DAQ Software Framework

Kurt Biery

18 June 2021

Conceptual Design Review of the Single-Phase Vertical Drift DAQ

Context

1. This talk expands on the ‘Core Software Components’ that Alessandro mentioned in the previous talk.

OS: Centos7/Scientific Linux 7

Basic tools: gcc, python, cmake…

External libraries and tools: boost, highfive, felix,…

DAQ – core components

- i.e. packages required to build modules and applications
- Logging, error reporting, application framework

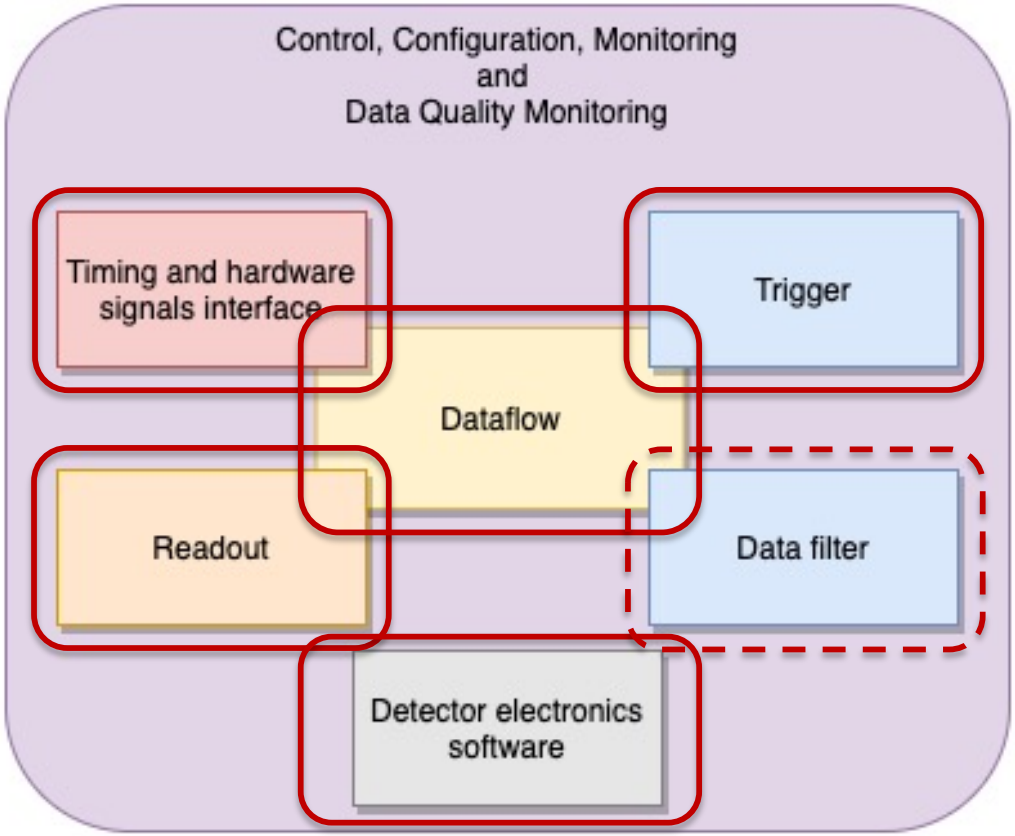
DAQ / Detector – application components

2. I realize that this is a *Conceptual* Design Review, but initial versions of a number of the core software components are already available.
3. We believe that the majority of the core software will be common between the Vertical Drift and Horizontal Drift DAQ systems.

Introduction

The software framework provides services and infrastructure for

- control, configuration, and monitoring of DAQ applications, and
- the development of software modules to run in DAQ applications

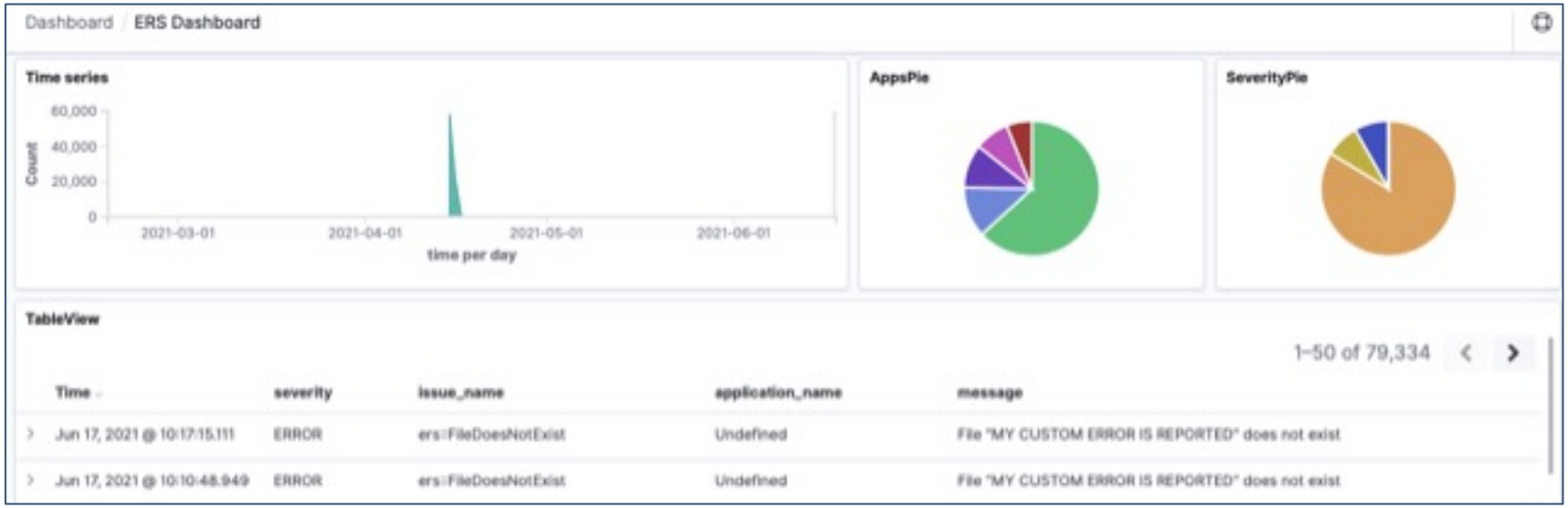


The goal of the framework is to support developers in the working groups.

CCM

CCM provides a set of services and interfaces that are used by DAQ applications to allow them to participate in a coherent system, from the points of view of control, configuration and monitoring.

CCM also provides the user interfaces, allowing humans to interact with the DAQ system.



Control

Run control application

- User interface to control state transitions; e.g. start & stop runs
- Console-based app currently; GUI-based app in the future

```
  waiting ──────────────────────────────────────────── 100% 0:00:00 0:00:01
[11:50:50] INFO     Sending resume to trigger (http://mu2edaq13:3333/command)
           INFO     Response: <Response [202]>
           INFO     Received reply from trigger to resume
                          Apps
```

name	host	alive	pings	last cmd	last succ. cmd
dataflow	mu2edaq13	True	True	start	start
hsi	mu2edaq13	True	True	start	start
ruemu0	mu2edaq13	True	True	start	start
trigger	mu2edaq13	True	True	resume	resume

```
⁝ waiting ──────────────────────────────────────── 73% 0:00:17 0:00:43
```

Command distribution within DAQ applications

- Developers write the code that gets executed when specific commands are received. The framework handles the delivery of the commands.

Configuration

User interfaces to assist users in specifying DAQ system layout and individual DAQ module configuration, including electronics configuration.

Software tools to allow developers to specify and make use of the configuration parameters that delivered to their modules at runtime.

Example in a few slides…

Monitoring

Logging

- A standard logging interface, both for centralized reporting and local debug messages.
- Tools to view and correlate messages.

Performance metrics

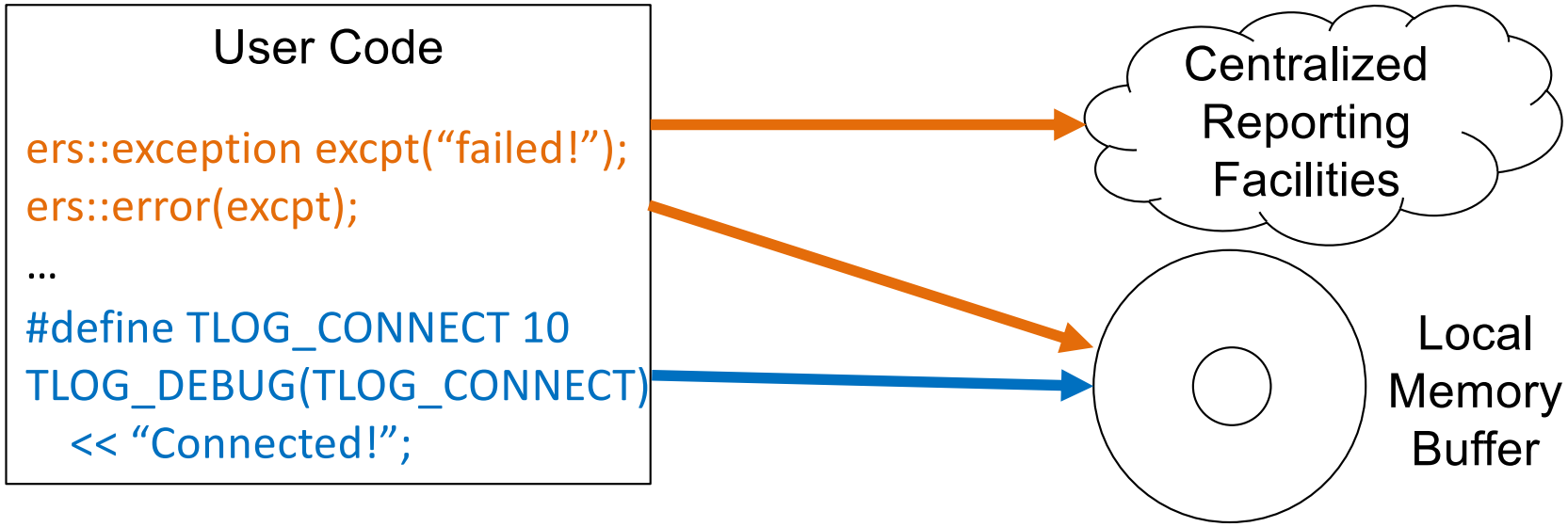
- A standard model for defining and reporting performance metrics and the software libraries to support them.
- Tools to view and correlate these.

Graphical display of metrics

Standardized logging and error reporting

The framework provides an integrated package that combines the ATLAS Error Reporting System and Fermilab TRACE

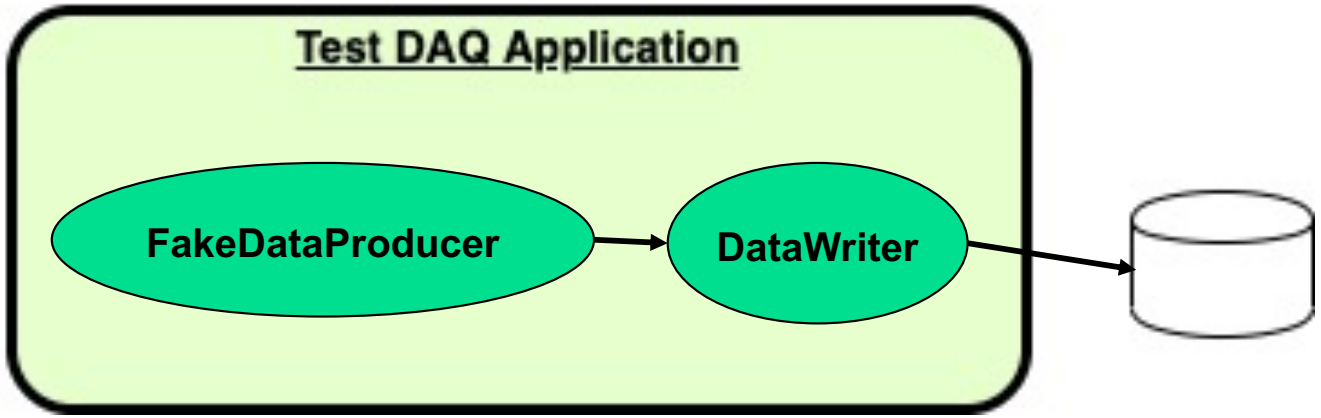
- C++ exceptions and ERS-style macros to report errors, warning, and informational messages
- TRACE-style macros to report debug messages
- Appropriate routing based on runtime settings



(SW) Module development tools

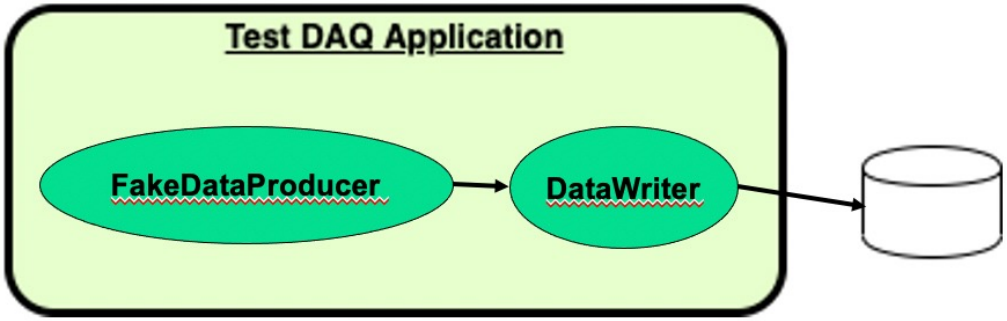
C++ classes that support the development of *DAQModules*.

- These modules are the components that developers create to implement the desired Timing, Readout, Trigger, Dataflow, and electronics communication functionality.
- They communicate via Queues or Inter-Process Messaging tools that are provided by the framework.



Simple module and queue configuration

```
{ // pseudo-JSON
  "queues": [
    {"kind": "FollySPSCQueue",
     "capacity": 100,
     "queue_name": "trigger_record_q"
    }
  ],
  "modules": [
    { "name": "FakeDataProducer",
      "plugin": "FakeProducerModule",
      "queue_infos": [
        {"direction": "output",
         "queue_name": "trigger_record_q",
    },] },
    { "name": "DataWriter",
      "plugin": "DataWriterModule",
      "queue_infos": [
        {"direction": "input",
         "queue_name": "trigger_record_q",
    },] }
  ]
}
```



Framework-based apps so far

Readout:

- WIB data emulator; readout module; sample SNB local storage

Timing:

- Interfaces to electronics; emulated hardware signals interface

Trigger:

- Multiple levels of processing, including module-level trigger

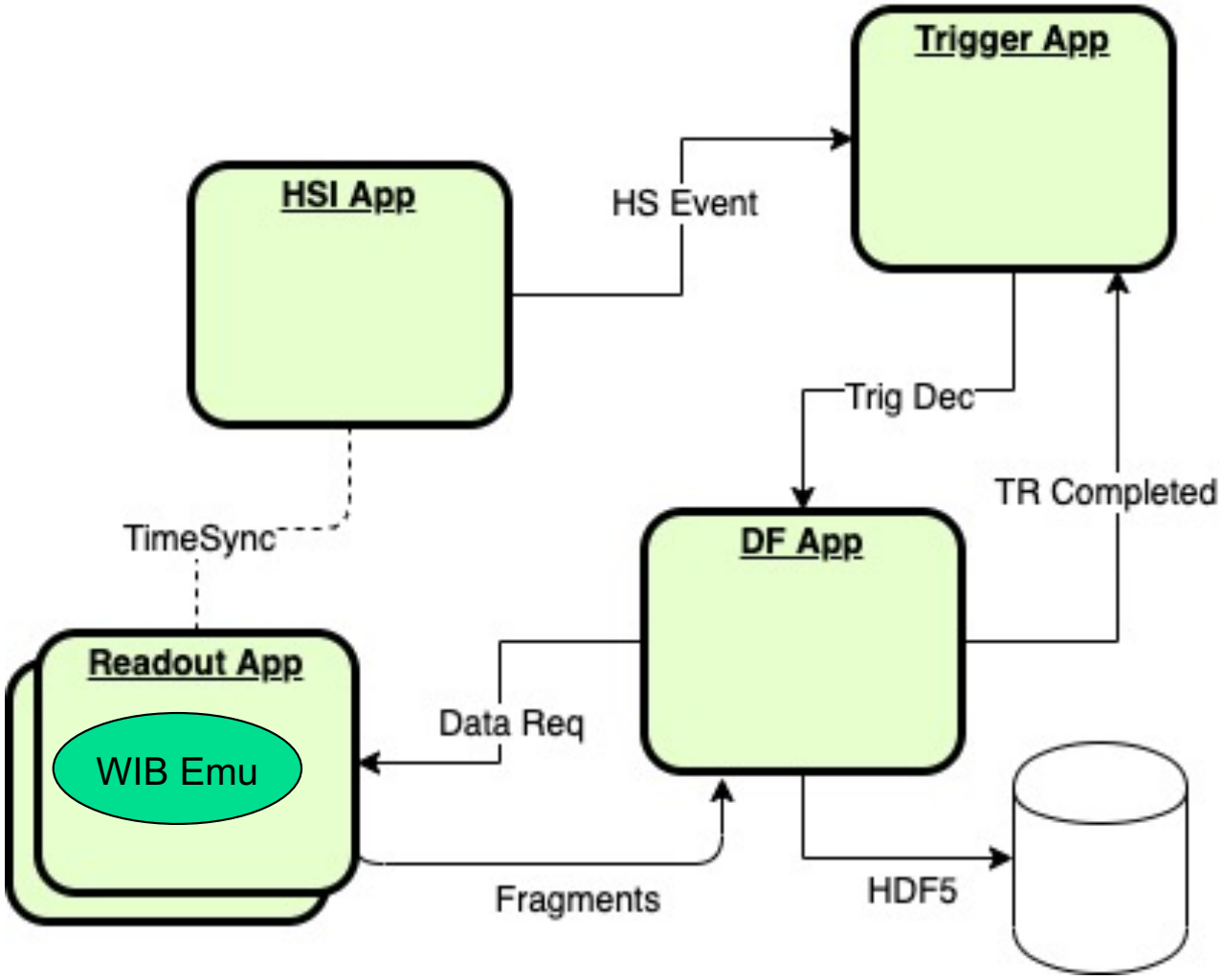
Dataflow:

- Event building; data storage

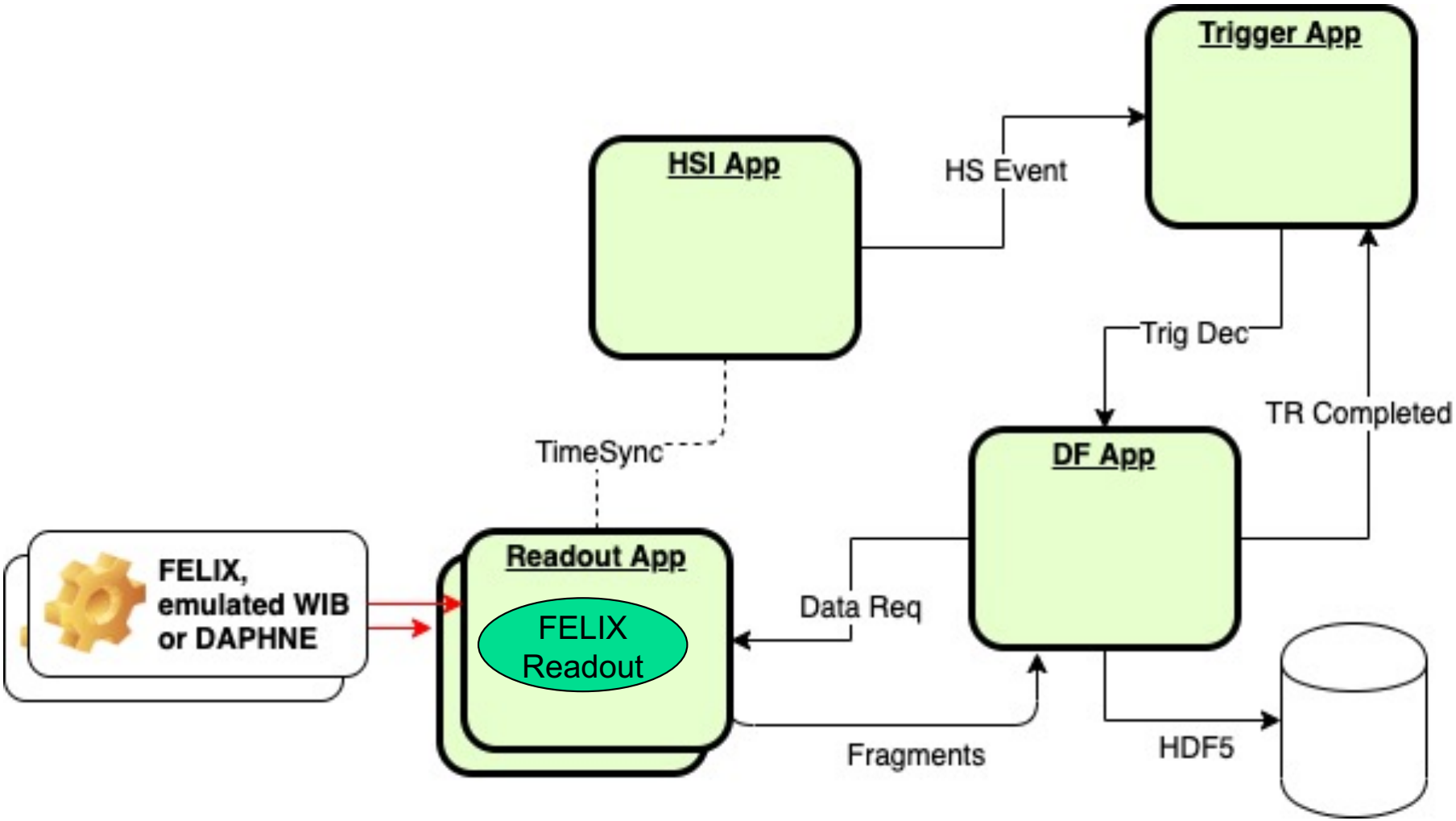
Electronics debugging:

- Configure, control, and monitor emulated WIB2 electronics

MiniDAQ system with HW emulators



MiniDAQ system with FELIX card



Status and plans

A solid core of module development functionality (application framework) already exists.

CCM components are in an early stage of prototyping.

The DAQ software development team is small, and this means that developments are prioritized and carried out somewhat sequentially.

The aim is to have a complete system ready for ProtoDUNE II.

Intermediate versions of the DAQ will add incremental improvements and new features in line with what is needed for the integration with the VD detectors electronics. Giovanna's 'development plan' presentation later today will talk more about this.