FRIB HIGH-LEVEL SOFTWARE ARCHITECTURE*

P. Chu#, E. Berryman, T. Brown, R. Gaul, S. Peng, V. Vuppala, FRIB, East Lansing, MI 48824, USA

Abstract

The Facility for Rare Isotope Beams (FRIB) is setting up its high-level application software architecture. The architecture consists of back-end data storage, client/service infrastructure, control system connectivity, supporting libraries and front-end Graphical User Interface (GUI). The relational database and services are based on the Integrated Relational Model of Installed Systems (IRMIS) design. The GUI is based on Control System Studio (CSS) framework. Libraries, service, data access and GUI tools will be available as Application Programming Interface (API). The infrastructure and technologies chosen here will utilize the robustness and performance for applications, as well as support quick prototyping for physicists. The overall architecture and some prototypes are described.

INTRODUCTION

The architecture for FRIB High-level Applications (HLAs) is to utilize feasible and cost effective technologies for the software needs during the FRIB design, installation, commissioning and operation. For a modern accelerator, the software complexity is way beyond simple personal programming effort; therefore architecture reflecting all modern software needs is necessary as a systematic approach. On the other hand, a comprehensive architecture design allows developers to take advantage of new technologies in computer software, hardware and network. The architecture design serves as a standard applied to the FRIB Accelerator Systems with extension to the Experimental Systems. A standard architecture can avoid unnecessary software complication such as hard-wiring or special codes, and duplicating efforts in various applications.

ARCHITECTURE OVERVIEW

HLA architecture has to fulfil both functional and non-functional requirements. Functional requirements include database as backend storage, online physics model, data integrity and control system connectivity support; while the non-functional requirements should cover performance, reliability and scripting support.

The data flow view of the architecture is shown in Fig. 1. Starting from the bottom of the diagram, all static data are stored in a MySQL-based global database (Global DB) similar to the IRMIS (Integrated Relational Model of Installed Systems). Physics routines, machine tuning algorithms and other utilities are supplied as software libraries in the left-hand side of the diagram. XAL Toolkit [1], optimization routine, and data plotting package are examples of many supporting library packages. Device controls are done via EPICS (Experimental Physics and Industrial Control System). Client applications are the user interfaces to the entire control systems. Examples of client applications are electronic logbook (E-log), machine settings save and restore (Save/Restore) and beam tuning applications. In the middle, a group of services provide all links among the database, API libraries, interaction with control systems, and user controlled interfaces. The services are distributed as their functions.

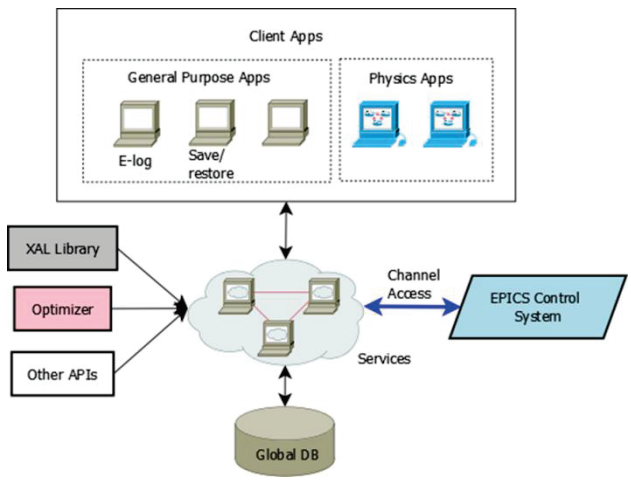

Figure 1: Top-level FRIB HLA architecture diagram.

The architecture design should take into account of all possible software needs such as data sharing, correlation, data exchange among applications, code reusability and cost efficiency. It will be difficult to deal with many ad hoc applications, if the overall architecture is not well structured. Design considerations are described briefly here.

Simplicity

A simple architecture system is more cost effective and easy to maintain. Typically a simple system can provide better performance with less overhead.

Service-oriented

Standalone applications typically need to prepare large amount of data initialization, to perform heavy computation, to communicate in real time, and to display in high repetition rate of updates. The result of such heavy applications is poor performance and less reliability.

*Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661.
#chu@frib.msu.edu

One way to solve the performance and reliability problems is to implement a Service-Oriented Architecture (SOA) for high level applications [2]. Servers will handle heavy computation for better performance. Reliability can be improved because functions are distributed to various services which have the same standard. Properly designed service architecture also provides better flexibility and extensibility because they can be highly modularized and each can be swapped easily. SOA is not needed for day one and can convert to service as needed. Because of the architecture's modularization, it is easy to switch to new technologies when they are available.

Up-to-date Technologies

The architecture should be based on currently widely available technologies such as Web services as Web compatibility, EPICS v4 as new EPICS community trend and Eclipse plug-in architecture as package management. These technologies are just examples for the architecture design consideration. As new technologies are available and matured, the team can consider the possibility of early adoption. Any new technologies will be used for production have to be proven robust and easy to maintain. Many new technologies such as web approach are optional and presently not in the baseline requirements.

Re-usability and Extensibility

Existing software such as XAL, IRMIS schema/services can be reused at FRIB in the way of callable API. In addition, some GUI components such as orbit display, embedded e-log screen, and "Save to e-log" button should be callable API.

Furthermore, common functionality can be shared among applications. If any application has to implement every function in it without sharing common functions among applications, the maintenance overhead is high and application code might be overly complicated. Also, even if some functions are not exactly the way application intending to use, it is easy to extend the existing functions for new needs.

Portability

Applications should be possibly adopted by other accelerator facilities. Any library components or packages developed here should not include FRIB specific hardwired code. It is recommended to use configuration files to handle site specific data. This will enable community-wide collaboration.

Security

To ensure uninterrupted operation, security is a vital part of the architecture. Security consideration should be a balance between access convenience and safeguarding of the systems.

Collaboration

With limited resources and budget, it is practically not possible to develop a full blown software system with one institute alone. On the other hand, many accelerator institutes face similar issues for high-level applications. Collaboration can be efficient to share work load. Properly dividing the entire software system can then distribute work effectively among collaborators.

SERVICES

Several services have been identified for the FRIB commissioning and operation needs. They are described below.

Magnet Service

Magnet service handles all magnet PVs for reading, setting and monitoring. Client applications do not have to worry about EPICS Channel Access connection. This service should have the following features:

- Updating set values and read-back values for all magnets with monitoring capability.
- Providing magnet statuses.
- Unit and magnet name conversion between physics and engineering (EPICS) names.
- Handling out-of-tolerance exception when trying to set a magnet.
- Knowing how to gracefully roll back magnets if any failed to set and logging problems.
- Providing other magnet attributes such as location, polarity.
- Handling multiple channels in parallel for best performance.

RF Service

Phase and amplitude for RF cavities should be locked to their desired energy profile. A feedback-like program runs continuously to track RF settings such that they are not drifting away from the desired energy profile. Also, running programs such as LEM and recovering from cavity quench requires careful RF setting, i.e. ramping in slow speed; all these considerations should be built in the RF Service.

Model Service

Model service runs online model periodically and makes up-to-date model data available for clients. This model server can be extended to cover not only XAL model but many other modelling codes with uniform API. Details for the model server will be described in a separate contribution [3].

LEM Service

Linac energy manager (LEM) for maintain certain beam optics due to energy change should run all the time to allow fast optics correction and restoration.

Save/Restore Service

Save/Restore has to connect to many channels. It could be very heavy load to IOCs and network to have each client having its own connection to all those channels. A service can connect all these channels and monitor them

ISBN 978-3-95450-115-1

all the time. Whenever a SCORE snapshot is requested, all data can be served up immediately.

BPM Service

BPM (Beam Position Monitor) service collects all the BPM data and can be buffered for clients to consume. All BPM data should be time-correlated in one collection record. Typical clients for this service are Orbit Display and Orbit Correction.

Beam Loss Monitor Service

This service is similar to the BPM Service but, instead, for monitoring Beam Loss Monitors (BLM). A client for this service is display for beam loss across the entire machine.

Steering Service

This service provides continuous orbit correction solution. It can be a back-end for transverse feedback. This service is a client of the BPM Service.

Directory Service

Directory Service provides information about the control system including its services. It is mainly showing related information from the database with searchable capability.

Alarm Service

The service provides access to alarm information and alarm configuration data.

Authentication Service

Authenticates users, and authorizes access to services, devices, instrument access, various logbooks and even access to database are included in the Authentication Service.

Logbook Service

Logbook Service is to manage entries in the logbook.

DATABASE

The Global Database inherits major concepts and functionalities from IRMIS and its services. IRMIS subschemas cover many areas such as devices (components) and documentation (electronic document Traveller). In general, users do not need to access the database directly and all access is through services. Business logic is mostly embedded in the service layer to avoid complication at the database level. For each subschema, there is at least one corresponding service for managing data access as well as additional data processing. The Global DB has similar to IRMIS' subschema structure.

So far, there are 17 subschemas identified for the Global DB. The "core" of this schema is the Installation which is based on accelerator design. Within Installation, Components hold data for any entity or building blocks of FRIB such as magnets, power supplies, cavities etc.; Configuration then represents the entities that exist on the blueprint or configuration of the FRIB facility. As mentioned above, one purpose to modularize the entire schema is to distribute work among collaborators. On the other hand, all subschemas are loosely coupled to ensure data correlated properly. Note that the control data is mostly residing within control systems as EPICS records, not in the relational database. However, the control systems should be initialized from the relational database. All the other subschemas are based on applications.

FRIB Naming Convention will be supported in the Components subschema and service. FRIB official parameter should be a by-product of the Installation subschema.

A Brief description for each subschema is listed below.

- Alarm – maintain alarm settings.
- Authentication and authorization – user authentication, group and role mapping information.
- Cables – cable connection information.
- Directory Service – organized device information for easy lookup.
- Installation – physical and logical information about the machine and its component systems.
- Interlocks – interlock hierarchy or dependency.
- Inventory – spare parts and stock items.
- Lattice – position, length, default setting and other physics related information.
- Logbook – electronic logbook entries.
- Maintenance – preventive maintenance data and scheduling, failure analysis, and lifetime analysis.
- Model – physics model data.
- MPS – machine protection system (MPS) state dump, MPS faults for post-mortem analysis.
- Operations – beam statistics, run hours, beam on target, shift summary, downtime, and bypass records.
- Physics – results from physics applications or experiments.
- PV – EPICS Process Variable (PV) information.
- Save and restore – machine snapshot and restore condition (certain signals cannot be restored unconditionally).
- Traveller – work flow control and process tracking for measurements, calibration and test data.

REFERENCES

[1] J. Galambos et al, "XAL Application Programming Structure," p. 79, Proceedings of 2005 Particle Accelerator Conference.

[2] G. Shen et al., "A Novel Approach for Beam Commissioning Software Using Service Oriented Architecture," PCaPAC 2010, Saskatoon, Saskatchewan, October 2010, WEPL037, p. 100 (2010).

[3] P. Chu et al, "Online Physics Model Platform", TUPPC048, these proceedings.