



Bluesky's Ahead: A Multi-Facility Collaboration for an *a la Carte* Software Project for Data Acquisition and Management

Daniel Allan, Thomas Caswell, Stuart Campbell & Maksim Rakitin

To cite this article: Daniel Allan, Thomas Caswell, Stuart Campbell & Maksim Rakitin (2019) Bluesky's Ahead: A Multi-Facility Collaboration for an *a la Carte* Software Project for Data Acquisition and Management, Synchrotron Radiation News, 32:3, 19-22, DOI: [10.1080/08940886.2019.1608121](https://doi.org/10.1080/08940886.2019.1608121)

To link to this article: <https://doi.org/10.1080/08940886.2019.1608121>



Published online: 23 May 2019.



Submit your article to this journal [↗](#)



Article views: 92



View related articles [↗](#)



View Crossmark data [↗](#)

Bluesky's Ahead: A Multi-Facility Collaboration for an *a la Carte* Software Project for Data Acquisition and Management



DANIEL ALLAN, THOMAS CASWELL, STUART CAMPBELL, AND MAKSIM RAKITIN

National Synchrotron Light Source II, Brookhaven National Laboratory, Upton, New York, USA

Introduction

To address the growing scale of data at user facilities, a multi-facility collaboration is developing a collection of Python libraries, which are co-developed but may be used *a la carte*, to leverage existing open-source scientific software in general and the scientific Python software ecosystem in particular to improvise cutting-edge experiments and data analysis at the beamline.

The growing velocity and volume of data from third-generation synchrotron light sources and radiation facilities are exposing a data *variety* problem that is particular to user facilities. User facilities manage a large and often changing collection of instruments with a wide span of data rates, structures, and access patterns. Measurements are analyzed with a mix of well-established procedures and improvised techniques. To manage these challenges, user facilities must leverage the growing ecosystem of open-source scientific software, sharing tools and ideas with our counterparts in astronomy, climate science, and particle physics. Tools like GitHub [1], BinderHub [2], and the communities that have arisen around them reduce the friction of distributed collaboration within and between facilities and science domains.

The Bluesky Project is an end-to-end solution [3], encompassing hardware integration [4], experiment specification and orchestration [5], online visualization and analysis, data export [6], and data access [7]. The software is currently in use at all beamlines at the National Synchrotron Light Source II (NSLS-II), a Department of Energy (DOE) Office of Science User Facility located at Brookhaven National Laboratory. It has some adoption at the Advanced Light Source (ALS), another DOE Office of Science User Facility located at Lawrence Berkeley National Laboratory, and is formally planned for wide adoption at the Linear Coherent Light Source II (LCLS-II), located at SLAC National Laboratory, and at the beamlines of the Advanced Photon Source Upgrade (APS-U), located at Argonne National Laboratory—both DOE Office of Science User Facilities. All of the software components have also been employed successfully by an electrical engineering lab [8], working at the lab-bench scale rather than the facility scale, to benchmark hardware performance.

To drive wide collaboration and to overcome “not-invented-here”-ism, the Bluesky Project comprises components with well-defined boundaries that are co-developed but separately useful, and which can be adopted piecemeal. Drawing inspiration from the numpy project, which is the array representation at the core of the scientific Python

ecosystem, Bluesky embraces the idea of *protocols* to drive interoperability. As in the case of numpy, protocols enable individually useful parts to be repurposed and extended in ways unforeseen by the original authors. Facilities and groups can meet their own needs and deadlines, while collaborating on a shared core. Figure 1 illustrates the roles and relationships of the project’s software libraries.

The data variety problem at user facilities


In a typical experiment at a user facility, data and metadata are scattered. Some critical context for interpreting the data is stored only in people’s memories. Metadata may be recorded cryptically in filenames or written in paper notes. Data may be encoded across a variety of formats—some proprietary—which can only be read by specialized software. This approach is manageable up to tens of data sets, but it does not scale well when the number of data sets rises with the data rates from faster detectors and brighter light sources.

What are the specific limitations of the status quo? The data are not machine-readable or searchable, and the relationship between any two pieces of data is unclear. It is difficult to automate away the rote steps of analysis, and code is difficult to generalize or reuse for future experiments. Multimodal analysis, involving plumbing together different techniques with different conventions, is labor-intensive. The traditional approach is also not usually streaming-friendly, which precludes exciting possibilities like observing partial results in real time or performing adaptive experiment steering.

To scale to modern data velocities and volumes, software must systematically track all of the data and metadata necessary for later analysis. This includes:

- Timestamps;
- Secondary measurements (e.g., motor positions, sample temperature);
- “Fixed” experimental values (e.g., distance from sample to detector);
- Calibration data;
- Hardware settings (e.g., exposure time);
- Hardware diagnostics;
- Physical details of the hardware (e.g., physical scale of a pixel).

It encompasses information about the sample and the purpose of the measurement, including:



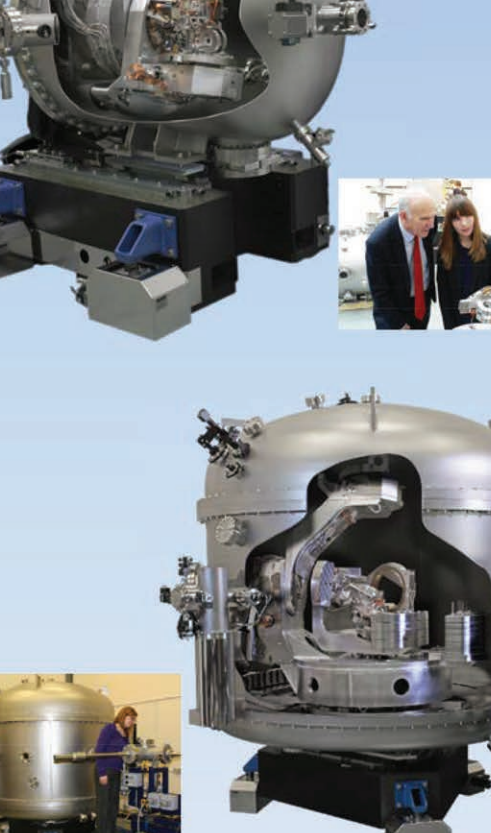
FMB

FMB Oxford

Tailor-made technology

ENDSTATIONS

BIG SOLUTIONS FOR BIG PROBLEMS



- Beamlines
- Monochromators

- Mirrors
- Components

FMB Oxford Unit 1 Ferry Mills Osney Mead
 Oxford OX2 0ES UK +44 (0)1865 320300
www.fmb-oxford.com Sales@FMB-Oxford.com

- Identification of the sample;
- Description of how it was prepared;
- References/links to external materials databases;
- The intent of the measurement (why are we looking at this sample?).

Finally, it includes bureaucratic information about data management:

- Where is the data and how can it be accessed, both on-site and off-site?
- Who took the data?
- Who owns or can access the data?
- How long will the facility store the data?

Bluesky's data model aims to create a place for all of this context without tying itself to a specific facility, a fixed set of science domains, or a sprawling taxonomy of names.

Data model

Bluesky represents data and metadata in a small variety of “documents,” key-value mappings that adhere to a minimal but formalized

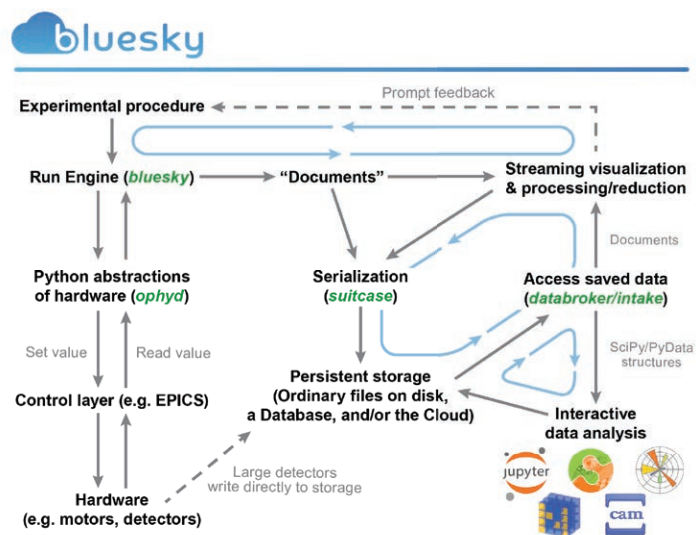


Figure 1: The green labels name some Python libraries that constitute the Bluesky Project. The scientist initiates data collection with an experimental procedure, a sequence of steps that may incorporate adaptive logic that depends on previous measurements. All communication with hardware occurs through an orchestration engine that captures readings and metadata and handles any errors safely. Data flow from the control system (e.g., EPICS) through a Python hardware abstraction layer and are organized by Bluesky in “documents” (key-value mappings) that are dispatched to any number of consumers. The documents are typically saved to a database (e.g., MongoDB) and additionally exported to ordinary files (e.g., CSV or TIFF) if desired. They are typically also sent directly to a live data processing and visualization pipeline. After the experiment, the data may be retrieved either as standard data structures suitable for use with existing scientific Python libraries or as Bluesky “documents,” suitable for piping back through the same pipelines that were used on the live data.

schema. A sequence of documents maps human-readable labels to measurements and associated timestamps. Other documents provide room for hardware configuration and metadata from the facility, instrument, user group, or individual scientist. The essential core is thin and straightforward. Additional facility-specific and technique-specific schemas may be layered on top of it and enforced as desired in accordance with established standards or policies.

The data model is emphatically not a file format—Bluesky is unopinionated about file formats—but the documents may be exported (i.e., serialized and transported) to any format and by any mode desired. To support traditional file-based workflows, ready-to-use exporters for common generic formats (e.g., CSV, TIFF) and specialized formats (e.g., SPEC [9], NeXus [10]) are available, and writing new ones requires low effort. But there is no need to write the documents' contents to files on disk before analyzing them. During data acquisition, documents are made available in a streaming fashion and can be routed directly into software for “online” visualization, reduction, and analysis. This code may run directly in the same process with acquisition, in a separate process on the same machine, elsewhere on the local network, or on remote HPC cluster. The involved software interfaces are straightforward and carefully delineated to make it easy to build bridges to new or existing software, formats, and network protocols.

Replay the experiment

Bluesky's design enables the same data reduction, processing, visualization, or analysis code to be run on “live” data during an experiment and on saved data. By writing code to Bluesky's *documents*, represented as standard data structures in memory, rather than *files*, the important scientific logic is separated from the input/output details. The design has a “streaming first” orientation: consumers of these documents do not need to know whether they are receiving data directly from the acquisition process or data that are being “replayed” from disk. Saved data may be stored in any file format or database and later deserialized back into the Bluesky document representation.

Interactive data analysis

The ultimate goal of this project, as stated at the beginning of this article, is to tackle the data “variety” challenge at user facilities by leveraging existing open-source software projects. In the scientific Python ecosystem, projects are able to interoperate and build on one another—addressing broad problems from image analysis [11] to machine learning [12] and narrower applications from particle tracking [13] to designing psychology experiments [14]—by using standard protocols and data structures, such as the numpy array (a container for N-dimensional arrays), the pandas DataFrame (a spreadsheet-like object for labeled tabular data), and the xarray Dataset (labeled N-dimensional data).

The Bluesky Project includes a component that provides the user with their data and metadata in these standard structures, regardless of how the data were saved to disk. This essentially allows us to define a programmatic interface for the data, rather than trying to standardize on a single data file format. From the users' point of view, reading a multi-

Tailor-made technology


FMB Oxford

ENDSTATIONS

BIG SOLUTIONS FOR BIG PROBLEMS



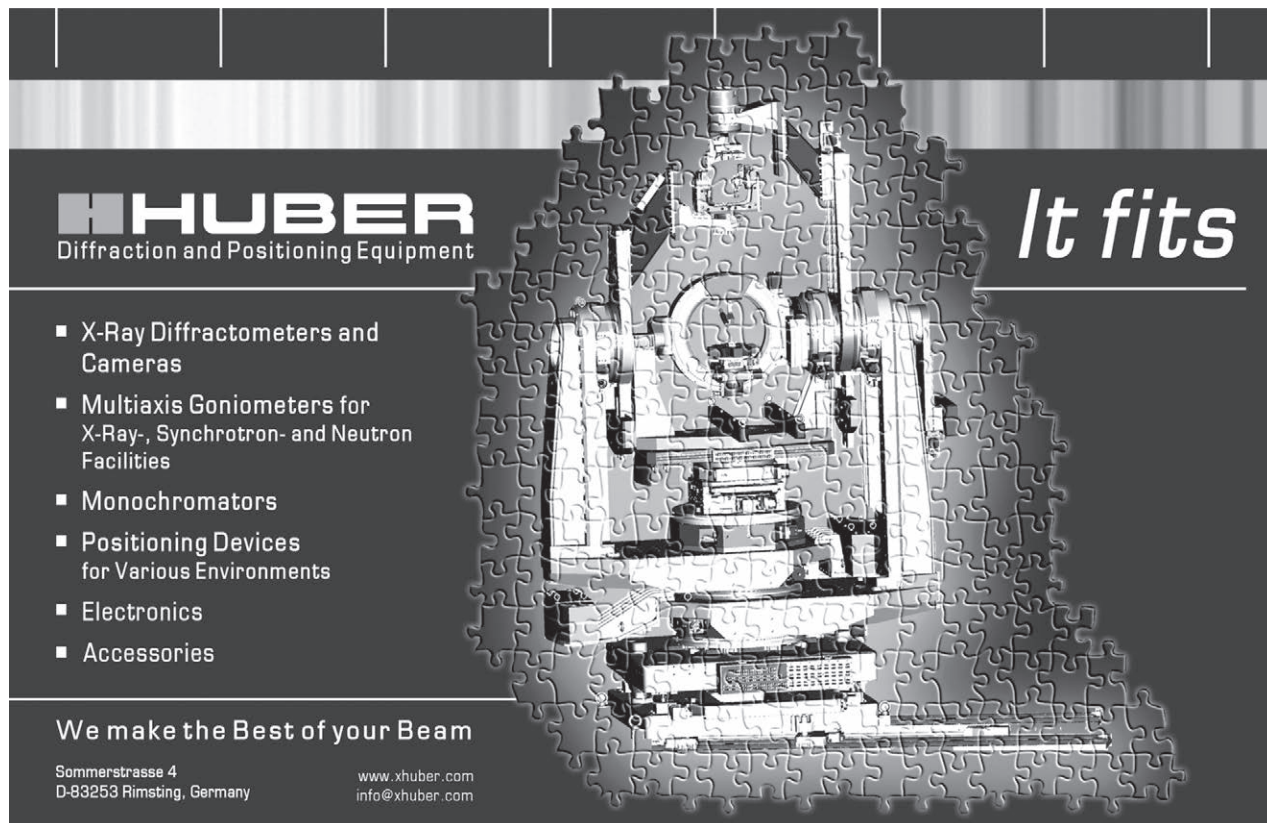


- Beamlines
- Monochromators

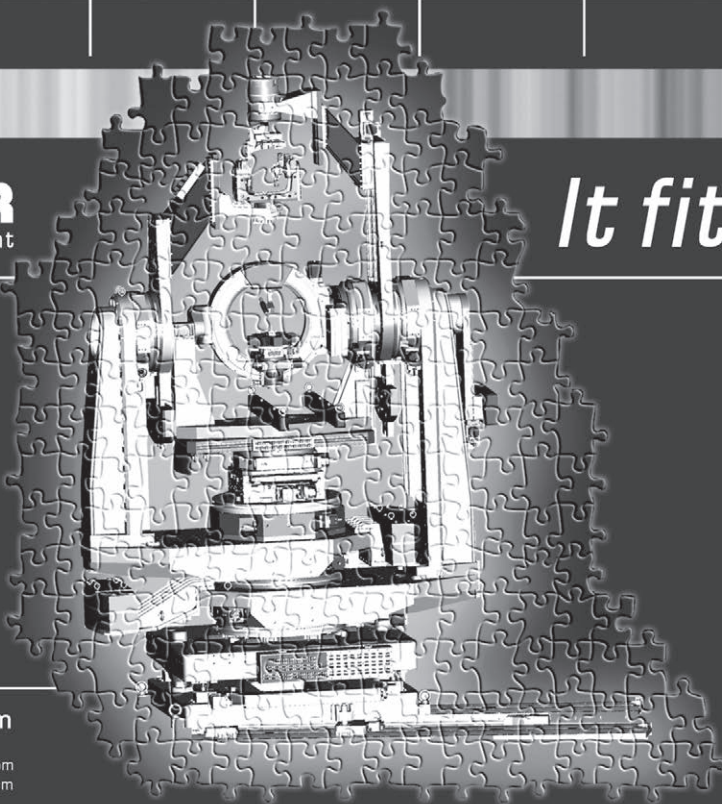
- Mirrors
- Components

FMB Oxford Unit 1 Ferry Mills Osney Mead
Oxford OX2 0ES UK +44 (0)1865 320300

www.fmb-oxford.com Sales@FMB-Oxford.com



It fits



- X-Ray Diffractometers and Cameras
- Multiaxis Goniometers for X-Ray-, Synchrotron- and Neutron Facilities
- Monochromators
- Positioning Devices for Various Environments
- Electronics
- Accessories

We make the Best of your Beam

Sommerstrasse 4
D-83253 Rimsting, Germany

www.xhuber.com
info@xhuber.com

dimensional dataset is as easy and as straightforward as reading a single metadata value, replacing tens of lines of brittle data-importing code with simple invocations along the lines of “dataset = catalog[scan_number].primary.read()” Whether the data are stored locally, remotely, in one or more files, or in one or more different formats, is an internal implementation detail that does not influence the user experience. By keeping data input/output details out of user code, that code becomes easier to reuse, reapply, and maintain.

A practical consequence of this approach is that it was a small amount of work to integrate the Xi-Cam [15] software package to read data from this system. This capability was first developed for Bluesky as a new Python library, dubbed *databroker*. In the spirit of reusing existing tools and building bridges to other scientific communities, databroker is currently in the process of being rewritten to reuse an existing project with very similar design goals, *intake* [16]. This will immediately provide the Bluesky Project with some exciting capabilities that were planned for databroker but not yet implemented.

The Bluesky Project enables creativity at beamlines by integrating easily and robustly with existing software and leaving space for deep customization at every level through well-defined software interfaces. Building on the principles that drove the popularity of scientific Python, it is empowering individual users and groups to work more productively by sharing software components within and between facilities and science domains.

The collaboration is eagerly seeking new collaborators. Visit <https://bluesky.github.io> to learn more and to give Bluesky a try. A browser-based sandbox enables visitors to run example experiments and data analysis in their web browser—no software installation needed. All code and related discussions are in the open at <https://github.com/bluesky>. ■

References

1. <https://github.com>
2. <https://binderhub.readthedocs.io/en/latest/>
3. <https://nsls-ii.github.io>
4. <https://nsls-ii.github.io/ophyd>
5. <https://nsls-ii.github.io/bluesky>
6. <https://nsls-ii.github.io/suitcase>
7. <https://nsls-ii.github.io/databroker>
8. L. J. Koerner et al., *IEEE Transactions on Instrumentation & Measurement* (submitted).
9. <https://certif.com/spec.html>
10. M. Konecke et al., *J. Appl. Cryst.* **48**, 301–305 (2015).
11. S. Van der Walt et al., *PeerJ* **2**, e453 (2014). doi:10.7717/peerj.453
12. F. Pedregosa et al., *J. Machine Learning Res.* **12**, 2825–2830 (2011).
13. <https://soft-matter.github.io/trackpy>
14. <https://www.psychopy.org>
15. R. J. Pandolfi et al., *J. Synchrotron Rad.* **25**, 1261–1270 (2018). doi:10.1107/S1600577518005787
16. <https://intake.readthedocs.io>