BEAMLINE INSERTIONS MANAGER AT JEFFERSON LAB [*]

M. Johnson[†], Jefferson Lab, Newport News, VA, 23606

Abstract

The beam viewer system at Jefferson Lab provides operators and beam physicists with qualitative and quantitative information on the transverse electron beam properties. There are over 140 beam viewers installed on the 12GeV CEBAF accelerator. This paper describes an upgrade consisting of replacing the EPICS based system tasked with managing all viewers with a mixed system utilizing EPICS and high level software. Most devices, particularly the beam viewers, cannot be safely inserted into the beam line during high-current beam operations. Software is partly responsible for protecting the machine from untimely insertions. The multiplicity of beam-blocking and beam-vulnerable devices motivate us to try a data-driven approach.

The beamline insertions application components are centrally managed and configured through an object-oriented software framework created for this purpose. A rules-based engine tracks the configuration and status of every device, along with the beam status of the machine segment containing the device. The application uses this information to decide on which device actions are allowed at any given time.

VIEWER SYSTEM

CEBAF Operations at Jefferson Lab run continuously 24/7 to support a program of experimental physics and accelerator studies. Control system software supports this effort in part by providing operators with the tools needed to diagnose and resolve issues with accelerator hardware and instrumentation. The software system that controls the insertion and retraction of viewers and other devices, dubbed "Insertables", has been rewritten using object-oriented programming methods and deployed to give operators and support personnel the ability to diagnose and resolve problems at all hours.

This new software system provides better encapsulation, separating device configuration from device behavior. A comprehensive view of insertable beamline components makes it easier to find configuration anomalies, such as device channel assignments that do not follow established patterns. Device failures during off hours have been diagnosed using debug screens and resolved quickly. Overall, faster problem diagnosis and resolution has resulted in a reduction in down time caused by this type of component.

[*] Notice: This manuscript has been authored by Jefferson Science Associates, LLC under Contract No. DE-AC05-06OR23177 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non- exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

[†] michaelj@jlab.org

In the following sections, we describe the context, requirements, and techniques employed in implementing this system.

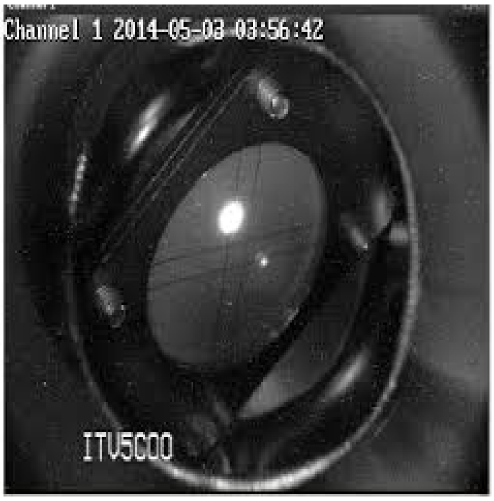

Figure 1: Beam Spot on Viewer

SOFTWARE REQUIREMENTS

The operation of the CEBAF accelerator at Jefferson Lab relies on a wide variety of devices which must be physically inserted into the beam line when needed. Beam viewers measure beam profile and provide information about transverse beam characteristics. Faraday cups measure beam intensity at several points around the accelerator. Insertable dumps provide device protection and control of beam transport to the experimental end stations. These devices and a variety of others are critical to the delivery of a precisely tuned electron beam.

To support this heterogeneous and frequently changing hardware configuration, the software system must:

- Provide a hardware-domain description of each device in a format useful to engineers
- Reduce the duty-cycle of the beam in response to a request to insert a vulnerable device
- Change device-protection strategy when accelerator zones change between active and inactive.
- Allow an inactive experimental end-station to work with viewers without interfering with an active experiment
- Respond to unauthorized device motion by protecting the machine
- Track device motion and find faults in behavior (broken switches, inactive actuators)

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPWI044

- Retract a viewer when a new viewer is inserted
- Automate repetitive behaviors, such as turning on lights and routing video signals when viewers are inserted, but allow some of these automated actions to be manually disabled.

DESIGN

The control systems at Jefferson Lab are based on the Experimental Physics and Industrial Control System (EPICS). [1] EPICS is a set of open source software tools used to create distributed real-time control systems. EPICS provides abstractions for prescribing automated behavior and supervisory functions. A control system in EPICS is created using a special database comprised of records called Process Variables (PVs). Several graphical tools provide the ability to instantiate and link PVs in a way that describe the control system behavior.

Visual programming and databases make it difficult to create new abstractions, such as a hierarchy of device types. A variety of insertable device types is easily realized in an object-oriented framework. Tangible elements such as "viewer" or "light" are represented by classes, while other classes take responsibility for coordinating activities or implementing control strategies. To take advantage of this more flexible programming model, we first need to interface with EPICS.

EPICS Interface Classes

The application decouples from EPICS by using PVs for only a few necessary functions, and by including a layer providing a simplified interface to PVs through channel access. PVs are conveniently divided into hardware records, control screen records, and accelerator condition records. Hardware records communicate directly with hardware, and Device objects subscribe to these records and reflect changes in their state. Control screen and accelerator records are monitored by an activity coordinating object, which also aggregates sets of device objects and monitors and mutates their state.

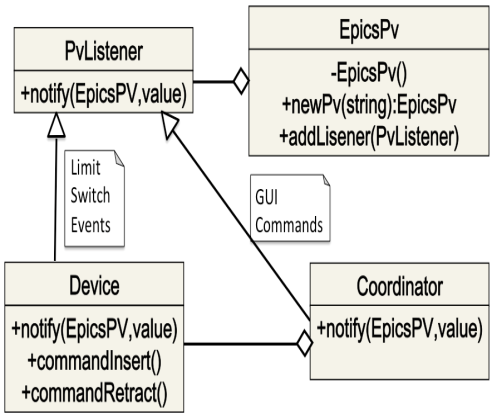

Figure 2: EPICS Facade Classes

Application Classes

A Device class holds all state and behavior associated with an insertable device. This provides us with an immediate improvement over earlier versions of the software: By keeping a command history and monitoring the state of an individual device, the object has a way to build more descriptive failure messages for faulty hardware. Error messages to the operators distinguish between a device that will not move, one that moves errantly, and one that moves but never reaches its destination. The Device class hides these internal details and exposes to the rest of the system only commands to insert or retract a device, route its video, or check its status or configuration.

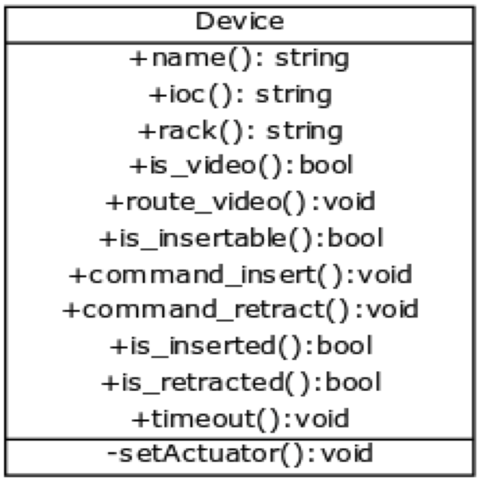

Figure 3: Device Class

The Coordinator class subscribes to EPICS channels and aggregates collections of Device objects. This class is responsible for dealing with changes to device state, as well as responding to operator commands. Having a centralized coordinator makes it possible for our control algorithms to respond to commands, check the state of the accelerator and all devices, adjust the duty-cycle of the beam, and insert and retract devices using a global knowledge of the system.

CONFIGURATION

XML Specification

A data-driven approach greatly improves maintainability of the system as a whole. All device and channel information is consolidated into a single structured XML file. All information about a device, its classification, and its channels can be found within a few lines of configuration data.

Recently, efforts have been underway to move this configuration into the CEBAF Element Database. [2] Managing this data in a centralized database helps multiple users to coordinate and agree on a single view of the configuration, and allows users to access its configuration without contacting a specialist. Collecting data in this manner also helps with validation of the system as a whole. Hardware engineers,

```
<device="ITV0I02" ioc="iocin3" type="viewer">
<insert_limsw card="IN01S19" channel="9"/>
<retract_limsw card="IN01S19" channel="10"/>
<solenoid card="IN01S18" channel="5"/>
<camera card="IN01S21" channel="5"/>
<light card_name="IN01S18" channel="16"/>
</device>
```

Figure 4: XML Configuration Data

with this data in hand, can find spare channels and duplicate entries.

Deployment

This new system has been completed and deployed in the commissioning of the newly upgraded 12GeV CEBAF beam line. Our experience with the system has demonstrated some benefits of this approach.

Identifying Inactive Hardware

Explicit configuration information consolidated in an easily-accessible format yields insight to users. Instrumentation engineers have been able to use this new system to identify devices that were physically removed from the accelerator long ago but were still left in the software configuration as an artifact. Using the older, pure-EPICS software, inactive or missing hardware was usually found only when engineers looked for disconnected ports on IO cards.

Programmable Behavior

Older versions of this software limited beam activity during device insertion by assuming the worst case: active beam in all areas of the accelerator. Each class in the new software system expressions a single type of functionality, making it relatively easy to understand and modify the behavior of the modules. The Coordinator class has been updated to take into account the presence or absence of beam in each accelerator zone. It uses this information to make more educated decisions about when and how to protect insertable devices, allowing inactive zones to accommodate hardware installation while active zones are carefully managed.

After Hours Support

On more than one occasion during commissioning, an insertable device has failed in some way due to hardware malfunction, such as a faulty limit switch or an obstruction in the insertion path. Operators were able to easily identify the faulty hardware using operations screens showing explicit error messages containing device ID, state, and recent activity. This enabled operations personnel to temporarily disable faulty hardware and continue operating the accelerator until engineers arrived on site the next morning to repair damage.

CONCLUSION

The Insertables System now serves as our replacement for the Viewers System. Safety protocols have become more sophisticated. When experimental end stations are being protected by beam stopper devices, the Insertables System will not try to protect devices in those end stations using software. This has the effect of reducing the amount of time we spend limiting beam current to experiments when others are reconfiguring.

Removing a monolithic state machine from an IOC and replacing it with software running on a UNIX host takes away a critical piece of code from a frequently-rebooted IOC. The server process runs for weeks at a time without interruption.

Static configuration data is more readily accessible. Our system generates hardware-centric configuration tables for hardware and software engineers to use as collaboration tools. Verbose dynamic status information is also available in the system. Each device now supports a status string detailing its last known state. Faulty limit switches and other hardware can be found more easily.

Future Work

This new system is well positioned to maintain and enhance. The modular design allows us to quickly implement upgrades once they are envisioned by accelerator scientists and operators. More sophisticated beam current limiting rules are being explored. Beam stopping devices and laser configuration are being considered as criteria for device protection.

New devices can be supported with relatively few changes. A new device type is easily added to the configuration data, and the Coordinator module dynamically adds new devices to appropriate sets and monitors their behavior. More device types may be added to this system in the future, allowing us to remove these duties from other systems and reduce overall code complexity in the facility.

ACKNOWLEDGEMENTS

Many thanks are owed to many people in the Instrumentation and Control Group, the Accelerator Operations Group, the Controls Software Group, and the Analysis-High Level Apps Group at Jefferson Lab.

REFERENCES

[1] EPICS Home Page, `http://www.aps.anl.gov/epics/index.php`

[2] M. Joyce, M. Keesee, C. Slominski,, R. Slominski, D. Turner, T. Larrieu, “The CEBAF Element Database and Related Software Development," (2015) IPAC 2015, Richmond, Virginia, USA, MOPWI045