FRONT-END SOFTWARE ARCHITECTURE [FESA]

Michel Arruat, Leandro Fernandez, Stephen Jackson, Frank Locci, Jean-Luc Nougaret, Maciej Peryt, Anastasiya Radeva, Maciej Sobczak, Marc Vanden Eynden

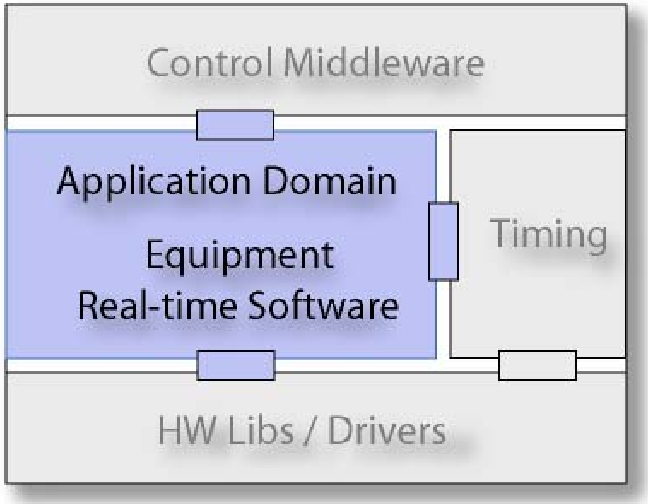
Accelerators & Beams Department, CERN

Outline

- What is FESA ?
- How FESA ensure Equipment Software portability across CERN Accelerator
- Quick turn in the FESA developer's shoes
- How FESA handle evolution
- Some recent extensions
- Conclusions

What is FESA ?
Application Domain

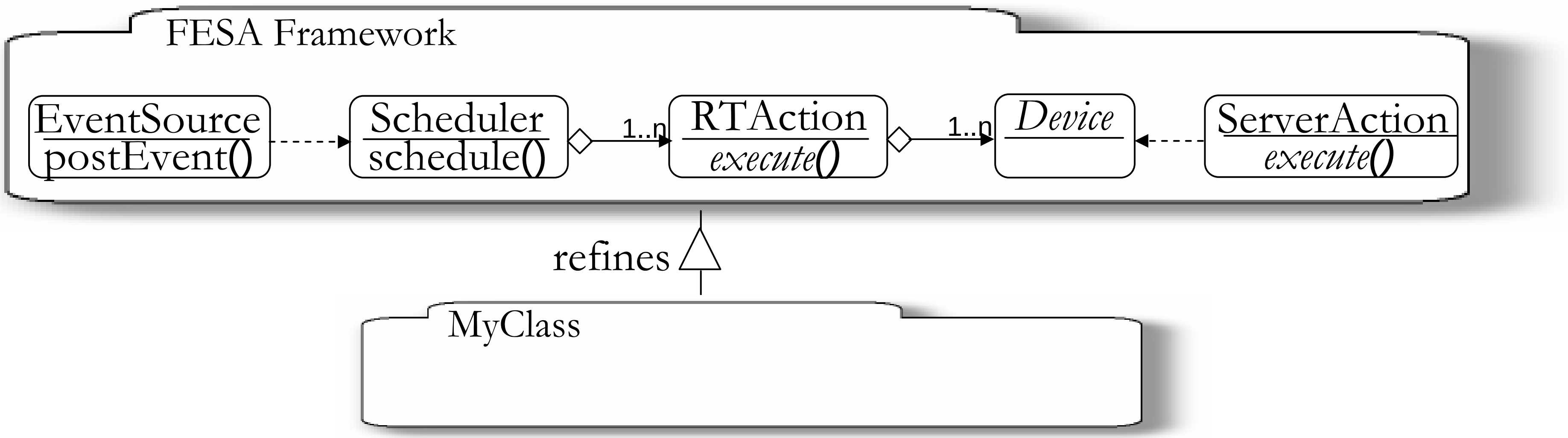
- Equipment Software running on front-end computers: FESA class.
- Surrounding Software Components:
  - Control Middleware: communication infrastructure
    - Device/Property model.
    - Narrow API: same calls for all equipment classes (access methods: get/set/subscribe).
    - A device belongs to a Device Class
  - Timing: timing events are distributed over a dedicated network to local timing receiver.
    - Handles timing hardware.
    - Provides interface for timing events connection.
  - Hardware:
    - Standard modules comes with drivers and /or libraries



[see: "Remote Device Access in the New CERN Accelerator Controls Middleware" ICALEPCS' 01]
[see: "The CERN LHC Central Timing, a Vertical Slice"]

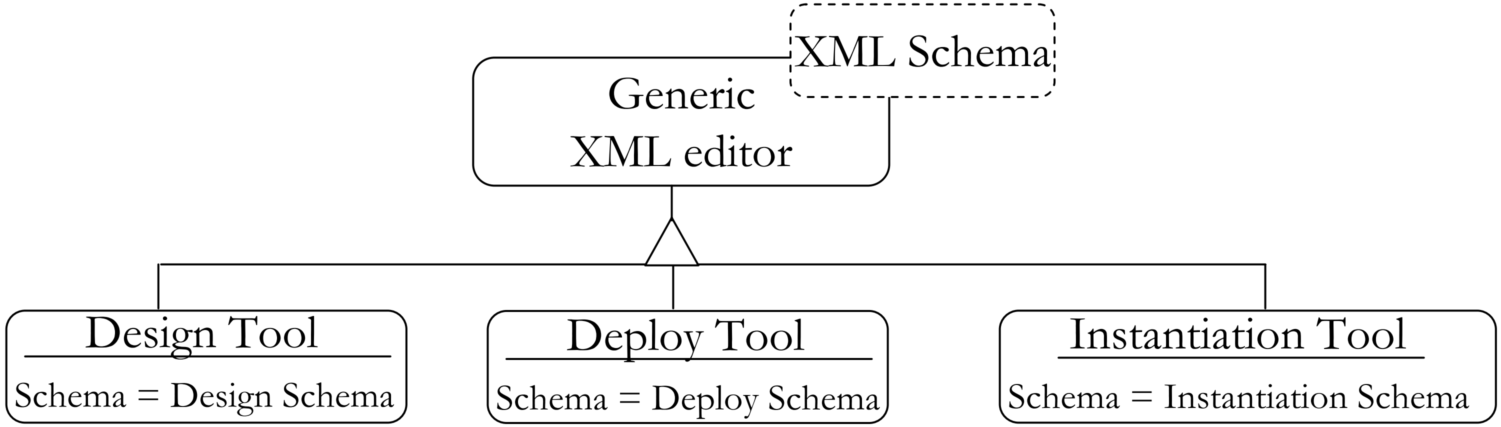
What is FESA ?
Real-time Framework

- Object-oriented real-time Framework:
  - Captures the structure and the control flow of the application domain.
  - Defines the application domain’s design pattern.
  - The Framework is the application.
  - Equipment Specialist provides application-specific behaviour.



What is FESA ?
Series of Graphical Tools

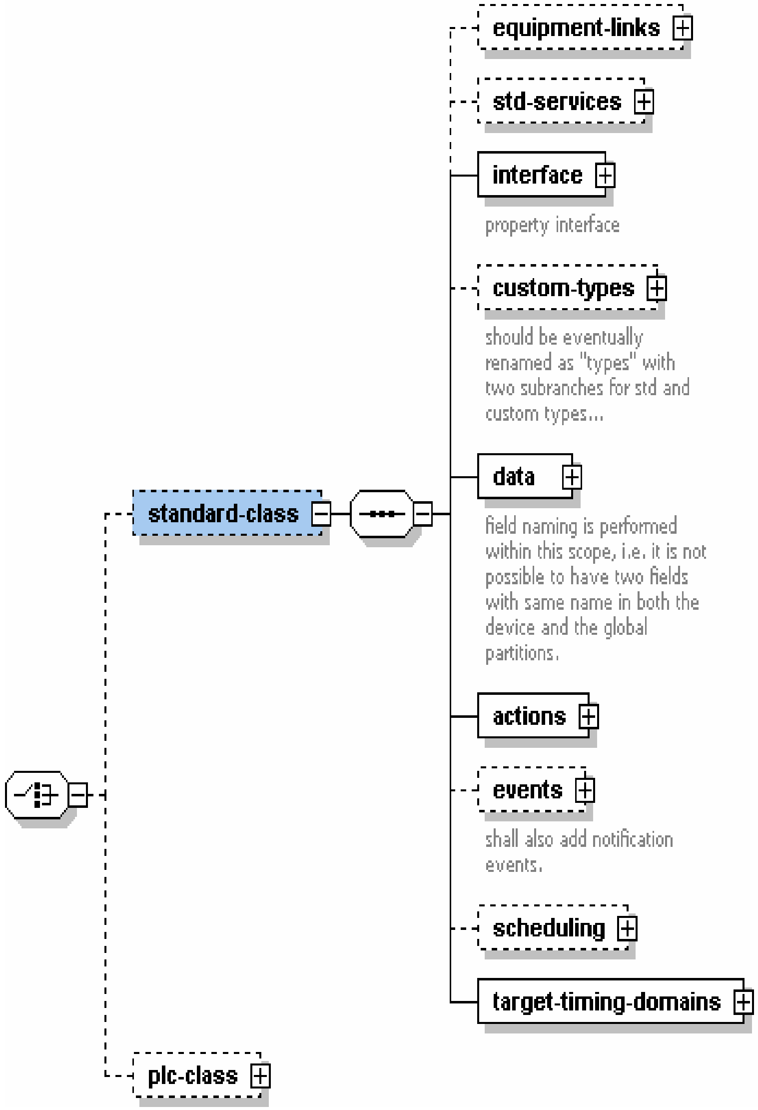
- Developing a FESA class requires the developer to produce three XML documents: Design, Deployment, Instantiation.
- Dedicated tool based on a Generic XML editor (Java application) configured by dedicated W3C's XML Schema are used to supply each XML document. The XML Schema is used to express the data model to which the XML document must conform.



- Benefits of this architecture:
  - Java code remains unchanged.
  - Evolution is handled by the XML Schemas.

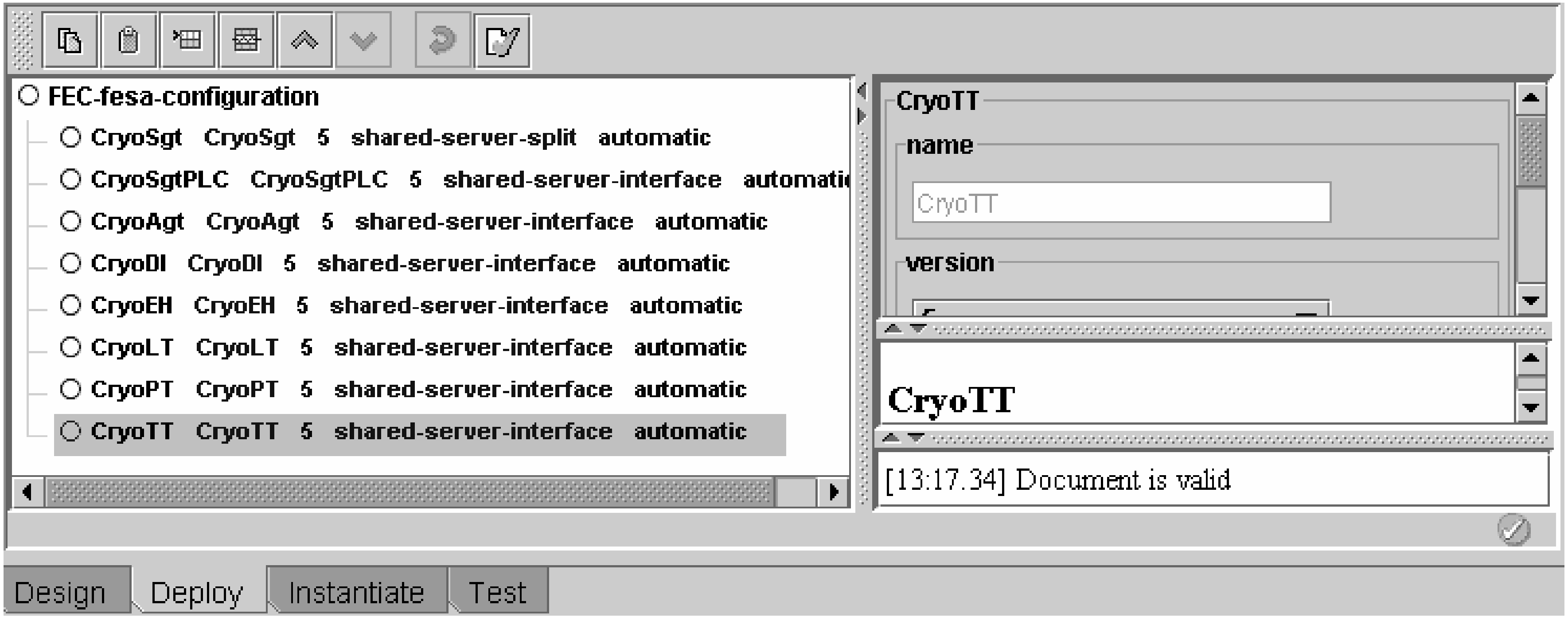
What is FESA ?
Design Tool

- Driven by the FESA Design Schema that encodes the meta-model.
  - XML is used as a high level modelling Language.
- Equipment specialist thinks equipment's design in terms of:
  - Public interface: properties.
  - Device-model: software abstraction of the hardware.
  - Server actions.
  - Real-time actions.
  - Logical events.
  - Scheduling: triggering rules.



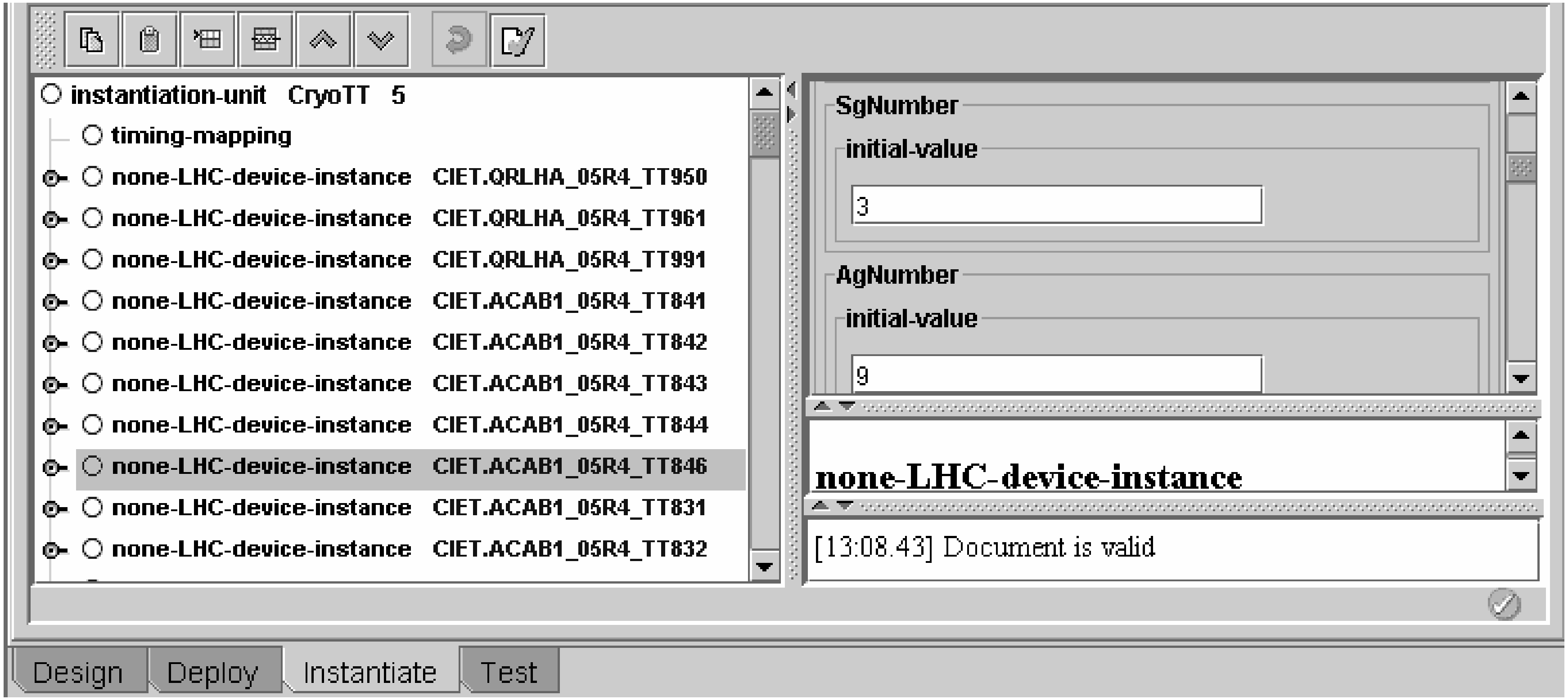
What is FESA ? Deployment Tool

- Deployment : driven by the Deployment schema. Generated on the fly based on the current list of FESA classes.
  - FESA classes can be merged in the same server or deployed as individual server.
  - Start-up mode: automatic or manual.

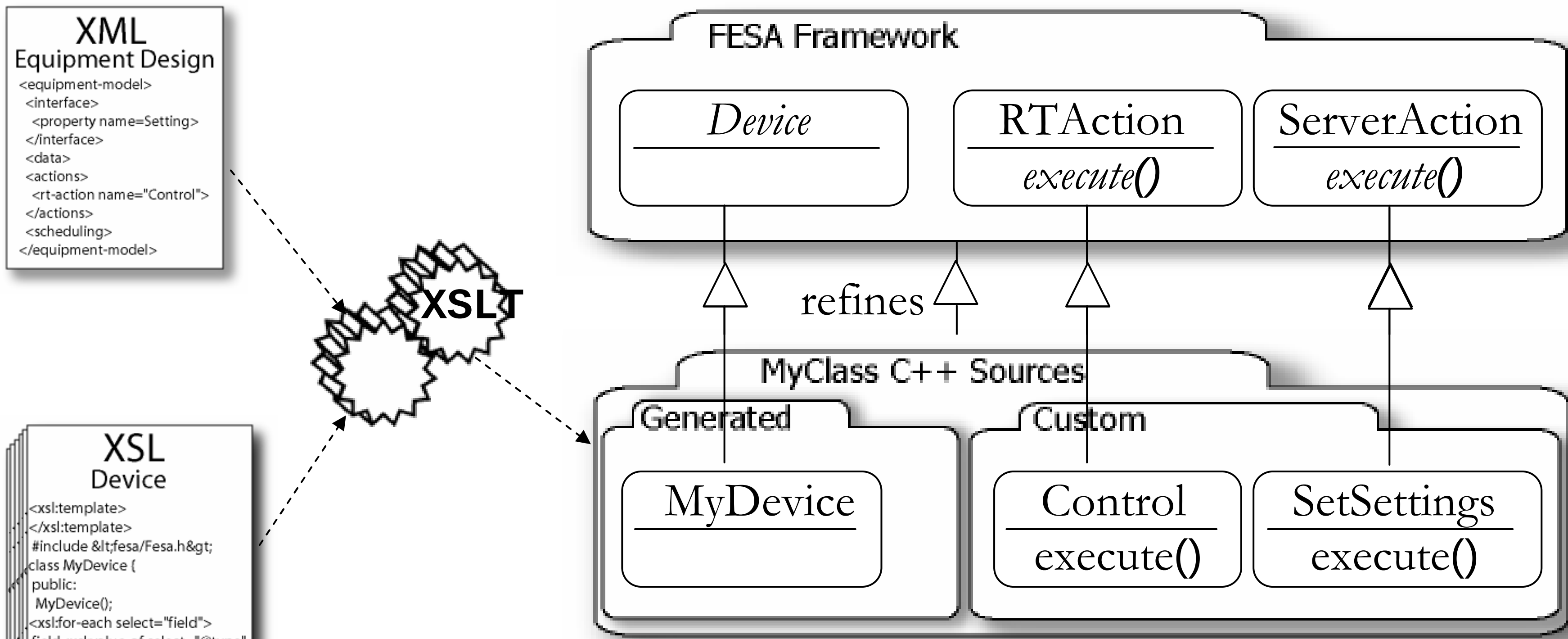


What is FESA ?
Instantiation Tool

- Instantiation : driven by the Instantiation schema. Generated on the fly based on the Design document.
  - Used to configure a set of devices on a front-end.



What is FESA? Code Generation



- Generated Code: entirely handled by FESA
- Custom Code: Action skeletons are generated by FESA

- Thanks to the formal language used to design equipment software the framework is refined by automatic code generation rather than hand-coding.

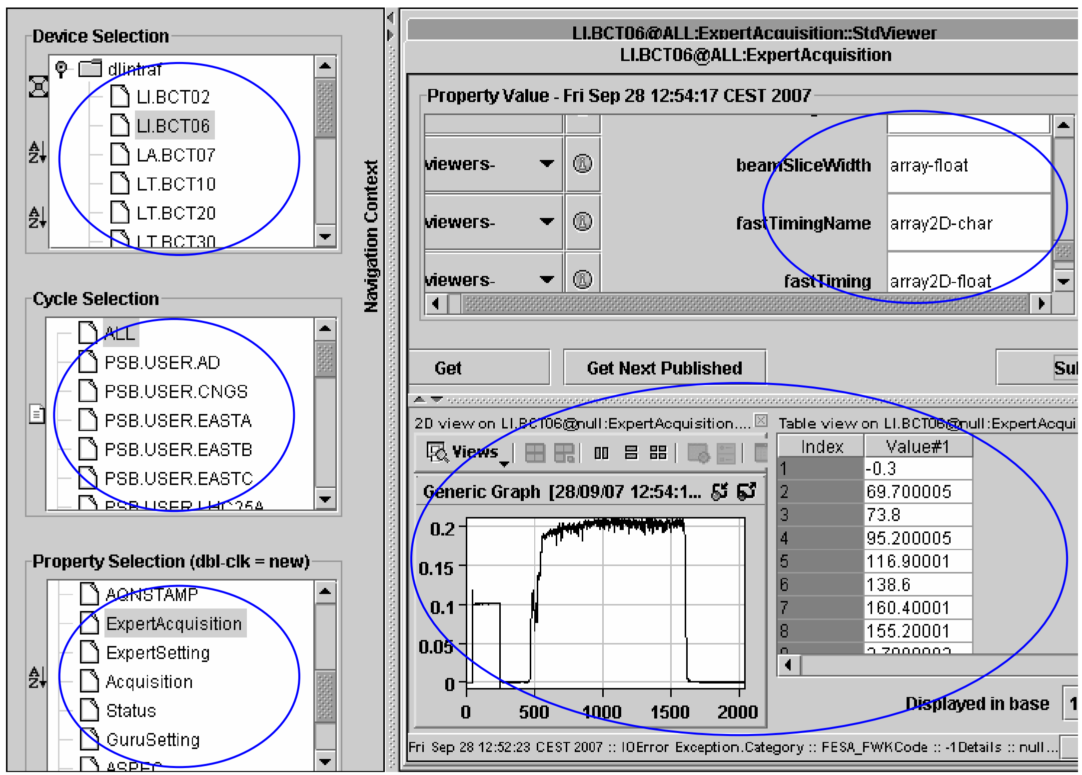
What is FESA ?
A Testing tool

- It's a complete generic Java application: XSL templates convert Design and Instantiation documents into Java property.

Front-end and devices list (Instantiation)

Cycle Selector (Timing)

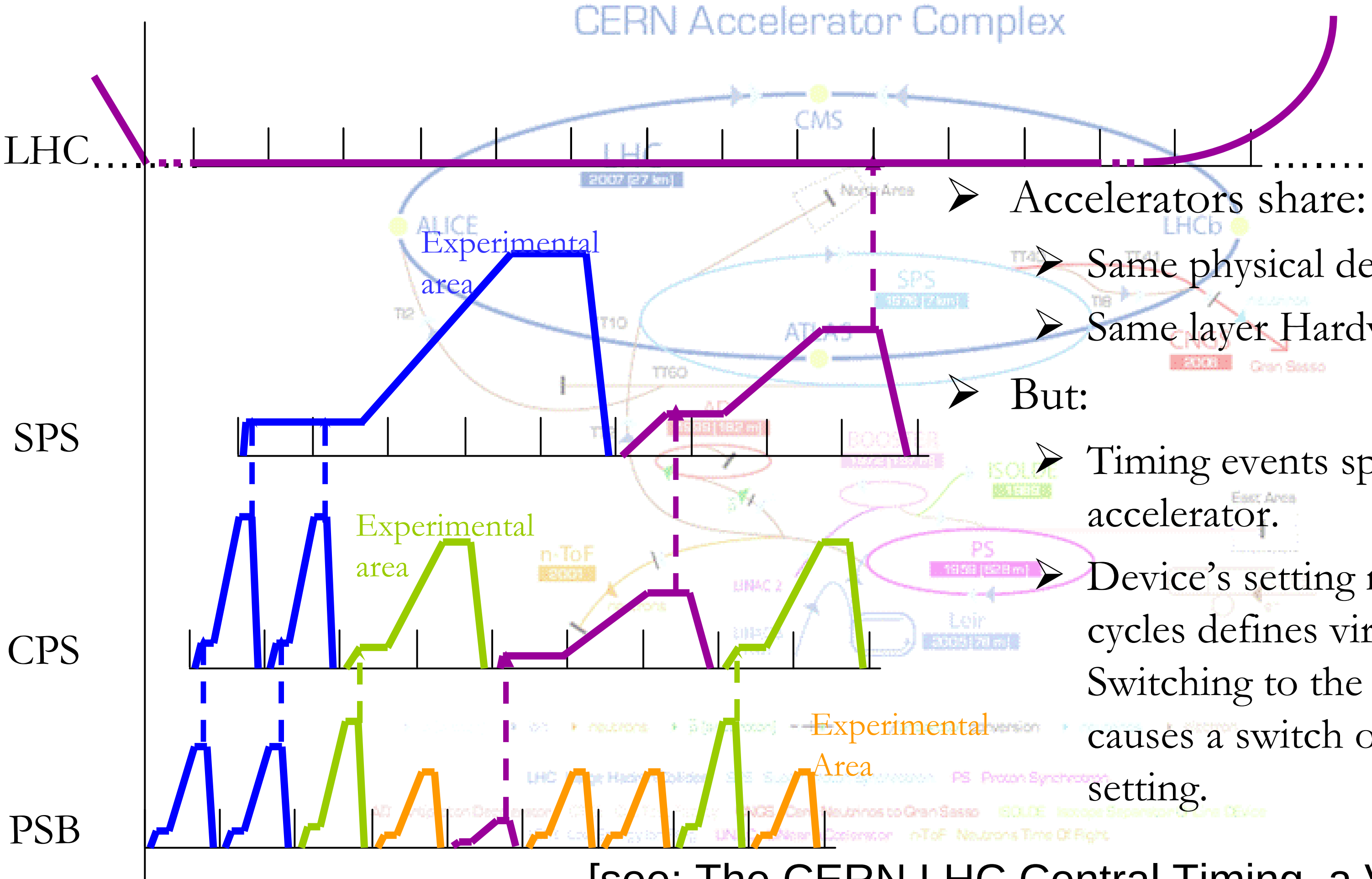
Property list (Design)



Property detail (Design)

Generic viewers

How FESA ensure Equipment Software portability across CERN Accelerator



- Accelerators share:
  - Same physical devices.
  - Same layer Hardware.
- But:
  - Timing events specifics by accelerator.
  - Device's setting multiplexing: cycles defines virtual devices. Switching to the next cycle causes a switch of device's setting.

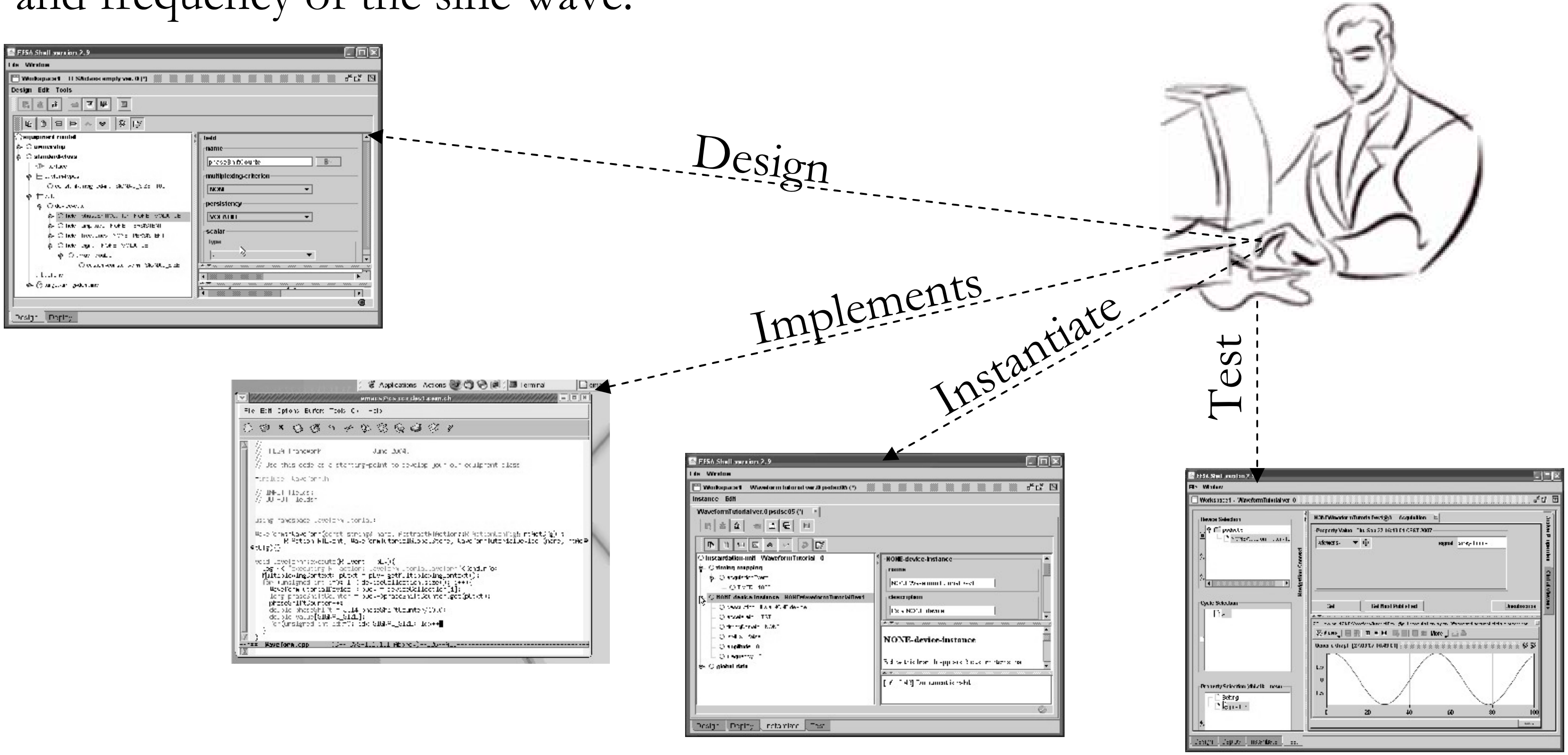
[see: The CERN LHC Central Timing, a Vertical Slice]

How FESA ensure Equipment Software portability across CERN

- In order to ensure FESA classes portability across CERN Accelerator, FESA provides a complete abstraction of the timing at different levels:
  - Design: design is not accelerator's timing dependent .
    - Timing events are logical events: concretization into accelerator events is done at the instantiation stage.
  - Implementation: multiplexing device's setting are managed by the framework transparently for the custom code.
- In this way FESA classes can be reused across all CERN accelerators. Design and Implementation are accelerator independent.

Quick turn in the FESA developer's shoes

- Case study: developing a FESA class which generate periodically a sine wave of 100 sampling with a phase shift. It shall be possible to change amplitude and frequency of the sine wave.



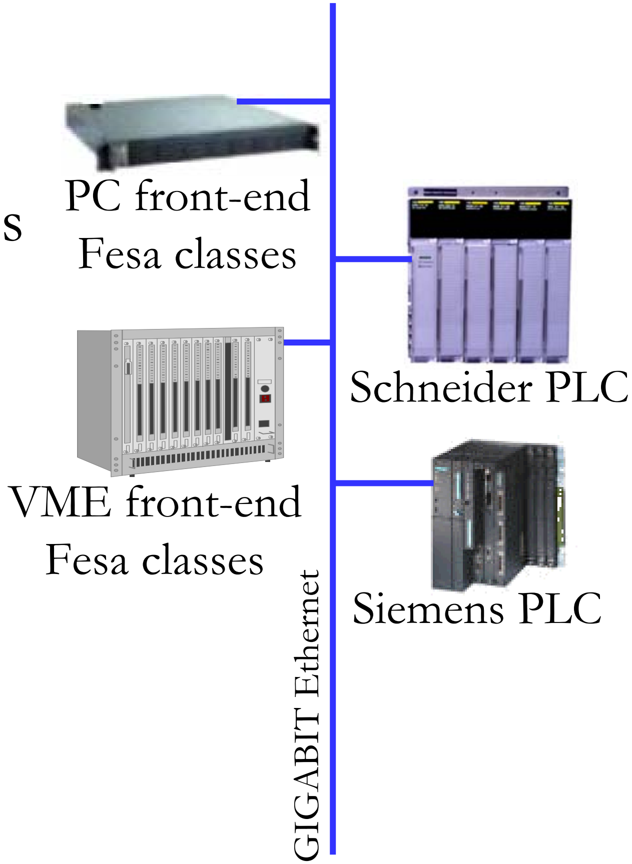
FESA

How FESA handle evolution?

- FESA evolution is mainly driven by requirements coming from Equipment Groups.
- Integrating new features in FESA requires to modify:
  - The meta-model XML Schema.
  - XSL templates used to generate the custom code.
  - Framework source code: implementation of the new features.
- What remains unchanged:
  - FESA tools: java code is really stable.
  - Equipment software custom code.
- FESA Release policy:
  - Three operational release: the new one makes obsolete the oldest one.
  - Retrofit tool: completely automatic. Upgrade the different XML documents, and the code generation.

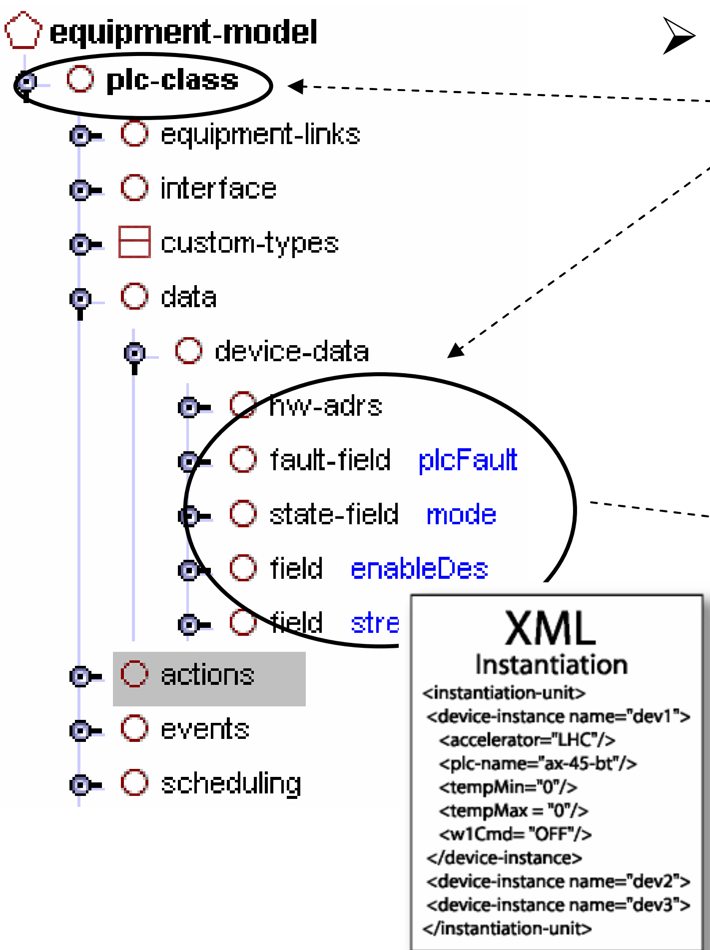
Recent extensions:PLC integration

- More and more accelerator devices are connected to PLC.
- Control hardware layer for complex devices can be a mixture of VME modules and PLC.
- Requirements:
  - PLC programmers are not FESA expert and have no desire to deal with Linux or C++.
  - No additional work.
  - No new concept or complexity
  - No duplication of description

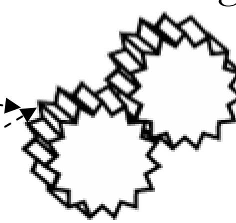


Recent extensions: PLC integration

- FESA meta-model defines plc-class as a restriction of a standard class model



- Integrated a PLC into FESA consist of:
  - Instantiate a plc-class design.
  - Design the device-data (everything-else is pre-configured).
  - Instantiate the device instances
  - Load in the PLC development tool the device data structure automatically generated
  - Develop the PLC logic as usual



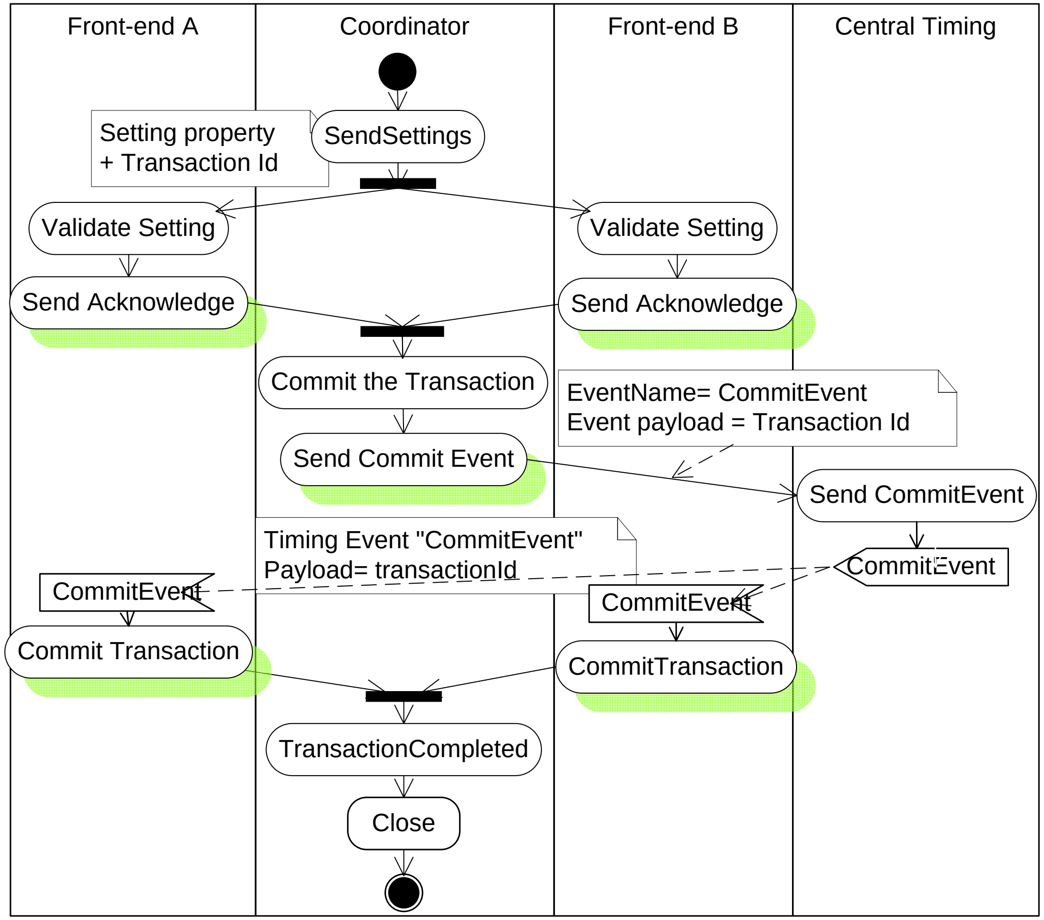
Device
data model

Automatic Generation of
the device data structure
exchanged between
PLC and front-end

FESA

Recent extensions: Transaction

- Requirement:
  - "to guarantee that several settings acting on different devices deployed on various front-ends will be taken in account at the same time or none of them in case of error".
- Implementation:
  - Two phase commit transaction.
  - Property has to be flagged as "transactional".
  - Requires the timing system to fire "Commit" or "Roll-back" event.
- Result:
  - Completely handled by the Framework.
  - Custom Code has only to supply a "ValidateSetting" method.



Other extensions

- Composition relation-ship between FESA classes.
  - Façade: complex FESA class can decomposed into sub-FESA classes.
  - Composition: “Has-a” relationship.
- Critical Settings Management: guarantees setting integrity for critical parameters. Implementing using public-key cryptography and a digital signature.
- Run-time diagnostic: “topic-oriented” diagnostic to have a finer granularity in the trace options.
- Monitoring: permanently survey the scheduling and the control flow of any equipment software.

Again all these extensions have been managed transparently for the Equipment Software.

Conclusion

- In spite of the huge diversity of devices, FESA has successfully standardized a high level language and an object oriented framework. About 250 FESA classes deployed on ~ 600 front-end computers (most of them on the LHC, but also on the LHC injectors).
- FESA reduces the time spent developing and maintaining equipment software and brings a strong consistency across all equipment software.
- The FESA development environment is based on a modelling tool, and keeps model and implementation synchronized.
- FESA provides an "XML-Centric Equipment Software design" approach.
- All the recent extensions have proven the flexibility and the capability of the complete FESA infrastructure to handle evolution.