Integration of Renovated Networking Middleware into a Running Control System Environment

U. Krause

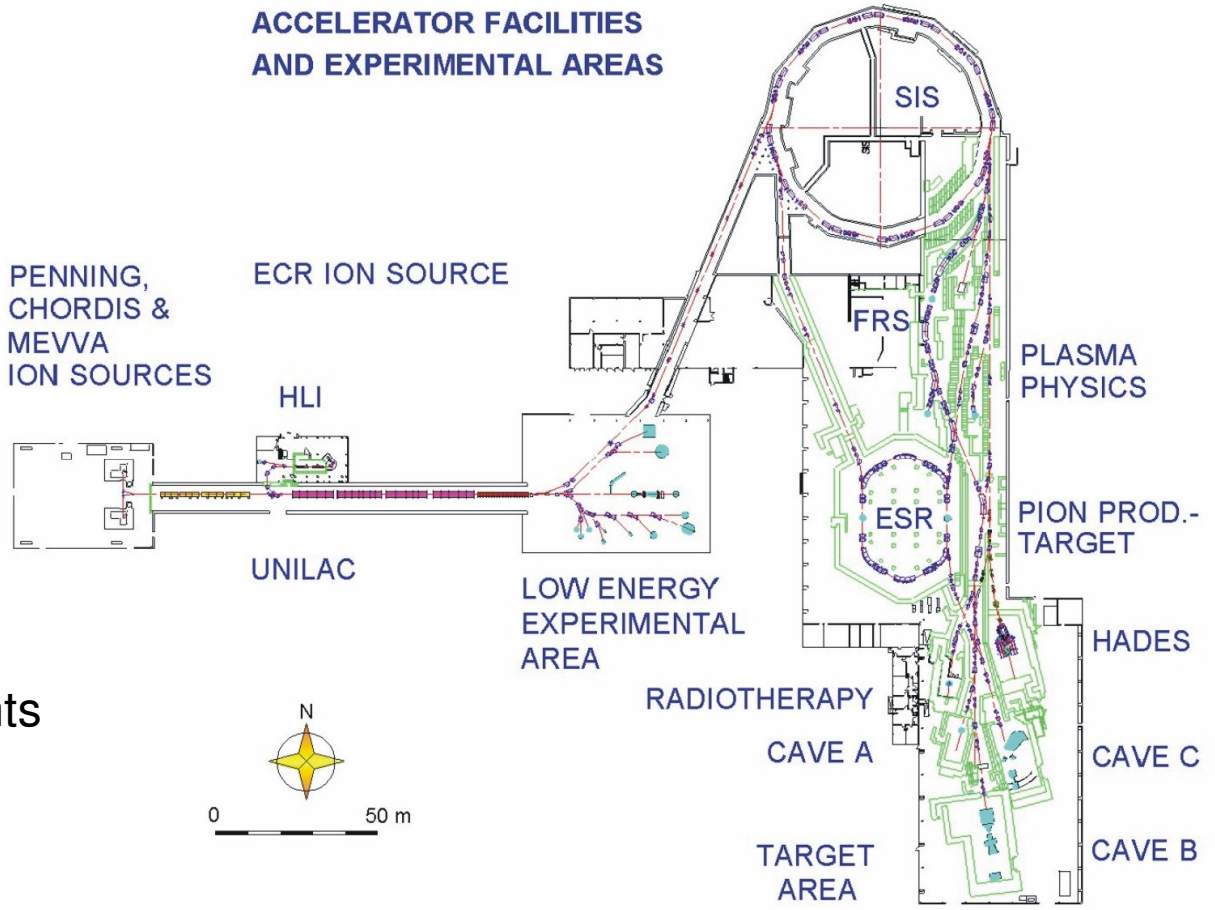
GSI Darmstadt

GSI Accelerators

Elements:

H ... U

- UNILAC
  - 15 MeV/u
- SIS
  - 2 GeV/u (Ne)
  - 1 GeV/u (U)
- ESR
  - In-beam experiments

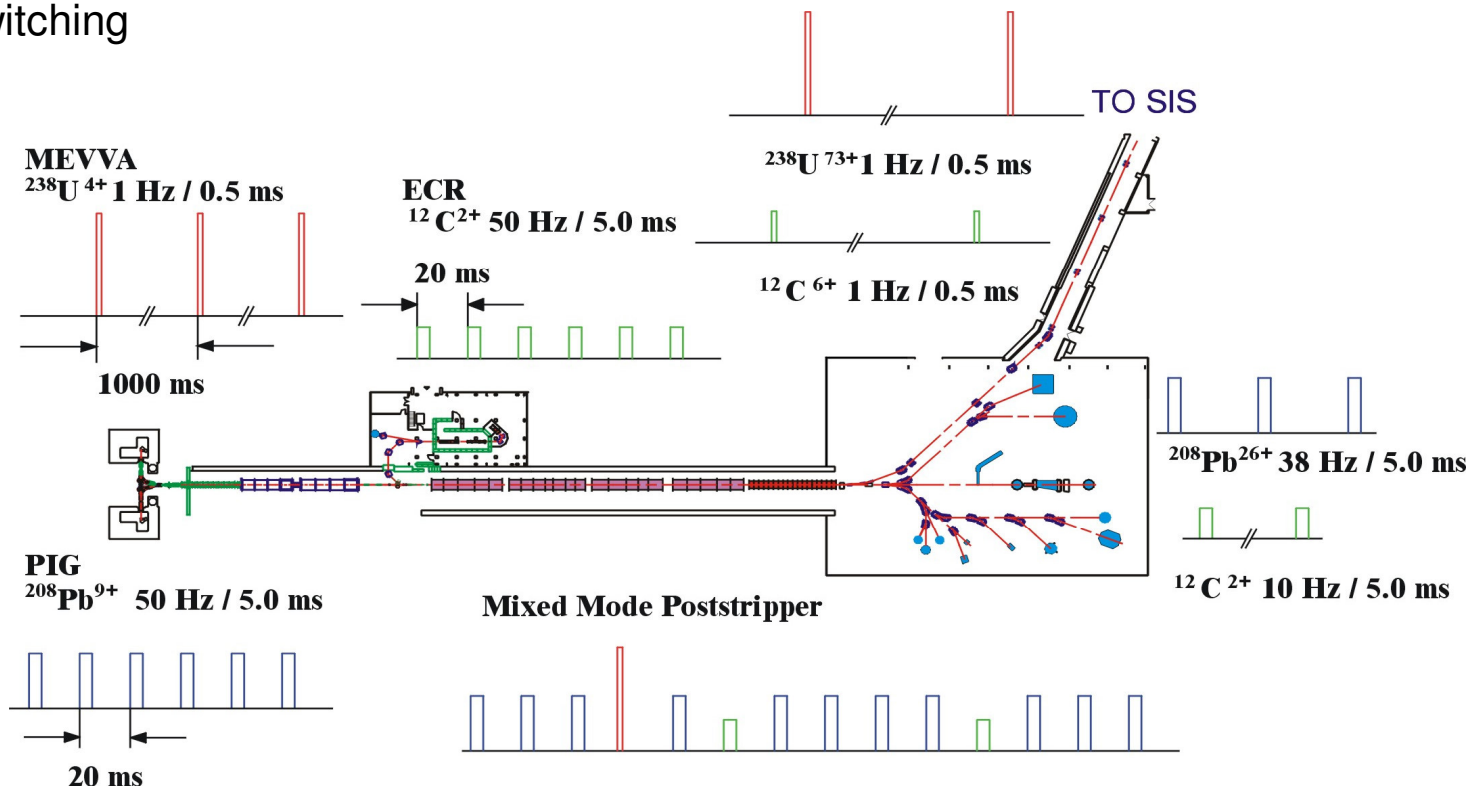


Accelerator Operation

~ 5 experiments in parallel

Pulse-to-pulse switching

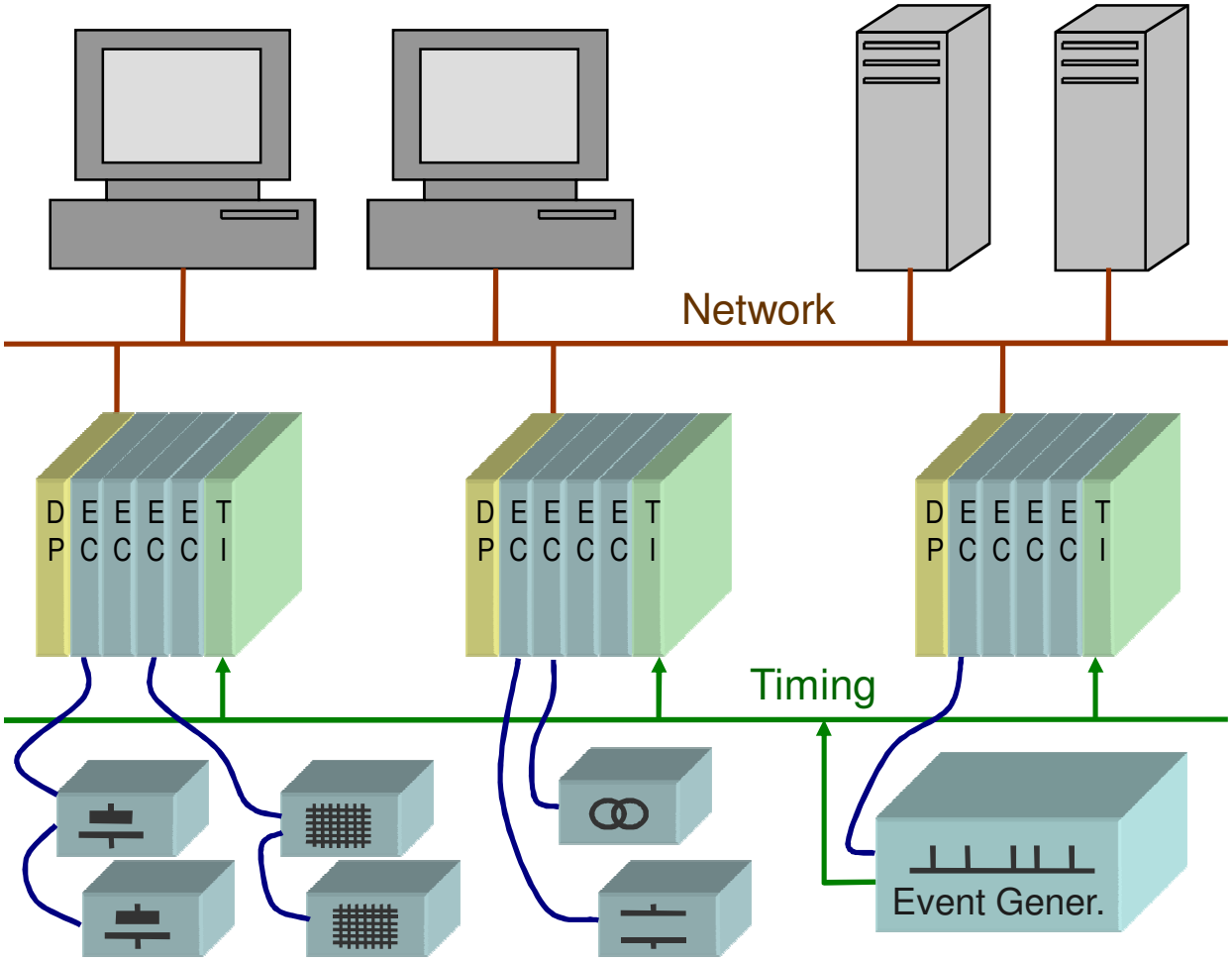
Different beams



Control system: Support operation

Operation's applications, equipment control

Control System Outline



Operation

OpenVMS

Operator GUIs

Services

Logging,
Databases,
…

Front-End

VME

Device **P**resentation

Equipment Modeling
Remote Access

Equipment **C**ontrol

Precise Real-Time

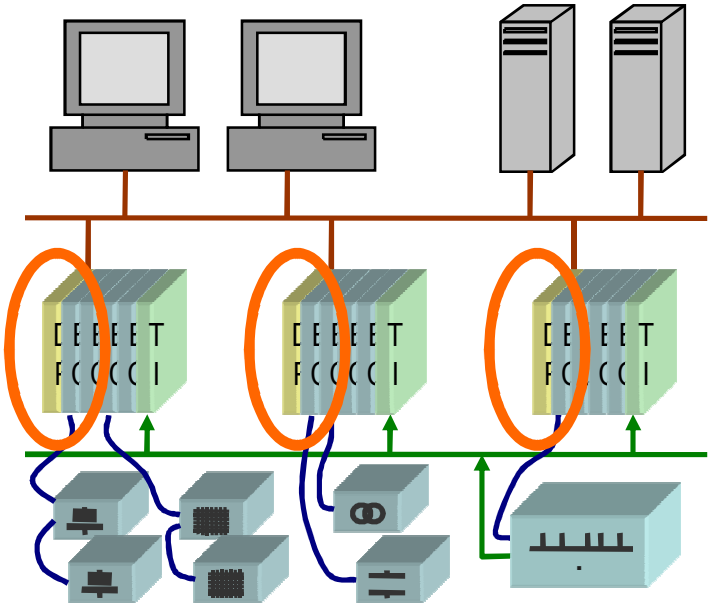
Timing **I**nterface

Devices

Fieldbus (*MIL 1553*)

Starting Point

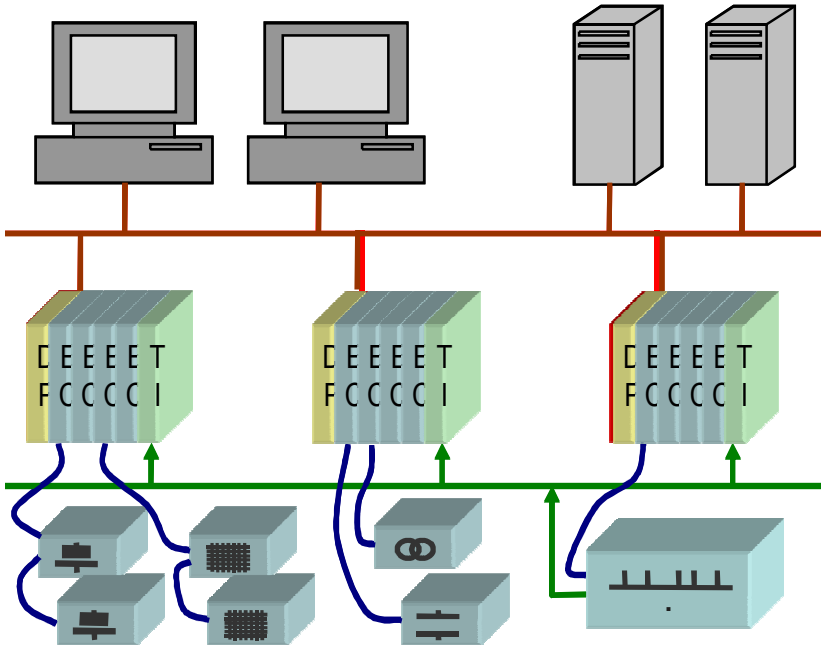
- Controls commissioned 1987
- Accelerator operation refined
  - Operation's applications
  - Equipment handling
    - Front-end SW adjusted, tuned
- Modernizations postponed
  - Cross connections
    - Change one area, adapt rest of system
  - In-house components
    - Home-made network protocol
- Limited to original platforms
  - Specific VME-boards (M680x0)
  - OpenVMS



- Device Presentation
  - Out of stock since long
  - Replacement urgent

Device Presentation Controller

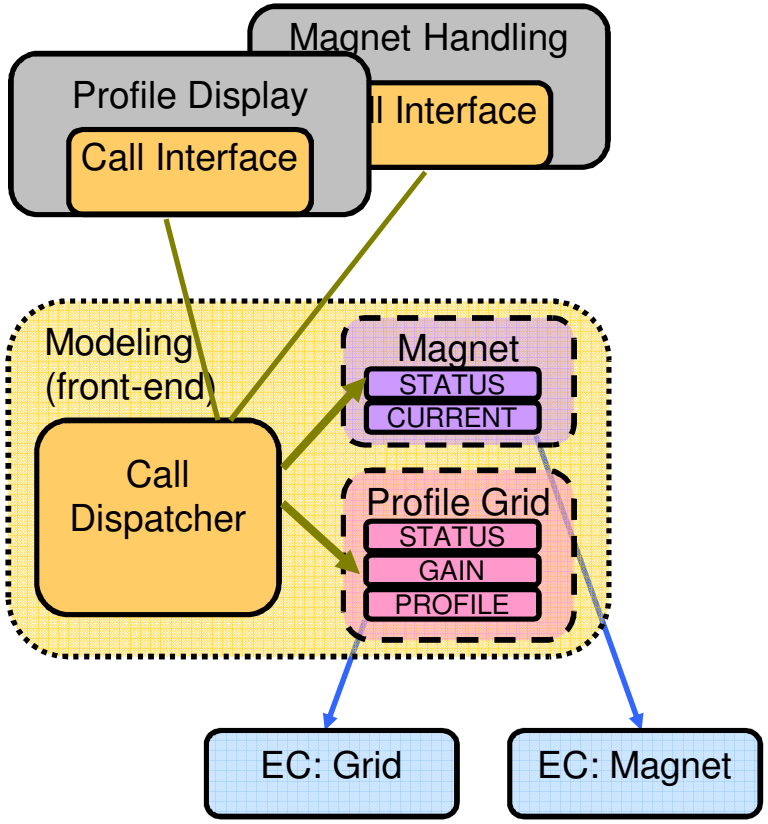
- Replace board, keep system?
  - Keep old structure
  - Conserve limitations
- New control system?
  - Re-Implement specifics again?
    - Front-end
    - Operations applications
- Re-build communication layers
  - Device presentation SW
  - Network SW
- Keep other controls elements



Software Outline

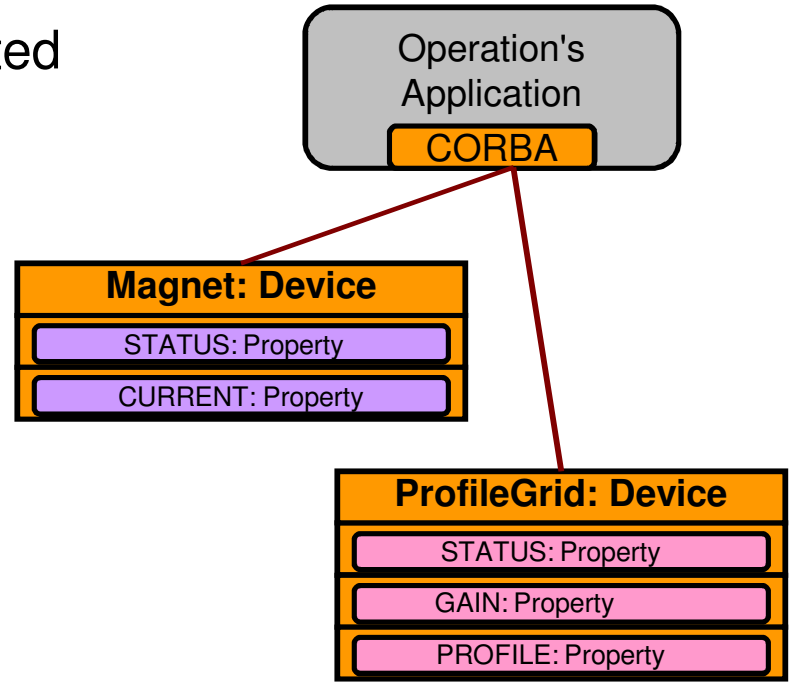
- Operation's applications
  - Common call interface
- GSI equipment modeling:

  Represent as
  - Devices
    - Magnets, profile grids, …
  - with properties
    - STATUS, CURRENT, ...
- Property-implementation
  - Functions on device presentation board
    - Equipment specifics
  - Tightly connected to equipment controllers



Rebuild Communications

- Equipment modeling: Truly object-oriented
  - Device-class
  - Hosting property-objects
- Remote access: CORBA
- Front-end: PowerPC / Linux



Fit into remaining elements:

- Operation's applications
- Equipment controller

Use existing property implementation

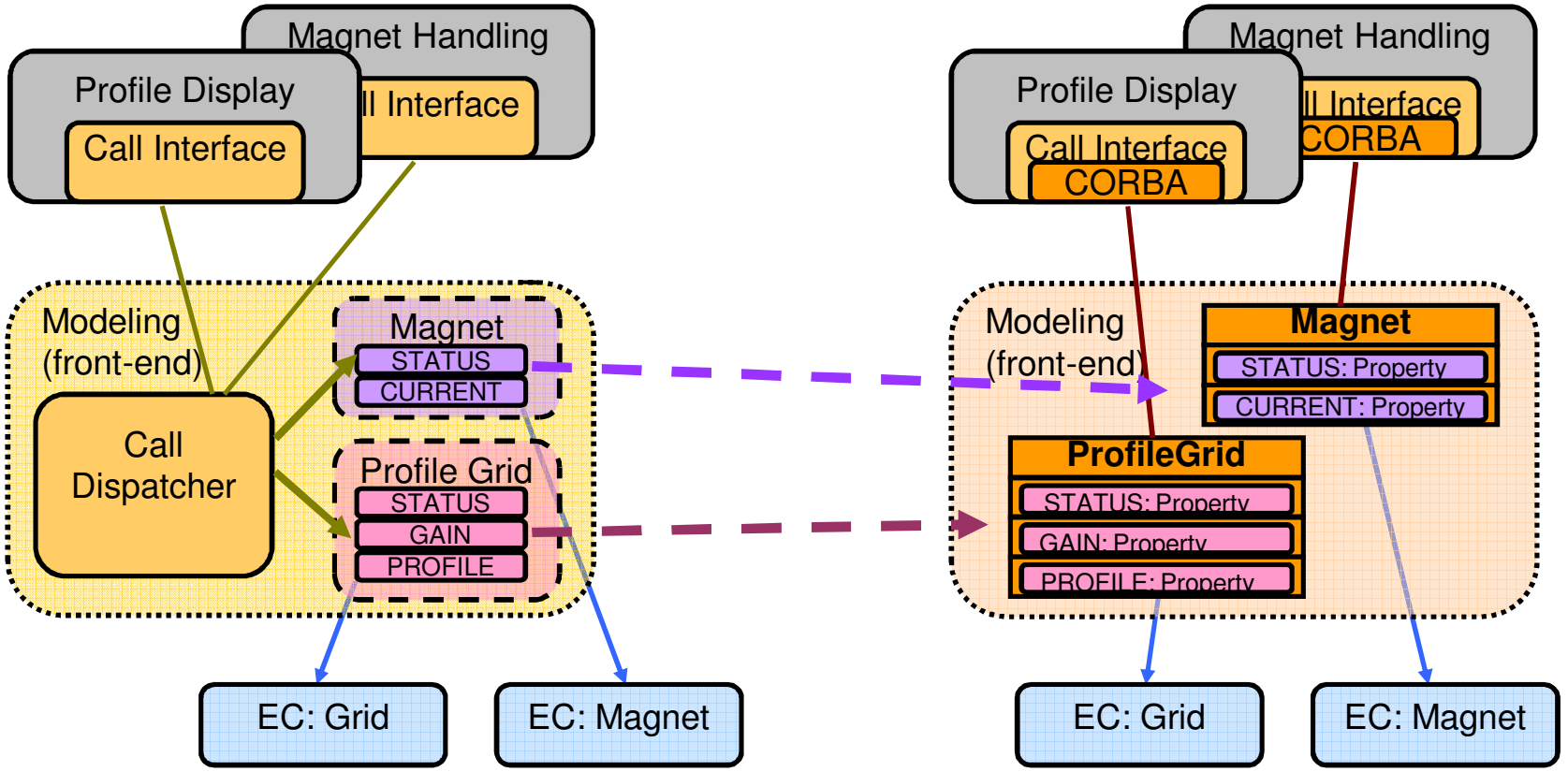
Integration

Previous call interface:

Shell on top of CORBA

Existing property implementation:

Execute-method of property-object



Status

- New communication core is operational
  - Installed in 6 out of 43 VME crates
    - Handling 7 out of 70 device types
- Negligible impact on accelerator operation
- Installation crate by crate
  - In shutdown-times (4 per year)
  - After front-end SW is adapted

Achievement

- Established functionality provided again
  - Existing functionality was primary goal
- Using modern VME-board

- Hard work - Progress?

- Development with nowadays knowledge
  - and nowadays technology
- New communications core:
  - Drawbacks of previous implementation avoided
  - Clearer, more solid
  - Higher flexibility
  - Simpler handling, ease of development

Platforms

Front-end server:

- Target: Linux PowerPC-VME
  - Joined with equipment controller
- Devices without timing control:
  - Dependency on EC removed
    - Trigger: Fast hack, for easier testing
  - Linux server
  - Windows server

Client access:

Linux, Windows

C++, Java, Python

Former call interface:

OpenVMS (Linux)

Fortran, Pascal, Modula-2, C

microIOC M-Box: Motion control

- Embedded Linux controller
- Installed: GSI front-end server

First device, directly connected to network

Established Property Data

- Wide spectrum of data types
  - 8, 16, 32 bit signed / unsigned int
  - 32, 64 bit float
- Single value and array
- Automatic conversion client ⇔ server
  - Byte order client/server
  - Any type to any type
- Added: Mixed types
  - Different types in same access
  - Raw binary format
  - Conversion by application

Wish:

Comprehensive set of device data

In one access

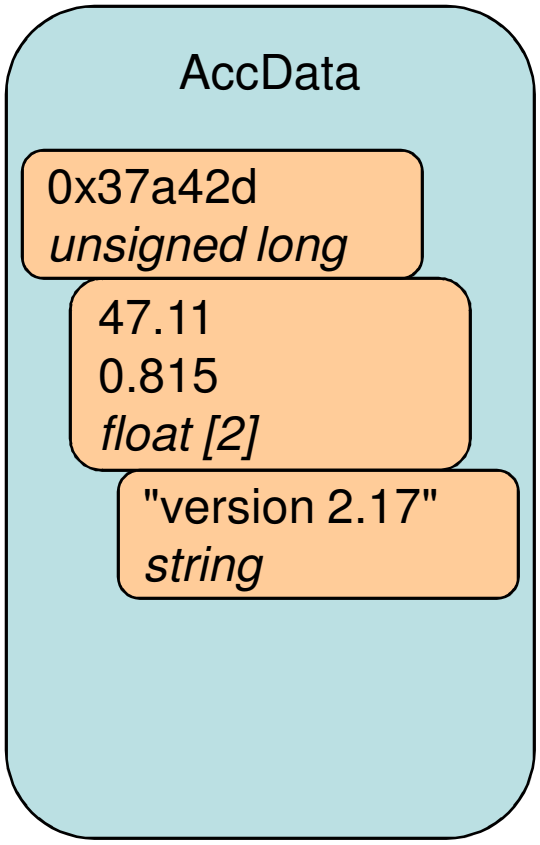
Often: int *and* float

Element	Type
status	unsigned long
currents (ref)	float
currenti (act)	float
. . .	. . .

Data Container

Device data: Container Object

- Any sequence of base types
- Insert / extract data
  - Single elements
  - Arrays
- Data combined with type-information
- Extraction:
  - Type-check
  - Automatic conversion, if requested



Code Generator

- GSI: 70 equipment types
  - ~20 specific properties each
  - Adaptation fully hand-made?
- Formal property-description
  - Name, data types, …
  - As XML-Files
- Generate automatically
  - Frames for connection to device-classes
  - Setter/getter for property data
    - Address by name
  - Paper-documentation
    - For operations developpers

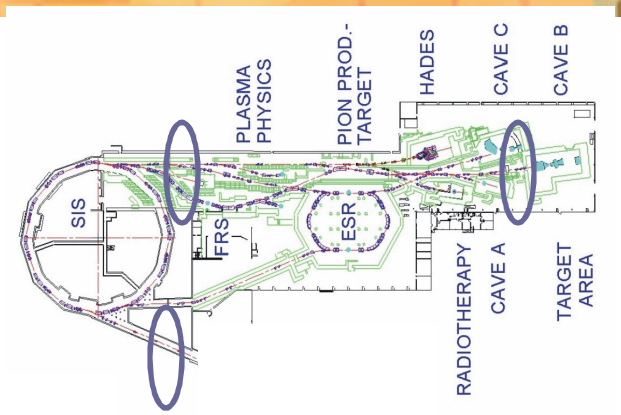
```
<data>
<value type="ULong"
   name="masterStatus"></value>
<value type="Float32"
   name="currents"></value>
<value type="Float32"
   name="currenti"></value>
</data>
```

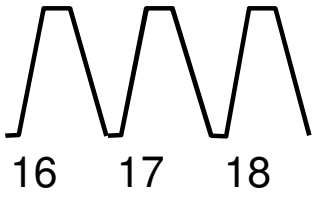
```
dataP->masterStatus(mdp->mSts.ulong);
dataP->currents(sRfcP->origCurrentS);
dataP->currenti(locCurrent);
```

Subscription Service

- Need for data correlation
  - Different devices
  - Same cycle
- Previously: Side effects used
  - Block-commands
    - Several devices in one access
    - Devices on same VME crate
  - No longer possible

- Subscription service:
  - Arrange subscribed data
  - According to cycle ID
  - On top of access interface





Device A	Device B	Device C
Cycle 16	Cycle 16	
Cycle 17	Cycle 17	Cycle 17
Cycle 18		Cycle 18
Cycle 19	Cycle 19	Cycle 19
	Cycle 20	Cycle 20
	Cycle 21	

Conclusion

- Evolution of long existig system
  - Evolution: Not always small step
    - Big modification
  - Change is localized
    - Most controls parts kept untouched
    - Compatible: Installation are by area
  - Low risk for accelerator operation
- Significant step foreward
  - Flexible state-of-the-art core
- Solid basis for future
  - Further upgrades
    - Migration OpenVMS $\Rightarrow$ Linux
  - Interoperation with other systems
    - Integration into FAIR facility

Thanks to team

L. Hechler

K. Herlo

K. Höppner

P. Kainberger

S. Matthies

G. Schwarz