HIGH-LEVEL PHYSICS CONTROLS APPLICATIONS DEVELOPMENT FOR FRIB

Tong Zhang
Controls Physicist
Accelerator Physics Department
Facility for Rare Isotope Beams, Michigan State University
E-mail: zhangt@frib.msu.edu

Presented at 17th International Conference on Accelerator and Large Experimental Physics Control System
5-11 October 2019, New York, NY, USA
1. Architecture
   - Introductions
   - PHANTASY Project

2. Application Development
   - UI Widgets
   - Showcases

3. Conclusions
Outline

1 Architecture
   - Introductions
   - PHANTASY Project

2 Application Development

3 Conclusions
Introductions to High-level Physics Applications

Purpose: have robust and functional beam tuning algorithms
Solution: sofware environment for high-level physics controls

High-level Physics Applications = Physics Algorithms + Controls Sofware
**Accelerator System**
- particle source, beam transport, experimental end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units (Channel Access): EPICS input & output controllers (IOCs)

**High-level Physics Applications**
- **Final goal**: operating accelerator facility
- **Purpose**: have robust and functional beam tuning algorithms
- **Solution**: software environment for high-level physics controls

High-level Physics Applications = Physics Algorithms + Controls Software
**Fundamental Requirements**

- **Quick prototyping**: dynamic programming language, powerful & friendly development environment
- **Features**: plenty of third-party packages
- **End-users**: good UI/UX with proven physics algorithms
- **Agile development**: develop → build → test → deploy

*Software Solution upon Python Programming Language*
PHANTASY
Physics High-level Applications and Toolkit for Accelerator System

FEATURES HIGHLIGHT

- Device configuration management: maintainability, portability
- Device abstraction: object-oriented
- Online modeling: physics model-depends machine representation
- Python interactive scripting environment for high-level controls: development and control
- Virtual accelerator based on EPICS and physics model: test physics algorithms
- Web service integration (channelfinder, scanserver, UNICORN ...): extendability

DEPLOYMENT

- Target OS: Debian 8, 9, 10
- Meta package: phantasy (~20 packages)
- Physics model engines: FLAME, IMPACT, TRACK
Architecture of PHANTASY Project
Architecture of PHANTASY Project

Code Facts
- Python 2 and 3, Debian OS
- phantasy (core) ~19k LOC
- UI widgets: ~11k LOC (incl. dataviz, excl. ui)
- Apps: ~26k LOC (excl. ui)

Dev Guideline
- Device → Python object
- List of devices → Lattice
- Lattice manipulation
- Scripting/GUI apps (facility-agnostic)
- Package → deployment
Device Abstraction

- Standardize device description data file
- Handle EPICS Channel Access (CA) in OO level
- Benefits: WORA, facility-agnostic script/apps

- Manage physics and engineering units by dedicated web app
- REST APIs for units interpretation
- Benefits: availability and extendability

Read/write policy from/to device could be defined with REST APIs.

WORA: Write Once, Run Anywhere.
I/n.sc/t.sc/e.sc/r.sc/a.sc/c.sc/t.sc/i.sc/v.sc/e.sc P/y.sc/t.sc/h.sc/o.sc/n.sc S/c.sc/r.sc/i.sc/p.sc/t.sc/i.sc/v.sc E/n.sc/v.sc/i.sc/r.sc/o.sc/n.sc/m.sc/e.sc/n.sc/t.sc

Continue with more complex work...
Outline

1 Architecture

2 Application Development
   - UI Widgets
   - Showcases

3 Conclusions
GUI App Development with Python and Qt5

- Standardized UI style (package: phantasy-ui)
- Standardized app development workflow
- Publish-ready quality figure
- High productivity
- Facility-agnostic apps
- Streamline deployment (package: phantasy-apps)

FRIB Collection: lattice, element, etc.
DataViz Widgets: curve, curve w/ errorbar, image, etc.
Commands: run_designer, frib_designer, makeBasePyQtApp

Package: mpl4qt
- Toolbar system
- Configuration system
- Qt-designer widgets

> mpl4qt: Data visualization widgets implemented with matplotlib for Qt5.
App Launcher: Manage All Apps in One Place

Features

- Global entrypoint: right-clicking context menu → ‘Phyapps Apps’
- Configuration file (.ini) controlled for easy maintaining
- Run app by double-clicking, w/ or w/o console, keep logs or not
App Launcher: Manage All Apps in One Place

Features:

- Global entrypoint: right-clicking context menu → ‘Phyapps Apps’
- Configuration file (.ini) controlled for easy maintaining
- Run app by double-clicking, w/ or w/o console, keep logs or not
**FLAME: Fast Linear Accelerator Model Engine**, envelope simulation code (C++/Python) developed at FRIB.

FRIB
Facility for Rare Isotope Beams
U.S. Department of Energy Office of Science
Michigan State University
**Online Central Trajectory Correction: Based on response matrix**

**Trajectory Viewer App**
- Load lattice, Tools → Load Lattice
- Select BPMs and Correctors
- Visualize trajectory

**Optics Response Matrix App**
- Set up correctors alter ranges
- Measure optics response matrix, save/load
- Correct trajectory with optics response matrix

Easy way to change COR scan range
Online Central Trajectory Correction: Based on response matrix

- Gradually suppress trajectory
- Secure tuning range of correctors
- Visual confirmation for each solution
- Stop button

- Keep all the history settings of correctors
- Go back and forth at any history settings
- Save/load settings, matrix
**Online Central Trajectory Correction: Based on Response Matrix**

Trajectory Correction with ORM from MEBT to FS1A
2019-09-24

Measured Response Matrix for LS-1
**GENERAL PARAMETER CORRELATION ANALYSIS AND VISUALIZATION**

Scan parameters by mouse clicking

Curve w/ errorbar widget

MPS signal interlock

Element single/multi selection

Tools → Load Lattice Work with different lattice

**FRIB Lattices**

- **Machine**: FRIB VA
- **Segment**: LS1
- **PV Prefix**: None

String Prefixed to All PVS

Loaded FRIB VA/LS1.
Extend to Higher Dimensions

Multi-charge states study

Completed at 2019-04-12 19:04:32
SCAN Duration: 304.73 s
1 Architecture

2 Application Development

3 Conclusions
Conclusions

- Developed Python-based software framework (**PHANTASY**) for high-level physics controls at FRIB
- PyQt5 based modularized generic GUI apps are developed for FRIB
- Efficient machine tuning is achieved with the developed apps for FRIB LINAC commissioning
- Apps based on **PHANTASY** also work with other EPICS-based facilities, no physics model dependency
- Development of physics model progresses as the commissioning advances

**Future Development Plan**

**DATA**  Unify and standardize data interface of all physics apps, streamline the data post-processing

**GUI**  Develop GUI toolkit for high-level physics apps development by any developer

**MODEL**  Based on the data flow from **PHANTASY** framework, refine physics model by machine-learning

**COLLABORATION**  [https://github.com/phantasy-project/](https://github.com/phantasy-project/)
Dylan Maxwell, Martin Konrad, Daron Chabot, Eric Berryman, Steven Beher, Masanori Ikegami, Bruno Martins, Diego Omitto, Scott Cogan, Steven Lidia, Michael Davidsaver, Guobao Shen, Kei Fukushima, Tomofumi Maruta, Takashi Yoshimoto, Jonathan Wong, Alexander Plastun, Qiang Zhao, Yue Hao, Steve Lund, Peter Ostroumov

Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University.
Thank you for your attention!