ASSONANT: A BEAMLINE-AGNOSTIC EVENT PROCESSING ENGINE FOR DATA COLLECTION AND STANDARDIZATION

Paulo Baraldi Mausbach*, Eduardo X. Miqueles, and Allan Pinto
Brazilian Synchrotron Light Laboratory (LNLS),
Brazilian Center for Research in Energy and Materials (CNPEM),
Zip Code 13083-100, Campinas, Sao Paulo, Brazil.

Abstract

Synchrotron radiation facilities comprise beamlines designed to perform a wide range of X-ray experimental techniques which require complex instruments to monitor thermodynamic variables, sample-related variables, among others. Thus, synchrotron beamlines can produce heterogeneous sets of data and metadata, hereafter referred to as data, which impose several challenges to standardizing them. For open science and FAIR principles, such standardization is paramount for research reproducibility, besides accelerating the development of scalable and reusable data-driven solutions. To address this issue, the Assonant was devised to collect and standardize the data produced at beamlines of Sirius, the Brazilian fourth-generation synchrotron light source. This solution enables a NeXus-compliant technique-centric data standard at Sirius transparently for beamline teams by removing the burden of standardization tasks from them and providing a unified standardization solution for several techniques at Sirius. The Assonant implements a software interface to abstract data format-related specificities and to send the produced data to an event-driven infrastructure composed of streaming processing and microservices, able to transform the data flow according to NeXus*. This paper presents the development process of Assonant, the strategy adopted to standardize beamlines with different operating stages, and challenges faced during the standardization process for macromolecular crystallography and imaging data at Sirius.

INTRODUCTION

For decades, data standardization has been one of the main concerns of industry and research institutes toward providing a better solution to find, transport, and interchange information across different computational platforms. Looking at the main progresses in computer science and engineering, retrospectively, we notice that data standardization brings significant progresses in disciplines such as telecommunication networks, processing, database technology, and web technologies. In telecommunication, data standardization plays a crucial role in communication protocols, which are fundamental for exchanging data between computing devices [1]. Data standardization also contributed to advances in database technology, which provides tools for storing, retrieving, updating, and deleting data from a collection of objects. For instance, the DICOM standard has been established to facilitate data exchange between medical database systems in digital format [2]. Data standardization was also paramount for the emergence of large X-ray-related datasets [3, 4]. Thus, standardized data formats and structures are critical for efficient and effective data management. Data standardization ensures that data stored within a database follows a consistent format and structure. This consistency simplifies data management, reduces errors, and makes it easier to enforce data integrity constraints [5]. Finally, data management systems often interact with various applications, and standardized data format enables seamless integration and data exchange among systems.

Nowadays, with the increasing use of machine learning and data science pipelines in the industry, data standardization is still crucial to enable novel processing technologies by helping the automation of data analysis pipelines and business needs. A common practice of data scientists who works on data lakes repositories, i.e., the pull of structure and unstructured data that came from multiple sources with different structure and format, consists of extracting, transforming, and loading (ETL) such data towards combining them and prepare them in a more suitable data format to facilitate further analysis. While ETL is an effective data integration method, the main drawback of this approach is the generation of specialized scripts for extracting meaningful data for further analysis, which is time-consuming due to the learning curve needed to understand the data and algorithms for efficient mining data. Furthermore, it is challenging monitoring the quality of data in a data lake repository, which is an essential aspect of the Findability, Accessibility, Interoperability, and Reusability (FAIR) principles [6] , since the main goal of FAIR is to optimize the reuse of data.

In synchrotron radiation facilities, data management is challenging mainly due to the heterogeneity of data sources, which are devices that compound the beamlines which in turn are designed to perform a specific X-ray experimental technique (e.g., X-ray powder diffraction, nanotomography). Besides, there are some proprietary devices in the beamlines that do not allow the modification of generated data structure, which also imposes several challenges for managing the data. Sirius is a fourth-generation synchrotron source of radiation which are still under development [7]. The Sirius beamlines were designed towards offering to the users a wide range of X-ray experimental techniques by providing a modern scientific instrument. Nowadays, Sirius is in its first phase of operation with 14 experimental stations, of which six are already fully operational and work with different experiment

* paulo.mausbach@lnls.br

control systems including proprietary software, five are in scientific commissioning also working with different experiment control systems, and two are under construction. As a fourth-generation synchrotron source radiation, Sirius can expand the capacities of X-ray experimental techniques (e.g., X-ray micro- and nanotomography) in terms of data acquisition speed, accuracy, and resolution. A stronger beam of photons may allow researchers to acquire information from a sample fast enough to study its structure over time, making *in situ*, *in vivo*, and *in operando* analyses feasible. This plays a huge impact on data management and processing as the software infrastructures, workflows, and algorithms need to deal with huge datasets efficiently, and in a minimum time as possible. Finally, researchers may perform an investigation in multiple beamlines, which also brings up the need for consistent data standardization across beamlines and techniques to make further data analysis easier [8].

Assonant is a beamline-agnostic event processing engine for data collection and organization, designed to work with different experiment control software. The Assonants offers a Python package that can easily be incorporated in beamlines' experimental workflow toward making the data standardization and collection abstract to the beamline groups. Alongside, the Assonant takes advantage of a publish-subscribe messaging service that will allow us to trigger services such as data ingestion (persistence) in a high-performance storage system, generation of views of data to be used in further processing applications, sending data to the cataloging system, among others possibilities.

RELATED WORK

The BeamLine Instrumentation Support Software (BLISS) is a control system developed by the ESRF team [9, 10] in the scope of the ESRF Extremely Brilliant Source upgrade program (ESRF-EBS). The BLISS project started in 2015 and, nowadays, is fully operating in several ESRF beamlines. The BLISS supports X-ray experimental techniques by controlling devices such as motors, detectors, and other acquisition devices. In summary, the BLISS is composed of five main components: (i) the Beacon, a configuration server that relies on the Redis database [1], dedicated to centralizing the configuration of the beamline and acts as a message broker that implements shared state and distributed lock; (ii) communication helper which implements several protocols to make the access to equipment easier; (iii) hardware control; (iv) scanning engine; and (v) data management layer. Focusing on the last feature, the BLISS provides a data management layer that allows users to adapt the data acquisition process to match the data policy rules that take place. After the scanning, the BLISS stores the data temporarily in a Redis pub/sub channel, called channel data, which makes the data available for the applications registered in that channel, besides archiving the collected data in Nexus HDF5 format [11]. Although BLISS provides an entire solution for data acquisition and management, the strong couple between experiments control and data management components does not allow the adoption of data management components in beamlines that already have an experiments control system in place.

[1] `https://redis.io`

Bluesky is also a well-known experiment control system that comprises a hardware control alongside components that allow users to perform online visualization and analysis, besides providing ways of accessing and exporting the data being acquired [12]. Bluesky relies on a data representation schema named as document that implements a simple and formalized data model based on key-value mapping such as JSON documents and key-value databases as Berkeley DB and Redis. This feature makes managing data collected from experiment control easier, since its structure is known and standardized regardless its origin device. Bluesky also provides a framework for data collection which allows the user to live and streaming data, which is convenient for online processing and data management pipelines, acquire a rich data associated to experiments and supports plugable I/O modules, which allow export the data into a desired format, database or message broker. According to the Bluesky' authors, the data acquisition is the main objective of the Bluesky. For this reason, Bluesky captures useful data associated to an experiment into a fine-grain time scale, by online streaming information to a data broker, which is responsible to maintain a time ordering of the collected documents. The documents are created by a run engine, specified by an execution plan. A common execution plan will generate a *"start document"*, which contains data known at the beginning of the experiment such as start time, data associated to the user, an unique ID that identifies the experiments, among others. Once the experiment is started, the execution plan begins generating event documents, which contains the actual measurements and their timestamps. Finally, at the end of experiment, the execution plan generates a *"stop document"*, which contains data as the end time and exit status, i.e., if the run is successfully completed, failed or if it was aborted. At the end, all the documents collected are gathered into an unified dataset.

Finally, Mukai *et al.* [13], from The European Spallation Source (ESS), also proposed a software architecture for data aggregation and streaming for neutron data using open source solutions such as EPICS [14], Apache Kafka [2], NeXus, and Google Flatbuffer [3]. Taking into account that neutron instruments performs neutron data acquisition in event mode, the authors proposed the use of a the Kafka cluster, a messaging system for data streaming, that receives event data which come from the Event Formation Unit (EFU) software. The EFU software is connected to the detector readout interface to acquire event data alongside with a pixel identifier-timestamp pair from the detector electronics. Besides the EFU data, the Kafka cluster also receives EPICS data via Kafka Forwarder application, which subscribes

[2] `https://kafka.apache.org`
[3] `https://flatbuffers.dev`

EPICs devices to send data from multiple Input/output Controllers (IOCs) to the Kafka cluster. Thus, the proposed data aggregation and data streaming comprises a distributed producers and consumers that exchange data via Apache Kafka using the Google Flatbuffer. Finally, the authors used NeXus to persist the data on storage. Although the authors used a well-known solution for data serialization, the Google Flatbuffer, a clear description about the adopted data model in their solution could not be found model on their article.

We observed that current state-of-the-art solutions for data acquisition and data management relies on a well-defined data model and data schema to manipulate the data throughout subsystems. Both the Bliss and Bluesky solutions, adopted a simple and yet effective data model based on key-values mapping which is very convenient to perform data serialization easier. Moreover, all solutions adopted the use of messaging system to manage data streaming. It is important to notice that Bliss use Redis database in different flavors (e.g., publication of acquired data and management of distributed lock and state sharing) and the ActiveMQ [4] to handle raw data. In turn, the Bluesky uses the Apache Kafka as a messaging system to gather a sequence of documents which contains data associated to the experiment, while the raw data is persisted directly on disk. Finally, all solutions supports NeXus data format as an end solution for data format to be delivered to the user. The main drawback found in these solutions that prevent its adoption as a unified solution for data collection and management at the Sirius is the strong coupling between experiment control and data management software layers. Since a decoupled solution for data management that attend beamlines with different experiment controls is needed, the proposed solution presented in this article offers to the beamlines a data management solution integrable with the current experiment control of the beamlines from Sirius.

METHODOLOGY

This section presents in details the proposed method for data collection and standardization at the Sirius, the Assonant – A Beamline-agnostic event processing engine for data collection and organization. Assonant is a flexible python package that can be easily integrated into the experiment control systems operating at the Sirius, without changing the beamlines' experimental workflow. The experiment control systems current operating in Sirius' beamlines are: MxCube [15, 16], SPEC (Certified Scientific Software, Cambridge, MA, USA), Bluesky [12], and the Py4Syn [17], an in-house python-based solution for data acquisition, device manipulation and scan routines.

The Assonant was developed to be agnostic to the beamlines as the data produced by them are being standardized according to their experimental technique(s) (e.g., tomography, crystallography, x-ray powder diffraction, XAS), taking into account the NeXus standardization and definitions for x-ray-based experiments [11]. NeXus is a common data format for neutron, x-ray, and muon science, developed as an international standard by scientists and programmers that represents major scientific facilities around the world.

[4] `https://activemq.apache.org`

Finally, Assonant is referred as an engine since the goal is to empower Sirius' beamlines with an easy-to-use tool to promote a comprehensive data management and processing infrastructure as transparent as possible to users. Thus, the Assonant focuses on providing to the beamline groups, software developers and researchers easy-to-use tools that facilities the use of Sirius' data-related infrastructure and data-driven services, and let burdensome and complex data tasks to be done behind the scenes.

Design Premises

The design of Assonant is guided by some premises which are presented as follow:

Data Standardization The Assonant tackle the problem of data standardization by defining common standards for synchrotron experimental techniques in order to guarantee a holistic solution for the Sirius. The Assonant abstracts the data standardization problem of the beamline researchers by providing an easy-to-use tool for beamline teams to send the data to a data management system and at same time it provides a comprehensive data collection and modeling tool for a software group specialized in data science and management, which defines standards for the data. The implementation of the solution should be done in the most abstract way possible from beamlines teams so they can focus in their crucial tasks in beamline operation without needing to bother with data standardization. Finally, the standard applied should be centralized in the experimental technique-related to the data and not with the beamline that generated it, as it enables a better way to manage data among beamlines. We call this a technique-centric approach instead of the currently beamline-centric one.

Beamline Agnostic Assonant should be usable by any beamline group at the Sirius. For this reason, the Assonant must be designed to be easily integrated on any experiment control currently in operation at the Sirius, which a minimum change as possible on the beamline scripts toward avoiding major refactors in their experimental workflows. In cases of the integration being unfeasible, for instance in proprietary software, the Assonant should be usable as an Extract, Transform, and Load (ETL) solution.

Data Model A data model must be used to create a standard data representation among beamlines to facilitate how data is accessed and manipulated. The data model must be able to handle the heterogeneous nature of data existent in synchrotron facilities. In this sense, NeXus seems to be a natural choice to be used as it defines a data format suitable for X-ray experimental data and can be extended for other types of experiment.

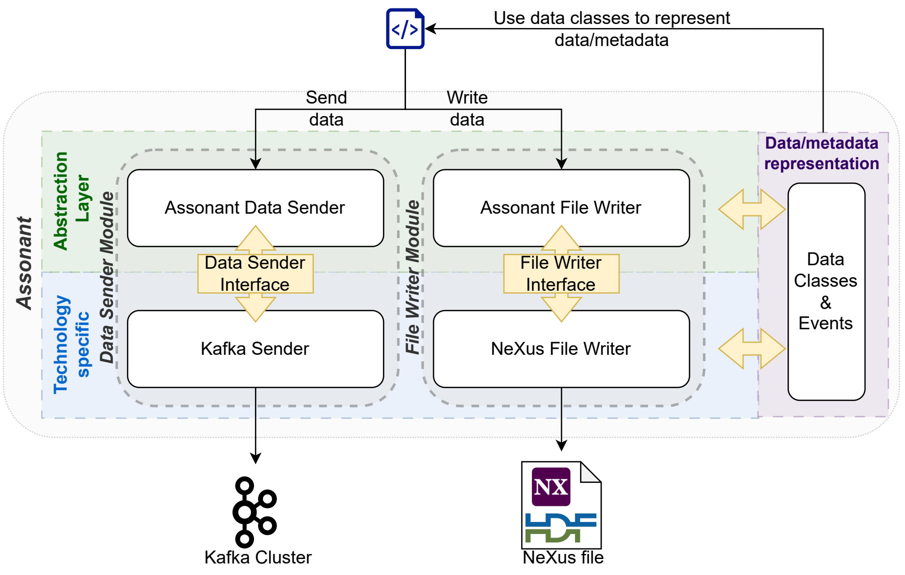


Figure 1: Assonant modular architecture overview with currently implemented modules. Further development may implement new modules and technologies.

Development The development process of the software must follow good practices of software engineering toward having a software highly extensible and maintainable. Thus, the Assonant package was organized in different modules which handle specific tasks. This modules should be generic enough to make possible the implementation changes of specific technology's variations (e.g., a module to write files must allow supporting different file formats).

Architecture Overview

Figure 1 illustrates the main abstraction layers of the Assonant, which are explained in detail in next sections. It is important to notice that Assonant is not restricted only to that modules as it is open to be extended with new modules and technologies.

Data Representation Layer The Data representation layer is responsible to define data schemes to allow representing data in a common way to access and exchange data inside Assonant engine. The Assonant uses the Pydantic [18] which is python library built on top of Python data classes, feature introduced in Python 3.7. The Pydantic offers a convenient way (i) to model data taking into account their types and default values, (ii) to validate simple and complex data structures by defining constraints and rules to specify fields requirements (e.g., optional fields, minimum and maximum length), and (iii) to parse data into python object which allows to easily serialize the data to JSON schema and other formats, and send them to another application using a message broker system, for instance. Next, we discuss the *Data Class* module in detail.

Data Classes. The Data Model used in Assonant is a hierarchical class-based data model that used class composition to represent its information. Internally, entities are represented as classes that gather information related to their scope, defined by the class name. Furthermore, they can be used to compose higher level classes that represent more generic or abstract scopes. For example, a Detector class is responsible for storing any type of information acquired from some kind of detector and a Sample class handles all information related to a sample being used at the experiment and they together can be inserted into a Experiment class to represent the data acquired during an experiment. That said, the Data Classes module implements Python classes powered by the Pydantic package to allow the usage of this Data Model into all Assonant modules creating a standard way to manipulate and exchange data between them.

Data Classes module implements a class called AssonantDataClass which all classes inside the module inherits from it. Its main goal is to allow defining common configurations and properties among implemented Data Classes. Above it, there are 3 abstract classes called Instrument, NonInstrument and Data Handlers that inherits from it and are used to logically divide data classes. Here follows some detailed explanation about each type of data class:

Instrument. Group of data classes that represent any kind of device existent in the synchrotron facility which data is collected, for example: Detector, Monochtromator, Mirror, (...).

NonInstrument. Group of data classes that represent scopes to gather non-instrumental data in it, for example, Sample data or User data. It also contain a group of classes called Entries, which are used to wrap up data classes into logical scopes that represent specific collection moments, e.g. data acquired during a callibration step done before sample exposure to synchrotron light.

Assonant Abstraction Layer The Assonant abstraction layer is responsible for exposing an easy-to-use interface to allow the usage of available services without bothering with lots of specific technology details. This way users can have access to complex services related to their data by adding simple calls to their codes. The development of this software layer was inspired in some design patterns intended to build decoupled pieces of software such as the *facades pattern* and *abstract factory pattern* and plays an important role in the overall architecture of Assonant. As this layer abstracts the low-level and complex implementation details of services and functionalities, the Assonant provides a simple interface to users and allow triggering subsystems toward execute complex data workflows. Furthermore, changes of current workflows occurs transparently to users. Next, it is discussed about two modules that implements services related to data.

File Writer. This module was developed with the intention of providing an easy way to write data contained in data classes into files. Its implementation on Assonant Abstraction layer exposes a simple interface for users to call underlying functionalities from the Technology specific layer that deal with specific file formats. Currently, the only available format is NeXus as it has been chosen as the standard data format that will be used in Sirius. Nevertheless, the design pattern applied in its development, the factory pattern, enables flexibility to easily extend it to support new data formats if needed.

Data Sender. The implementation principles of this module is similar to the File Writer, following the same used design pattern, flexibility, extensibility and easy-to-use interface. The biggest difference between them is that *Data Sender* module purpose it to allow exchanging data represented in Assonant Data Classes over some kind of data transfer interface, for example, REST APIs, message queues, data brokers. At the moment, the only supported exchange interface is by packing data into messages, serializing it and sending through a Kafka Cluster. Assonant Data Classes being developed using Pydantic-enhanced data classes come in hand here, as it intrinsically implements a Json serialization method for data classes, which is used internally at Technology Specific Layer from Data Sender module.

Technology Specific Layer The Technology specific layer, as the name suggests, deals with implementation details of specific technologies being used. For example, connection configuration and serialization needed to use a message queue platform or maybe converting information to another representation to use a specific package to persist data in a specific format.

EXPERIMENTS AND RESULTS

The performed experiment aimed to test the viability of integrating Assonant at beamlines and its capability of representing beamline data in a new and centralized format defined for the Sirius, instead of the one created specifically for each the beamline. The experiments were performed in MOGNO and MANACA beamlines and are presented in the following subsections:

Experiment in the MOGNO Beamline

”MOGNO beamline is a high-energy imaging beamline dedicated to *in situ* and *in operando* experiments in heterogeneous and hierarchical samples” [19]. It is currently under scientific commissioning and it has been designed to execute a variety of tomography techniques, in micro and nano resolution and also under different image regimes.

MOGNO showed to be a candidate for testing Assonant integration in a beamline as its experiment control system was an in-house development done by the beamline staff in Python which it is possible to add Assonant modules in it. The beamline being at commissioning stage allowed easier interaction with MOGNO developers and enabled the execution of tests using the beamline infrastructure.

The test consisted on executing a modified version of the tomography experiment control script using Assonant modules in it. The changes done in the experiment flux were minimal and consisted basically in wrapping up acquired metadata as Assonant Data Classes, changing the code line that used to write file into a HDF5 file to use the File Writer module and signalize the end of the experiment using the Data Sender module. Due to the size of data and throughput requirements of the beamline, the data is directly written into the centralized storage by the beamline detector instead of sending this information to the control system, returning only the path where the file was saved. Regarding that, data was represented in the final file as paths to acquired frames and a solution for standardizing this is currently being designed. Figure 2 illustrates an overview of the experimental setup and Fig. 3 shows the changes required in the control script to integrate Assonant in it.

Experiment at MANACÁ Beamline

MANACÁ was the first X-ray beamline to start operating for users at Sirius and is dedicated to performing Macromolecular Micro and Nano Crystallography experiments [20]. Among all the beamlines operating or planned for Sirius so far, it is one of its kind as it's the only one to execute this type of experiment. Due to that, integrating Assonant in it was not only an integration experiment but also the opportunity to standardize all data generated at Sirius for that type of technical experiment.

Initially, the plan was to execute at MANACÁ a similar experiment to the one done at MOGNO beamline but by integrating Assonant inside MANACÁ's MXcube [15, 16] instance. However, due to development priorities related to the beamline operation, the integration test was postponed. With that, MANACÁ became a possible case for testing situations where it isn't possible to integrate Assonant directly into the experiment control flux but use it as a tool for ETL scripts to deal with generated data. Figure 4 illustrates the applied experimental setup at MANACÁ beamline.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-WE3BC006

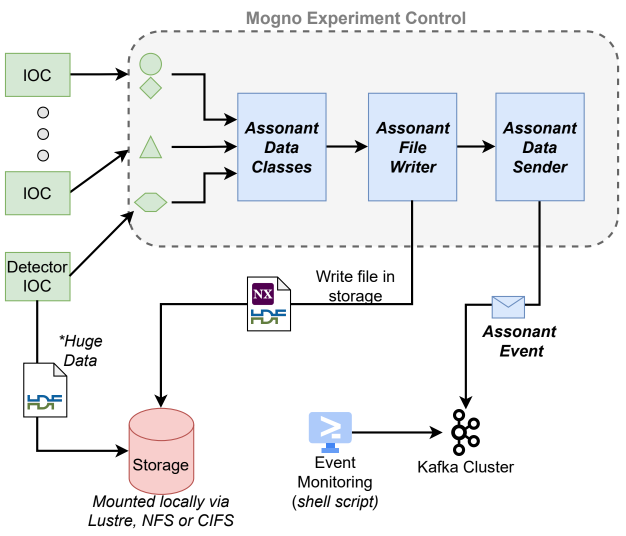


Figure 2: Experimental Setup applied on MOGNO. Data was collected using the experimental control and Assonant Modules were add to it in order to represent data as Assonant Data Classes, write it in storage as a NeXus file using Assonant File Writer and signalized that experiment finished using Assonant Data Sender.

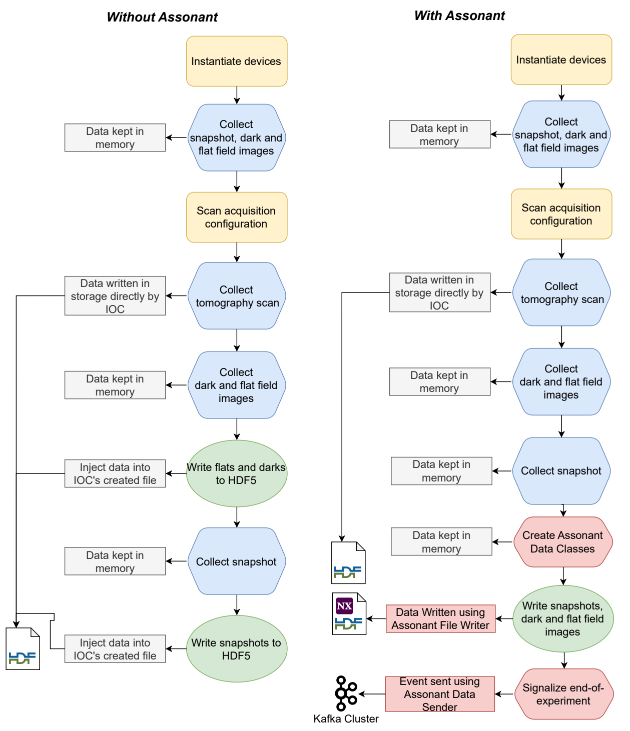

Without Assonant
With Assonant
Instantiate devices
Collect snapshot, dark and flat field images
Data kept in memory
Scan acquisition configuration
Collect tomography scan
Data written in storage directly by IOC
Collect dark and flat field images
Write flats and darks to HDF5
Inject data into IOC's created file
Collect snapshot
Write snapshots to HDF5
Create Assonant Data Classes
Write snapshots, dark and flat field images
Data Written using Assonant File Writer
Signalize end-of-experiment
Event sent using Assonant Data Sender
Kafka Cluster

Figure 3: Experiment control's workflow executed at MOGNO beamline to perform a tomography experiment. In the left it is shown how it was done before integrating Assonant and in the right after.

Results and Discussion

Figure 5a shows data generate before and after integrating Assonant at MOGNO experiment control script and shows

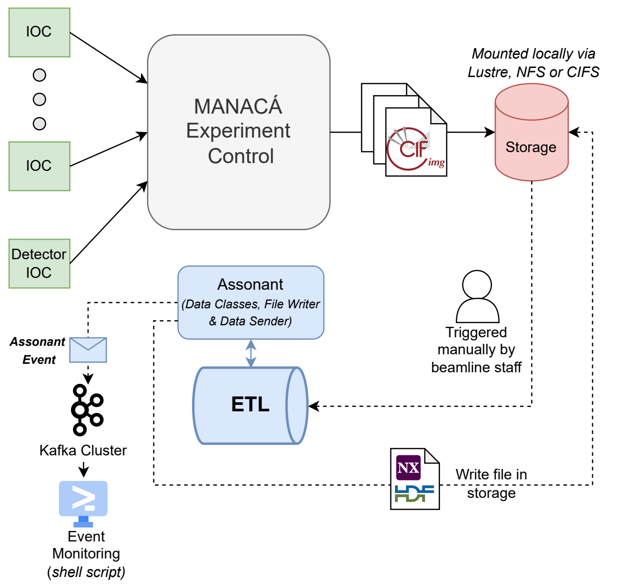
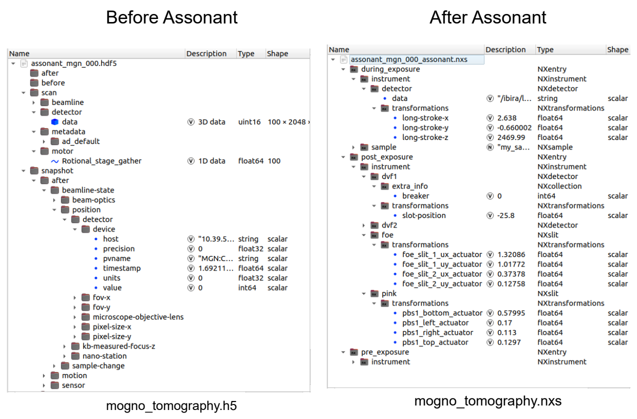

IOC
IOC
Detector IOC
MANACÁ Experiment Control
Mounted locally via Lustre, NFS or CIFS
Storage
Assonant (Data Classes, File Writer & Data Sender)
ETL
Triggered manually by beamline staff
Assonant Event
Kafka Cluster
Write file in storage
Event Monitoring (shell script)

Figure 4: Experiment setup applied on MANACÁ. After an experiment is executed, the beamline staff is responsible to manually trigger the execution of an ETL script which uses Assonant modules to convert data stored in CBF files into a single NeXus file and signalize that conversion was done by sending an Assonant Event to the Kafka cluster.

that before Assonant integration at MOGNO, its data was stored in their own data model representation and after it, it was transformed into NeXus data model. That is positive as it reveals that Assonant is capable of enabling a beamline to generate data into NeXus format with minimal changes to their experiment workflow when integrated directly into the beamline experiment control script. We use the Silx View [21] to open and visualize files in NeXus and HDF5 formats.

Additional to that, as illustrated in Figure 5b, the experiment performed at MANACÁ also showed that in situations where it is impossible to integrate Assonant into the experiment control, its modules can still be used to facilitate the development of ETL script to convert generated data into NeXus format. Before using the ETL with Assonant, MANACÁ data was stored into many CBF files, where each file contained data and metadata related to frames taken by rotating the sample 0.1 degree, after it, it became a single NeXus file with everything stored inside it.

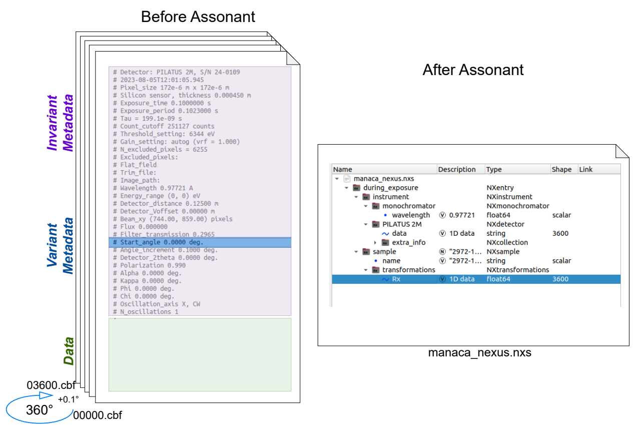
Additionally, both experiments showed Assonant capabilities to handle data standardization at totally different beamlines in terms of design, experimental technique and control systems generating in the end a standard format to store both tomography and macromolecular crystallography. It is important to highlight that current NeXus file is compliant to a set of NeXus data model definition but still require improvements that will be implemented along with its integration in beamlines.



```
Name                               Description   Type     Shape
assonant_mgn_000.hdf5
  after
  before
  scan
    beamline
    detector
      data                         3D data       uint16   100 × 2048 ›
    metadata
      ad_default
    motor
      Rotional_stage_gather        1D data       float64  100
  snapshot
    after
      beamline-state
        beam-optics
        position
          detector
            device
              host                 "10.39.5…     string   scalar
              precision            0             float32  scalar
              pvname               "MGN:C…       string   scalar
              timestamp            1.69211…      float64  scalar
              units                0             float32  scalar
              value                0             int64    scalar
            fov-x
            fov-y
            microscope-objective-lens
            pixel-size-x
            pixel-size-y
          kb-measured-focus-z
          nano-station
        sample-change
      motion
      sensor
```

```
Name                               Description   Type                Shape
assonant_mgn_000_assonant.nxs
  during_exposure                                NXentry
    instrument                                   NXinstrument
      detector                                   NXdetector
        data                       "/ibira/l…    string              scalar
        transformations                          NXtransformations
          long-stroke-x            2.638         float64             scalar
          long-stroke-y            -0.660002     float64             scalar
          long-stroke-z            2469.99       float64             scalar
    sample                         "my_sa…       NXsample
  post_exposure                                  NXentry
    instrument                                   NXinstrument
      dvf1                                       NXdetector
        extra_info                               NXcollection
          breaker                  0             int64               scalar
        transformations                          NXtransformations
          slot-position            -25.8         float64             scalar
      dvf2                                       NXdetector
      foe                                        NXslit
        transformations                          NXtransformations
          foe_slit_1_ux_actuator   1.32086       float64             scalar
          foe_slit_1_uy_actuator   1.01772       float64             scalar
          foe_slit_2_ux_actuator   0.37378       float64             scalar
          foe_slit_2_uy_actuator   0.12758       float64             scalar
      pink                                       NXslit
        transformations                          NXtransformations
          pbs1_bottom_actuator     0.57995       float64             scalar
          pbs1_left_actuator       0.17          float64             scalar
          pbs1_right_actuator      0.113         float64             scalar
          pbs1_top_actuator        0.1297        float64             scalar
  pre_exposure                                   NXentry
    instrument                                   NXinstrument
```

(a) MOGNO beamline.



```
# Detector: PILATUS 2M, S/N 24-0109
# 2023-08-05T12:01:05.945
# Pixel_size 172e-6 m x 172e-6 m
# Silicon sensor, thickness 0.000450 m
# Exposure_time 0.1000000 s
# Exposure_period 0.1023000 s
# Tau = 199.1e-09 s
# Count_cutoff 251127 counts
# Threshold_setting: 6344 eV
# Gain_setting: autog (vrf = 1.000)
# N_excluded_pixels = 6255
# Excluded_pixels:
# Flat_field:
# Trim_file:
# Image_path:
# Wavelength 0.97721 A
# Energy_range (0, 0) eV
# Detector_distance 0.12500 m
# Detector_Voffset 0.00000 m
# Beam_xy (744.00, 859.00) pixels
# Flux 0.000000
# Filter_transmission 0.2965
# Start_angle 0.0000 deg.
# Angle_increment 0.1000 deg.
# Detector_2theta 0.0000 deg.
# Polarization 0.990
# Alpha 0.0000 deg.
# Kappa 0.0000 deg.
# Phi 0.0000 deg.
# Chi 0.0000 deg.
# Oscillation_axis X, CW
# N_oscillations 1
```

```
Name                       Description   Type                Shape    Link
manaca_nexus.nxs
  during_exposure                        NXentry
    instrument                           NXinstrument
      monochromator                      NXmonochromator
        wavelength         0.97721       float64             scalar
      PILATUS 2M                         NXdetector
        data               1D data       string              3600
        extra_info                       NXcollection
    sample                 "2972-1…      NXsample
      name                 "2972-1…      string              scalar
      transformations                    NXtransformations
        Rx                 1D data       float64             3600
```

(b) MANACÁ beamline.

Figure 5: Results of beamlines' data before and after the integration of Assonant in the experiment control script.

CONCLUSIONS AND FUTURE WORK

This article presents the Assonant, a solution for data collection and standardization at the Sirius. The Assonant was designed to be a beamline-agnostic solution and a easy-to-use tool for beamline researchers for collecting, modeling, and sending metadata produced during experiments through data-related infrastructure and data-drive services, without enter in specificity of the underlying infrastructure or complexity of tasks to be performed with data. As a result, the article introduces the Assonant data classes that allows to implement pragmatically the data model proposed by the Assonant, which is inspired on the NeXus data format. NeXus was chosen as our standard data format since it provides a set of principles, definitions, and tools to standardize data produced for a wide range of X-ray experimental techniques and can also be extended for other types. The design of the software architecture took inspiration of some design patterns as factory and facades patterns to bring to the system flexibility to adapt to technology choices available for manipulating data, scalability to make the implementation of new functionalities easier, and maintainability to decrease efforts of maintenance activities of the software. The experimental results shown in this article for a tomography and a crystallography beamline, show the robustness of the proposed solution to deal with data acquired with different experiment control systems, and with a minimal changes in the current experimental workflows of beamlines.

Directions for future works include: the integration of the Assonant in other tomography beamlines; the extensions of the Assonant to support other experimental technique such as X-ray powder diffraction (XRD), X-ray fluorescence (XRF), Small-angle X-ray scattering (SAXS), Angle-resolved Photo Emission Spectroscopy (ARPES); the implementation of the NeXus application definition to produce data views of the collected metadata to be consumed by specific users and applications; and development of data-drive services to consume the data produced by the beamlines such as data catalog systems such as the ICAT system.

ACKNOWLEDGEMENT

We would like to thank the Brazilian Ministry of Science, Technology, and Innovation (MCTI) for funding this work through the Brazilian Center for Research in Energy and Materials (CNPEM). We also thank (a) Lucca Bavia Cuenca Campoi from MOGNO Beamline; (b) Andrey F. Z. Nascimento and Evandro A. de Araujo from MANACÁ Beamline; (c) members from the Scientific Computing Group (GCC); and (d) members from Control Software Group (SwC). All of them provided valuable suggestions during the development of the Assonant.

REFERENCES

[1] Zigbee specification (zigbee document 053474r06, version 1.0), `https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf`

[2] C. Parisot, "The dicom standard", *Int. J. Card. Imaging*, vol. 11, no. 3, pp. 171–177, 1995. `doi:10.1007/BF01143137`

[3] S. R. Hall, F. H. Allen, and I. D. Brown, "The crystallographic information file (CIF): a new standard archive file for crystallography", *Acta Crystallogr. A*, vol. 47, no. 6, pp. 655–685, 1991. `doi:10.1107/S010876739101067X`

[4] F. R. N. C. Maia, "The coherent x-ray imaging data bank.", *Nat. Methods*, vol. 9, no. 9, pp. 854–855, 2012. `doi:10.1038/nmeth.2110`

[5] D. International, *DAMA-DMBOK: Data Management Body of Knowledge (2nd Edition)*. Technics Publications, LLC, 2017.

[6] M. D. Wilkinson *et al.*, "The fair guiding principles for scientific data management and stewardship", *Sci. Data*, vol. 3, no. 1, p. 160 018, 2016. `doi:10.1038/sdata.2016.18`

[7] L. Liu, R. T. Neuenschwander, and A. R. D. Rodrigues, "Synchrotron radiation sources in brazil", *Philos. Trans., Math. Phys. Eng. Sci.*, vol. 377, no. 2147, p. 20 180 235, 2019. `doi:10.1098/rsta.2018.0235`

[8] D. Hesterberg *et al.*, “Micro- to nano-scale soil and rhizosphere processes analyzed using multiple beamlines at the Sirius synchrotron”, in *Fundam. Res. Vivo Stud.*, 2023. https://eprints.soton.ac.uk/477658/

[9] M. Guijarro *et al.*, “BLISS - Experiments Control for ESRF EBS Beamlines”, in *Proc. ICALEPCS’17*, Barcelona, Spain, 2018, pp. 1060–1066. doi:10.18429/JACoW-ICALEPCS2017-WEBPL05

[10] V. Michel *et al.*, “BLISS - Experiments Control for ESRF Beamline”, in *Proc. PCaPAC’18*, Hsinchu, Taiwan, 2019, pp. 26–29. doi:10.18429/JACoW-PCaPAC2018-WEP02

[11] M. Könnecke *et al.*, “The NeXus data format”, *J. Appl. Crystallogr.*, vol. 48, no. 1, pp. 301–305, 2015. doi:10.1107/S1600576714027575

[12] S.C. Daniel Allan Thomas Caswell 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

[13] A. Mukai *et al.*, “Architecture of the data aggregation and streaming system for the european spallation source neutron instrument suite”, *J. Instrum.*, vol. 13, no. 10, p. T10001, 2018. doi:10.1088/1748-0221/13/10/T10001

[14] L. R. Dalesio *et al.*, “The experimental physics and industrial control system architecture: Past, present, and future”, *Nucl. Instrum. Methods Phys. Res. A*, vol. 352, no. 1-2, pp. 179–184, 1994. doi:10.1016/0168-9002(94)91493-1

[15] J. Gabadinho *et al.*, “Mxcube: A synchrotron beamline control environment customized for macromolecular crystallography experiments.”, *J. Synchrotron Radiat.*, vol. 17, no. 5, pp. 700–7, 2010. doi:10.1107/S0909049510020005

[16] M. Oskarsson *et al.*, “MXCuBE3 Bringing MX Experiments to the WEB”, in *Proc. ICALEPCS’17*, Barcelona, Spain, 2018, pp. 180–185. doi:10.18429/JACoW-ICALEPCS2017-TUBPL05

[17] H. H. Slepicka, H. F. Canova, D. B. Beniz, and J. R. Piton, “*Py4Syn*: Python for synchrotrons”, *J. Synchrotron Radiat.*, vol. 22, no. 5, pp. 1182–1189, 2015. doi:10.1107/S1600577515013715

[18] S. Colvin *et al.*, *Pydantic/pydantic: V2.4.2 2023-09-27*, version v2.4.2, 2023. doi:10.5281/zenodo.8384169

[19] N. L. Archilha *et al.*, “Mogno, the nano and microtomography beamline at sirius, the brazilian synchrotron light source”, *J. Phys.: Conf. Ser.*, vol. 2380, no. 1, p. 012 123, 2022. doi:10.1088/1742-6596/2380/1/012123

[20] A. Nascimento *et al.*, “Launch of the manacá beamline at sirius: First protein crystallography structures and new opportunities for pharmaceutical development using synchrotrons”, *Synchrotron Radiat. News*, vol. 34, no. 5, pp. 3–10, 2021. doi:10.1080/08940886.2021.1994310

[21] T. Vincent *et al.*, *Silx-kit/silx: 1.1.2: 2022/12/16*, version v1.1.2, 2022. doi:10.5281/zenodo.7446362