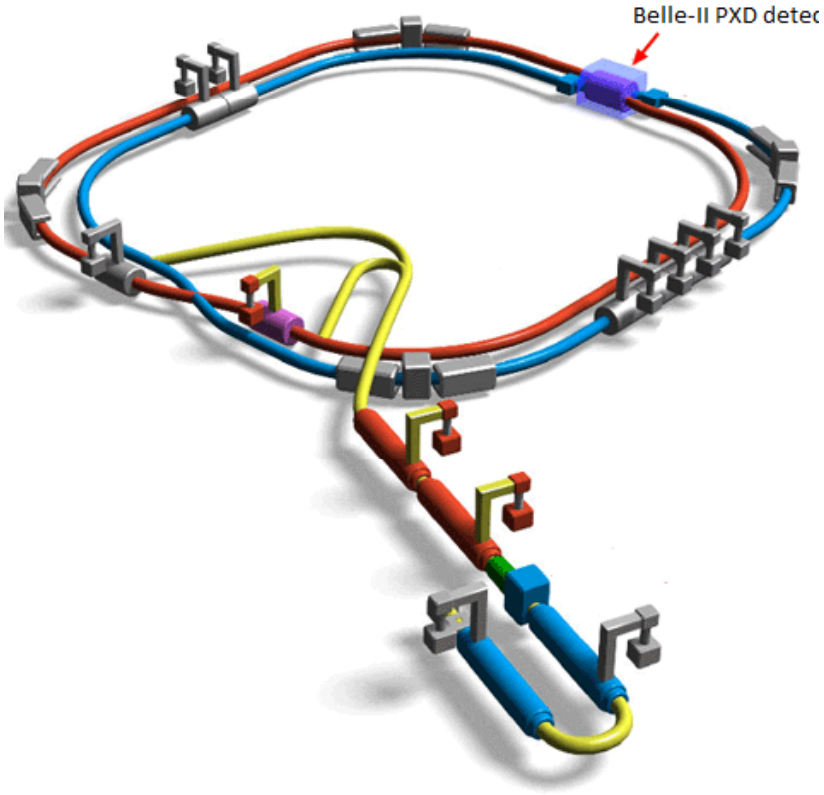
Belle II PXD Slow Control Considerations



Prof. Dr.-Ing. Alois Knoll

Stephan Sommer

Robotics and Embedded Systems
Department of Informatics
Technische Universität München

www6.in.tum.de

18 Jan 2012

Outline

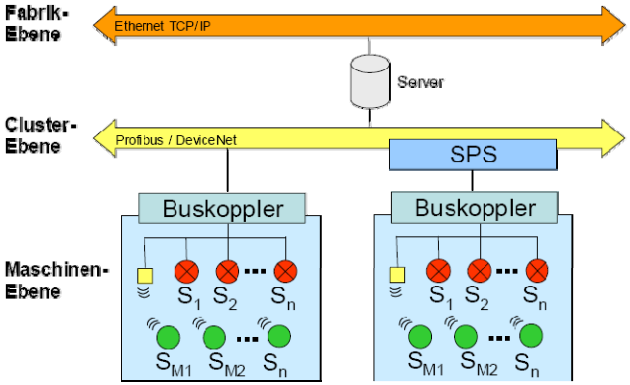
- Introduction
  - Sensor networks for process monitoring and control
  - Fault tolerant systems
- General Slow Control Design Rules
- Slow Control for the Pixel Vertex Detector
- PVSS II – The Slow Control System for the PDX
  - Overview
  - Architecture
  - Selected Modules
- Summary

Multifunk: Autonomous, Self-Organizing Sensor Networks for Process Monitoring

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Goals:

- **Wireless sensor network**-based monitoring of industrial production processes for **quality assessment** and reduction of **energy consumption**
- **Model-driven design tools** for high-level system specification
- Optimized **firmware code generation**
- Automatic implementation of extra-functional **quality-of-service requirements**
- Network **self-organization**



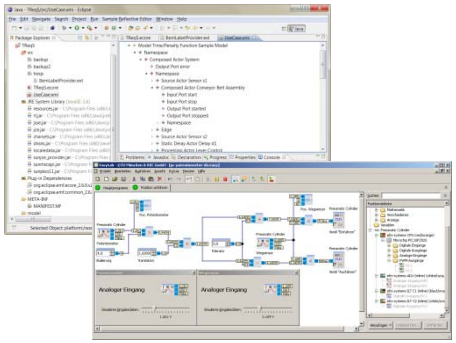
Sample Multifunk Network Topology

Prototypical applications:

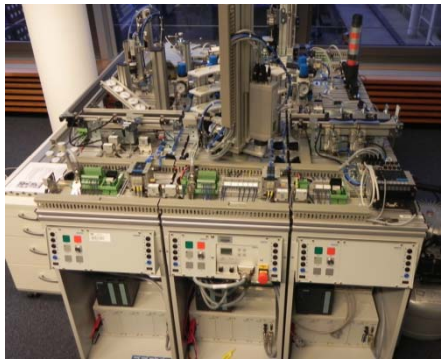
- Development of a monitoring sensor network for waste treatment facilities
- Development of a generic automation line monitoring sensor network

Technologies: Eclipse Modeling Framework (EMF)

Platforms: PC, ARM



Modeling Tools



Demonstration Platform

FTOS: Model-Driven Development of Fault-Tolerant Systems

- Creation of a programming framework for fault-tolerant, distributed, real-time system design with a sound formal basis
- Focus on programming applications that have traditionally been designed without or with just minimal degrees of fault tolerance
- It should be possible to handle **all types of software and hardware faults**
- **Full tool chain**, from specification to code generation for a variety of platforms

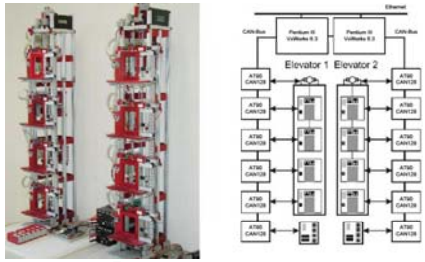
  The operating system must provide basic support for guaranteeing real-time constraints, supporting fault tolerance and distribution, and integrating time-constrained resource allocations and scheduling across a spectrum of resource types, including sensor processing, communications, CPU, memory, and other forms of I/O.
- **Reference**: Christian Buckl. *Model-Based Development of Fault-Tolerant Real-Time Systems (Chapter 3)*. PhD thesis, Technische Universität München, October 2008.
- **Partners**:

Kayser-Threde TUM – Lehrstuhl Prof. Broy

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Elektrik Car

Automation



Demonstrators

Slow Control Design Rules
(A. Knoll, C. Kiesling, J. Schieck)

- **Virtuality** to provide a “distributed virtual control room” to users
  - Users should be able to carry their “own”, customizable control room with them to support their work (e.g. iPAD, Laptop)
  - Data and control information should be available independently of any device
- Support **connectivity** to allow remote control by authorized users from all over the world
  - The location of an operator or researcher should not matter for accessing data and control information
  - But: Access must be securely restricted to authorized users and pre-defined modes of operation (e.g. maintenance)
- **Unified access** to all components in the same way to provide uniform “look and feel”
  - The same components should look and behave always the same, regardless the component developer and visualization device
  - e.g. iPAD application for maintenance and configuration (➔ virtual control room “to go”)
  - ➔ Design guide / templates for “common” devices

Slow Control Design Rules, ctd.
(A. Knoll, C. Kiesling, J. Schieck)

- **Abstraction** of concrete hardware devices and clustering of functional aspects
  - To control and monitor devices, only functional parameters are relevant
  - Devices and software components providing the same functions should be represented in the same way regardless of their concrete implementation; only functional aspects matter from the user perspective
- **Plug-and-Play** to allow component upgrade and exchange with minimum effort and operation interruption
  - Components (software and hardware) need to be exchanged or upgraded from time to time
  - A flexible infrastructure is required, where components subscribe and publish their data
  - ➔ Components should operate independently (as far as possible) and only communicate using "topic oriented" messages
- **Serviceability**, as a key feature to support "live documentation"
  - As software and systems are getting more and more complex, up to date documentation is indispensable, but:
    - Software is refined and hardware is parameterized ➔ Documentation must be kept in sync manually
    - ➔ Documentation (Model) should be the source of configuration and state as "live documentation"
- **Fault Tolerance** to allow reliable operation of the system
  - Based on the requirements, fault analysis is necessary and a concept needs to be developed to provide sufficient fault tolerance capabilities

Slow Control for the Pixel Vertex Detector

- Application of the design rules to the PXD Slow Control:
  - Creation of a homogeneous and scalable infrastructure for slow and probably run control
  - Integrated status monitoring and fault detection
  - Plug&Play support to integrate new or evolving software and hardware components
  - Security layer to support save off-site access to experiments and data
  - Tailoring and deployment of the whole system within a tight time schedule

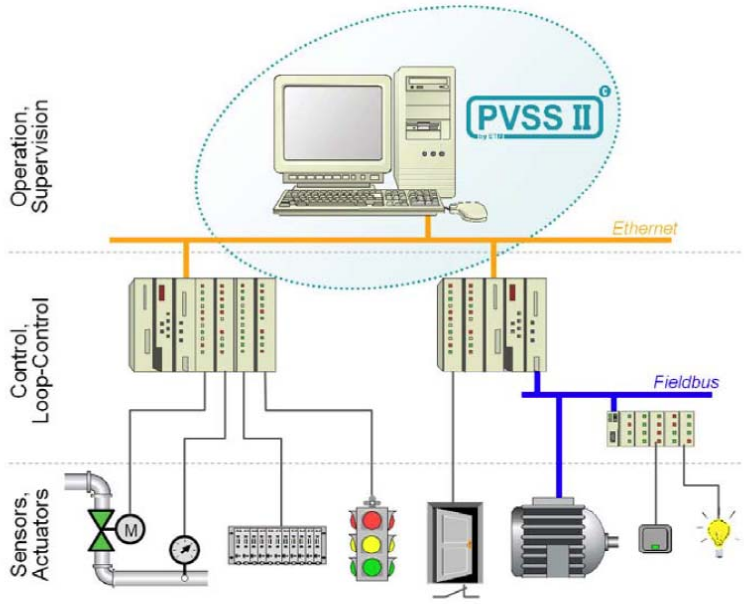
➔ A new solution satisfying all requirements can not be developed within reasonable time and budget

➔ An of the shelf product needs to be found which can be used as basis and provides sufficient options for extension

- Result of market analysis:
  - PVSS II (now WinCC Open Architecture) provides the necessary basic infrastructure and programming interfaces
  - PVSS II needs to be integrated anyway because it is used for the cooling

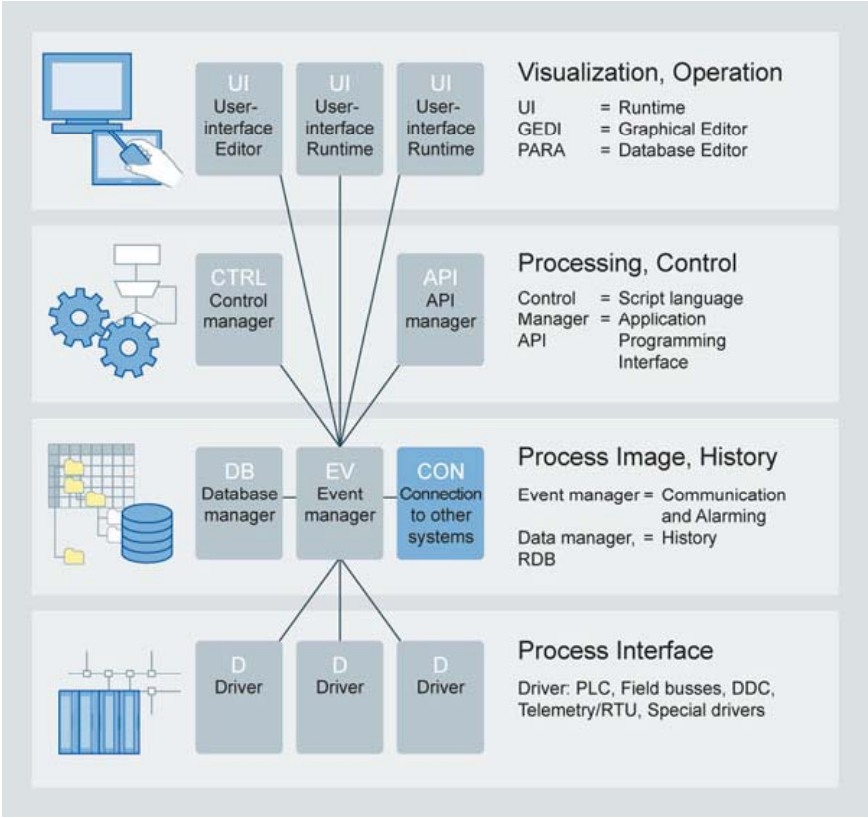
PVSS II – Short facts

- PVSS is used both in ATLAS (slow control) and LHCb (slow+run control)
- PVSS is a scalable solution that can be executed in a distributed system (both Linux and Windows are supported)
- User interfaces can be developed easily with PVSS II
- PVSS is extendable by additional software components
- Completely event-oriented, telegram-based communication
- Redundancy: hot standby system



PVSS II Architecture
(now WinCC Open Architecture (Siemens))

- **Event-Manager (EV)**
  - The heart of a WinCC Open Architecture system
  - Distributes all kind of events and stores them in a database
- **Driver-Manager (D)**
  - Processes are linked via any number of drivers
  - Interconnects devices and external software modules
- **Data- Manager (DB)**
  - Stores process changes in a high-speed database
- **Control-Manager (CTRL)**
  - Runtime environment to process programs written in the Control programming language
  - Multitasking and Event-driven
- **User Interface-Manger (UI) –**
  - Takes care of graphical display of process states
  - Data can be visualized in different views



http://www.etm.at

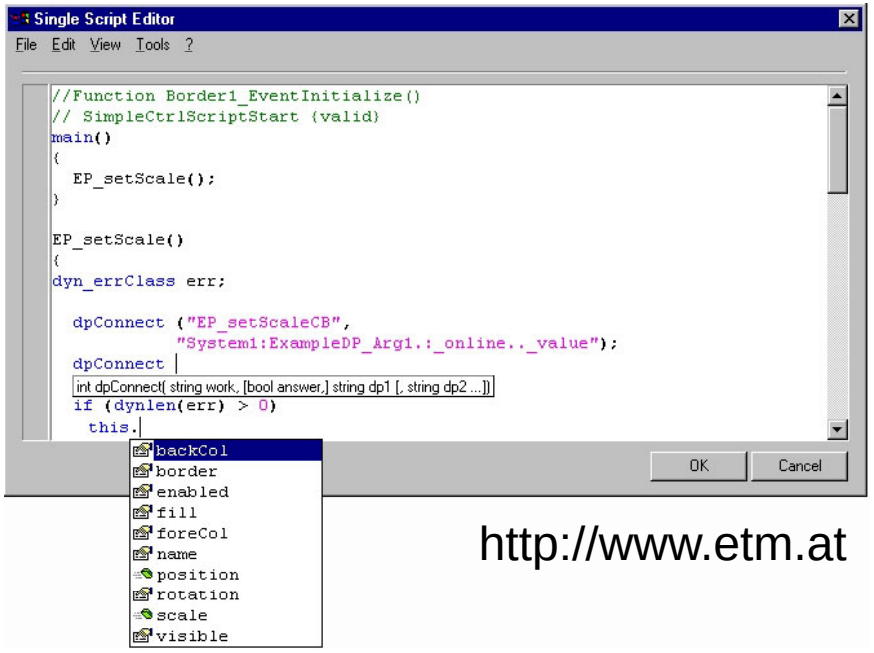
Driver Manager - Access to “real” Devices:

- Driver support for industry standard devices, otherwise C++ API
- Each external device in PVSS can be modeled as a (hierarchical) finite state machine
- Events are triggered and received by devices

- OPC UA (Client and Server)
- OPC Data Access (Client and Server)
- OPC Alarms & Events (Client and Server)
- Field bus:
  - S7 Communication
  - Profibus DP
  - TLS
  - MP
- Telecontrol systems:
  - SINAUT
  - DNP3
  - IEC 60870-5-101, -104
  - SAT-SSI
  - RK512, 3964R, etc
  - Dynamic Logic
- Applicom General Driver Interface (Ethernet)
- Direct TCP/IP drivers:
  - Industrial Ethernet S7
  - Modbus TCP
  - Teleperm
- Other drivers on request or via C++ API

Application Control

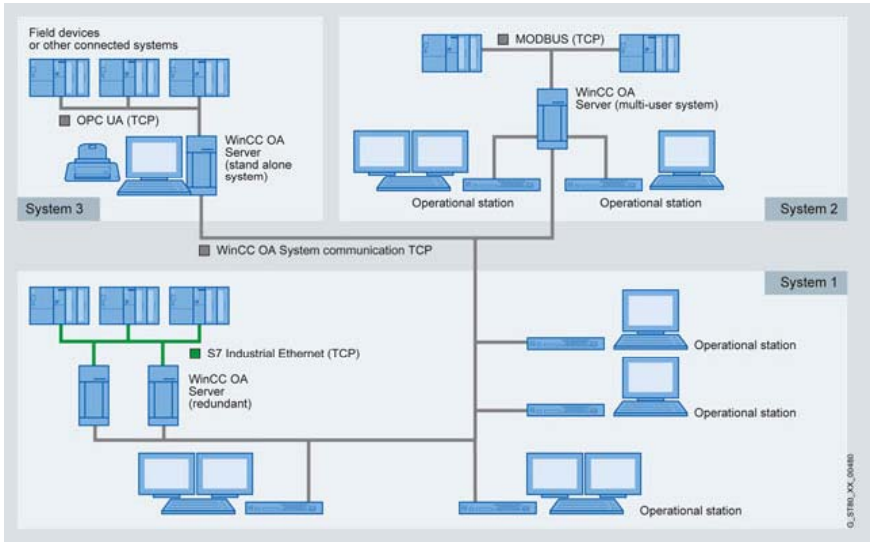
- Application Control is performed using Control Units:
  - Events are transmitted to Control Manager and trigger actions
  - System behavior is modeled using finite state machines or coded using script language:
    - External DLLs and shared libraries
    - Database interface to relational databases (e.g. Access, Oracle, MS-SQL)
    - Editor with syntax highlighting, function tool tips, Intellisence, syntax checking, etc
    - Extensive debugger for fast troubleshooting
  - Finally, events are emitted and distributed by the Event Manager



http://www.etm.at

Interconnecting System to a “System of Systems”

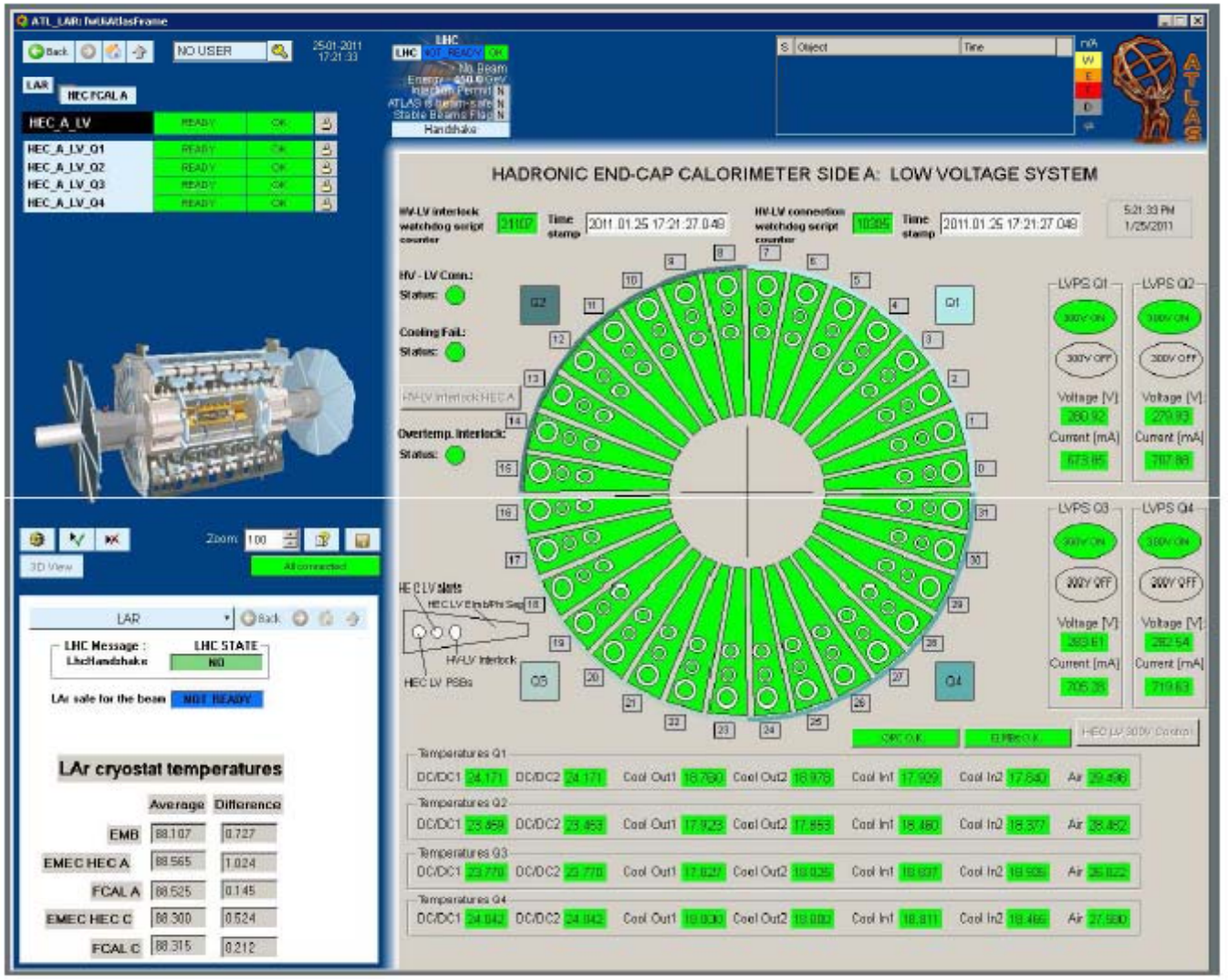
- Transparent access to all distributed system data
- Commands and alarms of a subsystem can be kept local or distributed to every connected system
- Sub-systems itself can also consist of many sub-systems
- Systems can be singular or redundant
- System can be parameterized easily and extended stepwise



➔ Each target system could use it’s own PVSS system, but systems can be treated as subsystems and interconnected

http://www.etm.at

CERN Atlas Detector Visualization using PVSS II

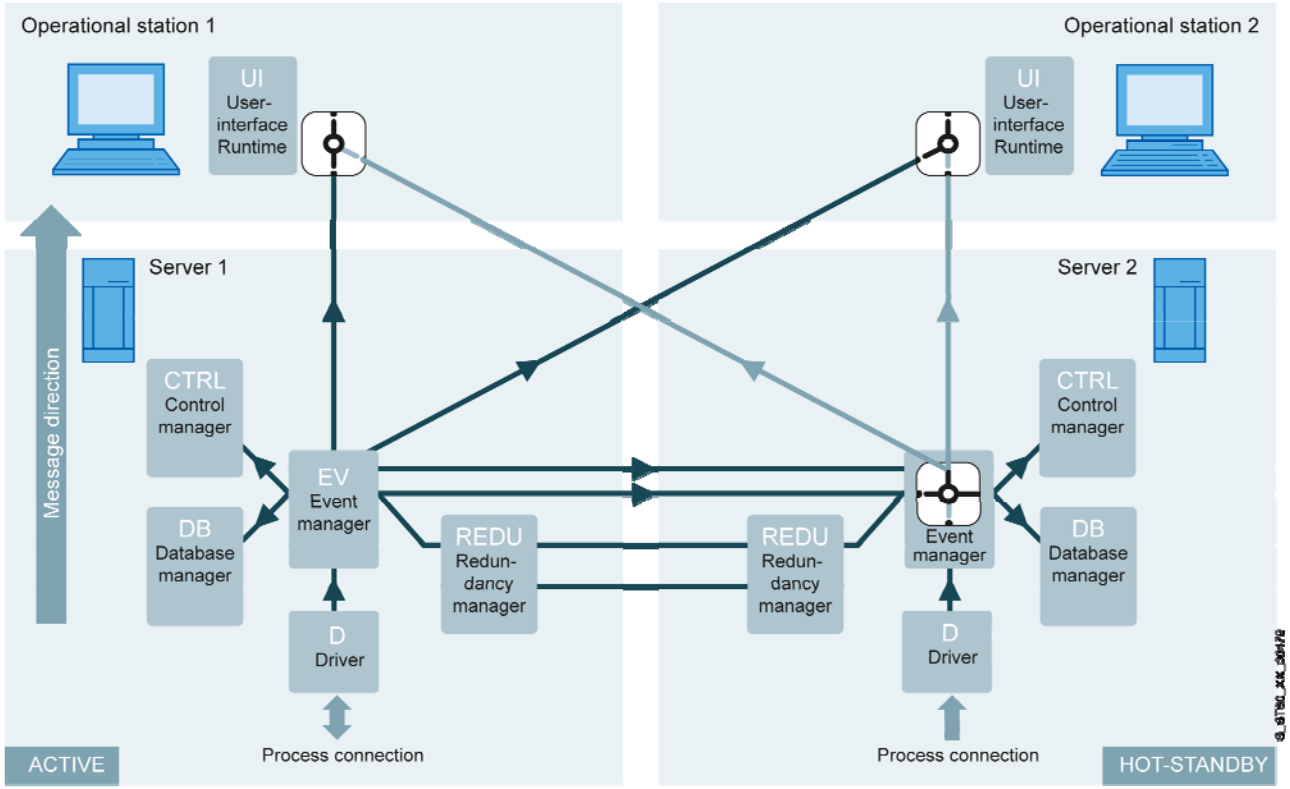


Summary

- Use of a (commercially) available solution most reasonable
  - A flexible and powerful control and monitoring environment can be deployed with reasonable effort, cost and risk.
  - A homogeneous look and feel throughout the whole application is guaranteed by the application's building blocks (templates), but someone should be responsible!
- Currently, PVSS II seems best choice (for PXD) because,
  - it is used in any case for PXD cooling (which is shared with additional component)
  - it provides a reasonable set of the required features
  - it supports a wide variety of industry protocols and is extendable using "open" interfaces
- Some open issues like desired architecture and cost for PVSS and database licenses
- PVSS II as Slow Control for the whole Belle II experiment?

Thank you for your attention!

Redundancy


Operational station 1
Operational station 2
UI
User-interface Runtime
UI
User-interface Runtime
Server 1
Server 2
Message direction
CTRL
Control manager
CTRL
Control manager
EV
Event manager
DB
Database manager
REDU
Redun-dancy manager
REDU
Redun-dancy manager
Event manager
DB
Database manager
D
Driver
D
Driver
Process connection
Process connection
ACTIVE
HOT-STANDBY