ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-WEPAF030

# HEPS HIGH-LEVEL SOFTWARE ARCHITECTURE PLAN*

P. Chu†, Y.S. Qiao, C.H. Wang, Institute of High Energy Physics, Beijing 100049, P. R. China
H.H. Lv, Shanghai Institute of Applied Physics, Shanghai 201800, P. R. China

*Abstract*

The High Energy Photon Source (HEPS) is a planned ultra-low emittance synchrotron radiation based light source which requires high precession control systems for both accelerator and beamlines. Such kind of accelerators will require extremely sophisticated high-level control software for both accelerator and beamline operation to achieve not only the demanded precision but also high reliability. This paper outlines the high-level application software architecture design including relational databases, software platforms, and advanced controls with machine learning (ML) techniques. Early plan for beamline control is also reported. For better quality control and easy maintenance, the high-level applications will be built upon matured software platforms. Also, the HEPS High-Level Software team will collaborate with EPICS community for improving the software platforms.

## INTRODUCTION

HEPS is a sub-100 pm·rad emittance, about 1360 m circumference 4th generation synchrotron light source which is designed and constructed by the Institute of High Energy Physics (IHEP), Beijing, China with a scheduled completion date at the end of 2024. The number of devices to be controlled, the required precisions, and the amount of experimental data will be collected are all unprecedentedly high. For such a large-scale light source, the control systems as well as high-level software should be proceeded with systematic approaches. Specifically, the high-level architecture starts with a data centric design. The HEPS high-level software architecture design considers not only for accelerator but also for optical beamlines while many software packages can be shared by both sides. Furthermore, data from both sides may need to be correlated for future Big Data analysis.

As shown in Fig. 1, basically the HEPS high-level software architecture can be divided into the followings: databases which including relational and non-relational ones, application programming interface (API) such as online model, optimizer, services, and channel access (CA) interface to EPICS control systems, and applications in various forms like desktop, web, and mobile apps. With this architecture design, each part can be replaced with difference technologies or implementations as upgrade may be necessary in the future. Also, because of the modular design and interchange ability for systems it is easy to collaborate with other institutes or projects which may have different system implementations. Details for each part in the architecture will be described below. Beamline control and programming language selections will also be mentioned.

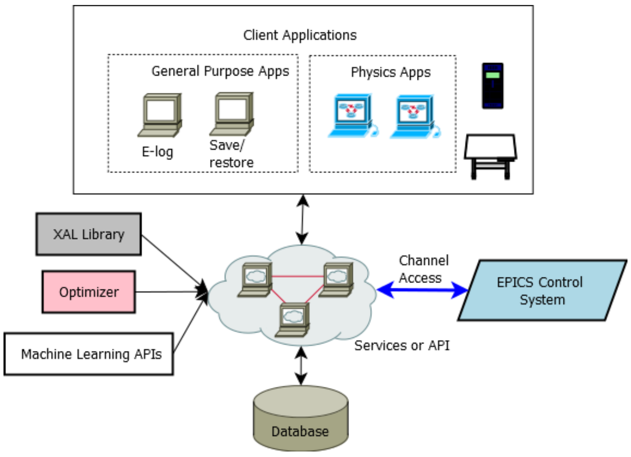


Figure 1: HEPS high-level software overview.

## DATABASES

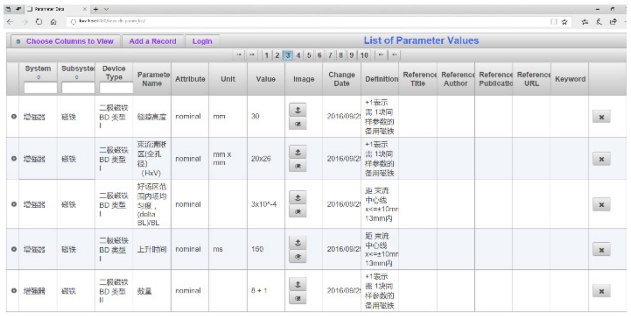
Any important data throughout the entire life cycle of an accelerator, including both static and runtime data, should be captured systematically and stored persistently. In addition to the data storage, applications to utilize the stored data are the foundations for increasing productivity, safety, and reliability. Based on the nature of the data, the data applications, and run-time efficiency, the data can be stored in either relational database (RDB) or other forms such as files or noSQL-based databases. A few databases needed for the present HEPS design and pre-construction preparation are under development or already in use will be described in the following subsections. In general, these databases are developed independently but with possible integration in mind. In the future, depending on the needs and database performance, one can link multiple databases together via either primary key/foreign key relationship or services. For most of the database linkages, the device ID is the joint point. The general procedure for many databases is first to fill a template in Excel form for initial or bulk upload. Apache POI [1] is applied as the Excel parser. Follow-up modifications in database can be done via a Web-based user interface (UI).

### *Design Parameter Database*

It is sometimes hard to keep track of all important physics and equipment parameters consistently yet correctly for an accelerator during the busy design period. A database for storing the essential HEPS parameters has been established. The database schema was based on ESS design [2] with necessary modifications to fit HEPS own needs. An Excel template is for system managers to fill

* Work supported by the Chinese Academy of Science and the HEPS-TF Project.
† chuzm@ihep.ac.cn.

the parameters, and then a program parses the filled Excel file and uploads the information to the database.



List of Parameter Values

| | System | Subsystem | Device Type | Parameter Name | Attribute | Unit | Value | Image | Change Date | Definition | Reference Title | Reference Author | Reference Publication | Reference URL | Keyword | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 增强器 | 磁铁 | 二极磁铁 BD 类型 I | 磁隙高度 | nominal | mm | 30 | | 2016/09/2 | +1表示需 1块同样参数的备用磁铁 | | | | | | |
| | 增强器 | 磁铁 | 二极磁铁 BD 类型 I | 束流清晰区(全孔径)（HxV) | nominal | mm x mm | 20x26 | | 2016/09/2 | +1表示需 1块同样参数的备用磁铁 | | | | | | |
| | 增强器 | 磁铁 | 二极磁铁 BD 类型 I | 好场区范围内场均匀度，(delta BL/BL | nominal | | 3x10^-4 | | 2016/09/2 | 距 束流中心线 x<=±10m 13mm内 | | | | | | |
| | 增强器 | 磁铁 | 二极磁铁 BD 类型 I | 上升时间 | nominal | ms | 150 | | 2016/09/2 | 距 束流中心线 x<=±10m 13mm内 | | | | | | |
| | 增强器 | 磁铁 | 二极磁铁 BD 类型 II | 数量 | nominal | | 8 + 1 | | 2016/09/2 | +1表示需 1块同样参数的备用磁铁 | | | | | | |

Figure 2: Design parameter database web UI.

The Design Parameter Database also has a web-based UI which can perform parameter query with general search functions as well as new data update. The front page of the web UI is shown in Fig. 2.

## *Naming Convention Database*

For a large accelerator project like HEPS, everything has to be named according to strict rules or it is hard to ever figure out what a name's meaning. The HEPS Naming Convention will be applied to both accelerator and beamline experiment instruments, but not applicable to cables. As shown in Fig. 3, device names are composed by system/subsystem (SSSS_BBBB), and device (RII_DDDDQIII) with signal name extension (TTTTIIII_XXXX).

The naming rules are stored in the database where device or signal names can then be generated programmatically. The naming rules can be checked against the rules and enforced while device and signal databases are being processed.

## *Magnet Database*

Magnets are among the most important components for a charged particle accelerator. We would like to capture and store all essential data regarding a magnet in a database. At this moment, the HEPS magnet group is busy with design and early prototype test work; therefore, the Magnet Database is focused on these functions. The present Magnet Database schema is shown in Fig. 4 which includes support for a few particular magnet measurement methods. It should be easy to add support for more measurement methods at any time. Applications for this database will provide magnet excitation curves for the control system.

As magnets being installed and becoming part of the accelerator, the magnet operation data will then be collected and correlated to this magnet database which can be analysed for better operation and maintenance purposes. Again, the link between the Magnet Database and operation database is through the device ID.

## *Equipment Database*

An equipment database for storing all equipment including spared parts is needed even for the preliminary stage of HEPS, the Test Facility (TF). A schema cropped from Integrated Relational Model of Installed Systems (IRMIS) v3 [3] serves as the starting point for this database. Figure 4 shows the present IHEP implementation of the equipment database.

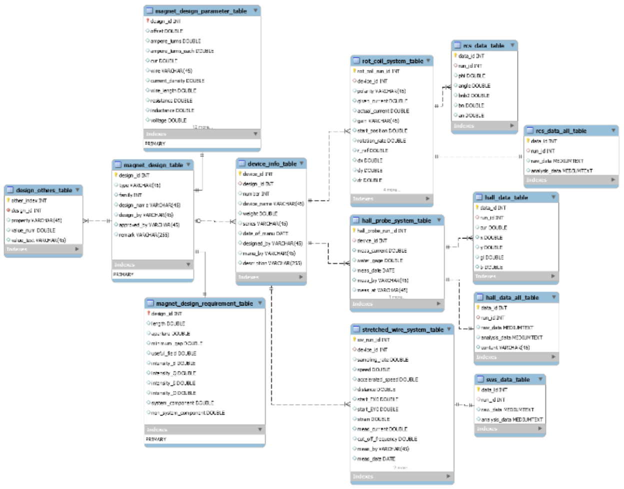


Figure 4: Magnet database schema.

## *Lattice and Model Database*

Lattice and Model Database is for keeping design lattices and their corresponding model calculation data. The Lattice and Model Database can accept various modeling formats with property name-value pair design. The lattice and model data is handled through Open XAL [4] API which will be described in the Software Platforms below.

## SOFTWARE PLATFORMS

As shown in Fig. 5, generally, application flow can be categorized into three layers: data, software API or services, and applications. For better software reusability, functions appears in multiple applications should be coded as regular callable API or service API form in the middle layer. Due to the nature of the functions, it is better to separate them in three groups so they don't mixed together and lose the flexibility: control system API, physics and general-purpose API, and machine learning API. Details for these APIs will be described in the following.

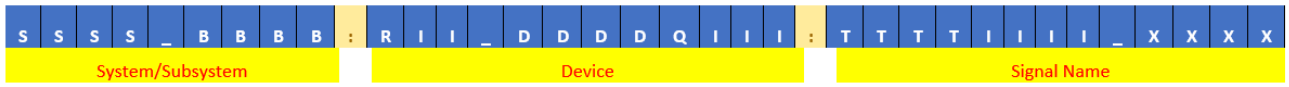

S S S S _ B B B B : R I I _ D D D D Q I I I : T T T T I I I I _ X X X X
System/Subsystem
Device
Signal Name

Figure 3: HEPS naming convention.

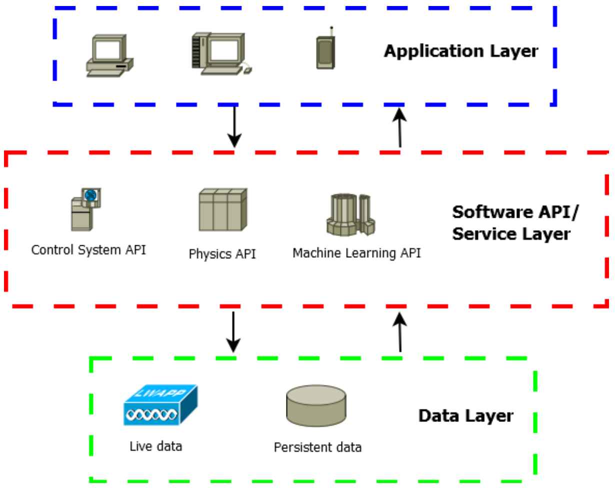


Figure 5: Software platform diagram.

### Control System Connectivity

HEPS accelerator as well as optical beamline is based on EPICS control systems. Instead that everyone has to deal with EPICS client calls, it is better to provide a set of easy API which can handle all the tedious work such as connection exceptions and operation loggings. If there is any need for another control system, we can simply replace the EPICS wrapper with a different one.

### Physics and General-Purpose API

Physics API provides physics specific functions such as online model while general-purpose API supports useful operations like parameter scan. Java-based Open XAL [4] toolkit includes not only definitive accelerator data structure but also quite complete functions for most accelerators. On the other hand, beamlines can have a similar platform for their physics.

### Machine Learning API

As the artificial intelligence (AI) and machine learning (ML) is making huge progress in many fields, it is also gaining attention for accelerators and accelerator related Big Data research. At this moment, the doorstep for utilizing AI tools is too high for any physicist who is not a seasoned AI player. It is convenient to provide a set of API to simplify ML application development. For instance, APIs for pre-processing raw data prior to applying the actual computation algorithms, and the computation result showing in visual form for easy read should be provided. All these can be done with simple APIs to cut development effort. Figure 6 illustrates the data cleaning processes in such a platform. The ML API should also provide wrappers for several popular open-source machine learning platforms such as Scikit-Learn and TensorFlow which, conveniently, are both support Python APIs. One can switch among ML platforms as well as algorithms for easy try-out and tests. The ML API will also be responsible for converting data format to suit these popular ML platforms.

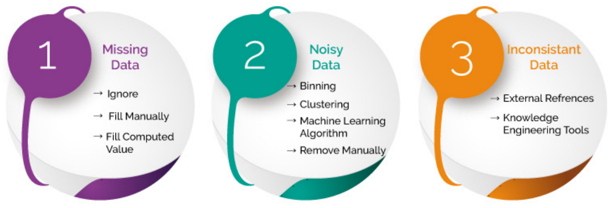


Figure 6: Machine learning API for data cleaning.

## APPLICATION PLATFORMS

For quick prototyping and development, applications are built upon platforms. If applications are well designed in Model-View-Controller (MVC) architecture, it is easy to share the model and controller codes between desktop and mobile versions of the same applications. The software platforms and APIs mentioned above will all have Java and Python API support for application development. As for mobile devices getting more and more popular, web and mobile app support will be provided and considered equally important as the desktop support. One can then easily assemble an application, regardless it is a desktop or mobile application.

## BEAMLINE CONTROL

For better manpower resources sharing, the HEPS optical beamline control is also handled by the same group as the accelerator controls. Also, it is not practical to have each beamline possessing its own database experts and handle all computing needs, for example. Many tools and platforms built for the accelerator can also apply to the beamlines. Standards like naming conventions and EPICS supported devices are also shared.

The beamline data may be much more structured than the accelerator data. Therefore, EPICS 7 which supports complicated data structures is being considered as the data protocol for packaging the beamline and experiment data. Still, the data structure has to be compatible for future mobile applications.

## CONCLUSION

The HEPS high-level software architecture has been sketched. Modern technologies will be applied to the actual implementation. With tremendous amount of work, it is fortunate that several domestic accelerator projects in China which are also at about the same stage as HEPS will be collaborating closely to share the development burden. For the software shareable between accelerators and optical beamlines as well as experiment stations, collaborations are also formed. Starting from project supports like databases and project management tools, the software team is also starting client application UI development such as mobile and web interfaces. All the databases are saved in GitHub repository for easy collaboration access [5]. This overall high-level software architecture design gives us the most flexibility and efficiency for software development yet ensure high-quality and reliable software products. Also the design is data centric which can be extended for future Big Data analysis.

## ACKNOWLEDGEMENTS

The authors would like to thank the Controls and Experimental Computing Support Groups of China Spallation Neutron Source, the HEPS Beamline Controls Group, and many other members of the IHEP Accelerator Center colleagues for fruitful discussions.

## REFERENCES

[1] http://poi.apache.org/.

[2] K. Rathsman *et al.*, "ESS parameter list database and web interface tools", in *Proc. 2nd Int. Particle Accelerator Conf. (IPAC'11)*, San Sebastian, Spain, Sep. 2011, pp. 1762-1764.

[3] http://irmis.sourceforge.net/.

[4] J. Galambos, et al, "XAL application programming structure", in *Proc. Particle Accelerator Conf. (PAC'05)*, Knoxville, TN, USA, May 2005, pp. 79-81.

[5] https://github.com/AcceleratorDatabase/.