ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THSDSC06

DEVELOPING A DIGITAL TWIN FOR BESSY II SYNCHROTRON LIGHT SOURCE BASED ON EPICS AND MICROSERVICE DESIGN

W. Sulaiman Khail*, P. Schnizer, M. Ries
Helmholtz-Zentrum Berlin, BESSY, Berlin, Germany

Abstract

Digital twins, i.e. theory and design tools connected to the real devices and machine by mapping of physics components to the technical correspondents, are powerful tools providing accelerators with commissioning predictions and feedback capabilities. This paper describes a new tool allowing for greater flexibility in configuring the modelling part combined with ease of adding new features. To enable the various components developed in EPICS, Python, C, and C++ to work together seamlessly, we adopt a microservice architecture, with REST API services providing the interfaces between the components. End user scripts are implemented as REST API services, allowing for better data analysis and visualization. Finally, the paper describes the integration of dash and ploty for enhanced data comparison and visualization. Overall, this workflow provides a powerful and flexible solution for managing and optimizing BESSY II digital twins, with the potential for further customization and extension to upcoming machines.

INTRODUCTION

In recent years, the concept of digital twins has gained significant traction as virtual representations of physical systems, enabling real-time monitoring, analysis, and optimization of intricate processes [1]. This technology has found widespread application across diverse sectors, such as aerospace, smart cities, healthcare, automotive, and energy, due to its ability to enhance efficiency, reduce costs, and improve safety [1]. Within the realm of scientific research, particularly in the domain of Synchrotron Light Sources, there is a burgeoning interest in harnessing digital twins to model and monitor the intricate behavior of these complex systems.

Synchrotron Light Sources, as described by the European Synchrotron Radiation Facility (ESRF), are expansive research facilities generating intense light beams for applications in material science, biology, and drug discovery [2]. These facilities operate through intricate setups involving linear accelerators, booster rings, and storage rings, necessitating seamless coordination of thousands of components to produce high-quality light beams. Given their complexity, digital twins emerge as invaluable tools, enabling real-time modeling and monitoring. By leveraging technologies such as the Experimental Physics and Industrial Control System (EPICS) and microservice design, Synchrotron Light Sources can achieve enhanced flexibility, scalability, and reliability in their operations [3, 4].

* waheedullah.sulaiman_khail@helmholtz-berlin.de

This paper presents the development of digital twins for two longstanding Synchrotron Light Sources, namely BessyII and MLS (Metrology Light Source). Utilizing EPICS as the standard interface and employing MongoDB in JSON format for machine structure storage, the implementation integrates REST APIs and Python scripts to facilitate seamless communication between system components. The integration of dash and plotly further enhances data visualization and comparison, offering a robust solution for real-time monitoring and optimization of Synchrotron Light Sources (Ref 5).

In summary, this paper delves into the integration of digital twin technology within the realm of Synchrotron Light Sources, showcasing its potential for revolutionizing the monitoring and optimization processes in these complex scientific facilities. Through the utilization of established frameworks and innovative approaches, the study not only presents a comprehensive solution for current facilities but also lays the groundwork for the customization and expansion of digital twin applications in upcoming machines.

TOWARDS AN ARCHITECTURE

Domain-driven design is based on the principle that knowledge obtained needs to be cast into business logic and surrounding components. However, this approach may be questionable if the domain of interest is subject to change.

At least one of the authors considers this approach invalid for the calculation of beam dynamics due to the field being impacted by progress in other sciences or paradigm shifts within the domain itself:

- Machines developed currently need to be controlled by complex procedures for first commissioning and periodic recommissioning (4th generation light sources).
- Machine learning requires fast evaluation of the model.
- Implementing propagation code on GPU codes makes calculations feasible for difficult tasks such as the interaction of particles within bunches or in different bunches (referred to as “collective effects”), and optimization of sextupole magnet settings to reduce beam loss (referred to as “dynamic aperture”).

Typical implementations today provide a full package dedicated to calculating beam propagation. This needs to be revisited for appropriate layering to be ready for these requirements, such as obtaining a fully differentiable model. Furthermore, the calculations need to be performed not only for values but also for objects representing dependencies.

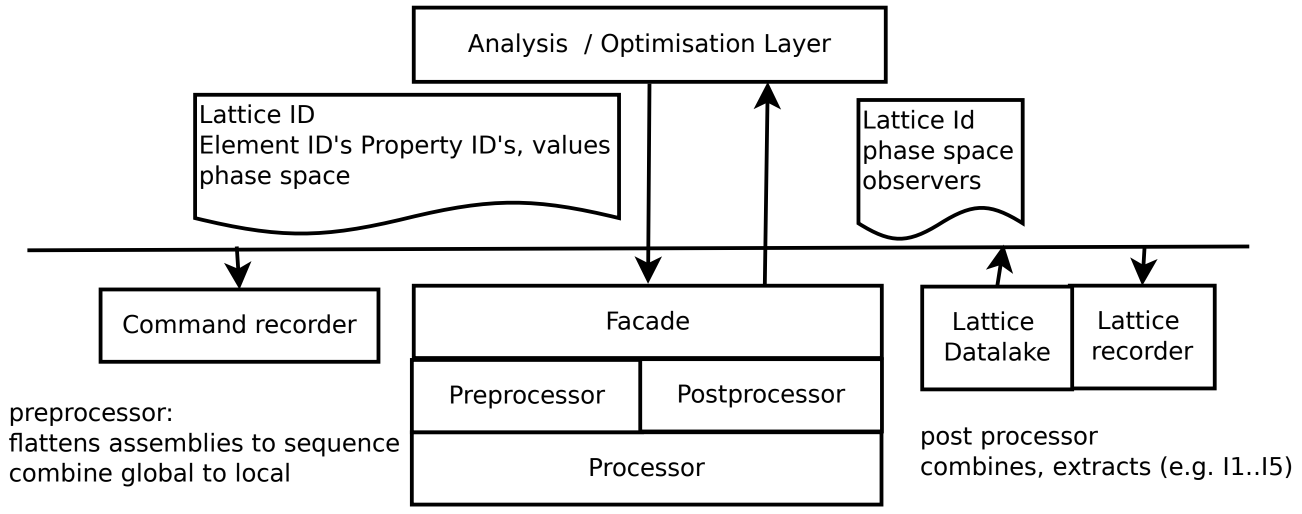


Figure 1: Architecture pattern.

Abstracting Calculation

The calculation and analysis capabilities provided by many particle tracking codes can be split into at least two layers (see Fig. 1):

- Beam dynamics propagation layer or "linear particle accelerator propagation processor",
- Analysis and control layers.

The communication between these layers can then be narrowed down to:

- Updates of the lattice: the property to change would be identified by a tuple of: (lattice id, device id, property id). The calculation engine would be tasked to create a new lattice and return the new id.
- Propagation of phase space: a phase space and "calculation configuration" would be handed to the lower layer. This would then respond with a new phase space reflecting the state of the beam particle after propagating through the lattice.

In this manner, one can develop the analysis and control script layer independently of the calculation engine.

The communication exchange needs to be addressed as well. The upper layer would control the configuration of the device. This setup is split up as shown in Fig. 2:

- Information required to place the device within the machine (e.g., Δx or Δy).
- Information on properties of the device (e.g., K value).

This configuration is then handled by the lower layer:

- Preprocessor would inspect the information and simplify, e.g., transformations for further processing. Sanity checks could be also implemented here.
- The processor itself executes the calculation: it will be detailed below.
- The postprocessor collects the information gathered by the observers and makes it available in a simpler manner.

The "processor" could be called a "propagator processor" or even simpler, a "kick processor". It would implement

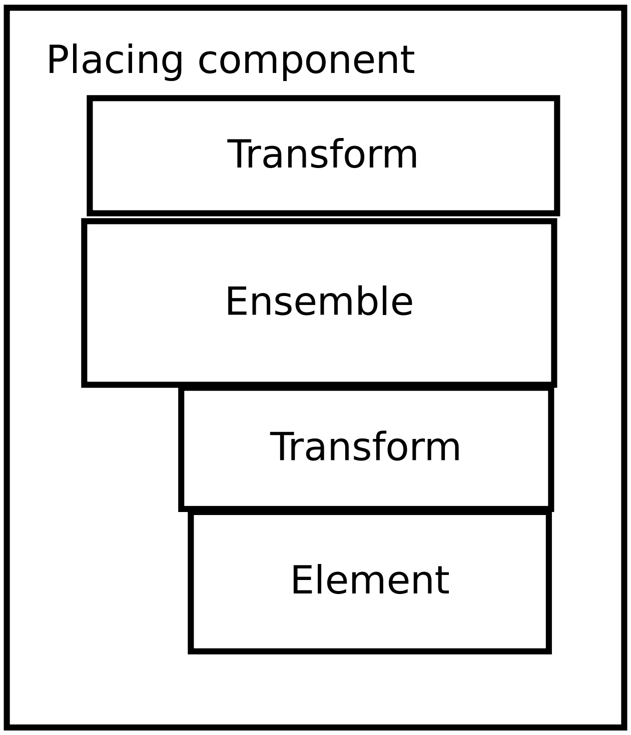


Figure 2: Handling device configuration.

different propagators (similar to instructions on a CPU). This is motivated as follows:

- For single-particle dynamics, different kicks are applied to the start phase space.
- For many independent particle dynamics, the kicks work for an ensemble of particles independently.
- Multiple particle dynamics typically treats the particles like independent particles for many kicks. At dedicated steps, the particle phase space is inspected. The mutual effect of the particles on their phase space is then handled at this step.

Common tasks of the analysis layer include:

- Finding the fixed point (e.g., for ring machines).
- Deducing properties of the overall machine (e.g., Twiss parameters).
- Studying when particles are lost.

Similar studies are run at this level. One can summarize all these tasks to a common study of:

- Making changes to some of the machine element properties and
- Seeing which effects it has on the propagated phase space.

It is important to note that these machine studies do not only use phase space of machine representable types but also of other types, e.g., truncated power series or symbolic computation. These are either used to:

- Represent the individual variables of the phase space or
- Represent some of the kicker parameters as knobs.

CPUs have dedicated Arithmetic Logic Units (ALUs) for handling integers or floating points. Here, the propagators would be expected to handle different types of phase space. A possible implementation could be:

- Using dynamically typed languages. Then the appropriate functions would be called automatically for the different types.
- Using statically typed languages with polymorphism: for example, if using C++, the elements would be templates of the knob. The propagators would then be implemented in a templated fashion to be prepared for different variable types used by the phase space and different knob types used by the element description.

Abstracting Machine Interface

Changing configuration parameters for the machine is typically nearly instantaneous or similar to that. For real machine settings, magnets can take in the order of seconds, but this step can typically be done in parallel. Therefore, the update of the lattice needs to be inspected to determine if:

- Implementing the update asynchronously and using futures to provide the caller a handle is appropriate, or
- Combining it as a vector of update commands resulting in a new lattice is better.

Further Extensions

Given the split interaction between these two layers, several additional applications are supported:

- **Digital Twinning:** A study could be made on a digital model. The updates made to the model can then be inspected, as these are available in the command recorder. These can then be reviewed or replayed on the real machine.
  Given a digital shadow of the machine that can predict the lower or upper confidence bound of the next command, one could also have an additional safety barrier added when the commands are replayed to the real machine
- **Fault Injection:** the commands could be passed through dedicated dispatchers: then the user can instrument the dispatcher to e.g. abandon or delay or modify a command: then the device would not respond in an appropriate manner. This could be used to see if the higher level application is able to handle such faults in an appropriate manner.

DIGITAL TWINS AND SYNCHROTRON FACILITIES

In this section, we detail the creation of the digital twin for BessyII and MLS, utilizing the EPICS control system and embracing a microservice architecture. The digital twin functions as a real-time model of the synchrotron light sources, enhancing the efficiency of monitoring and control for operators and researchers. Our implementation leverages standard EPICS as the interface, ensuring reliability within scientific facilities [3].

To facilitate flexible and accessible storage of the machine structure, we opted for MongoDB as our database, storing the structure in JSON format. This choice allows effortless updates and offers a more adaptable alternative to conventional lattice files. The machine structure is modeled using the open-source toolkit Bluesky [5] and Ophyd, providing a framework for defining and executing experiments. This approach enables versatile commissioning tasks, including the addition of online analysis during measurements.

Building on the Bluesky software stack [5], MongoDB storage, the thor scsi engine wrapped within pydevice [6], and a set of packages split up per functionality, our approach provides a comprehensive solution for managing the machine structure. The integration of MongoDB ensures efficient and flexible storage, allowing seamless updates and adaptability. Additionally, the use of Bluesky and Ophyd facilitates the execution of experiments, enabling diverse commissioning tasks and real-time analysis capabilities [5, 6].

For seamless communication among components, we employed REST API services as the interface. This method ensures smooth interaction between components developed in diverse languages and frameworks. Each microservice possesses a specific functionality, communicating with others through well-defined APIs. End user scripts are developed as REST API services, simplifying access to the digital twin.

A notable advantage of utilizing REST API services is the integration of advanced data analysis and visualization tools. We integrated the open-source libraries Dash and Plotly, enhancing data comparison and visualization. Dash, a Python framework, facilitates web application creation, ensuring accessibility from any device with an internet connection. Plotly, a versatile data visualization library, provides customizable tools like scatter plots and line charts, seamlessly integrated with Dash applications (Dash; Plotly; Ref 2).

The diagram in Fig. 3 illustrates the intricate data flow within the BessyII digital twin architecture. It showcases the interconnection of components, including BessyII, EPICS,

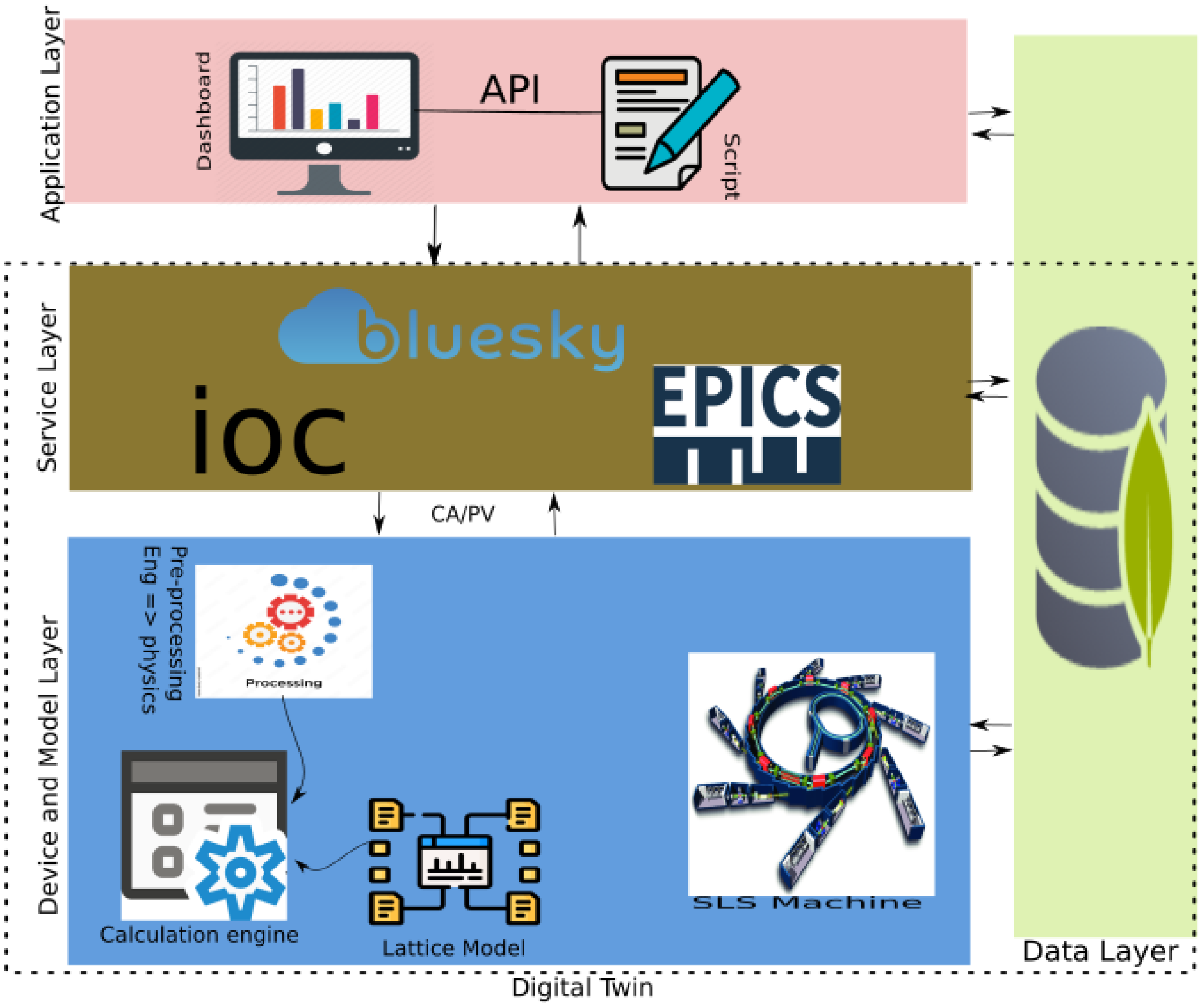


Figure 3: Top level overview of the digital twin architecture.

calculation engine, MongoDB, commissioning scripts, and data visualization tools. This visual representation highlights the communication pathways, emphasizing the smooth exchange of data between these elements, ensuring the digital twin's efficiency. EPICS acts as the interface for interacting with the synchrotron light sources, with data stored in MongoDB in JSON format. Commissioning scripts utilize REST API services for advanced data analysis and visualization, contributing to comprehensive monitoring and optimization (Fig. 3; Ref 3).

In summary, the digital twin implementation for BessyII and MLS offers a flexible, modular, and scalable architecture, accommodating the integration of new components seamlessly. Leveraging standard EPICS, microservice design, and REST API services ensures a robust and reliable framework for scientific facility control systems. This approach enhances the monitoring and control capabilities of synchrotron light sources, paving the way for more effective research and experimentation (Ref 4).

BEAM BASED ALIGNMENT (BBA) MEASUREMENT: A USE CASE

Within the realm of our digital twin, grounded in EPICS control system and Bluesky and ophyd frameworks, the Beam Based Alignment (BBA) measurement stands out as a testament to our system's robust capabilities. This pivotal process is seamlessly integrated into our digital infrastructure as a REST API, offering a user-friendly interface for executing BBA commissioning scripts remotely. Researchers and operators can conveniently access these scripts via REST API calls or dedicated REST clients, facilitating a highly flexible and efficient workflow.

What sets our approach apart is the incorporation of a sophisticated data model into our measurements. From the BBA measurement to its analysis and subsequent storage of results, the entire process is meticulously modeled based on object-oriented concepts. This strategic decision provides a holistic understanding of the measurement process, making

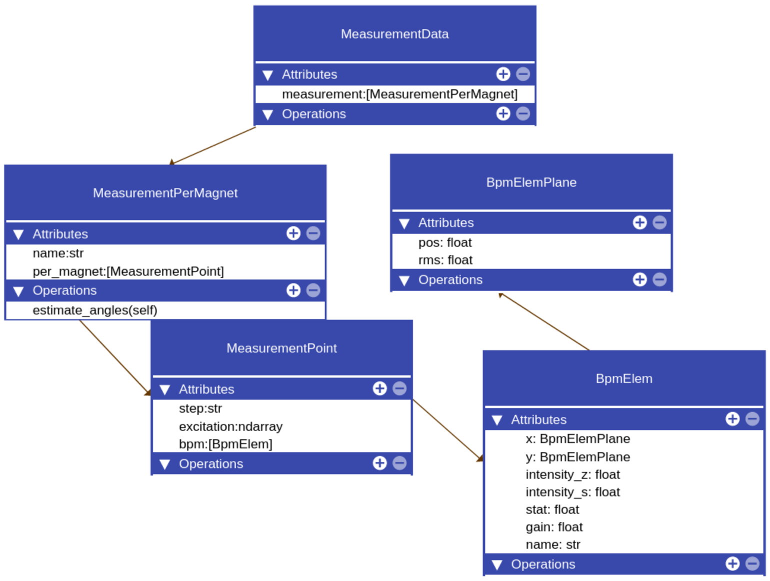
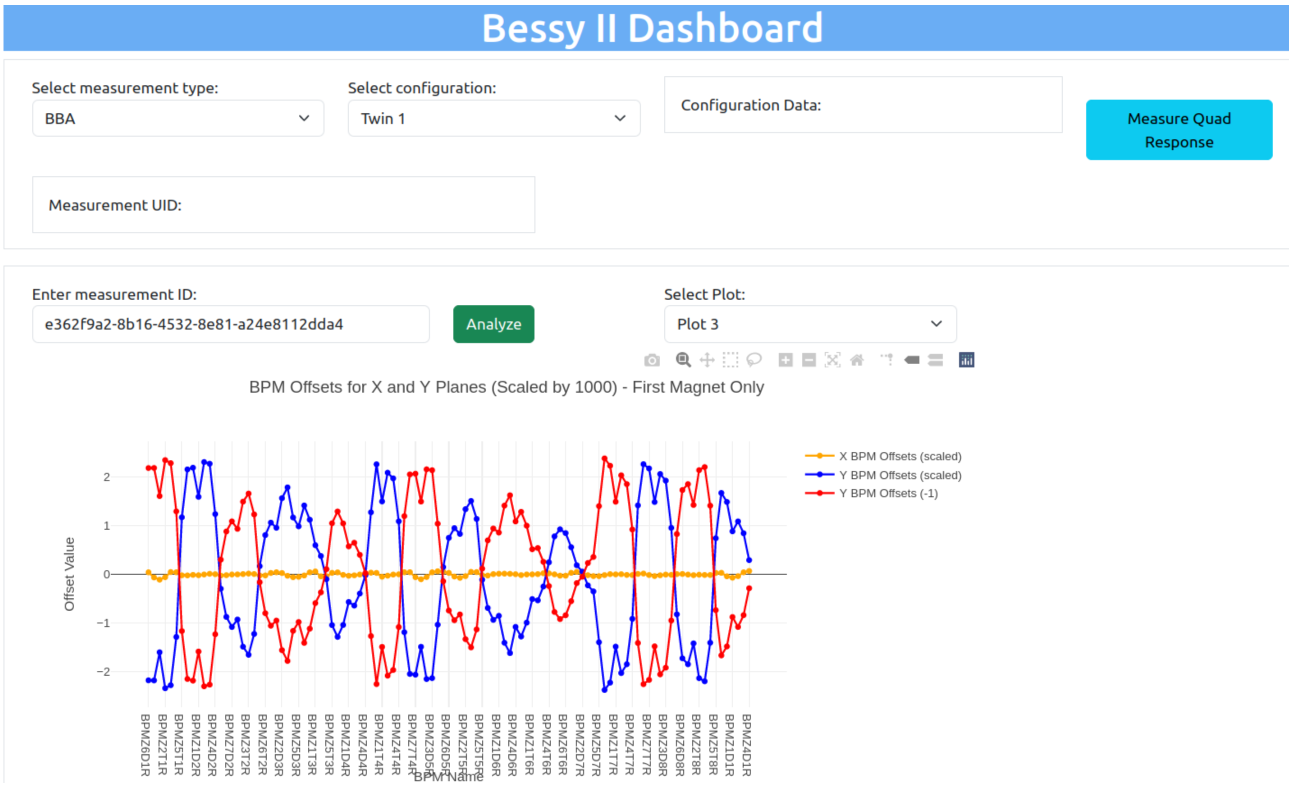


Figure 4: BBA Measurement Data Model.



Bessy II Dashboard

Select measurement type: BBA

Select configuration: Twin 1

Configuration Data:

Measure Quad Response

Measurement UID:

Enter measurement ID: e362f9a2-8b16-4532-8e81-a24e8112dda4

Analyze

Select Plot: Plot 3

Figure 5: Digital Twin Dashboard.

code maintenance significantly more straightforward. By structuring the entire workflow through an object-oriented paradigm, we enhance both the comprehensibility of the measurement process and the overall maintainability of our system.

Figure 4 further elucidates the intricacies of our data model for BBA measurement results, visually representing the underlying architecture. This model not only captures the raw data but also encapsulates the analysis parameters and outcomes, ensuring a comprehensive representation of the BBA process. Researchers benefit from a clear and structured overview, aiding in the interpretation of results and enabling data-driven decisions.

In tandem with our data modeling efforts, Fig. 5 showcases the BBA measurement results presented in a REST client. This visualization provides a real-time representation of the alignment variations, allowing researchers to interactively assess the outcomes of their experiments. The integration of visual feedback within the REST client enhances the user experience, offering a dynamic interface for result interpretation.

The harmonious integration of advanced data modeling, real-time visualization, and user-friendly interfaces within our digital twin not only optimizes the BBA measurement process but also serves as a paradigm for future developments in synchrotron light source facilities. This multifaceted approach not only streamlines current experiments but also lays the foundation for a data-driven future in scientific research at Bessy II.

CONCLUSION

In summary, the development of a digital twin for BessyII and MLS Synchrotron Light Sources using EPICS, microservice design, and REST API services will greatly enhanced system monitoring and optimization. The integration of EPICS and the adoption of a microservice architecture will enable seamless communication between components, improving interoperability and reliability. The digital twin provides a flexible and scalable solution for managing and optimizing synchrotron light sources, with potential for further customization and extension.

To conclude, the digital twin implementation will revolutionize system monitoring and optimization. The implementaion of end user scripts as REST API will empower users with advanced data analysis and visualization capabilities. Overall, the digital twin presents a flexible and scaleable solution for effectively managing and optimizing synchrotron light sources.

REFERENCES

[1] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art", *IEEE Trans. Ind. Inform.*, vol. 15, pp. 2405–2415, 2019. doi:10.1109/TII.2018.2873186

[2] P. Willmott, *An Introduction to Synchrotron Radiation: Techniques and Applications*. John Wiley & Sons Ltd., 2019. doi:10.1002/9781119280453

[3] Epics: Experimental physics and industrial control system, https://epics-controls.org/

[4] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Inc., 2021.

[5] 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

[6] K. Vodopivec, "Easy integration of Python into EPICS IOCs", presented at EPICS Collaboration Fall Meeting 2020, 2020. https://indico.fhi-berlin.mpg.de/event/52/contributions/555/