and a high-level implementation is available for *IDL*.

BALTRAD

15 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, Michelson et al. 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 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 (Section 3). 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 I/O, and the ability to chain well-defined processing algorithms in memory. Tools developed by different partners, some of which pre-date 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 RADVOL-QC (Ośródka et al. 2012). Together with asynchronous parallel processing, this allows for scaling of the applications to the European continental level (Henja and Michelson 2012).

Py-ART

The Python-ARM Radar toolkit (*Py-ART*) is an architecture for geophysical retrievals from radial and gridded remote sensing data. *Py-ART* was designed to add value to the Atmospheric Radiation Measurement, 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 Section 3); 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 the ARM program: (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-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 the ARM program. 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-D Doppler dealiasing scheme (James and Houze 2001).

wradlib

The development of *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 off-line 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 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*), as well as the open *OPERA BUFR* software.

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 on-line documentation (<http://wradlib.bitbucket.org>), with an infrastructure to keep code and documentation synchronized. This documentation not only provides a detailed API reference, but also allows newcomers to get started with tutorials, recipes and worked examples which also demonstrate how to combine the different functions in order to create complete workflows.

5 Lessons learned

In Section 3, 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 [Section 4](#), 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 [Section 3](#), 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 co-developers. 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 (a) establishing a central, sustainable and operationally viable architecture, and (b) 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, that 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 Section 4 that indicate trade-offs and the need for compromises:

(a) Real-time operations vs. research lab: 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 which 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.

(b) Linux vs. 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.

(c) Enterprise vs. 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 base.

(d) Turn-key solutions vs. flexibility: A common feature of the OSS tools presented in Section 4 is that successful application requires advanced computational skills for system configuration or application development. In contrast, proprietary software often comes as a complete “turn-key” solution which 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 Section 4 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.

6 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* are jointly organizing a workshop on Open Source Radar Software within the 8th European Conference on Radar in Meteorology and Hydrology (ERAD) on August 31st 2014 (<http://www.pa.op.dlr.de/erad2014>), including a first attempt of demonstrating 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 NASA 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, 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.

7 Conclusion

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 co-exist, 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.

In order 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 US Federal Aviation Administration. The development of *LROSE* is funded by the US 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 the ARM Program. 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 would like to thank three anonymous referees whose comments substantially contributed to improve this paper.

References

- Ackerman, T. P., and G. M. Stokes, 2003: The Atmospheric Radiation Measurement Program. *Physics Today*, **56**, 38-44.
- Barnes, S. L., 1964: A Technique for Maximizing Details in Numerical Weather Map Analysis. *J. Appl. Meteorol.*, **3**, 396-409.
- Barnes, S., 1980: Report on a meeting to establish a common Doppler radar data exchange format. *Bull. Amer. Meteor. Soc.*, **61**(11), 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, 2006. Available at SSRN: <http://ssrn.com/abstract=1656616>.
- Dixon, M., and G. Wiener, 1993: TITAN - Thunderstorm Identification, Tracking, Analysis, And Nowcasting - A Radar-Based Methodology. *J. Atmos. Ocean. Technol.*, **10**(6), 785-797.
- Dixon, M., W.-C. Lee, B. Rilling, C. Burghart, 2013: CfRadial Data File Format – Proposed CF-compliant netCDF Format for Moments Data for RADAR and LIDAR in Radial Coordinates. URL <http://www.eol.ucar.edu/system/files/CfRadialDoc.v1.3.20130701.pdf>
- Doviak, R. J., and D. S. Zrnic, 1993: *Doppler Radar and Weather Observations*, Second Edition. Dover Publications, Mineola, New York, 562 p.

Eaton, B., J. Gregory, B. Drach, K. Taylor, S. Hankin, J. Caron, R. Signell, P. Bentley, G. Rappa, H. Höck, A. Pamment, and M. Juckes, 2011: NetCDF Climate and Forecast (CF) Metadata Conventions Version 1.6. URL: <http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/cf-conventions.pdf>

Fornasiero, A., J. Bech, P. P. Alberoni, 2006: Enhanced radar precipitation estimates using a combined clutter and beam blockage correction technique, *NHESS*, **6**(5), 697-710, 2006.

Gabella, M., and R. Notarpietro, 2002: Ground clutter characterization and elimination in mountainous terrain. *Second European Conference on Radar Meteorology (ERAD), Delft University of Technology, The Netherlands, Copernicus Gesellschaft*, pp. 305-311. URL: <http://www.copernicus.org/erad/online/erad-305.pdf>.

German Weather Service, 2004: Projekt RADOLAN – Routineverfahren zur Online-Anreicherung der Radarniederschlagsdaten mit Hilfe von automatischen Bodenniederschlagsstationen, Final Report, Offenbach, Germany. URL: <http://www.dwd.de/RADOLAN> (in German).

Germann, U., G. Galli, M. Boscacci, M. Bolliger, 2006: Radar precipitation measurement in a mountainous region. *Q. J. R. Meteor. Soc.*, **132**, 1669-1692.

Germann, U., M. Berenguer, D. Sempere-Torres, and M. Zappa, 2009: REAL – Ensemble radar precipitation estimation for hydrology in a mountainous region. *Q. J. R. Meteor. Soc.*, **135**, 445–456.

Giangrande, S. E., R. McGraw, and L. Lei, 2013: An Application of Linear Programming to Polarimetric Radar Differential Phase Processing. *J. Atmos. Ocean. Technol.*, **30**, 1716-1729.

Gill R. S., M.B. Soerensen, T. Boevith, J. Koistinen, M. Peura, D. Michelson, and R. Cremonini, 2012: BALTRAD dual polarization hydrometeor classifier. *ERAD 2012 - The Seventh European Conference On Radar In Meteorology And Hydrology, Toulouse, France, Meteo France*.

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.

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

Henja A., M. Szewczykowski, S. Ernes, and D. Michelson, 2010: The BALTRAD technical platform. Proc. ERAD 2010. Meteo-Romania, Sibiu, Romania.

Henja A. and D. Michelson, 2012: Improving the quality of European weather radar composites with the BALTRAD toolbox. *ERAD 2012 - The Seventh European Conference On Radar In Meteorology And Hydrology, Toulouse, France, Meteo France*.

Hitschfeld, W., and J. Bordan, 1954: Errors Inherent in the Radar Measurement of Rainfall at Attenuating Wavelengths. *J. Atmos. Sc.*, **11**, 58-67.

Hubbert, J. C., M. Dixon, S. M. Ellis, 2009: Weather Radar Ground Clutter. Part II: Real-Time Identification and Filtering. *J. Atmos. Oceanic Technol.*, **26**, 1181–1197.

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.

Kraemer, S., H. R. Verworn, 2009: Improved radar data processing algorithms for quantitative rainfall estimation in real time. *Water Sci. Technol.*, **60**(1), 175-184.

Lin, Johnny Wei-Bing, 2012: Why Python Is the Next Wave in Earth Sciences Computing. *Bull. Amer. Meteor. Soc.*, **93**, 1823–1824.

Mather, J. H., J. W. Voyles, 2013: The Arm Climate Research Facility: A Review of Structure and Capabilities. *Bull. Amer. Meteor. Soc.*, **94**, 377–392.

Michelson D., 2006: The Swedish weather radar production chain. *Proceedings of ERAD 2006 – Fourth European Conference on Radar in Meteorology and Hydrology, Barcelona, Spain, CRAHI*, URL: <http://www.crahi.upc.edu/ERAD2006/proceedingsMask/00101.pdf> [Accessed October 2013].

Michelson D., R. S. Gill, M. Peura, and J. Szturc, 2010: Community-based weather radar networking with BALTRAD. *Proceedings of ERAD 2006 – Fourth European Conference on Radar in Meteorology and Hydrology, Sibiu, Romania, Meteo-Romania*. URL: http://www.erad2010.org/pdf/oral/wednesday/dataex/01_ERAD2010_0170.pdf

Michelson D. B., I. Holleman, H. Hohti, and M. Salomonsen, 2003: HDF5 information model and implementation specification for weather radar data. COST 717 Working Document WDF_02_200204_1. 27 pp., URL: http://www.smhi.se/cost717/doc/WDF_02_200204_1.pdf

Michelson D., J. Koistinen, T. Peltonen, J. Szturc, and M. R. Rasmussen, 2012: Advanced weather radar networking with BALTRAD+. *ERAD 2012 - The Seventh European Conference On Radar In Meteorology And Hydrology, Toulouse, France, Meteo France*. URL: http://www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/NET_073_ext_abs.pdf

Michelson D. B., R. Lewandowski, M. Szewczykowski, H. Beekhuis, and Haase G., 2014: EUMETNET OPERA weather radar information model for implementation with the HDF5 file format. Version 2.2. EUMETNET OPERA Deliverable.

URL: http://www.eumetnet.eu/sites/default/files/OPERA2014_O4_ODIM_H5-v2.2.pdf

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

Ośródką K., J. Szturc, and A. Jurczyk, 2012: Chain of data quality algorithms for 3-D single-polarization radar reflectivity (RADVOL-QC system). *Met. Apps.*, early online release, doi: 10.1002/met.1323.

Peura, M., 2002: Computer vision methods for anomaly removal. *Second European Conference on Radar Meteorology (ERAD), Delft University of Technology, The Netherlands, Copernicus Gesellschaft*, pp. 312–317, URL: <http://copernicus.org/erad/online/erad-312.pdf>.

Peura, M., 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 (ERAD2006), Copernicus*, pp. 563-566.

Peura, M., 2010: The living composite, *Proceedings of the Sixth European Conference on Radar in Meteorology and Hydrology (ERAD2010), Vol. 1 (Advances in Radar Applications)*, pp. 350–354.

Peura, M., 2012: Rack - a program for anomaly detection, product generation, and compositing, *7th European Conference on Radar in Meteorology and Hydrology. ERAD 2012 - The Seventh European Conference On Radar In Meteorology And Hydrology, Toulouse, France, Meteo France*.

URL: http://www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/DQ_304_ext_abs.pdf

Raymond, E. S., 1999: The Cathedral and the Bazaar, O'Reilly Media, 241 pp., URL: <http://www.unterstein.net/su/docs/CathBaz.pdf>.

Robles, G., 2004: A Software Engineering Approach to Libre Software. In: Gehring, R. A., Lutterbeck, B. (Eds.): Open Source Jahrbuch 2004, Technical University of Berlin, Germany. URL: http://www.opensourcejahrbuch.de/download/jb2004/chapter_03/III-3-Robles.pdf

Ryzhkov, A., M. Diederich, P. Zhang, 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.

St. Laurent, A. M. (2008): Understanding Open Source and Free Software Licensing. O'Reilly Media, Sebastopol, CA, USA.

Szturc J., D. Michelson, J. Koistinen, G. Haase M. Peura, R. Gill, M. Sørensen, K. Ośródką, and A. Jurczyk, 2012: Data quality in the BALTRAD+ Project. *ERAD 2012 - The Seventh European Conference On Radar In Meteorology And Hydrology, Toulouse, France, Meteo France*. URL: http://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.

Villarini, G., W. F. Krajewski, 2010: Review of the Different Sources of Uncertainty in Single Polarization Radar-Based Estimates of Rainfall. *Surveys in Geophysics*, **31**(1), 107-129.

Vivekanandan, J., D. S. Zrnic, S. M. Ellis, R. Oye, A.V. Ryzhkov, and J. Straka, 1999: Cloud Microphysics Retrieval Using S-Band Dual-Polarization Radar Measurements. *Bull. Amer. Meteor. Soc.*, **80**, 381–388.

Von Krogh, G., and S. Spaeth, 2007: The open source software phenomenon: Characteristics that promote research. *J. Strat. Inf. Sys.*, **16**(3), 236-253.

Vulpiani, G., M. Montopoli, L. D. Passeri, A. G. Gioia, P. Giordano, 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.

Wang, Yanting, V. Chandrasekar, 2009: Algorithm for Estimation of the Specific Differential Phase. *J. Atmos. Oceanic Technol.*, **26**, 2565–2578.

WGLS (2000): Free Software / Open Source: Information Society Opportunities for Europe? *Report of the Working group on Libre Software, Version 1.2*, URL: <http://eu.conecta.it/paper.pdf>

Tables

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	Pyart-users, Facebook, Twitter	-	wradlib-users, wradlib-dev
Documentation	Online, Doxygen	Online	Online, Sphinx	Online	Online, Sphinx

Table 2: Supported radar file formats.

	BALTRAD	LROSE / TITAN	Py-ART	RSL	wradlib
ARM NetCDF		✓	✓		
CfRadial		✓	✓		
DORADE		✓			
DWD-DX					✓
DWD-RADOLAN					✓
EDGE NetCDF					✓
GAMIC HDF5		✓			✓
Lassen			✓	✓	
McGill			✓	✓	
MDV		✓	✓		
ODIM_H5	✓	✓			✓
ODIM_BUFR	✓				✓
RAINBOW v. 5		✓			
RAPIC		✓			
Sigmat		✓	✓	✓	
UF - Universal Format		✓	✓	✓	
WSR-88D		✓	✓	✓	

Table 3: Algorithms available.

	BALTRAD	LROSE / TITAN	Py-ART	RSL	wradlib
Echo classification and clutter	Peura 2002 Ośródkka et al. 2012 Szturc et al. 2012 Gill et al. 2012 Michelson and Henja 2013	Vivekanandan et al. 1999 Hubbert et al. 2009		Wolff and Kelley 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	Ośródkka et al. 2012, Gill et al. 2012		Gu et. al. 2011		Hitschfeld and Bordan 1954, Kraemer 2009, Jacobi et al. 2011, Vulpiani et al. 2012
VPR	Networked VPR correction				Average VPR
Advanced compositing	Peura 2010 Henja & 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				