

# THE EUDAQ DATA ACQUISITION SYSTEM.

## Introduction to Concepts and Integration

```
#include "eudaq/Utils.hh"
#include "eudaq/OptionParser.hh"
#include "eudaq/ExampleHardware.hh"
#include <string>
#include <vector>

// A name to identify the raw data format of the events generated
// Modify this to something appropriate for your producer.
static const std::string EVENT_TYPE = "Example";

// Declare a new class that inherits from eudaq::Producer
class ExampleProducer : public eudaq::Producer {
public:
    // The constructor must call the eudaq::Producer constructor with the name
    // and the runcontrol connection string, and initialize any member variables.
    ExampleProducer(const std::string & name, const std::string & runcontrol)
        : eudaq::Producer(name, runcontrol),
          m_run(0), m_ev(0), stopping(false), done(false) {}

    // This gets called whenever the DAO is configured
    virtual void OnConfigure(const eudaq::Configuration & config) {
        std::cout << "Configuring: " << config.Name() << std::endl;

        // Do any configuration of the hardware here
        // Configuration file values are accessible as config.Get(name, default)
        m_exampleparam = config.Get("Parameter", 0);
        std::cout << "Example Parameter = " << m_exampleparam << std::endl;
        hardware.Setup(m_exampleparam);

        // At the end, set the status that will be displayed in the Run Control.
        SetStatus(eudaq::Status::LVL_OK, "Configured (" + config.Name() + ")");
    }

    // This gets called whenever a new run is started
    // It receives the new run number as a parameter
    virtual void OnStartRun(unsigned param) {
        m_run = param;
        m_ev = 0;
        std::cout << "Start Run: " << m_run << std::endl;

        // It must send a BORE to the Data Collector
        eudaq::RawDataEvent bore(eudaq::RawDataEvent::BORE(EVENT_TYPE, m_run));
        // You can set tags on the BORE that will be saved in the data file
        // and can be used later to help decoding
        bore.SetTag("EXAMPLE", eudaq::to_string(m_exampleparam));
        // Send the event to the Data Collector
        SendEvent(bore);

        // At the end, set the status that will be displayed in the Run Control.
        SetStatus(eudaq::Status::LVL_OK, "Running");
    }
};
```

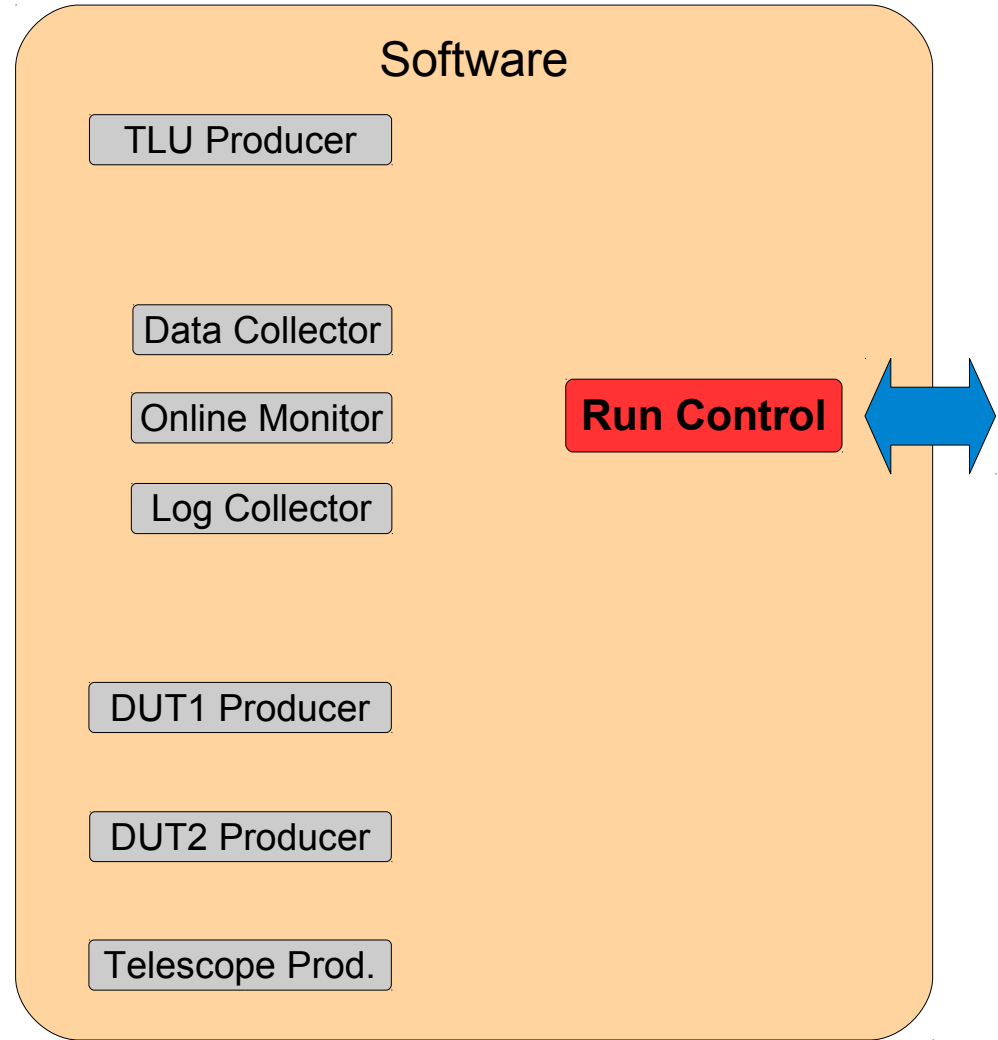
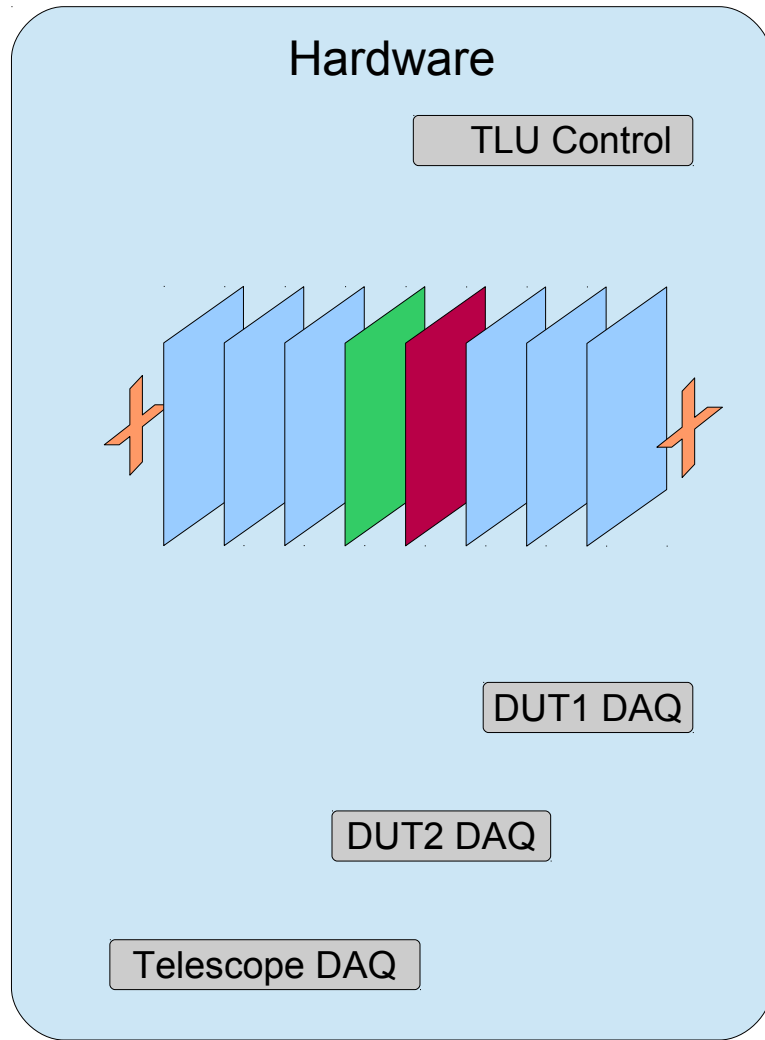
Simon Spannagel  
EUTelescope Workshop  
DESY Hamburg, March 25-27 2013

# Features of the EUDAQ Data Acquisition System

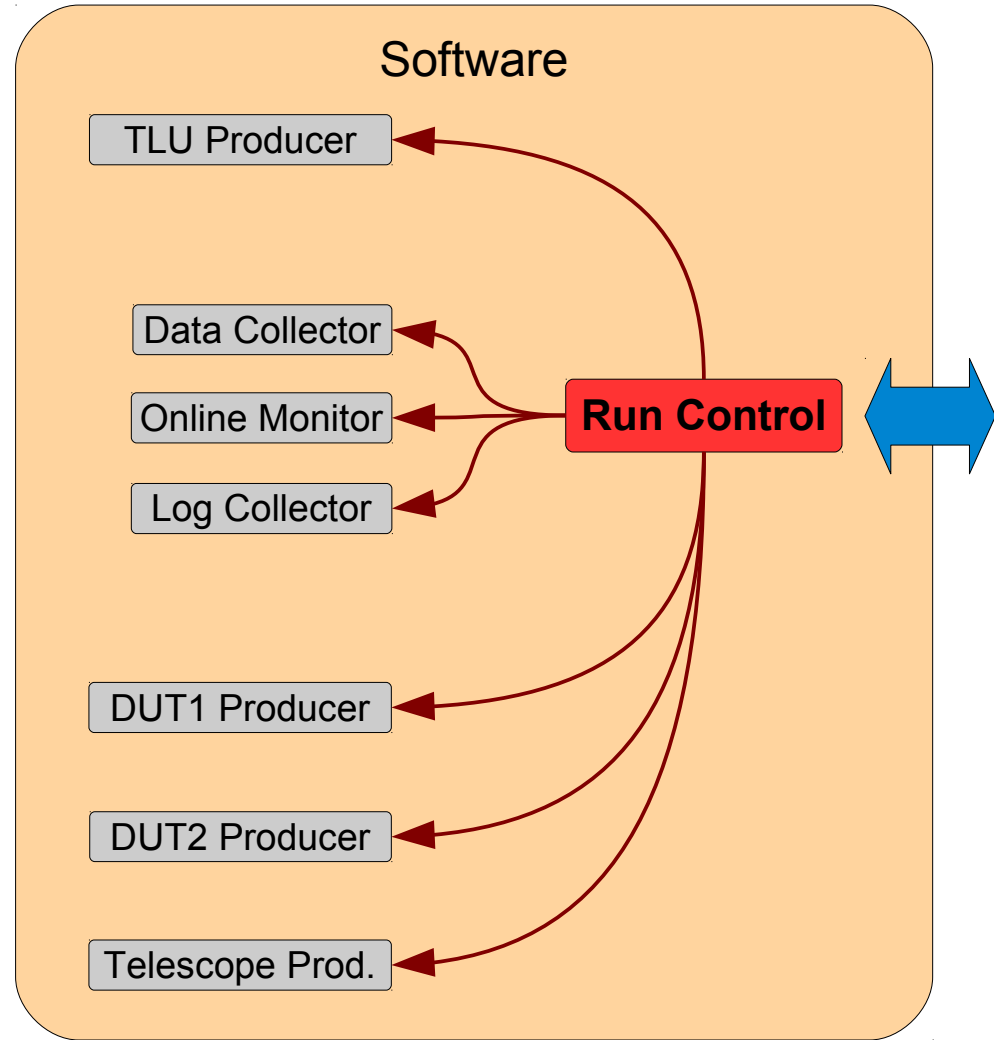
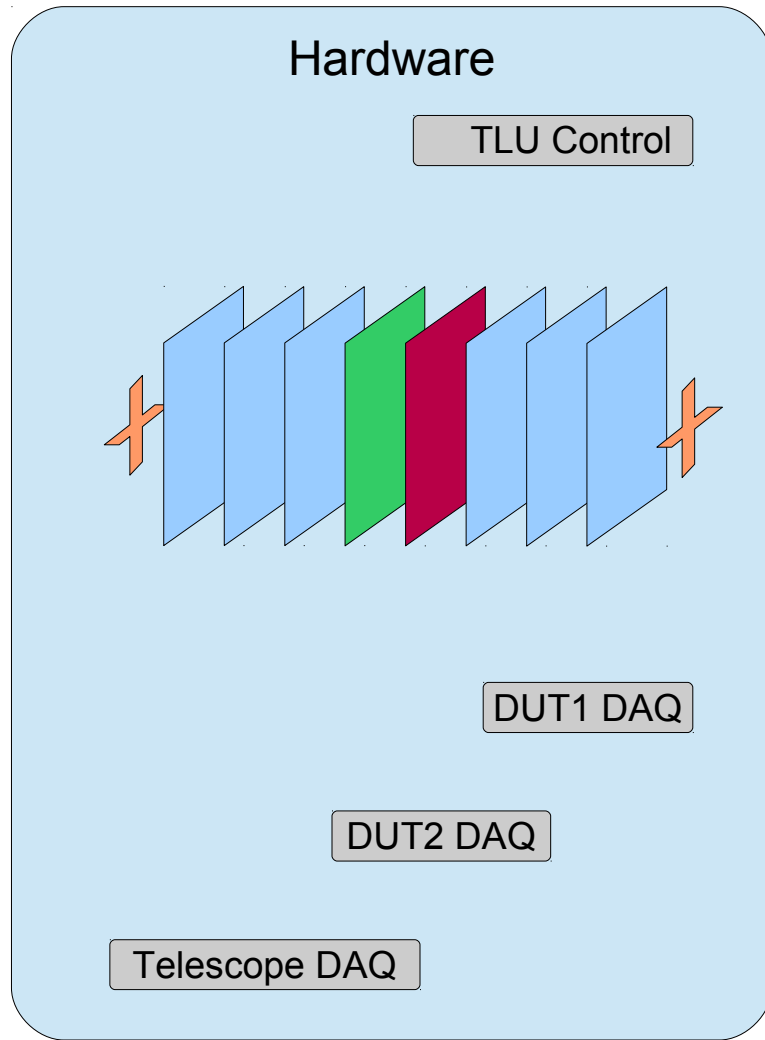
- Generic framework for data acquisition
- OS independent: Linux, Mac OSX, (Windows)
- Allows full integration of device under test (DUT) independent of its technology including pre-existing DAQ systems
- Modular and flexible design
- DAQ control via GUI, but CLI interface also available
- Online DQM using the OnlineMonitor
- All hardware communication done by “Producers” with equal rights
  - e.g. “NiProducer” for the MIMOSA26 sensors via National Instruments Crate
- Used by many groups:
  - Altro (Bonn), APIX (Atlas Pixels), Atlas (TRT), CMS Pixel (DESY), DEPFET (Bonn), FORTIS/SPIDER (Bristol), MimoRoma (INFN), MVD (DESY), PixelMan (Freiburg), SITRA (Santander), Taki (Mannheim), Timepix (Bonn)
  - and more (NA62, Alfa, Alice, etc.) ...



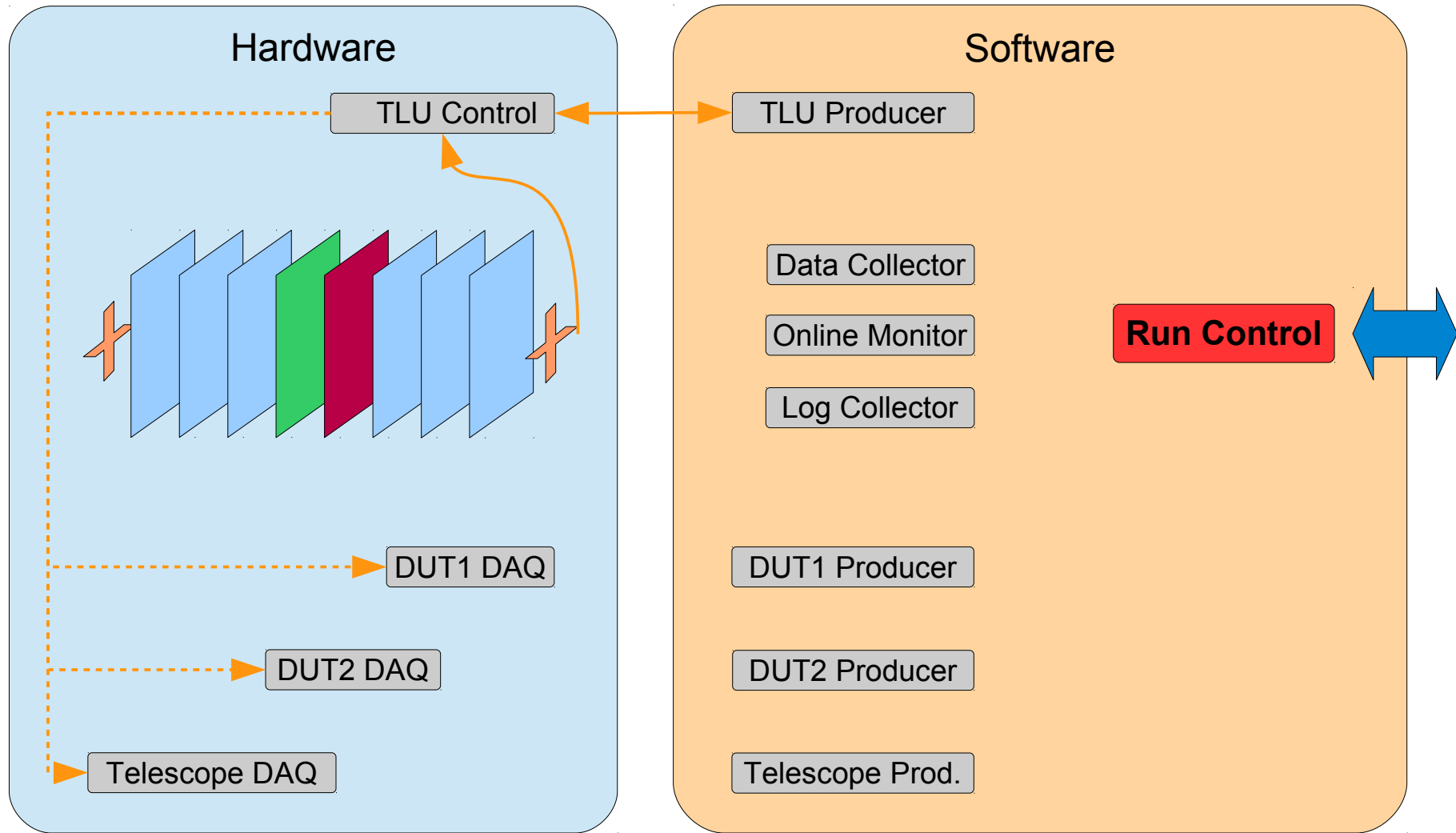
# The EUDAQ System Architecture



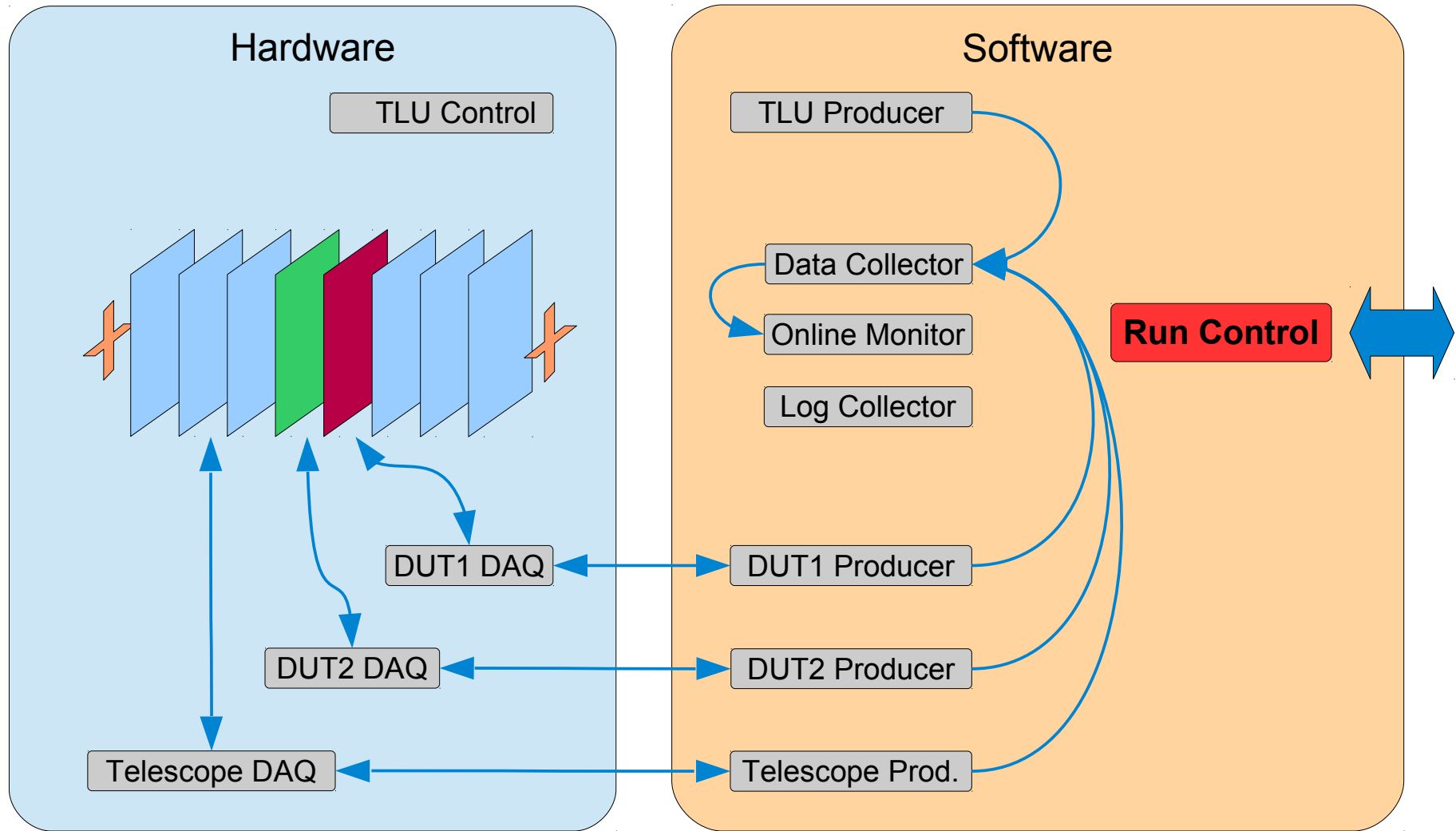
# The EUDAQ System Architecture



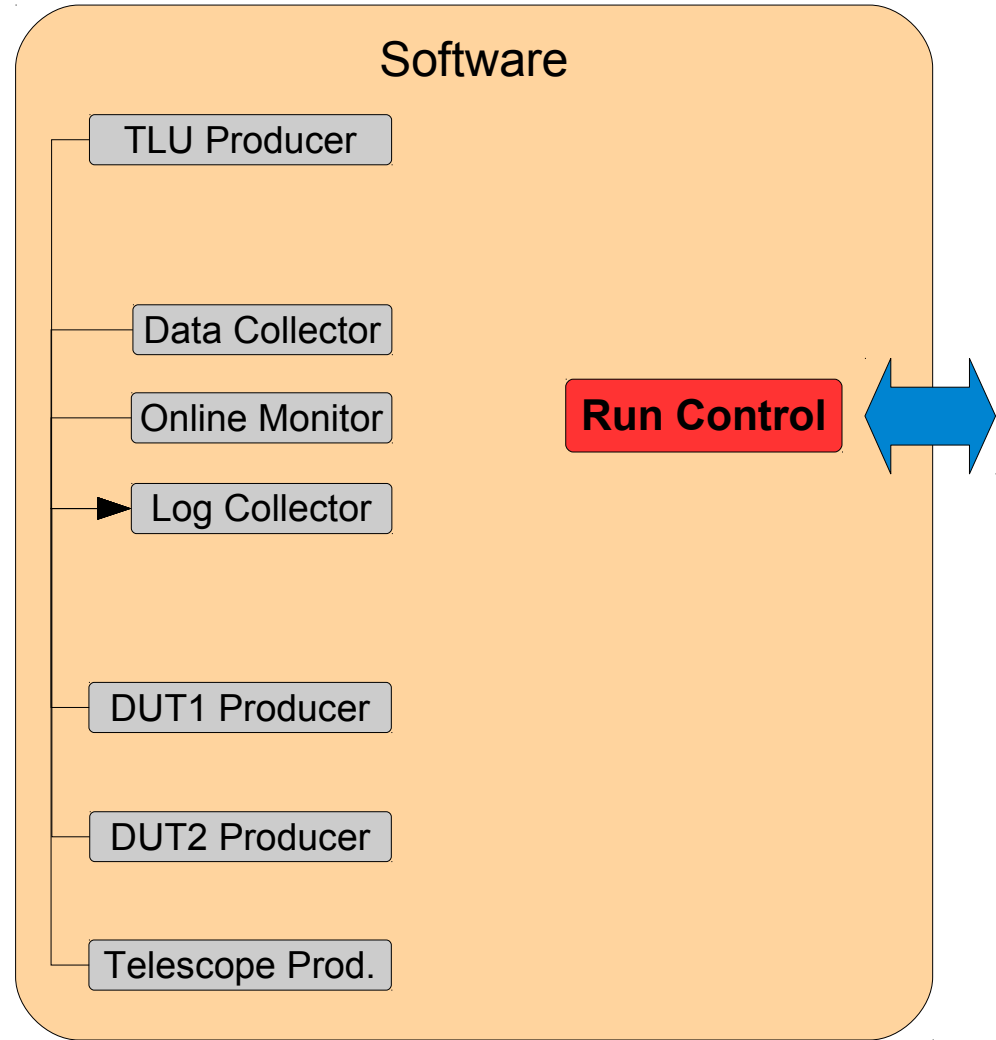
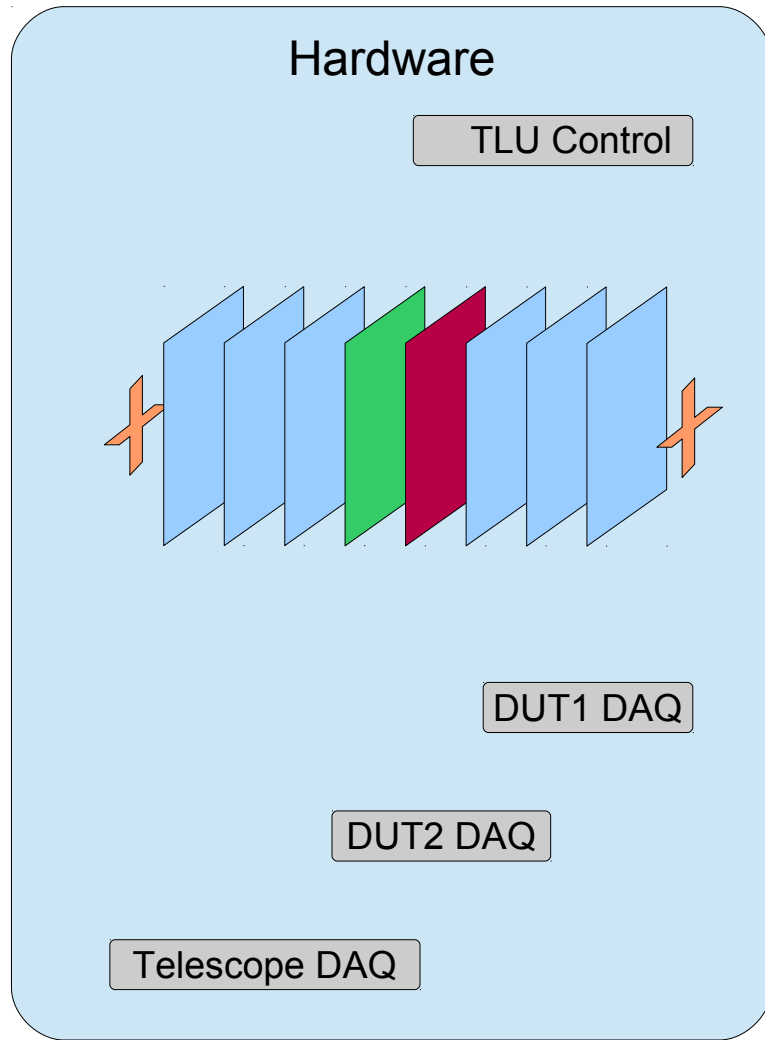
# The EUDAQ System Architecture



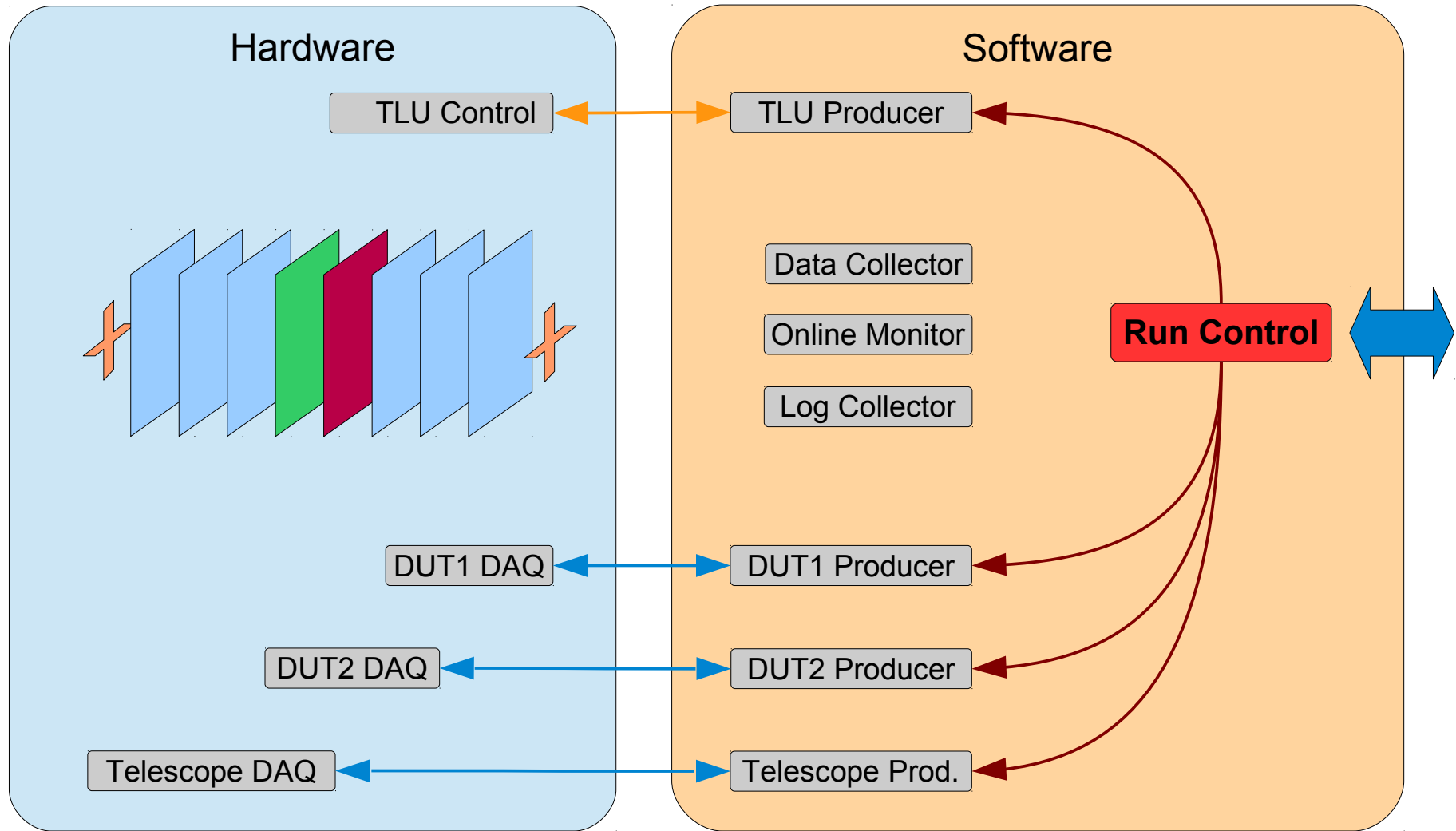
# The EUDAQ System Architecture



# The EUDAQ System Architecture



# The EUDAQ System Architecture





# Running EUDAQ

- All components communicate via TCP stack
  - can be run on different networked machines
- User interaction via Run Control
- Processes:
  - Run Control
  - Log Collector
  - Data Collector
  - Producers for hardware communication (e.g. TLU, Telescope, DUT)
  - [Online Monitor]
- Start scripts are provided which fire up all necessary processes



# Running EUDAQ

### eudaq Run Control

Control

Config:

Run:

Log:

GeoID:

---

Status

Run Number: 1569      Events Built: 0

Rate:                    Triggers: 0

File Bytes: 1209 B      Particles: 0

TLU Status: --,00,--,00,--, (0,0)      Scalers: 0, 0, 0, 0

---

Connections

type	name	state	connection
DataCollector		OK	192.168.2.3:55187
LogCollector		OK	192.168.2.3:55182
Monitor	OnlineMon	OK	192.168.2.3:55196
Producer	TLU	OK: Started	192.168.2.3:55194
Producer	MimosaNI	OK: Started	192.168.2.3:55191

---

#### Data Collector

```

Connect: Producer.MimosaNI (192.168.2.3:49940)
Connect: Producer.TLU (192.168.2.3:49944)
Configuring (ni_cms_coins)...
...Configured (ni_cms_coins)...
Configuring (ni_cms_coins)...
...Configured (ni_cms_coins)...
Complete Event: 1563,0
    
```

### EUDET Telescope Online-Monitor 1.0beta2I

- MMOSA26
- Correlation
- MMOSA26 0
- MMOSA26 1
- MMOSA26 2
- MMOSA26 3
- MMOSA26 4
- Monitor Performance
- EUDAQ Monitor

run: 1542      Curr. event: 15276      Analysed events: 15277

### EUDAQ Log Collector

Level:  From:  Search:

Received	Sent	Level	Text	From	File	Function
11:16:56.721	11:16:56.721	4-INFO	Connection from Producer.TLU (192.168.2.3:49944)	DataCollector	DataCollecto...	OnConnect(const eudaq::ConnectionInfo&)
11:16:56.722	11:16:56.721	4-INFO	Connection from Monitor.OnlineMon (192.168.2.3:57631)	LogCollector	euLog.hh:97	
11:20:11.874	11:20:11.874	4-INFO	Configuring (ni_cms_coins)	RunControl	RunControl.c...	Configure(const std::string&, int)
11:20:14.205	11:20:14.205	4-INFO	Configured (ni_cms_coins)	Producer.TLU	TLUProducer.c...	OnConfigure(const eudaq::Configuration&)
11:33:17.380	11:33:17.380	4-INFO	Configuring (ni_cms_coins)	RunControl	RunControl.c...	Configure(const std::string&, int)
11:33:19.762	11:33:19.762	4-INFO	Configured (ni_cms_coins)	Producer.TLU	TLUProducer.c...	OnConfigure(const eudaq::Configuration&)
11:33:36.477	11:33:36.477	4-INFO	Configured (ni_cms_coins)	Producer.Mim...	NiProducer.c...	OnConfigure(const eudaq::Configuration&)
11:33:36.736	11:33:36.736	4-INFO	Configured (ni_cms_coins)	Producer.Mim...	NiProducer.c...	OnConfigure(const eudaq::Configuration&)
11:33:52.635	11:33:52.635	4-INFO	Starting Run 1569:	RunControl	RunControl.c...	StartRun(const std::string&)
11:33:53.637	11:33:53.637	4-INFO	Preparing for run 1569	DataCollector	DataCollecto...	OnPrepareRun(unsigned int)
11:33:59.651	11:33:59.651	4-INFO	Starting run 1569	Monitor.Onlin...	Monitor.cc:68	OnStartRun(unsigned int)

## ➤ Basic EUDAQ event data format: RawDataEvent

- Generic container for unaltered, encapsulated detector response
- Data input: raw block of memory or vector
- Storage of additional information possible (custom tags, trigger numbers...)
- Correct data decoder is chosen by unique identifier string for each producer

## ➤ Alternative: StandardEvent

- Decoded detector data
- Consists of StandardPlanes with set pixel dimensions of the respective detector
- Can be read by e.g. the Online Monitor for direct processing

## ➤ Final analysis in EUTelescope: LCIO (Linear Collider I/O)

- Usually done within EUTelescope



# Integrating a DUT into EUDAQ

## > DUT Producer

- Talk to the DAQ hardware, receive events from there
- Receive commands from Run Control:  
OnConfigure, OnStartRun, OnStopRun, Terminate
- Send data to the Data Collector (either RawDataEvent or StandardEvent)
- Configure itself with parameters received before data taking
- Send log messages to the Log Collector

## > [DUT DataConverterPlugin]

- Convert the specific native detector data into StandardEvents
- Needed e.g. for online monitoring of the DUT
- Can be used to convert into LCIO for the final analysis using EUTelescope

> Example code showing the usage of the base classes is provided.



# Summary

## > EUDAQ

- offers a modular and flexible framework for data acquisition
- is well documented (see below)
- is used and supported within AIDA

> Both simple and full integration of a DUT and its DAQ possible (only producer or additional DataConverterPlugin)

> Also usable for detector DAQ w/o AIDA telescopes

> Future plans: add slow control elements for beam monitoring

- Enables e.g. the storage of beam energy settings into the respective data files

> SVN repository: <http://eudaq.hepforge.org/svn/trunk>

> Extensive documentation (~60p) available in SVN, build PDF using

- `cd doc/ && make manual`

