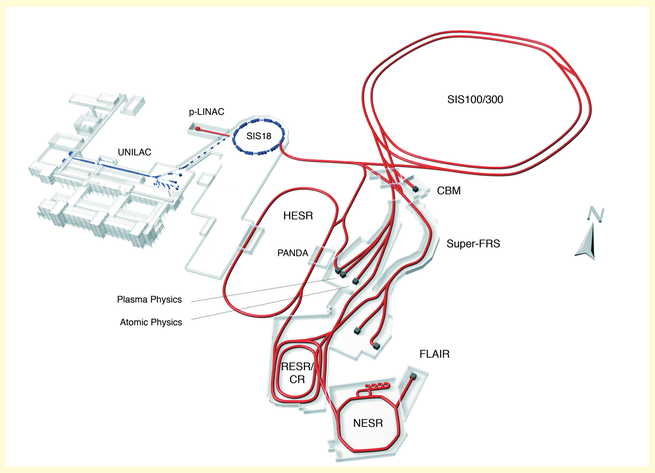
FESA
The Front-End Software Architecture at FAIR

T. Hoffmann
GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

PCaPAC 2008

Abstract
The planned Facility for Antiproton and Ion Research (FAIR) at GSI in Darmstadt is a very challenging task due to its dimension and complexity. Several new heavy ion accelerators have to be built and then operated in parallel and multiplexed modes. In order to cope with these unique requirements numerous collaboration partners are involved to add so-called ”in-kind contributions” to the project. Detailed guidelines and interface specifications have to be defined in advance to avoid an indefinite pool of different technologies which have to be handled by the future control system. For that purpose, GSI decided to use the Front-end Software Architecture (FESA) at the lowest level of the control system. FESA was developed by CERN and is already established for usage at LHC and its injectors. It is a framework to integrate any kind of equipment such as beam instrumentation devices, magnet power supplies, vacuum- and cryogenic components into the control system. A framework overview, its advantages, and boundary conditions provided by FESA are described.

Facility for Antiproton and Ion Reasearch (FAIR)



Scheme of the planned FAIR installation: Blue: existing accelerators UNILAC and SIS (heavy ion synchrotron) as injectors. Red: all new accelerator installations (p-Linac, SIS100/300, HESR, NESR, CR, RESR, pBar, FLAIR, SuperFRS plus experiments and beam transfer lines)

Primary beams:
10^{12}/s; 1.5-2 GeV/u; $^{238}U^{28+}$
Factor 100-1000 over present intensity
2(4)x10^{13}/s 30 GeV protons
10^{10}/s $^{238}U^{92+}$ up to 35 GeV/u
up to 90 GeV protons

Secondary beams:
Broad range of radioactive beams up to 1.5 - 2 GeV/u; up to factor 10 000 in intensity over present
Antiprotons 0 - 30 GeV

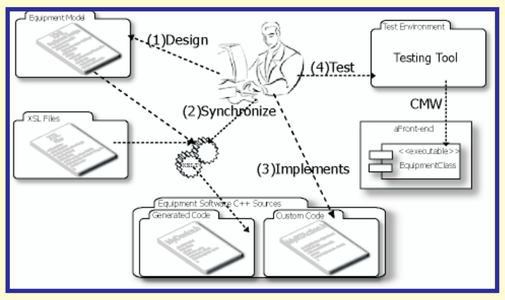
Storage and Cooler rings:
Radioactive beams
e^- – A (or Antiproton-A) collider
10^{11} stored and cooled 0.8 - 14.5 GeV antiprotons

Advantages of FESA

- high level of **maturity** (stable and tested) and availability right now.
- not only a control but also **data acquisition system**.
- allows integration of demanding applications such as video imaging or beam position monitoring.
- addresses also **non-control system** experts.
- the lack of manpower can be partly compensated.
- no need of specific knowledge on interfaces, timing, middleware access, visualisation.
- source code is **generated automatically**.
- re-usage of debugged and tested code **saves plenty of developing time** and prevents errors and inconsistencies.
- FESA community for sharing experience of about **70 active developers** is growing.
- high **standardization** of supported environment.
- changes to the design or code by the user, by others, and also ten years later, are much easier thanks to **uniform source code structure**.
- **safety features** such as role based design and test access, instantaneous syntax checks and CVS storage.

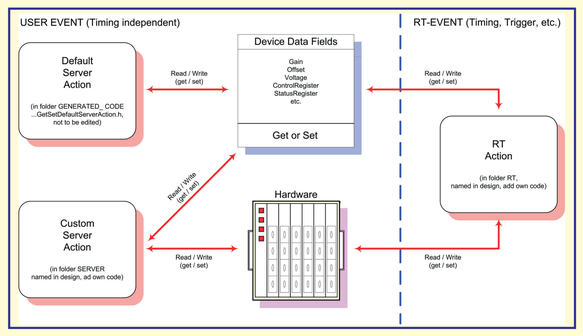
FESA Facts

- Supports: VME, cPCI/PXI, VXI, PCI
- PLC: Schneider & Siematic
- OS: Linux/LynxOS
- CPU: Motorola, PowerPC, Intel
- Devel:
  - Java based (Mac, Windows, Linux)
  - Linux access for source code development
  - creation of a binary for DAQ
  - included RDA server (CMW, omniOrb)
  - C/C++ for development
- Roots in CERN Beam Diagnostics group
- CERN features: multiplexed timing, logging, alarms, diagnostics



Successive steps to develop equipment software using FESA (deployment and instantiation are not shown). Picture from ICALEPS07,WOPA04, Arruat et.al.

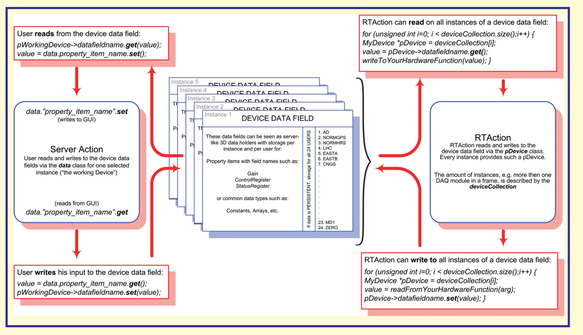
Action concept and data container



RT-Action:
Executes user-code on trigger (software, timer, external or machine trigger)
Used to access (read/write) hardware (high priority)

Default SERVER-Action:
Executes simple get/set operations from user to the system
Timing independent hardware access (low priority)

SERVER-Action:
Executes user-coded get/set operations from user to the system
e.g. Text-outputs, calculations, conversions etc.
Timing independent

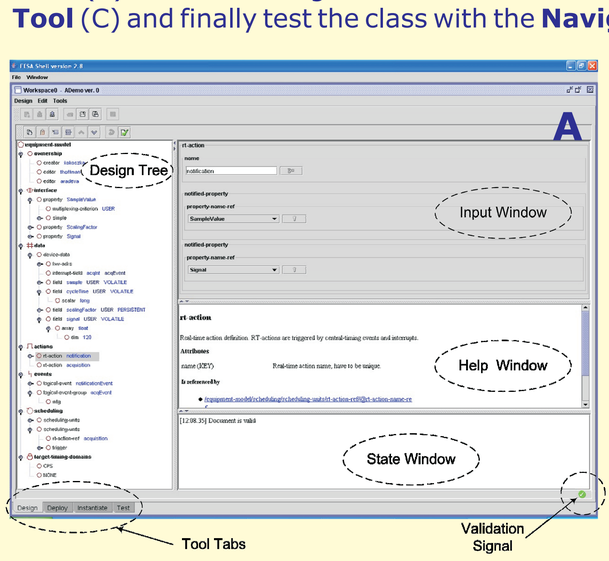


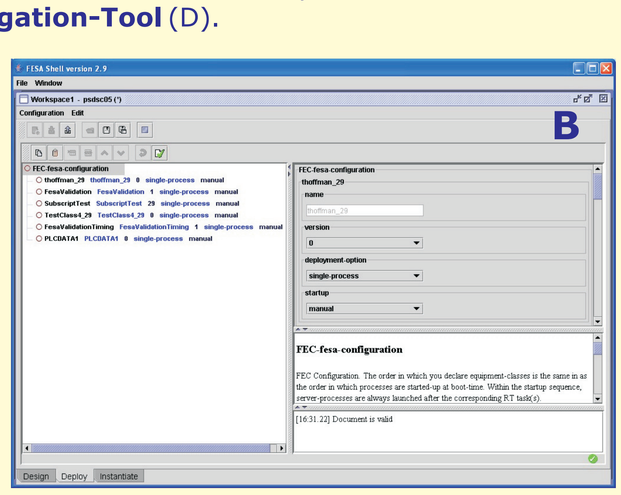
DEVICE DATA FIELD

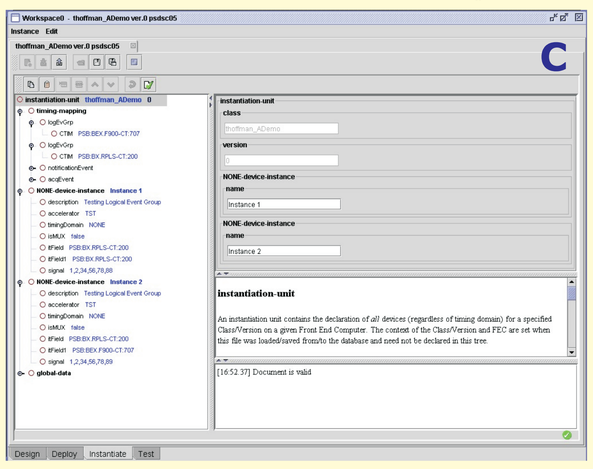
- Any kind of data (registers, bits, readout, constants, arrays) is represented
- Exchange container from hardware to user
- User has no direct access to hardware
- 3-D data container for multiplexed beam operation and different instances

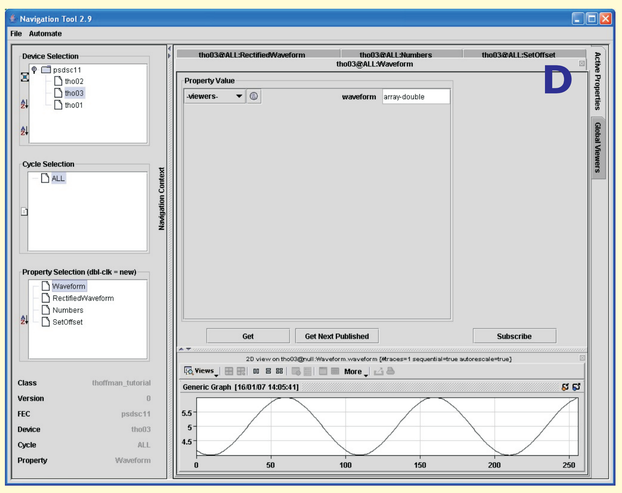
FESA-Tools

Develop a FESA class **step-by-step** using the **Shell** (Java-GUIs). Begin with the design of the class by using the **Design-Tool** (A). Bind the class to a front-end CPU with the **Deployment-Tool** (B). Define settings for different instances of the class with help of the **Instantiation-Tool** (C) and finally test the class with the **Navigation-Tool** (D).



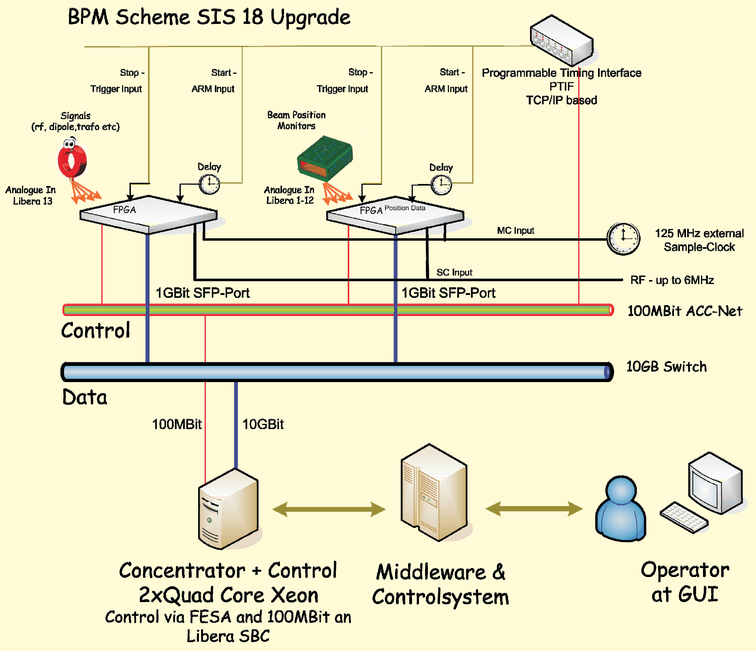






FESA Test-Project at GSI

Aim is to install a **bunch-to-bunch beam position monitoring system** at the SIS-18 at GSI. Therefore 12 **Libera Hadron** systems (I-Tech, SI) are installed and supervised by two powerful servers. With this installation the **complete hierachy** of the future control system, from the sensor and FESA up to the GUI in the control room, shall be tested.



- 12x **Libera Hadron** with 1GB data rate each (4x14Bit ADC, 125MS/s)
- 2x 10-Channel Libera **Clock-Splitter**
- 2x Dual Quad-Core Xeon 2GHz Server with 32GB RAM each
- 1x 20 Channel HP-Procurve **10GBit** Switch
- Distributed system layout
- Programmable GSI Timing (hardware coupled - no FESA)
- External **precise sampling clock** (about 5ps jitter, in development)
- 10GBit network for DAQ, 100MBit network for control

Installation of **complete FESA framework** at GSI
- NFS structure
- Middleware (CMW)
- CVS
- Database
- Timing Receiver Boards (only for VME, external trigger)

Not yet:
Installation of application level (Java-GUI)
Functional and performance checks of complete hierachy
Porting FESA and Timing Receiver Library to 64Bit