Wesley Ketchum (FNAL)
ProtoDUNE SP Run Control, Operational Monitoring, and Configuration, V1.1
ProtoDUNE SP DAQ Design Review Documentation
27 October 2016

Run Control, Operational Monitoring, and Configuration: ProtoDUNE SP DAQ Design Plans

The ProtoDUNE SP DAQ will use a collection of tools currently in use for accomplishing its run control, operational monitoring, and configuration management. Many of these tools are already in use by the *artdaq* software suite and benefit from support from *artdaq* and 35-ton DAQ experts, while others are in common use at CERN and benefit from existing operational expertise at the host lab. In this section, we describe in detail the design plan for implementing these tools into a fully operational DAQ system.

Run Control

Requirements

The run control software for protoDUNE SP needs to satisfy the following requirements:

- as *artdaq* is the core framework for the DAQ event-building software, it must start, stop, and monitor the *artdaq* processes, and push them through their states, ideally via the existing control structures currently in use in *artdaq*;
- it must provide a clean and simple interface to shifters/operators;
- and, it must allow for including and excluding individual components, and allow for partitioning to run two or more subsystems independently.

We have chosen to use the JCOP (Joint Controls Project) framework in common use at CERN, which uses the WinCC-OA (previously called PVSS) supervisory control and data acquisition system at its core. The JCOP system, with additional development of a DAQ interface to the *artdaq* system, satisfies the above requirements and carries with it the added benefit of expertise and support available at CERN.

artdaq control tools

The *artdaq* software suite contains tools for starting and stopping *artdaq* processes, and sending state transition requests to configure, start, and stop the data-taking functions of those processes. This section will contain a brief description of these elements, and describe the interface layer being developed to allow a connection to the JCOP run control system.

artdaq contains a common Process Management Tool (PMT) that can take a DAQ systems architecture configuration and start all the necessary *artdaq* processes (BoardReaders, EventBuilders, Aggregators, etc.). Inside the PMT, calls to start and manage these processes are made via *mpirun*, which assigns MPI ranks to each of the processes that in turn are used to control the data flow from BoardReaders to EventBuilders and EventBuilders to Aggregators. MPI, the Message Passing Interface messaging system used for passing data between independent threads or processes, is currently used as the data transfer mechanism in *artdaq*. In the future, as *artdaq* makes available alternative data transfer mechanisms, the underlying calls in the PMT may be modified to not use *mpirun*. The systems architecture configuration consists of a list of desired processes, which DAQ node those processes should run on, and a port number on that node to be used for sending and receiving control messages. The PMT, technically, is a simple Ruby script that can be launched from a command-line or as part of other scripts.

Control messages and responses are sent using the XML-RPC (eXtended Markup Language Remote Procedure Call) protocol. Typical control messages request state transitions to be accomplished or request the status (existing state) of the *artdaq* processes, including the PMT. Parameters are also passed in these control messages: for instance, the *daq.init* command is used to initialize and configure the *artdaq* processes, and takes as an argument a string in FHiCL format (see the "configuration" section below) to be used for configuration parameters. XML-RPC commands can be launched from the command line or as part of other programs---APIs for issuing XML-RPC calls exist for many languages, and Ruby-based control scripts exist as part of the *artdaq-demo* that use such APIs.

Typically, deployed *artdaq* systems use simple, experiment-specific interface programs in their run control systems, acting as a layer between an operator-facing program and the underlying PMT/XML-RPC calls. In the DUNE 35-ton prototype, this was simply called the DAQInterface. We plan to deploy a simple interface based largely on the 35-ton's and other *artdaq* experiments' experience, allowing for configurable clients for DAQ process input and output, and flexible transition command implementation. A demonstrator of this interface is in progress now, with initial implementation on the *artdaq-demo*.

JCOP Run Control

The JCOP framework is a collaboratively-build control systems framework in common use by the LHC experiments at CERN. It provides many features, including custom data elements to describe detector devices, hierarchical organization of devices, and control of that hierarchy via a finite state machine toolkit, process management tools, device and system configuration, automated control, alarms, GUIs for displaying status and issuing user-driven commands, and more. Because it is shared across the LHC experiments at CERN, many common device configurations and interfaces to the underlying WinCC-OA supervisory control system already exist and are available to be used by any experiment, simplifying the development time. Once a detector device type is defined, it is easy to build hierarchies of these devices to describe a full system.

JCOP has a number of options for communication with external processes, like the *artdaq* BoardReaders, EventBuilders, and Aggregators. We plan to use the DIM (Distributed Information Management) system as a communication mechanism to

JCOP. DIM is a simple server/client-based network communication system, that allows for the creation of servers that publish data and clients that read data and can issue commands to the servers. A separate name server is maintained for registering of services and provides the information to clients on how to subscribe to data streams. C++, Java, and Python APIs exist for DIM server and client actions. And, most important, a JCOP framework module, fwDIM, exists, allowing for easy integration into JCOP.

For the integration between *artdaq* and JCOP, see Figure 1 below for a sketch of the proposed architecture. We plan to implement a DIM service for the overall DAQ interface layer and for each *artdaq* process. The DIM service would allow for publishing of data out (status information), and the receipt of commands (like status requests and state transitions). Using the existing JCOP framework, corresponding DIM clients would be established as part of the JCOP run control. The run control would then receive and update status displays, and upon user action, would issue a command via the DIM client to the DAQ interface's DIM service. That command would be interpreted by the DAQ interface to make the corresponding PMT and/or XML-RPC commands to the *artdaq* processes.

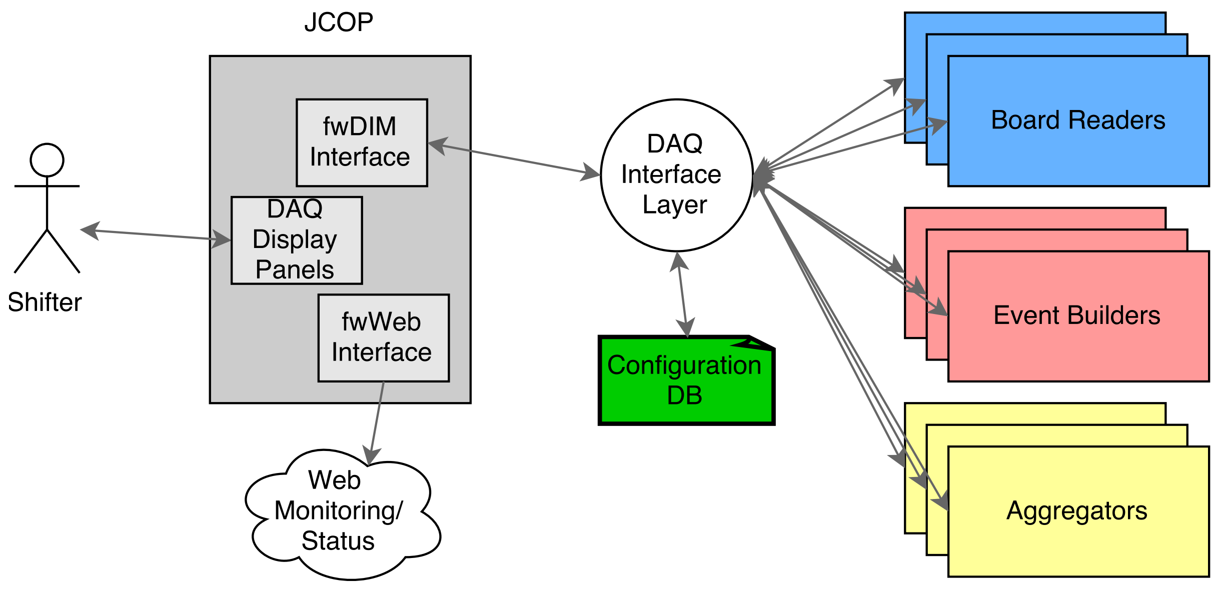

Figure 1: Sketch of the architecture of the run control system, with arrows representing state transition commands and status message reporting. A DIM service will run as part of the DAQ interface for the connection to JCOP, while the DAQ interface will send XML-RPC commands to the artdaq processes.

Thus, we envision several DIM-type devices to be defined in JCOP: generic BoardReader, generic EventBuilder, generic Aggregator, and one for the DAQ interface program. The collection of these processes can be combined together in a "DAQ process" hierarchy.

Operational Monitoring

Requirements

During the course of operations and data acquisition, we must monitor the status of both hardware and software processes, store status information over time, and provide capabilities to alarm on bad or unexpected status of both individual and groupings of components. The metrics used for this monitoring will need to be developed over time, and may change quickly during commissioning and early operations periods, and so a system that can flexibly accommodate such changes must be available. We also desire an integrated system with other slow controls/monitoring elements, to have a centralized collection of detector state data and monitoring. JCOP is perfectly capable of meeting these requirements, and has been chosen to serve as the basic detector status monitoring utility for protoDUNE SP.

Additionally, log files from each of the DAQ processes need to be collected and have some available display that can target specific components. The Elastic Stack of products, including log-stashing, searching, and visualization utilities for log messages has been previously integrated into JCOP control systems, and has been chosen as the log-monitoring suite of choice for protoDUNE SP.

DAQ Metric Monitoring tools

A “Metric Manager” exists in *artdaq* to allow for the communication of user-defined monitoring metrics, inside the DAQ code, to external monitoring programs. This is accomplished via the definition and configuration of “metric plugin” modules. These modules are registered with the Metric Manager on initialization of the *artdaq* processes (and are part of the basic configuration). Metric types and their values are reported to the Metric Manager during the course of the data acquisition with a single line of code, can be analyzed to determine statistical values over some periods of time, and are then distributed to the configured monitoring modules. Many metric plugins already exist, including reporting of metrics to a simple text file, to graphite and ganglia time-series databases, and to an EPICs control system.

To interface to the JCOP system, we plan to develop a DIM metric plugin using the existing DIM C++ API. Metrics would then be reported by DIM servers running as part of the *artdaq* processes and, using the previously mentioned JCOP framework DIM module, could be incorporated into the JCOP control system. In contrast to the control system architecture described in the previous section and illustrated in Figure 1, there would be no need for the intermediate DAQ interface: *artdaq* processes could, via the built-in Metric Managers, report directly to JCOP.

Metrics and the associated data elements in JCOP can be organized into logical devices: “hardware” status devices that can report hardware registers (collected inside *artdaq* Fragment Generator monitoring threads), “fragment generator” devices that can monitor internal software elements (incoming data rates, buffer sizes, data header values, error conditions, etc.) on the *artdaq* software interface to DAQ hardware, and more general BoardReader, EventBuilder, and Aggregator statuses (data throughputs, buffer occupancies, etc.). We will develop any necessary scripts to update data points as needed in JCOP, to allow for flexibility in the addition or change of monitoring metrics.

The JCOP “devices” described above can be combined with or declared in relationship to other devices. For example, DAQ hardware registers read by the *artdaq* processes could be associated directly with statuses of current draws for that hardware. Also, fragment generator devices could be associated to the common BoardReader device that is used for state control.

As multiple metric plugins can be run simultaneously, we will also have the ability to monitor the same metrics, or a different set, outside of JCOP should that be desired.

DAQ Process State/Status Monitoring tools

As mentioned in the Run Control section, DAQ process status and current state will be integrated into JCOP via DIM servers and clients.

DAQ Message logging and log-searching

Log files are created at the start of each *artdaq* process. Along with simple print statements in the DAQ software, *artdaq* contains two message-logging utilities: the debugging and time-stamping tool TRACE, and a more general Message Facility that allows logging and filtering of messages to configurable destinations. These messages are easily made to print log files, but messages from both utilities may also be viewed in an *artdaq* Message Viewer utility, which is based on QT, and allows for filtering of messages in the viewer based on severity.

We plan to incorporate elements of the “Elastic Stack” of products for storing, searching, and displaying log file information. The Logstash program allows for the storing and processing of log files, containing all kinds of structured or unstructured data as it is produced during a process. These logs can then be input to ElasticSearch, a search and data analytics engine, that allows for querying over unstructured or structured data and creating relevant time-series data. The results from ElasticSearch can then be visualized using the Kibana toolkit, which allows for custom results dashboards with different kinds of data visualizations. These tools have been interfaced to the JCOP framework, allowing for easier setup and configuration.

Electronic logbook

We have not yet made a decision on the final form of our electronic logbook, but currently proposed has been to use the Electronic Logbook for the Information Storage of ATLAS (ELisA).

Configuration

Requirements

The run configuration utilities for the protoDUNE SP DAQ need to satisfy the following requirements:

- provide configurations for DAQ system architecture (describing the data flow), DAQ hardware, and DAQ software;
- provide a utility for retrieving the next run number and assigning a user-defined configuration to that run;
- provide for partitioning/partial configurations of subsets of the detector;
- provide easy-to-use interfaces for viewing, modifying, and creating new configurations;
- encourage descriptive configurations by allowing for easy addition and change of configuration parameters; and,
- provide interfaces to retrieve configurations from the database per run, both online and in offline analysis.

Additionally, while the configuration management does not need to be controlled by the JCOP system, there must be a cohesive understanding of configurations by JCOP, especially in understanding what data elements are currently in use.

artdaq Configuration Management Toolkit

As mentioned above, the process management tool is configured with a list of processes, nodes on which they run, and port numbers. The *artdaq* processes then receive configuration information in the form of FHiCL-formatted (Fermilab Heirarchical Configuration Language) strings. These configurations contain both parameters for configuration of hardware devices and the software itself. The full FHiCL parameter set used for configuration can be stored as part of the raw data output file (with only small data overhead), and it becomes stored as part of the data provenance system common to the *art* event-processing framework. Thus, DAQ process configuration information is saved in all subsequent steps of data processing, and, in general, is available for extraction from the data file.

A configuration management toolkit is currently in development as part of the *artdaq* software suite to provide a comprehensive set of features that include the protoDUNE SP requirements. All configuration information can be stored in a MongoDB or file-based document system, allowing for straightforward queries of data and uncomplicated schema evolution. Global configurations will exist as a superset of sub-configurations, and a run number database will be maintained to track the history of configurations for each run. Configurations can be displayed, modified, and saved in a simple configuration viewer. Configuration information can also be used by the JCOP system to ignore unused detector elements in a number of ways, though the details of our plan to implement this are still under design. The DAQ interface will make the necessary requests for information from the configuration database, and pass configuration information along to the *artdaq* processes.

Interfaces to additional elements requiring configuration that are not part of the core DAQ processes can also be provided: they can be stored as separate configuration types in the database, and specialized configuration commands can be called by extra steps in the DAQ interface program if necessary.

References/Resources

- *artdaq*
  - *artdaq* project page: https://cdcvs.fnal.gov/redmine/projects/artdaq/wiki
  - *artdaq-demo* project page: https://cdcvs.fnal.gov/redmine/projects/artdaq-demo/wiki
  - Biery, K. et al. artdaq: An Event-Building, Filtering, and Processing Framework. *IEEE TNS* **60** 3764-3711 (2013).
    - https://cdcvs.fnal.gov/redmine/attachments/download/10253/artdaq_IEEE_TNS_Final.pdf
- JCOP
  - O Holme, et al. The JCOP framework *Proc. of the 10th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2005) Geneva, Switzerland 2005*
    - https://accelconf.web.cern.ch/accelconf/ica05/proceedings/pdf/O3_005.pdf
  - JCOP Framework project page: https://wikis.web.cern.ch/wikis/display/EN/JCOP+Framework
  - WinCC OA project page: https://wikis.web.cern.ch/wikis/display/EN/WinCC-OA+Service
- DIM
  - https://dim.web.cern.ch/dim/
  - PVSS (WinCC OA)-DIM manual: http://lhcb-online.web.cern.ch/lhcb-online/ecs/fw/fw_dim_description.html
- Elastic Stack
  - https://www.elastic.co/products
- ELisA (Electronic Logbook)
  - Radu, A. C. et al. The Electronic Logbook for the Information Storage of ATLAS Experiment at LHC (ELisA). *Journal of Physics: Conference Series*. **396** 012014 (2012).
    - http://iopscience.iop.org/article/10.1088/1742-6596/396/1/012014/meta