ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO2A002

A BEAMLINE AND EXPERIMENT CONTROL SYSTEM FOR SLS 2.0

K. Wakonig*, C. Appel, A. Ashton, S. Augustin, M. Holler, I. Usov, J. Wyzula, X. Yao
Paul Scherrer Institute, Villigen PSI, Switzerland

Abstract

The beamlines of the Swiss Light Source (SLS) predominantly rely on EPICS standards as their control interface but in contrast to many other facilities, there is up to now no standardized user interfacing component to orchestrate, monitor and provide feedback on the data acquisition. As a result, the beamlines have either adapted community solutions or developed their own high-level orchestration system.

For the upgrade project SLS 2.0, a sub-project was initiated to facilitate a unified beamline and experiment control system. During a pilot phase and a first development cycle, libraries of the Bluesky project were used, combined with additional in-house developed services, and embedded in a service-based approach with a message broker and in-memory database. Leveraging the community solutions paired with industry standards, enabled the development of a highly modular system which provides the flexibility needed for a constantly changing scientific environment. One year after the development started, the system was already tested during many weeks of user operation and recently received the official approval by the involved divisions to be rolled out as part of the SLS 2.0 upgrade.

INTRODUCTION

The beamline and experiment control system is defined as the software layer tasked with the orchestration and the monitoring of the data acquisition routines and acts as the primary interface for the beamline scientist and users of a larger research facility. At the Swiss Light Source (SLS), Paul Scherrer Institute, Switzerland, this layer was initially often merged with the underlying control system, e.g., EPICS [1] resulting in EPICS as a control system and orchestration system. However, throughout the operation of the SLS, beamlines frequently developed or adopted high-level tools as dedicated orchestration systems. Yet in contrast to other facilities, the SLS currently does not provide a standardized interface for all beamlines. As a result, a large variety of solutions are currently in operation, inter alia SPEC [2], DA+ [3], FDA [4], PShell [5], GDA [6].

As these solutions were mostly driven and maintained by the beamline scientists themselves, it has led to an environment in which the beamlines were able to quickly adapt their acquisition routines to changes of the overall requirements. However, it also created an environment where sharing devices, feedback loops and data analysis routines between beamlines was frequently hindered by incompatible interfaces of the bespoke orchestration system. Additionally, already challenging topics such as FAIR (Findable, Accessible, Interoperable, and Reusable) data, the integration of high-throughput devices or a more automated control of the beamline were hindered further by the increased complexity of supporting various high-level interfaces.

To remedy these shortcomings, a project was initiated to investigate the feasibility of leveraging the SLS 2.0 upgrade program and unifying these interfaces [7, 8]. Starting with the functional and non-functional requirements given by the beamline scientists, the users, and the support groups, the Beamline Instrument Support Software (BLISS) [9] and the Bluesky framework [10], two well-known experiment control systems were chosen to be investigated more thoroughly. Although these two solutions have a similar software architecture, their actual implementation and core concepts differ vastly. Fundamentally, both solutions rely on a Python-based, monolithic main component for steering and monitoring the acquisition. Events and metadata streams are forwarded to a dedicated message broker system (Redis [11] for BLISS, Kafka [12] for Bluesky). Another similarity is that the file writer on a production system should be listening to the message broker events instead of being attached to the core components directly to avoid performance penalties.

BLISS, marketed as an "all-in-one" solution [13], provides a plethora of features with a tight integration into the BLISS ecosystem. The Bluesky framework on the other hand provides fewer features out of the box but relies on a more modular environment, where components can be easily reused without committing into a fully-fledged ecosystem. BLISS is shipped with support for Tango [14] devices, yet often relies on direct device control interfaces. The Bluesky framework on the other hand uses Ophyd, a hardware abstraction layer on top of existing control systems. Although Ophyd was clearly designed with EPICS in mind and supports many different types of EPICS interfaces out of the box, Ophyd can also be used to integrate Tango devices and even custom communication protocols, bypassing the underlying control system.

As the SLS will keep using EPICS as a primary control system after the SLS 2.0 upgrade, a good support for EPICS devices is crucial for a successful adoption of a high-level interface. Although EPICS is predominantly used and is the only officially supported control system at PSI, various custom devices bypassing the EPICS layer have also been implemented. As a result, a hardware abstraction layer such as Ophyd provides a much-needed layer to standardize the diverse landscape at PSI.

ARCHITECTURE

Following the initial evaluation phase, a prototype project was launched to investigate the potential integration of certain Bluesky components into a service-oriented architecture. The reconfiguration of selected Bluesky components

* klaus.wakonig@psi.ch

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M02A002

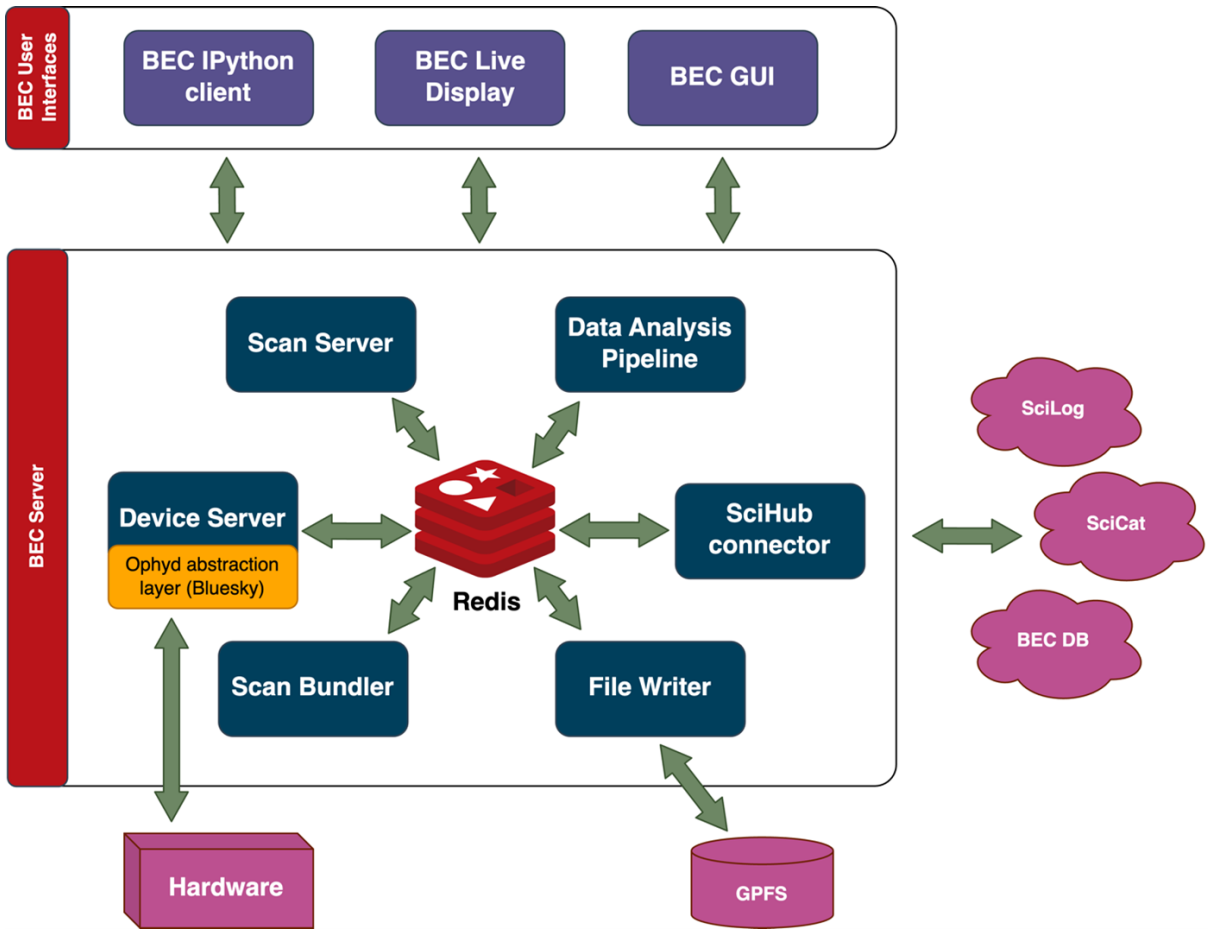

Figure 1: Architecture diagram. In contrast to other existing experiment orchestration systems, BEC relies on a service-oriented backend server with multiple smaller services and Redis as a central message broker and in-memory database. More-over, any client can be considered as another service to the entire system with access to the same message broker instance.

facilitated the creation of a client-server architecture, with a message broker serving as its core component. To streamline access to device data and status information, Redis was introduced as a replacement for Kafka. In this prototype architecture for the Beamline and Experiment Control (BEC), Redis not only assumes the role of a message broker but also functions as an in-memory database.

The adoption of a client-server architecture was deliberate, as it offers a distinct separation between the software stack of user interfaces and core services [8, 15–18]. This separation provides a high degree of flexibility for future modifications. Furthermore, the client-server approach simplifies the development of new interfaces by delegating the detailed orchestration tasks to the BEC server, Fig. 1. Consequently, interface developers can program against a client library without needing to delve into the intricacies of the underlying architecture. This approach not only accelerates development cycles for experienced professionals but also empowers less experienced contributors to create new interfaces tailored to their specific use cases.

In addition to adopting a client-server architecture, the central components have been broken down into a series of smaller, interconnected services that communicate through Redis. This approach provides a dynamic and modular framework that can swiftly adapt to evolving requirements. Its core services are:

- **Scan Server** The Scan Server acts as the primary interface for user requests, tasked with the choreography of the data acquisition. It checks the incoming requests for validity, assembles scan instructions, executes them and if necessary, publishes device instructions to Redis. It comprises dedicated components for the scan guard, the scan assembler, the scan queue and its scan worker. Although by default only the primary queue is set up, separate queues can be added, and their corresponding workers are added and removed automatically. This allows for multiple, independent scan queues if parallel activities are required.
- **Device Server** This service provides a thin layer on top of Bluesky's Ophyd library to support remote procedure calls (RPC) through Redis. It listens to device instructions sent to Redis and performs the specified operations on Ophyd objects. The available Ophyd objects are determined by loading a device configuration, either through the client, e.g., through yaml files, or from the connected database (BEC DB, Fig. 1).

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO2A002

Providing Ophyd objects through a service layer also facilitates sharing access to devices that require a direct socket connection, bypassing the EPICS layer.

- **Scan Bundler** Data streams in a control system are inherently asynchronous. Yet, to simplify the user feedback and data analysis, asynchronous readouts are often synchronized afterwards. The Scan Bundler creates such synchronization barriers based on metadata entries of the individual readouts (e.g., point IDs or time stamps) and broadcasts these synchronized readings as a new data stream to the BEC system.
- **File Writer** Beyond simply writing NeXus-compatible [19] metadata entries to disk, the file writer also adds external links to the NeXus master file to any other large data file, such as detector files. The internal NeXus structure can be adjusted using customizable plugins to comply with a desired NeXus application definition.
- **SciHub connector** A service to connect a BEC instance to external cloud services such as an electronic logbook (SciLog), a data catalogue and archiving solution (SciCat) and the BEC database.
- **Data Analysis Pipeline** While simple data processing routines such as live data fitting using e.g., LMfit [20], can be performed directly on the server, more computationally expensive operations can be controlled e.g., through Slurm [21] jobs. Alternatively, any process with access to Redis can react to live events and trigger analysis pipelines. Results or metadata thereof can be fed back into the BEC and potentially used to dynamically adjust the data acquisition.

PROTOTYPE DEVELOPMENT

The prototype development of the new software started in May 2022, primarily using device simulations and the LamNI [22] endstation. This setup was chosen as it covers many aspects of the desired BEC system and provides a high availability for offline tests without interfering with user operation. Beyond a sheer proof of principle, the aim was to develop the BEC system with rapid feedback loops from the beamline. However, to narrow the scope, the aim was limited to previously integrated devices in EPICS or SPEC and existing acquisition routines. Building a large collection of unit tests and end-to-end tests paired with continuous integration further allowed to create a stable system despite the rapid development phase.

Ptychographic Laminography Experiments

LamNI is a dedicated endstation for ptychographic laminography. Its primary focus lies on measuring high-resolution images of planar samples by means of ptychographic phase retrieval. Frequently, these measurements are combined with diamond quarter wave plates positioned before the setup, facilitating measurements using X-ray magnetic circular dichroism (XMCD) [23]. The primary components are bypassing the EPICS layer and thus direct communications are established to Galil, SmarAct and an in-house developed, Linux-based controller. All components were integrated as devices in Ophyd. The detector, an Eiger 1.5 M, in-vacuum detector (PSI), was integrated using a bootstrap approach: Leveraging Redis's in-memory database, Ophyd devices can set themselves up during "staging" cf. Ref. [24], for the upcoming measurement. This approach was extensively used during SAXS/WAXS imaging experiments described in the next section.

Although the LamNI endstation also supports step scans, the typical data acquisition during user operation of ptychographic laminography beamtimes leverages hardware-controlled acquisitions for improved performance. Within the Bluesky framework and in particular Ophyd, these acquisitions are supported through the additional fly scan interface. Using BEC, the simple "kickoff" functionality of Bluesky's Ophyd library was extended to read out all devices, including the "flyers" during the acquisition. The direct feedback also facilitated the development of graphical user interfaces showing the progress of the scan in real time. During the development and the integration phase, it was noticed that compared to the previous implementation using SPEC, the new system reduces the internal overhead significantly, sometimes leading to devices not yet being ready. These problems were remedied by adding additional status checks to the Ophyd devices.

In addition to the scan server plugins for LamNI-specific scan routines and Ophyd device classes, IPython-based client interfaces were added to simplify the set-up of new acquisitions.

After a first evaluation in August 2022, the decision was made to test the BEC system during user operation at the cSAXS beamline, SLS, in November 2022.

During a week of user operation, the laminographic imaging was paired with XMCD. Moreover, a heating element provided by the users was integrated ad-hoc and used to temperature-cycle magnets. After a successful week of operation, the beamline staff decided not to switch back to SPEC for subsequent LamNI beamtimes. As a result, BEC was used for four and a half weeks of user beamtime with five different user groups in April and May 2023.

SAXS / WAXS Imaging Experiments

To further explore and test BEC using other acquisition techniques, BEC was used for another set of experiments again at the cSAXS beamline in September 2023 right before start of its 18 months long shutdown. For two and a half weeks, five different user groups performed various imaging experiments. For the measurements, a focused X-ray beam of a few tens of microns was used to raster scan samples with extensions in the mm or cm range, while multiple different detectors were used to capture different signals, i.e. small angle X-ray scattering (SAXS), wide angle X-ray scattering (WAXS), X-ray fluorescence (XRF) and transmission signal of the direct beam. Furthermore, BEC was used to acquire SAXS tensor tomography datasets [25] in two of the above mentioned five beamtimes.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M02A002

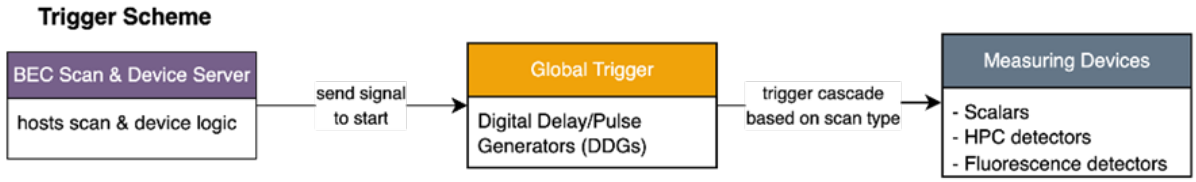

Figure 2: Trigger Scheme. Using a global trigger mechanism facilitates a high level of synchronization whilst minimizing the beamline-specific modifications to the BEC system.

All scans, including time scans, step scans, burst scans and fly scans were implemented using the same hardware configuration with synchronized data acquisition by using a universal trigger. SAXS, XRD and XRF detectors as well as scalar readouts were operated in gating mode where the acquisition is steered by TTL pulses generated by Digital Delay Generators (DDGs) as shown in Fig. 2.

For efficient raster scanning, BEC-controlled fly scans were implemented such that in contrast to the previously described LamNI setup, no dedicated hardware controller was used to steer the fly scan. The current implementation is compatible with any motor that allows control of acceleration time and target speed. This framework was further used to write a 2D grid fly scan. Compared to the previous implementation in SPEC, BEC reduced the internal overhead by around 60 %.

Another advantage in the integration of BEC for the current experiments was that the described trigger scheme of Fig. 2, facilitated to move any beamline specific logic to a device level and follow a bootstrap approach for initiating data acquisition. This approach not only works for any scan type, but it also simplifies the development of new scan routines.

Figure 3 illustrates the bootstrap approach for the data acquisition using the example of a fly scan: BEC starts a scan by first staging all devices, which is accomplished by publishing a message in Redis that is picked up by the device server and the instruction is performed on the devices accordingly. During staging, the devices pull the required information about the requested scan directly from Redis and prepare themselves for the upcoming acquisition. For detectors, this may include setting up the data backend and file write services. Once all devices have been set up successfully, BEC triggers the acquisition based on the type of scan. For the implemented software fly scan, BEC initiates the motion of the flyer motor before sending a second trigger with a delay calculated based on stage velocity to the delay generator that initiates the data acquisition. For the 2D scan, it further initiates the motion to the start of the next line before repeating the same type of motion again. The bottom right of Fig. 3 shows a microscope image of a 3D printed nanocomposite, in addition to the four different contrast modes that were measured simultaneously, SAXS/XRD/XRF/transmission. They clearly show that individual lines nicely follow sample features, even though the stages in use were not encoded.

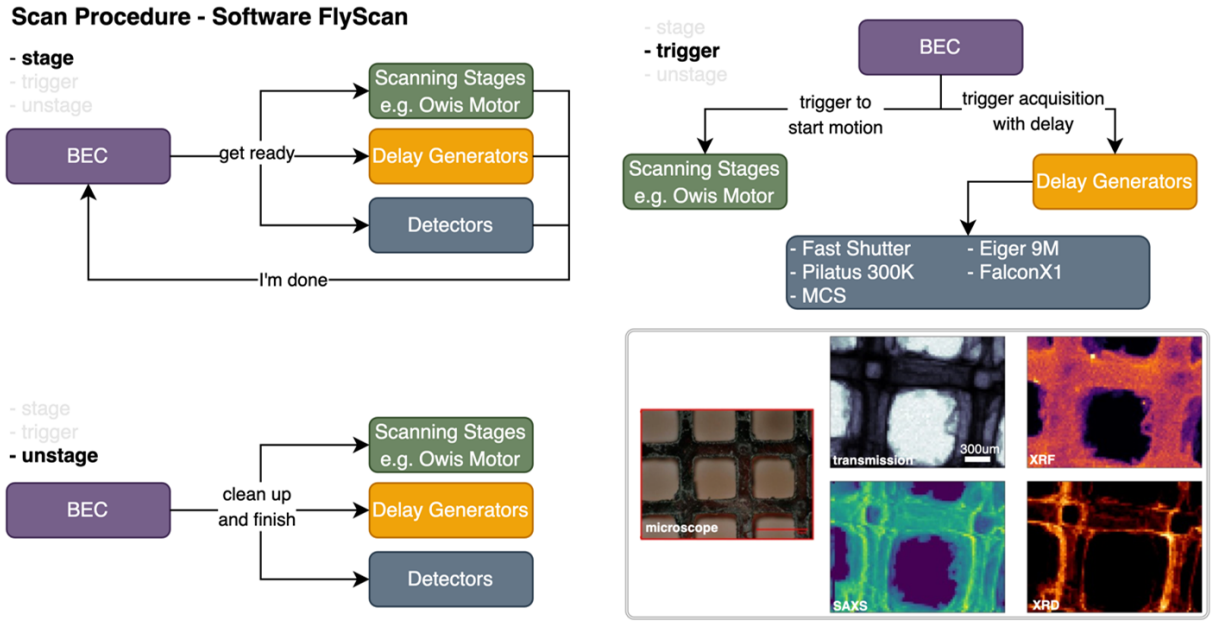

Figure 3: Scan procedure using a bootstrap approach. All devices are prepared for the data acquisition during "stage" instruction. To this end, the required information is pulled independently from Redis and used to initialize the device. The trigger mechanism following a global triggering scheme in which delay generators are used to synchronize the acquisition for all time-critical components. Finally, the during "unstage" all devices check independently their status and reset if necessary.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M02A002

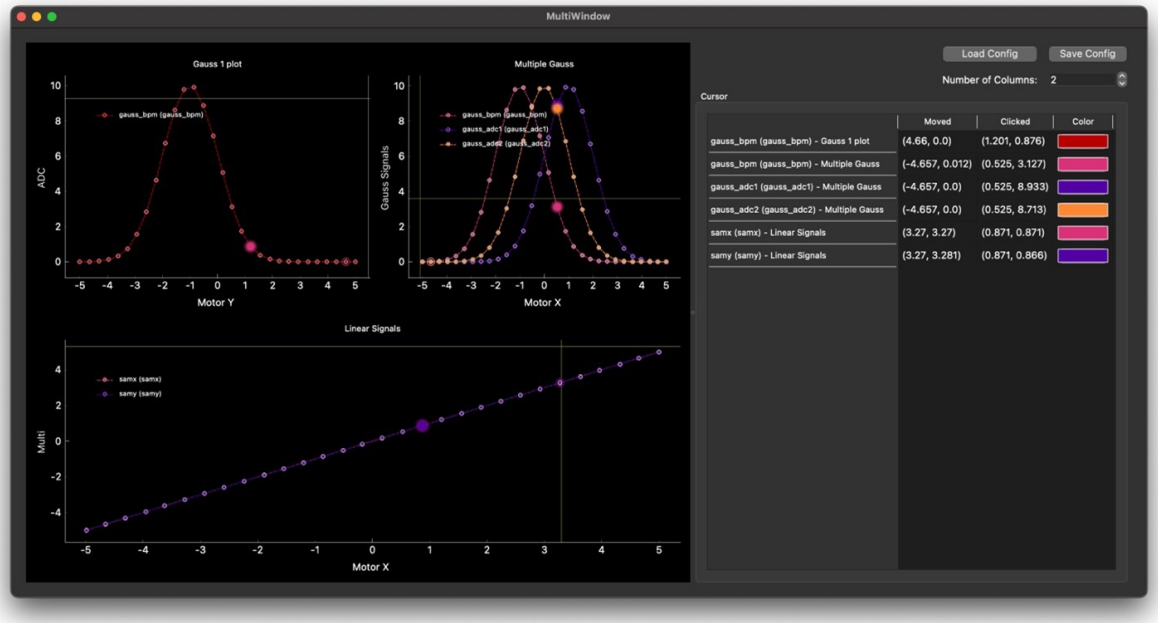

Figure 4: Live plot of wave forms and interpolated data.

At the end of a scan, BEC requests a clean-up procedure for all devices, unstage, which includes a report from all devices whether this procedure was successful. For detectors, this included for instance a check whether detector and data backend has acquired the expected number of points and allowed us to automatically decide whether BEC moves on to the next scan in the queue or initiates an automatic repeat of an unsuccessful acquisition. In principle, this may even host information or logic from fast live feedback systems that decide on their own whether the acquired data is sufficiently good for further data analysis.

INTEROPERABILITY

From the beginning, the BEC system was set up to facilitate the interaction with already existing or envisioned tools at PSI. In the following, two examples are described to demonstrate BEC's interoperability.

Live Feedback

A central component of any experiment orchestration system is the ability to provide live feedback on the ongoing measurement. While the command-line interface can achieve it by means of tables and progress bars, more detailed views are commonly moved to dedicated graphical user interfaces (GUI).

Previously, many control panels at PSI have been developed using Qt [26] and caQtDM [27]. Therefore, the Qt-Designer [26], a graphical, high-level tool to compose new graphical user interfaces by means of arranging widgets has been widely used and is familiar to many beamline scientists. For BEC, we aim to build on top of this existing knowledge and pair it with caQtDM's Python-based alternative PyDM [28]. In addition to providing a datasource plugin for BEC to enable PyDM widgets to communicate with BEC resources, customized interfaces can be developed using

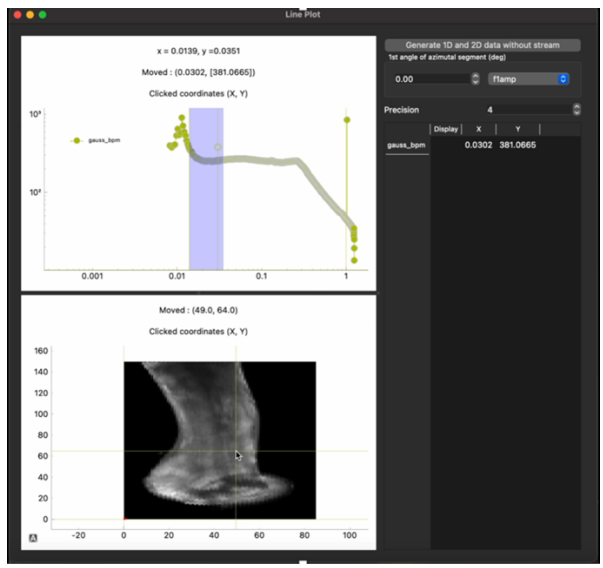

Figure 5: Live feedback for SAXS imaging. The azimuthally integrated data is plotted on the top left. By changing the location and width of the ROI selector, a different q-range can be used for the live preview (bottom left).

the same software stack of PyQt5 [29] and PyQtGraph [30]. These bespoke tools can often be composed from more generalized, modular widgets.

Beyond receiving events from BEC and merely plotting waveforms as shown in Fig. 4, GUI events can be used to emit new events to BEC, triggering new data acquisitions, movement of devices or even modifications to the data processing pipelines by means of using the client library. Figure 5 is an example where the live data processing is steered by the user's interaction with the graphical user interface: While the top left plot updates the incoming processed data stream of azimuthally integrated detector data, its region of interest selector can be used to select a q-range for the

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M02A002

live visualization of the sample as shown in the bottom-left plot. All components shown in Fig. 4 and Fig. 5 are modular plugins that are simply connected to a specific event stream of BEC, enabling customized solutions for beamline whilst minimizing the overall development effort.

SciLog

To facilitate recording the experiment for the users and staff BEC can also be integrate to an electronic logbook. At PSI, SciLog is available to assist users in capturing their intention during their experiments at large research facilities. As such, it provides a simple to use web interface to capture and annotate notes. The frontend is currently based on Angular 14, the backend uses Loopback 4 paired with MongoDB.

Even though logbooks are displayed as discrete entities in SciLog, underneath they are mere database queries that can be further adjusted. Therefore, users can query any of their entries in the database and display them in a new logbook view. To simplify the search, entries can be further annotated with tags.

Within BEC, a connection to SciLog and the logbook of the currently running experiment is established. Using SciLog's Python interface, the BEC can be used to send new messages to SciLog. During automated beamline checks, the SciLog interface is used to automatically document the beam loss and annotate scan repetitions, Fig. 6.

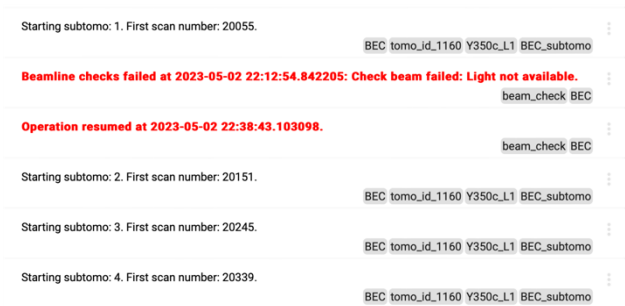

Figure 6: Logbook excerpt. Logbook entries can be further annotated using tags to improve the search results or filter the current logbook to only a subsection of entries. By integrating SciLog into BEC, beam checks can automatically report to SciLog and precisely track the time e.g., a beam loss occurred, simplifying the data analysis later on.

Furthermore, users can use the SciLog interface to modify their scripts and automatically send relevant data to the logbook.

OUTLOOK

With the SLS 2.0 upgrade the expected increase in coherent fraction and a general boost in photon flux will not only unlock new avenues in synchrotron research, but also introduce new challenges for data acquisition and data processing. As a result, a flexible and yet scalable solution is needed to cope with the rapidly evolving requirements. The BEC solution presented here was designed from the beginning with these aspects in mind and was already successfully stress-tested during many weeks of user operation. The client-server system paired with a service-oriented server provides a highly flexible framework that can be easily extended if needed.

Graphical user interfaces are currently under development iteratively with SLS beamlines. This will complement the already existing command-line interface for beamline scientists and users alike who would benefit from a higher level of abstraction and application specificity.

We expect that the development of web interfaces for monitoring and potentially at some point also controlling certain acquisition parameters will complement the desktop-based graphical user interfaces. Furthermore, a database for BEC events, currently under development, will enable beamline scientists to extract relevant diagnostics data by running simple queries, facilitated by standardized and well-annotated data provided by BEC.

The BEC system has recently obtained official approval from the relevant divisions for deployment at SLS 2.0. Consequently, we anticipate that a significant number of the beamlines, particularly those resuming operation immediately after the shutdown (referred to as phase 0/1 beamlines), will be utilizing BEC.

ACKNOWLEDGEMENTS

The authors would like to thank Oliver Bunk for initializing the project and his great support throughout the project. Moreover, we would like to thank our controls specialists Alvin Acerbo, Tine Celcer, Istvan Mohacsi, Xiaoqiang Wang and Elke Zimoch for their constructive feedback and Jonathan Stalberg for his great contribution to the live feedback interfaces during the first user tests. We would like to thank all SLS beamlines and the SLS 2.0 staff for their contribution to the project.

REFERENCES

[1] M. Clausen and L. Dalesio, "EPICS - Experimental Physics and Industrial Control System", *Beam Dyn. Newsl.*, vol. 47, pp. 56–66, 2008.

[2] SPEC, https://www.certif.com/

[3] J. A. Wojdyla *et al.*, "DA+ data acquisition and analysis software at the Swiss Light Source macromolecular crystallography beamlines", *J. Synchrotron Radiat.*, vol. 25, no. 1, pp. 293–303, Jan. 2018. doi:10.1107/S1600577517014503

[4] PSI in-house developed, "FDA".

[5] A. Gobbo and S. G. Ebner, "PShell: from SLS beamlines to the SwissFEL control room", in Proc. ICALEPCS'17, Barcelona, Spain, Oct. 2017, pp. 979-983. doi:10.18429/JACoW-ICALEPCS2017-TUSH102

[6] N. Rees, "The diamond beamline controls and data acquisitEqion software architecture", *AIP Conf. Proc.*, vol. 1234, pp. 736–739, 2010. doi:10.1063/1.3463315

[7] A. Streun *et al.*, "SLS-2 – the upgrade of the Swiss Light Source", *J. Synchrotron Radiat.*, vol. 25, pp. 631-641, 2018. doi:10.1107/S1600577518002722

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO2A002

[8] A. Ashton *et al.*, "Conceptual design report on controls and science IT for the SLS 2.0 upgrade project", Paul Scherrer Institut, Villigen PSI, Switzerland, Report No.: 21-03, 2021. https://www.dora.lib4ri.ch/psi/islandora/object/psi:39514

[9] M. Guijarro *et al.*, "BLISS - Experiments Control for ESRF EBS Beamlines", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 1060-1066. doi:10.18429/JACoW-ICALEPCS2017-WEBPL05

[10] D. Allan, T. Caswell, S. Campbell, and M. Rakitin, "Bluesky's Ahead: A Multi-Facility Collaboration for an *a la Carte* Software Project for Data Acquisition and Management", *Synchrotron Radiat. News*, vol. 32, no. 3, pp. 19–22, 2019. doi:10.1080/08940886.2019.1608121

[11] Redis, https://redis.io

[12] J. Kreps, N. Narkhede, and J. Rao, "Kafka: a Distributed Messaging System for Log Processing", in *Proc. NetDB*, vol. 11, no. 2011, pp. 1-7, 2011. https://notes.stephenholiday.com/Kafka.pdf

[13] About BLISS - Bliss documentation, https://bliss.gitlab-pages.esrf.fr/bliss/master/

[14] Tango Controls, https://www.tango-controls.org

[15] K. Kasemir and X. Chen, "CSS Scan System", in *Proc. ICALEPCS'13*, San Francisco, USA, Oct. 2013, pp. 1461-1464. https://jacow.org/ICALEPCS2013/papers/FRCOAAB01.pdf

[16] K. Kasemir and M. Pearson, "CS-Studio Scan System Parallelization", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 517-520. doi:10.18429/JACoW-ICALEPCS2015-TUA3O04

[17] X. Yao *et al.*, "UX focused development work during recent ORNL EPICS-based instrument control system upgrade projects", in *Proc. ICALEPCS'19*, New York, USA, Oct. 2019, pp. 818-823. doi:10.18429/JACoW-ICALEPCS2019-TUCPR05

[18] X. Yao *et al.*, "A Unified User-Friendly Instrument Control and Data Acquisition System for the ORNL SANS Instrument Suite", *Appl. Sci*, vol. 11, no. 3, Jan. 2021. doi:10.3390/app11031216

[19] M. Könnecke *et al.*, "The NeXus data format", *J. Appl. Crystallogr.*, vol. 48, no. 1, 2015. doi:10.1107/S1600576714027575

[20] M. Newville, A. Ingargiola, T. Stensitzki, and D. B. Allen, "LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python", Zenodo, 2014. https://zenodo.org/records/11813

[21] SLURM, https://slurm.schedmd.com

[22] M. Holler *et al.*, "LamNI - An instrument for X-ray scanning microscopy in laminography geometry", *J. Synchrotron Radiat.*, vol. 27, p. 730-736, 2020. doi:10.1107/S1600577520003586

[23] C. Donnelly *et al.*, "High-resolution hard x-ray magnetic imaging with dichroic ptychography", *Phys. Rev. B*, vol. 94, p. 064421, 2016. doi:10.1103/PhysRevB.94.064421

[24] Bluesky Project, https://blueskyproject.io/

[25] M. Liebi *et al.*, "Nanostructure surveys of macroscopic specimens by small-angle scattering tensor tomography", *Nature*, vol. 527, no. 7578, pp. 349–352, Nov. 2015. doi:10.1038/nature16056

[26] Qt, https://www.qt.io

[27] caQtDM, https://caqtdm.github.io

[28] PyDM, https://slaclab.github.io/pydm/

[29] D. Hess and M. Summerfield, "PyQt Whitepaper", Riverbank Computing Ltd., vol. 9, no. 4, 2013.

[30] PyQtGraph, https://www.pyqtgraph.org