DAQ software & developments for the EUDET Calorimeter

EUDET

Tao Wu

On behalf of UK Collaboration

Imperial College London

MANCHESTER 1824

Royal Holloway University of London

UCL

UNIVERSITY OF CAMBRIDGE

Outline

- DAQ targets
- Software skeleton of DOOCS
- Recent developments of DAQ software
- Future plans

EUDET DAQ Targets

- Aiming at generic DAQ system, e.g. ECAL/HCAL
- Provide well-defined interfaces between DAQ components to allow for minimizing costs and development cycles;
- A control system to easily integrate the rest of sub-systems of detectors
- A software to build events from bunch train data and disparate sources into single event data
- Manage network and data storage

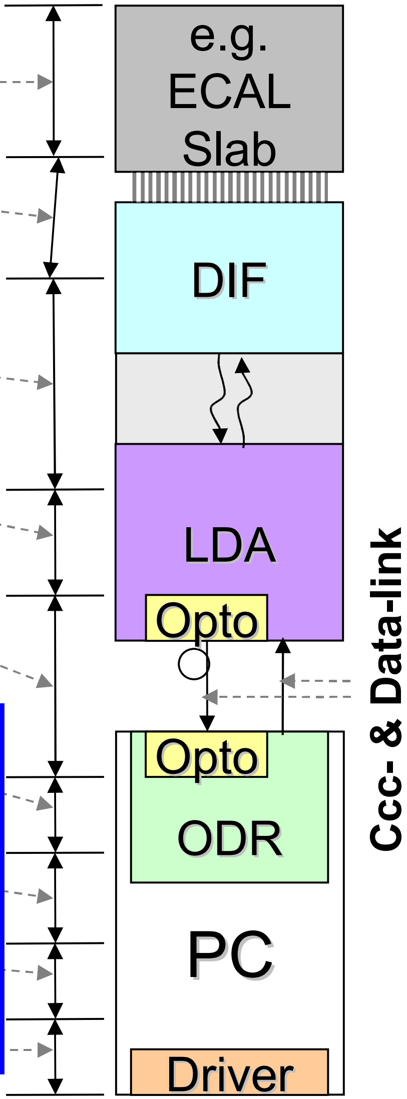
DAQ Software Design Targets

- Targeting at EUDET Modules, e.g. for ILC;
- Triggerless DAQ system: use C&C instead
  - All data are sent off detector within a bunch train
  - Use bunch struct. as advantages: 1ms in bunch train, read out data within 200ms of inter-train gap;
- DAQ system will also control power cycling of readout ASICs
- A funnel-like DAQ to collect, wrap and transmit data in stages before sending to central storage

DAQ Architecture Overview

- **Slab hosts VFE chips (ASICs)**
- **Detector Interface** connected to Slabs
- LDA servicing DIFs
- **Link/Data Aggregator**
- LDAs read out by ODR via opto-links
- **Off-Detector Receiver**
- PC hosts ODR
  - PCI-Express driver software
- Local Software DAQ
- Fully-bloomed Software

Covered by Valeria JRA3



DAQ Software: **DOOCS framework**

User Application layer

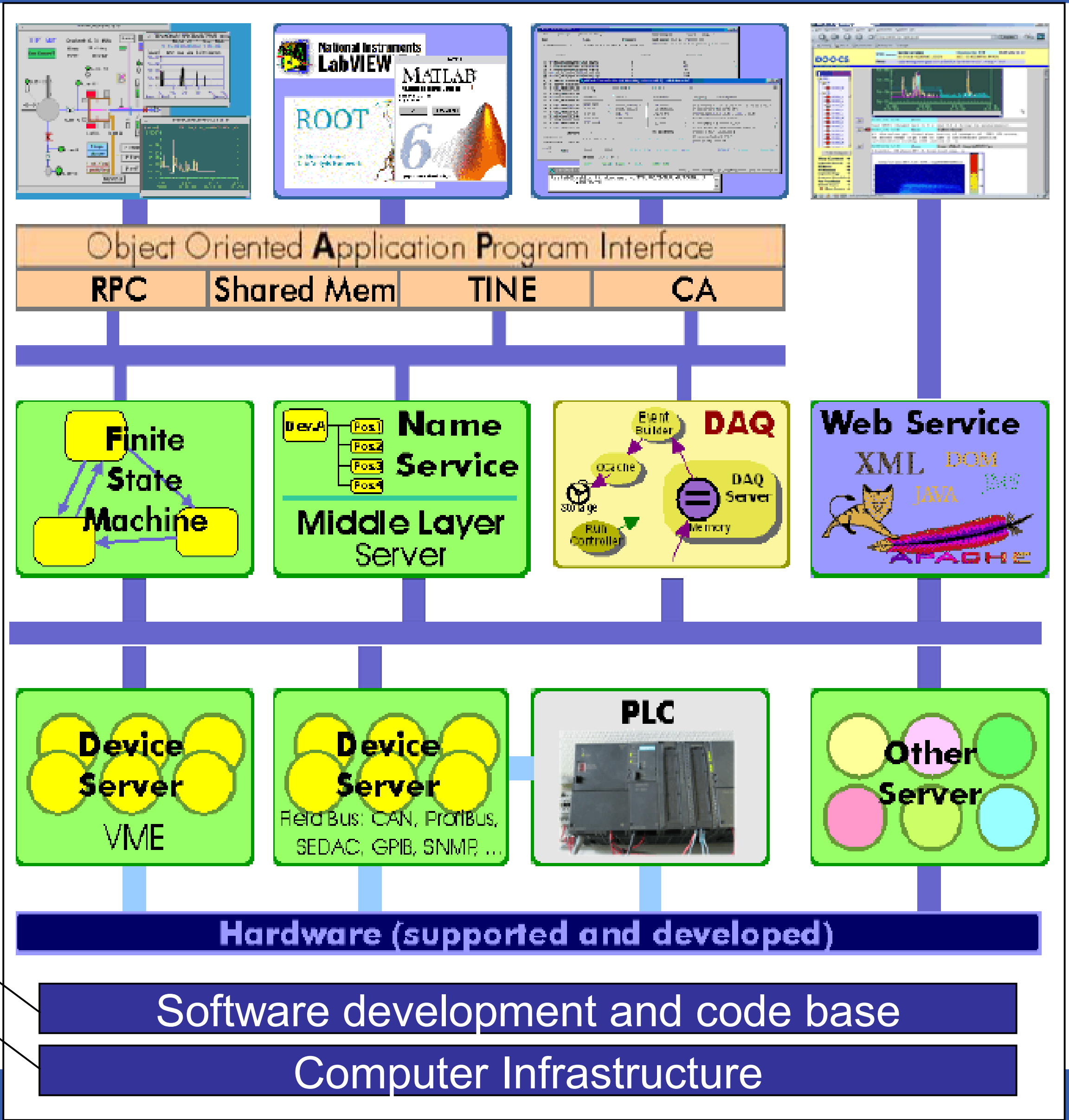
Communication

Middle layer

Hardware Interface Layer

Software Libs

Sun/Linux Cluster



Object Oriented Application Program Interface

RPC | Shared Mem | TINE | CA

Finite State Machine

Name Service

Middle Layer Server

DAQ

Web Service

XML DOM JAVA

Device Server

VME

Device Server

Field Bus: CAN, ProfiBus, SEDAC, GPIB, SNMP, ...

PLC

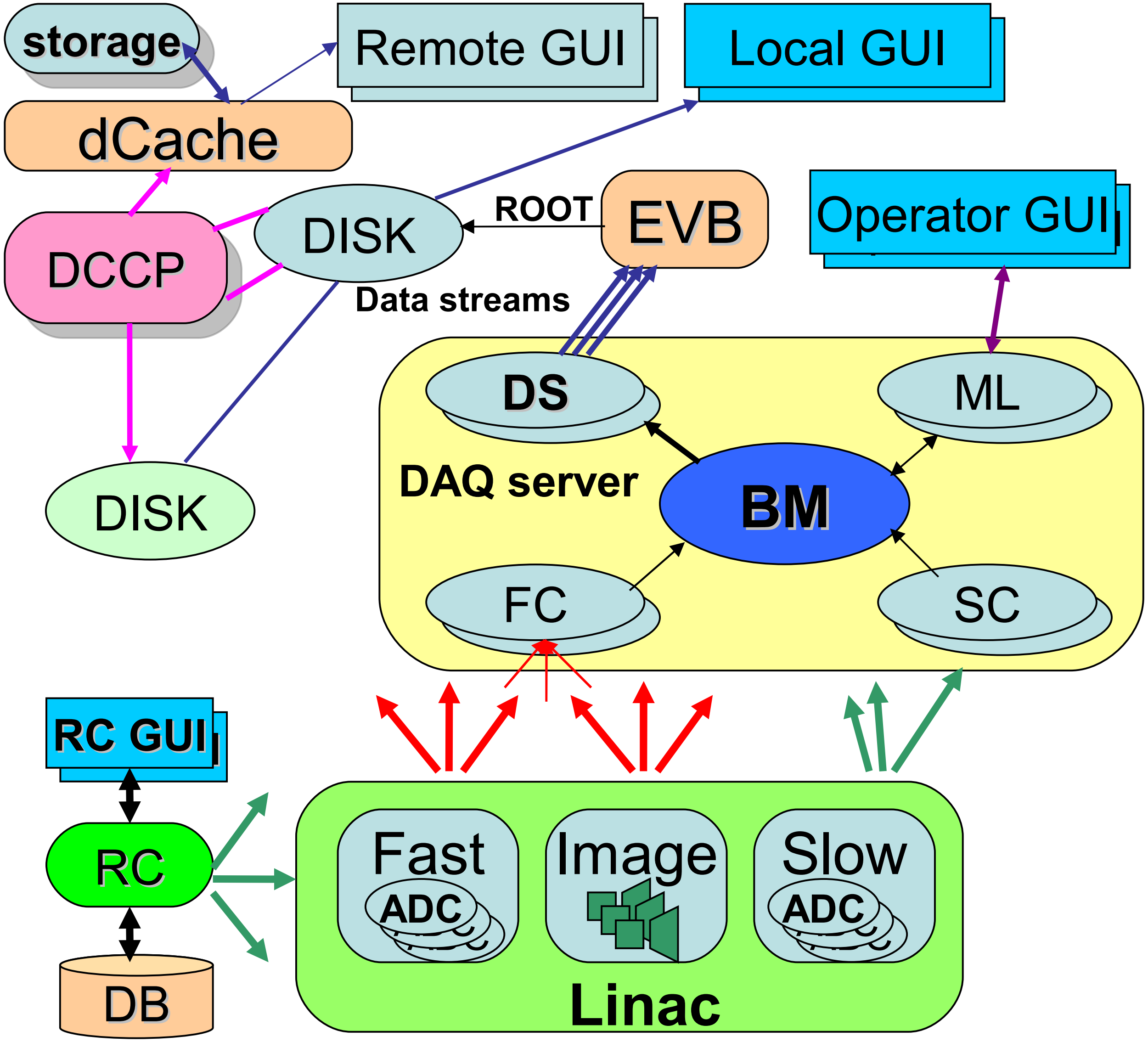
Other Server

Hardware (supported and developed)

Software development and code base

Computer Infrastructure

DAQ Software of DOOCS



FC/SC:
Fast/Slow
Collector

BM:
Buffer Manager

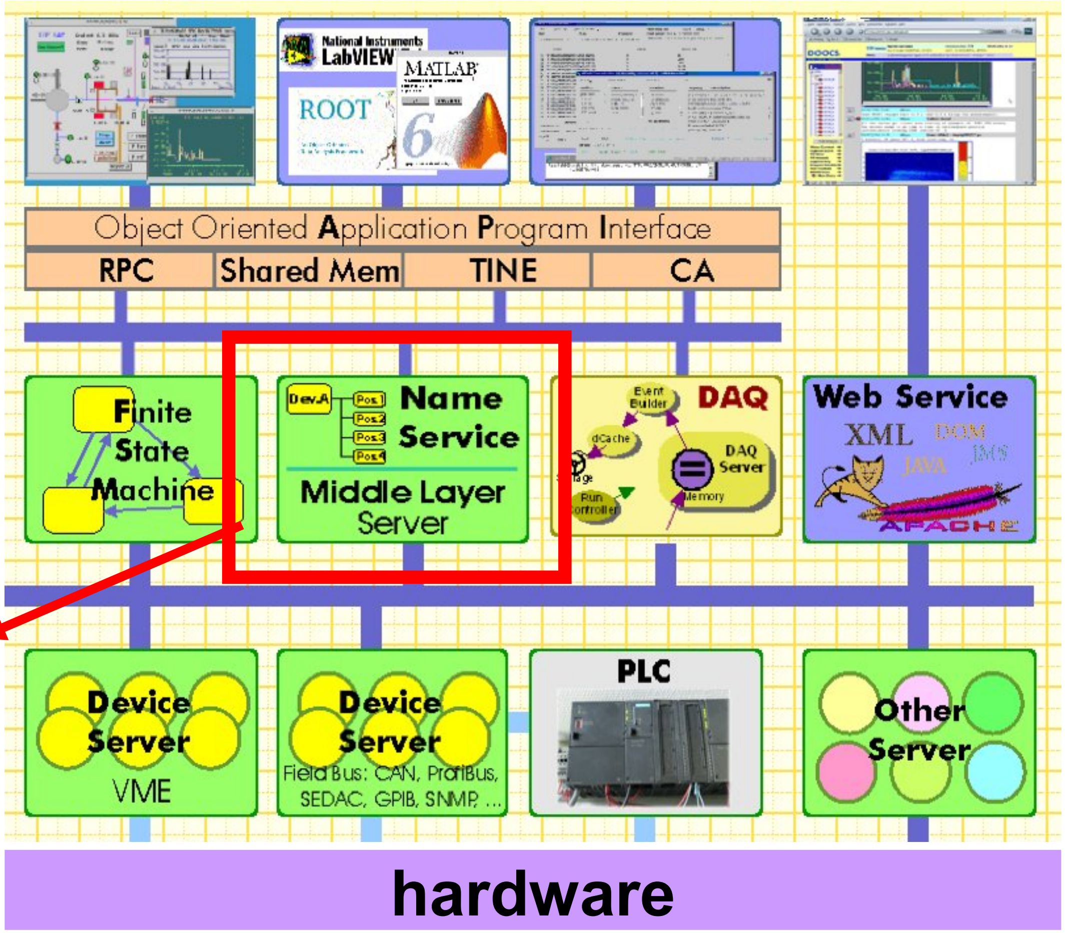
EVB:
Event Builder

DS:
DAQ Server

Adapting DOOCS to EUDET DAQ

- Modeling hardware card via device server
  - Existing: ODR device server ***Odr_server***
- **Equipment Name Server** (ENS):
  - Facility(**F**) / Device(**D**) / Location(**L**) / Property (**P**)
  - e.g. *CALICE.ECAL*/*ODR*/*ODR1*/*Status*
    - F: CALICE.ECAL, CALICE.AHCAL, CALICE.DHCAL
    - D: ODR, LDA, DIF, ASICs;
    - L: ODR1,ODR2,ODRX; LDA1,LDA2,LDAX; DIF1,DIF2,DIFX;
    - Property: X X X ?
- An interface talking to ODR has been built;
- To classify all properties and functionalities of each device for our EUDET DAQ system (DIF, LDA, ODR, etc.) is ongoing!

ENS naming service



User Interface

Communi-cation layer

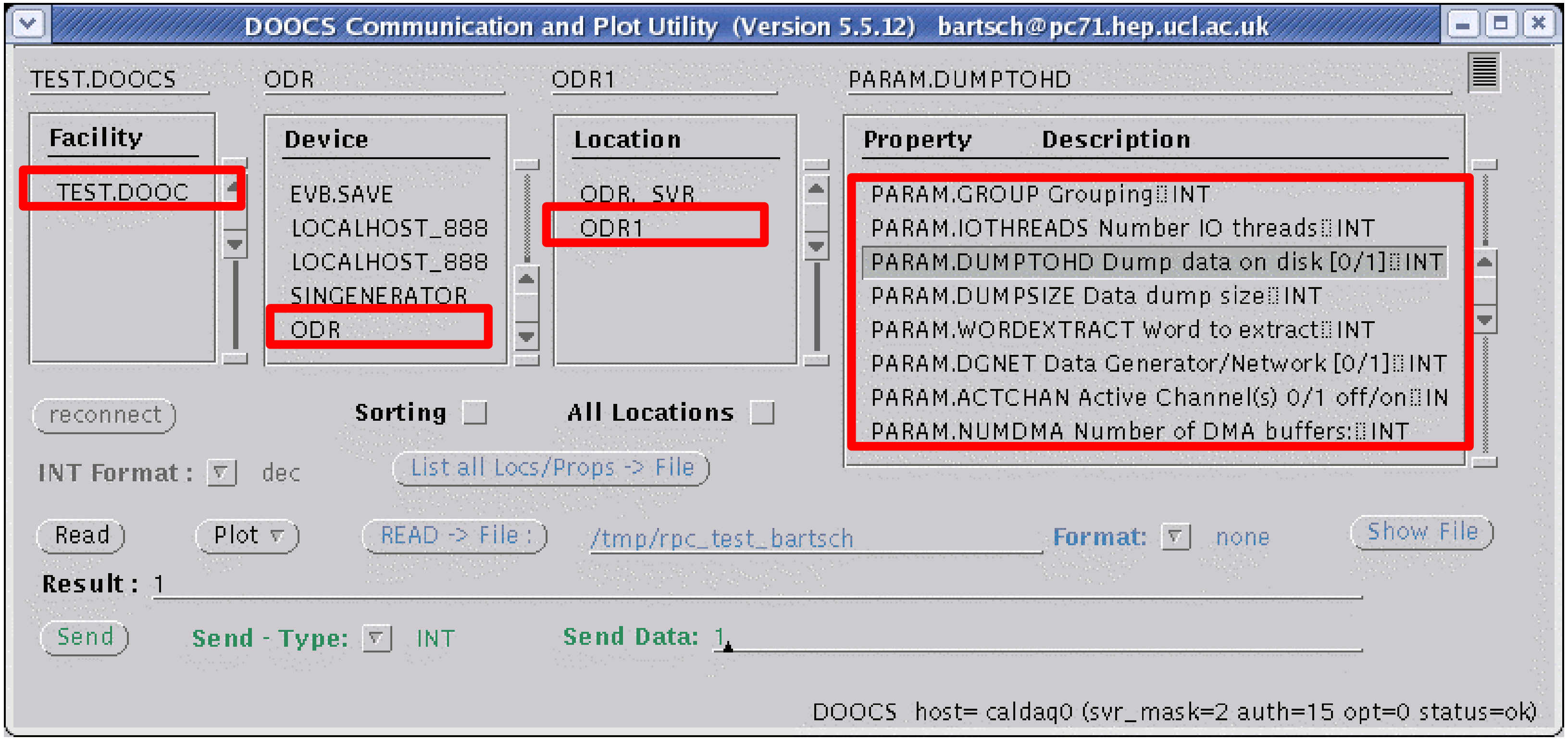
Middle Layer

Hardware interface

Provided by DOOCS and already in use for RPC communication between client and server

ENS naming service

Screenshot of the rpc_util GUI



DOOCS Communication and Plot Utility (Version 5.5.12) bartsch@pc71.hep.ucl.ac.uk

TEST.DOOCS ODR ODR1 PARAM.DUMPTOHD

Facility

TEST.DOOC

Device

EVB.SAVE
LOCALHOST_888
LOCALHOST_888
SINGENERATOR
ODR

Location

ODR._SVR
ODR1

Property **Description**

PARAM.GROUP Grouping⬚INT
PARAM.IOTHREADS Number IO threads⬚INT
PARAM.DUMPTOHD Dump data on disk [0/1]⬚INT
PARAM.DUMPSIZE Data dump size⬚INT
PARAM.WORDEXTRACT Word to extract⬚INT
PARAM.DGNET Data Generator/Network [0/1]⬚INT
PARAM.ACTCHAN Active Channel(s) 0/1 off/on⬚IN
PARAM.NUMDMA Number of DMA buffers:⬚INT

Sorting □ **All Locations** □

INT Format : dec

/tmp/rpc_test_bartsch **Format:** none

Result : 1

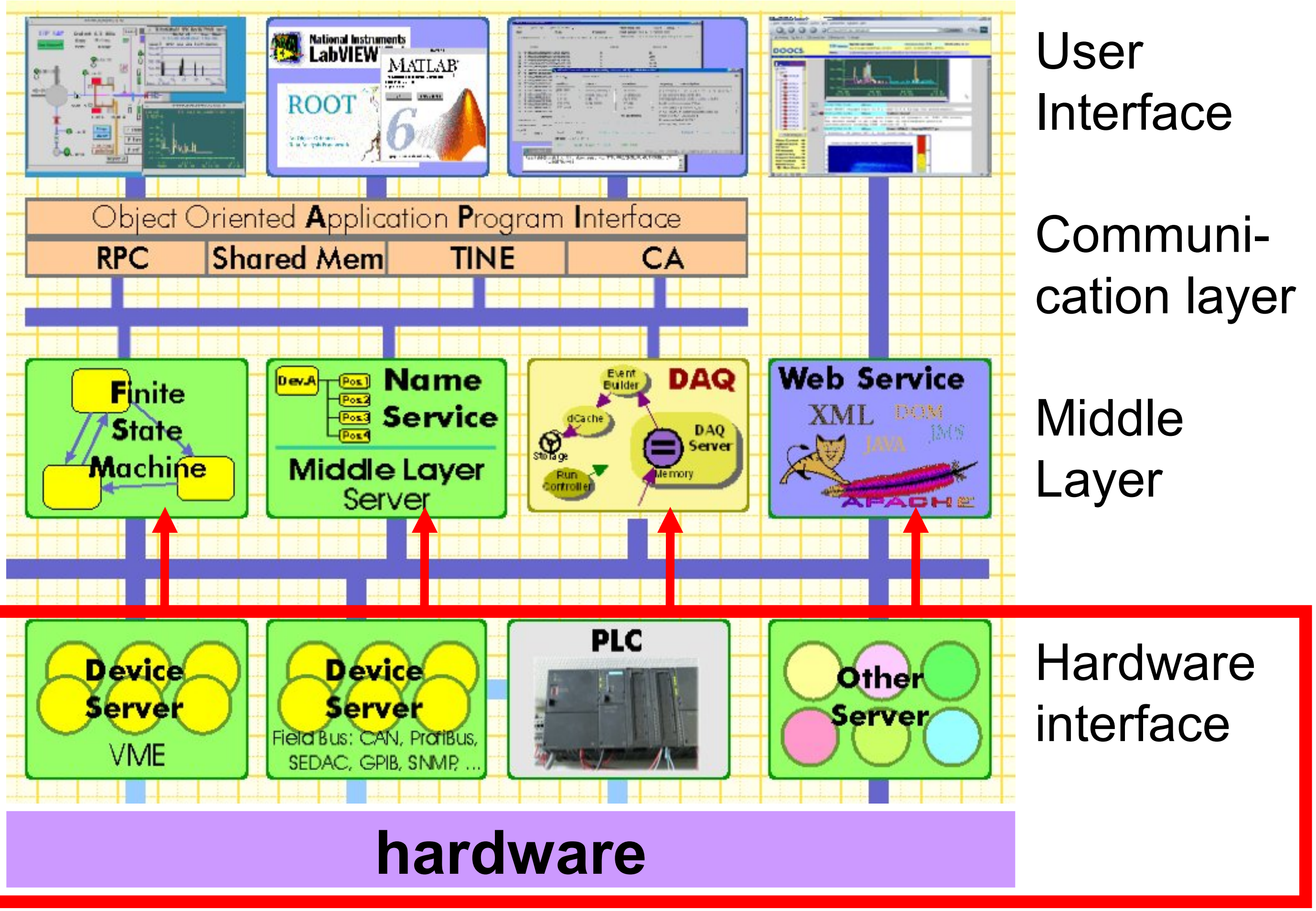
Send - Type: INT **Send Data:** 1

DOOCS host= caldaq0 (svr_mask=2 auth=15 opt=0 status=ok)

- Naming convention is already specified (similar for LDA, DIF and ASICs)
- Properties need input from hardware programmers

Hardware interface

Hardware interface most important at this stage



User Interface

Communi-cation layer

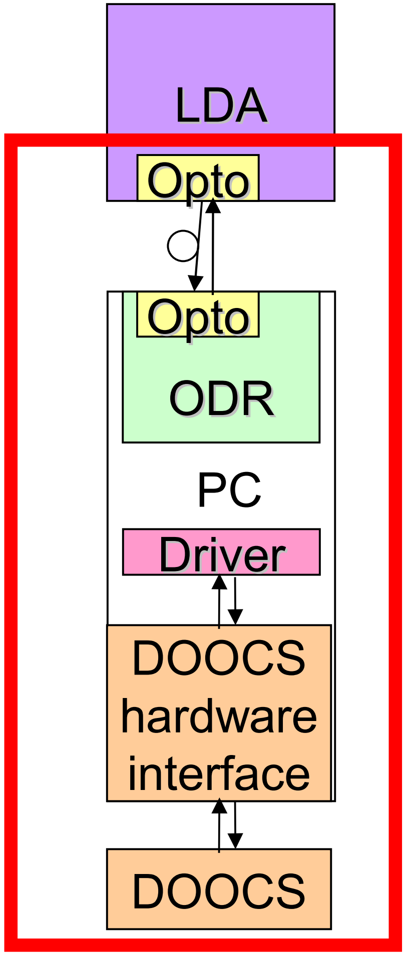
Middle Layer

Hardware interface

Existing examples: VME, Sedac, Profi-Bus; ODR server

Start hardware interface

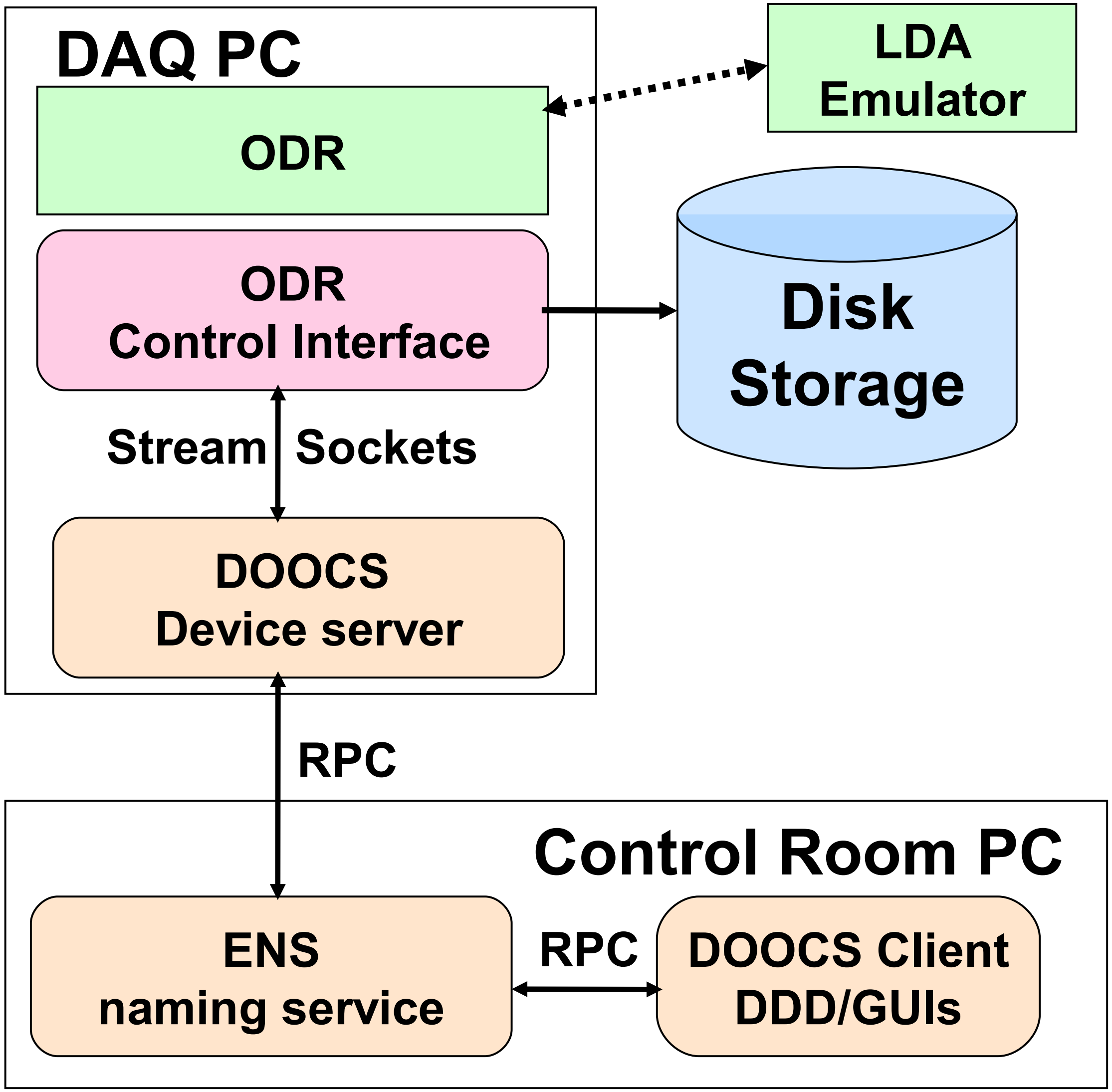
- Concentrating on the ODR interface:
  - because it is the first hardware layer to talk to DOOCS
  - the device is close to be ready
- Plan:
  - Will start with LDA and DIF from Oct
  - have the interfaces ready about end of the year



ODR hardware interface is existing !

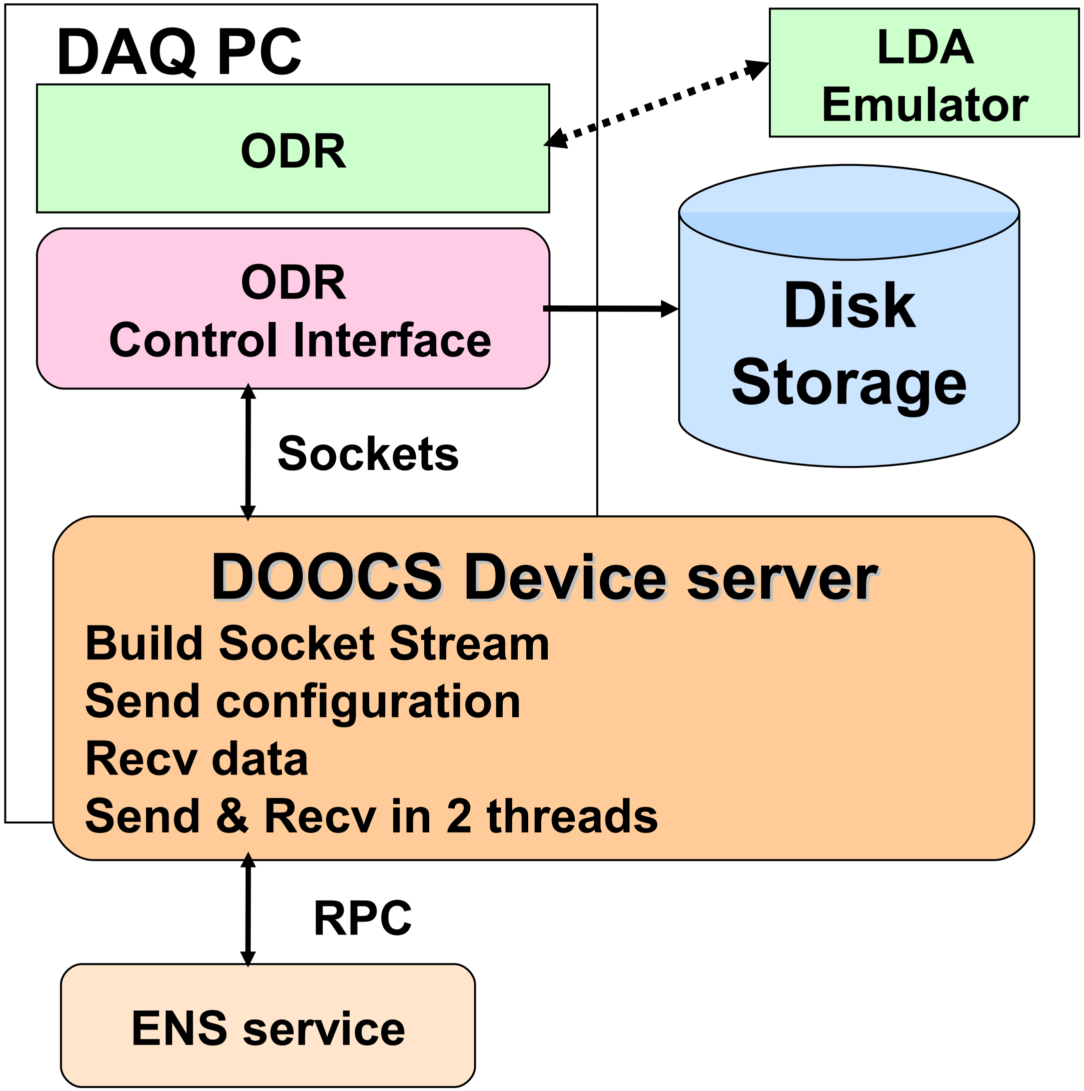
ODR-DOOCS interface

- Communication between different parts of DOOCS **Server/Client** by RPCs
- Configuration files are used when startup and to link different parts of the system



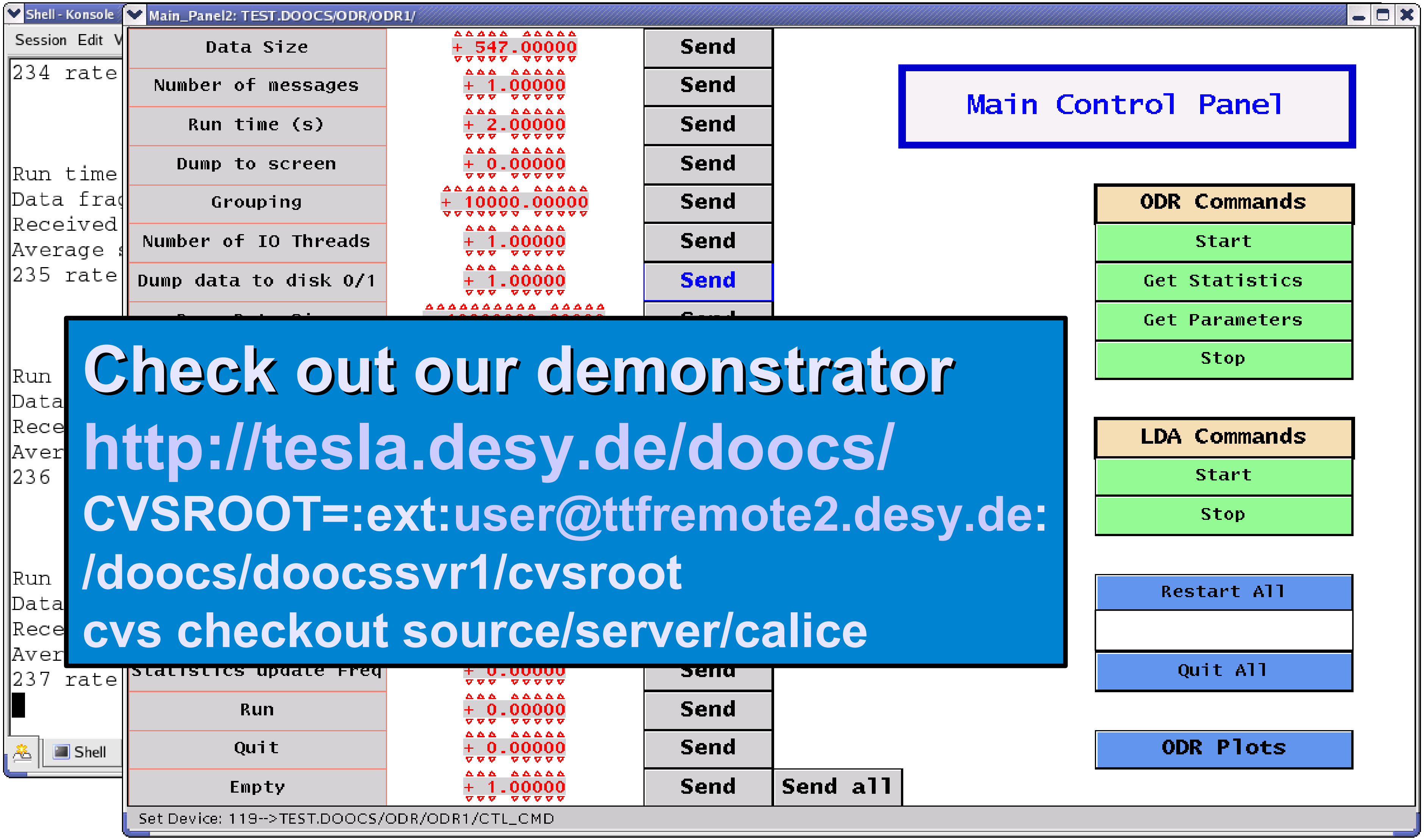
ODR-DOOCS interface

- One device server can serve many instances all connected via different ports and hostnames
- Using 2 threads: one for receiving, one for sending on the socket
- Sockets format is chosen to build an interface to the ODR and the LDA



Build Socket Stream
Send configuration
Recv data
Send & Recv in 2 threads

ODR interface at work



```
234 rate


Run time
Data frag
Received
Average s
235 rate



Run
Data
Rece
Aver
236



Run
Data
Rece
Aver
237 rate
```

Data Size	+ 547.00000	Send
Number of messages	+ 1.00000	Send
Run time (s)	+ 2.00000	Send
Dump to screen	+ 0.00000	Send
Grouping	+ 10000.00000	Send
Number of IO Threads	+ 1.00000	Send
Dump data to disk 0/1	+ 1.00000	Send
Statistics update Freq	+ 0.00000	Send
Run	+ 0.00000	Send
Quit	+ 0.00000	Send
Empty	+ 1.00000	Send

Set Device: 119-->TEST.DOOCS/ODR/ODR1/CTL_CMD

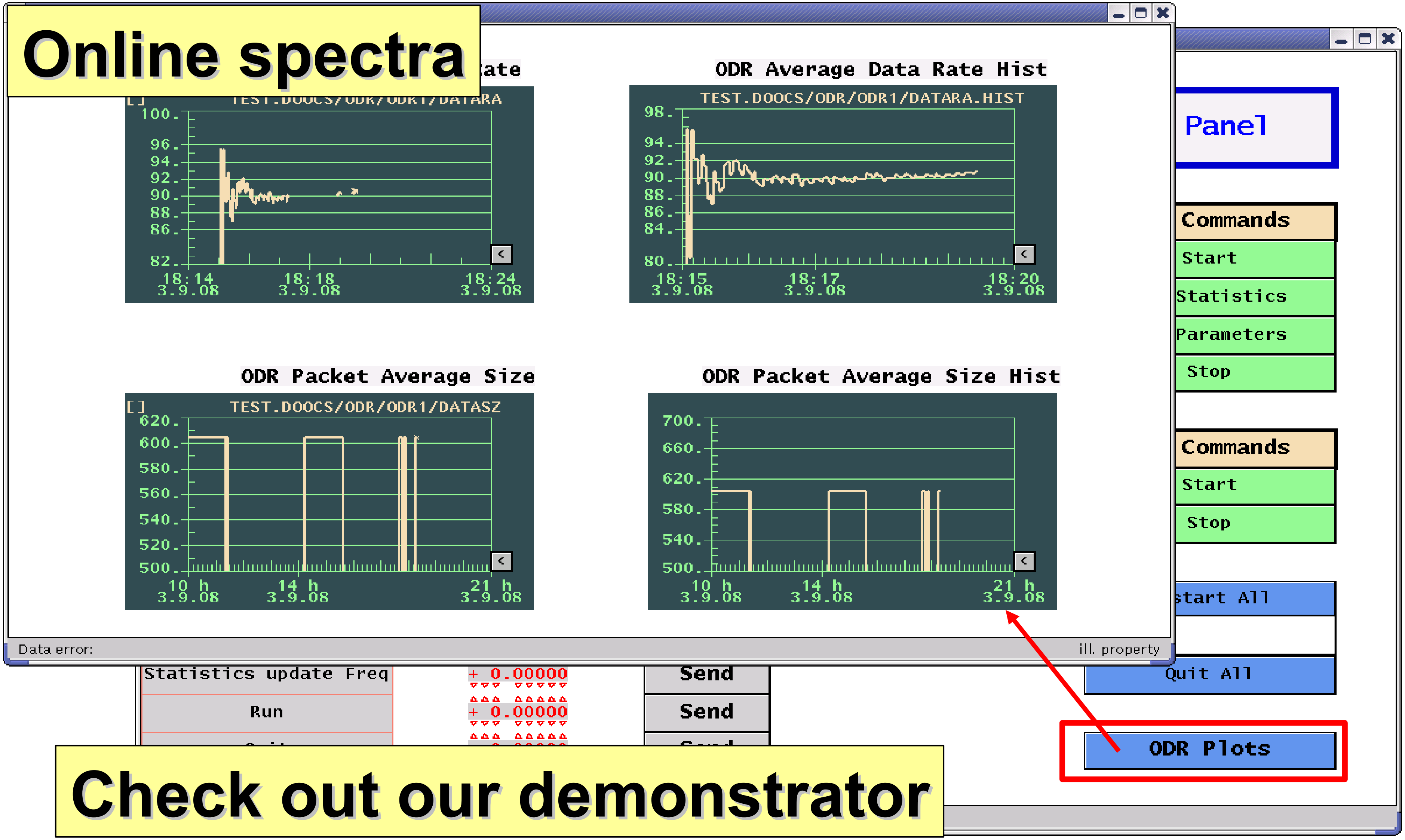
Check out our demonstrator

http://tesla.desy.de/doocs/

CVSROOT=:ext:user@ttfremote2.desy.de:/doocs/doocssvr1/cvsroot

cvs checkout source/server/calice

ODR interface at work



Data error:

ill. property

Statistics update Freq + 0.00000 Send

Run + 0.00000 Send

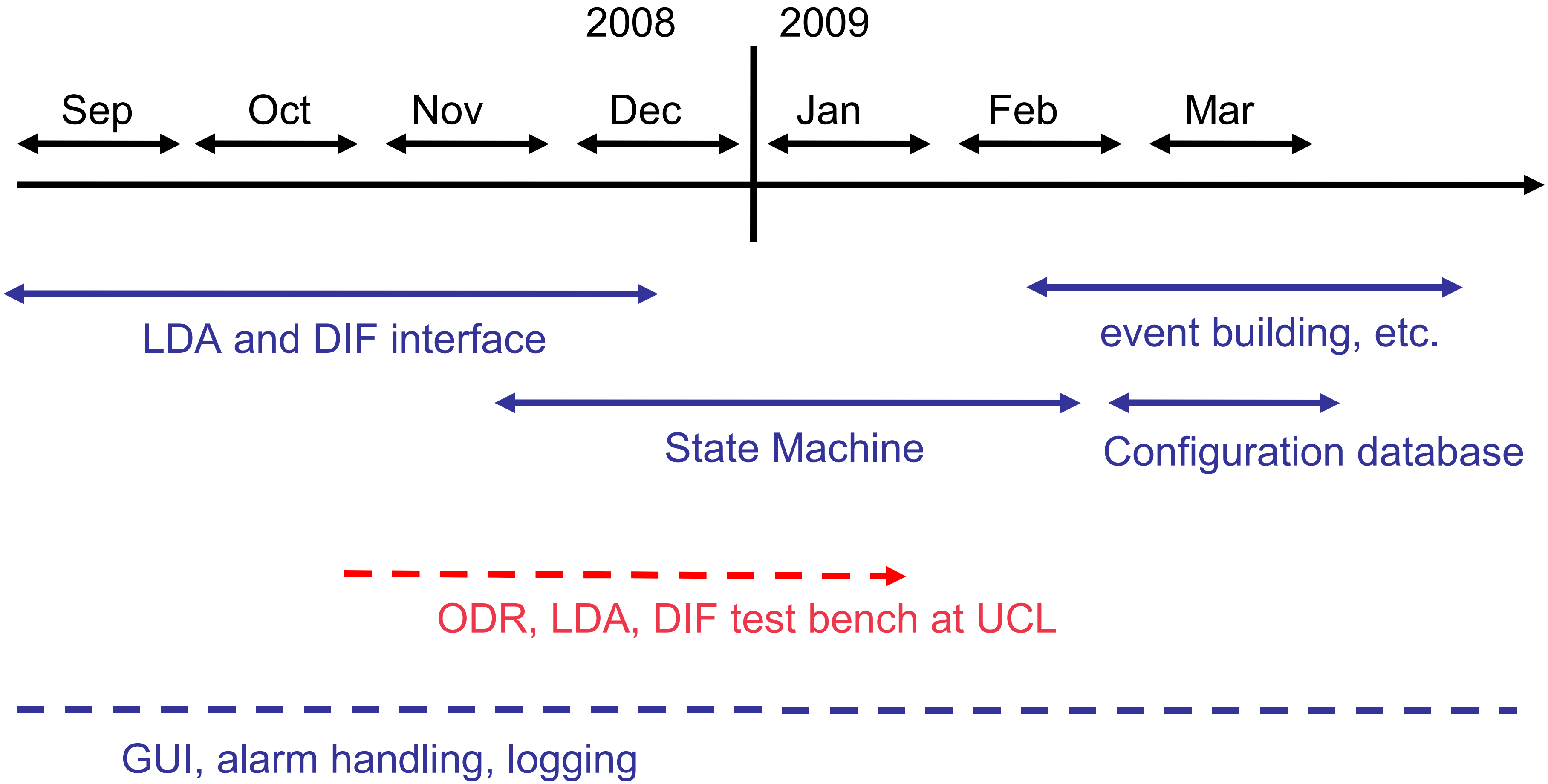
Online spectra

Check out our demonstrator

Future plans: to be implemented

- Continue to develop interface to hardware (ODR to LDA, LDA to DIF, etc.) talking to DOOCS;
- Investigate & define the DIF→LDA→ODR links;
- Start the event building;
- Think about the implementation of State Machine;
- Complete necessary properties and functions of all hardware components;
- Putting (all) components together, test…

Suggested Timescale



2008 2009

Sep Oct Nov Dec Jan Feb Mar

LDA and DIF interface

event building, etc.

State Machine

Configuration database

ODR, LDA, DIF test bench at UCL

GUI, alarm handling, logging

Summary

- DOOCS framework is reusable & suitable for our DAQ control system.
- DAQ software is in designing phase; Progresses have been making…
- Building the software for the whole link from ASIC→DIF→LDA→ODR is ongoing.

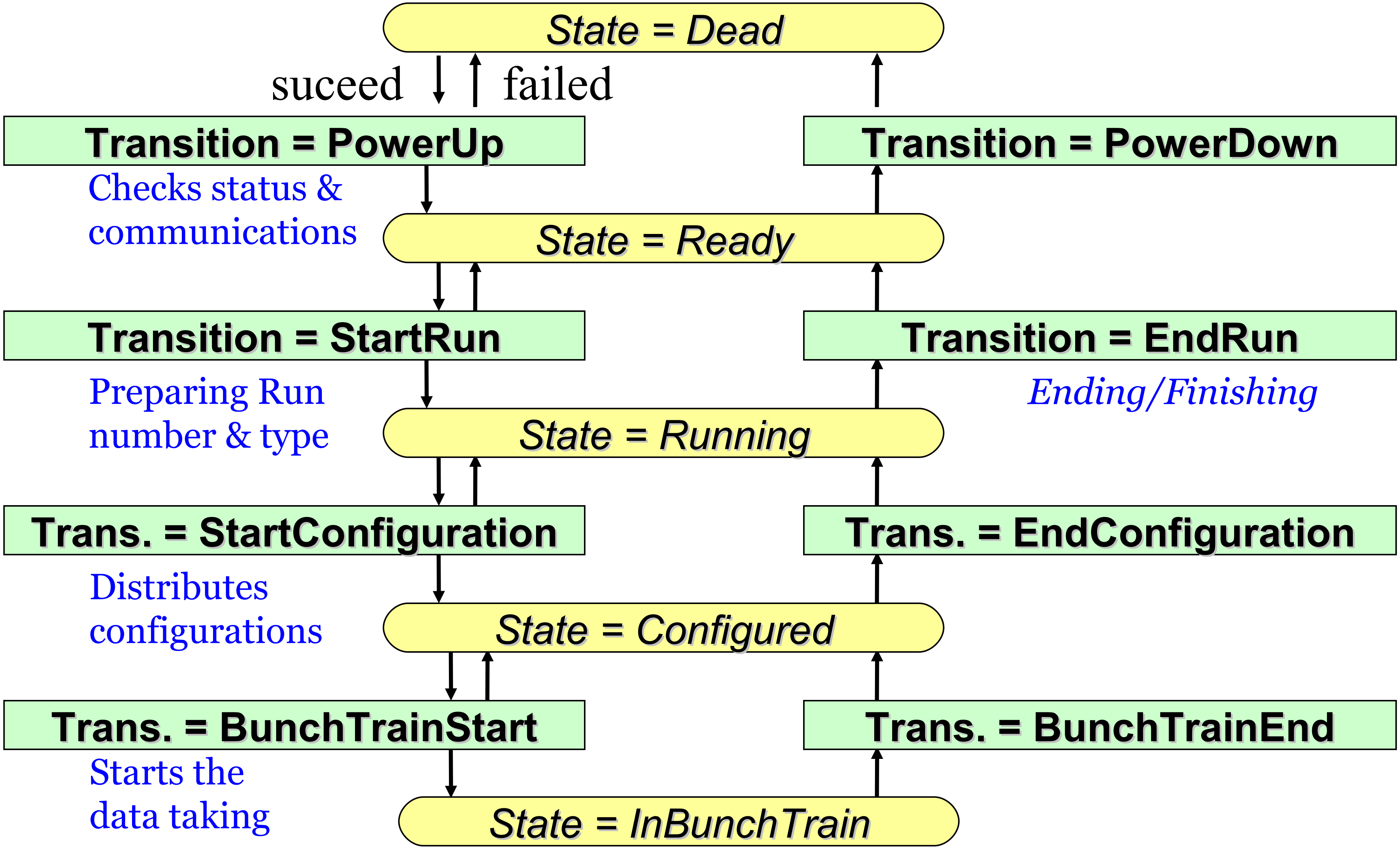
Thanks to all my colleagues in UK collaboration!

Open to public

- New ideas
- Comments
- Suggestions
- Existing experience ...

ALL Welcome !

EUDET DAQ software: State



State = Dead

suceed

failed

Transition = PowerUp

Checks status & communications

State = Ready

Transition = StartRun

Preparing Run number & type

State = Running

Trans. = StartConfiguration

Distributes configurations

State = Configured

Trans. = BunchTrainStart

Starts the data taking

State = InBunchTrain

Trans. = BunchTrainEnd

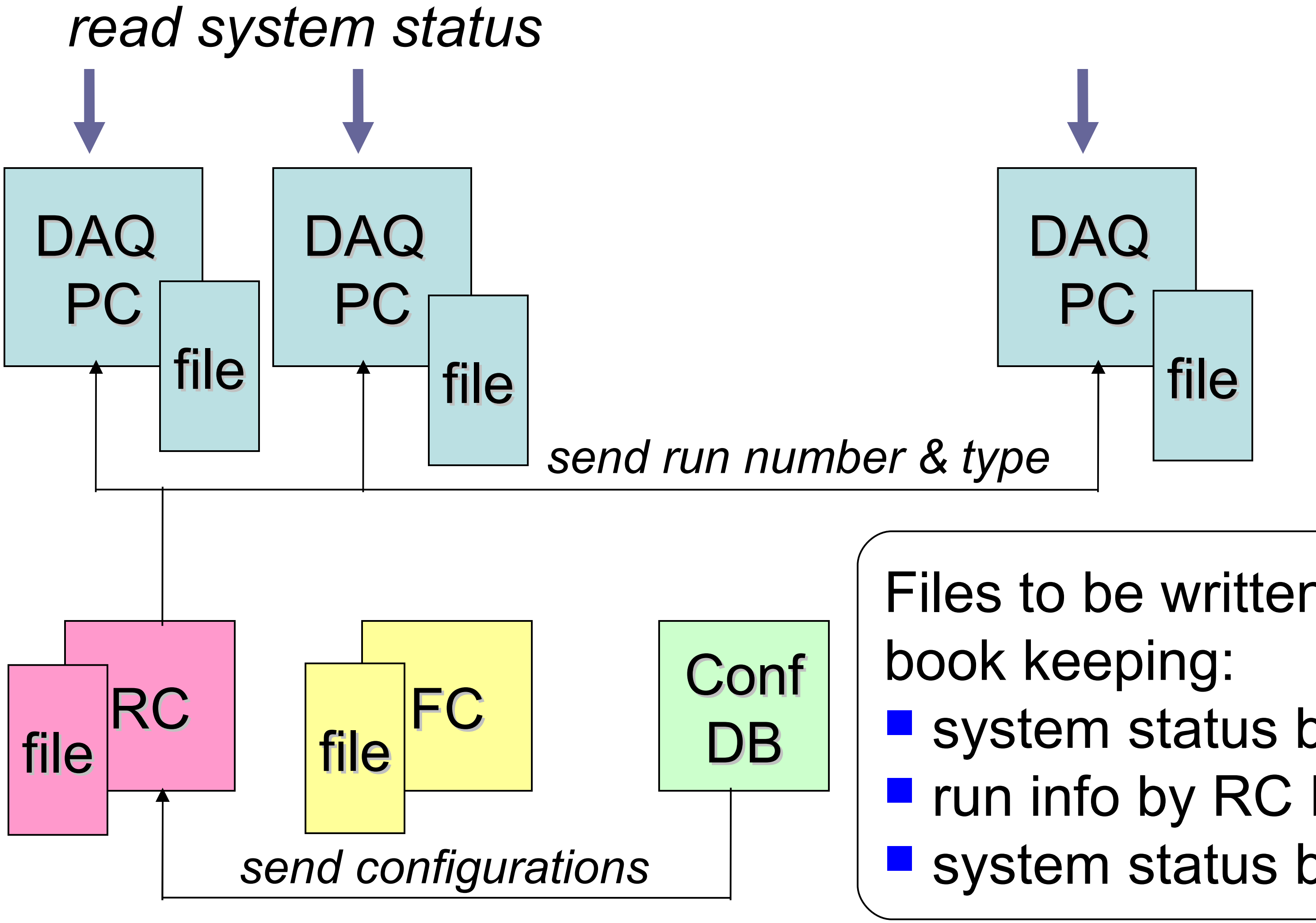
Trans. = EndConfiguration

Transition = EndRun

Ending/Finishing

Transition = PowerDown

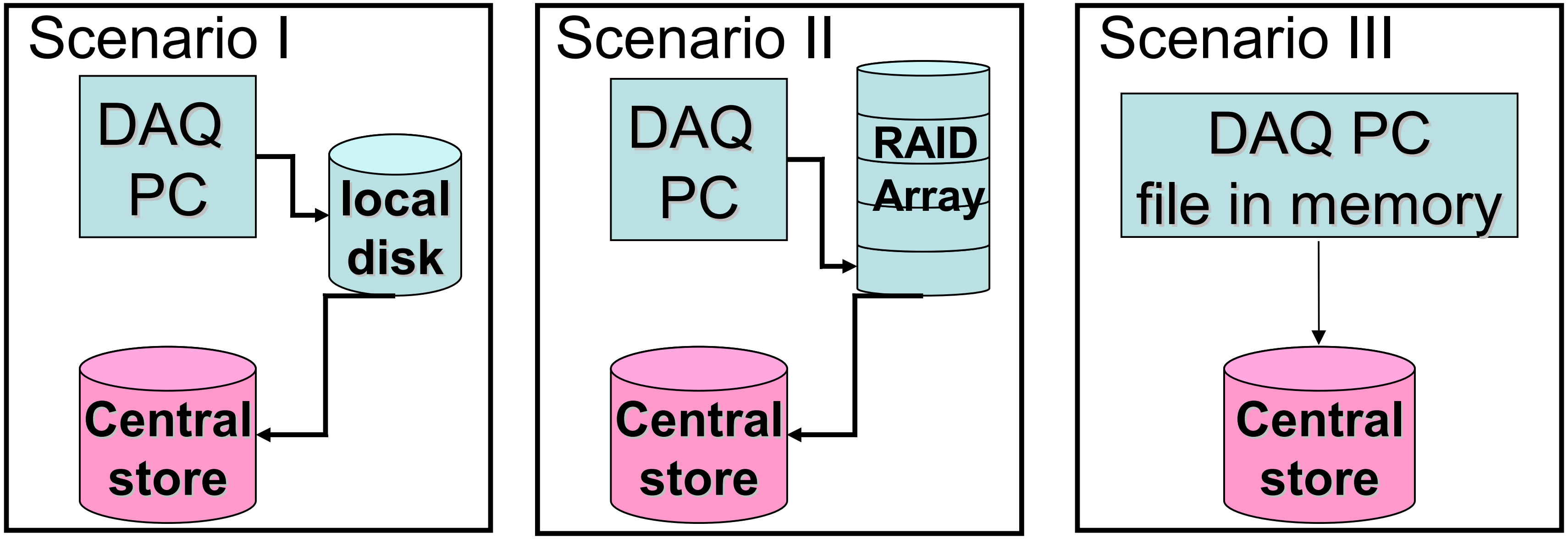
Transition: StartRun



Files to be written for book keeping:

- system status by DAQ PC
- run info by RC PC
- system status by FC

Data Storage



- Which scenario to choose depending on the bandwidth with which the data gets produced: **(1)** up to 200Mbit/sec, **(2)** up to ~1600Mbit/sec, **(3)** from there on

- Estimation of the data rate for the EUDET DAQ prototype has to cope with ~400Mbit/s, however it depends on the detector and the choice of VFE.

DAQ Architecture

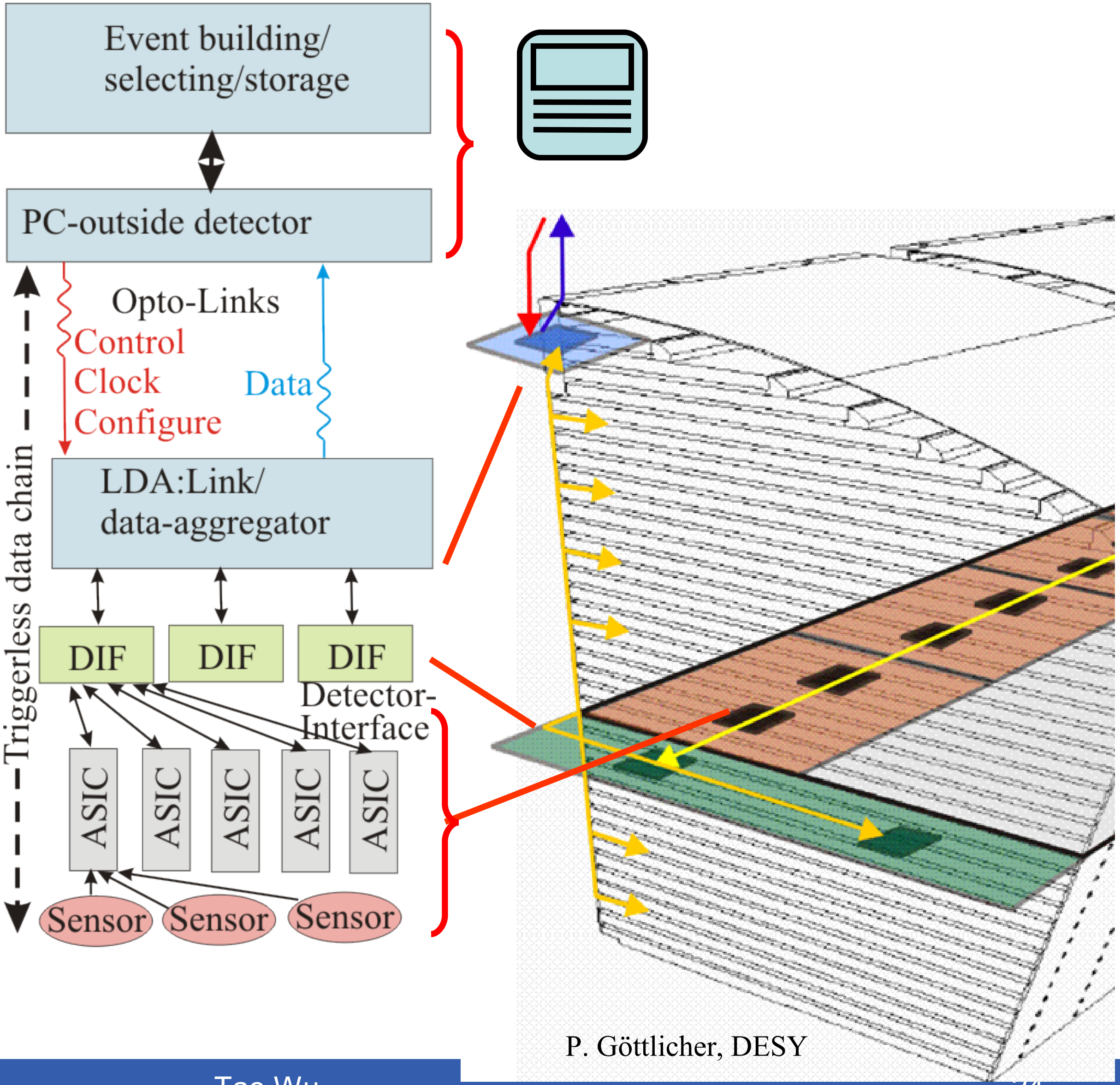
DAQ software

Off Detector Receiver (ODR)

Link Data Aggregator (LDA)

Detector Interface (DIF)

Detector Unit ASICs



P. Göttlicher, DESY