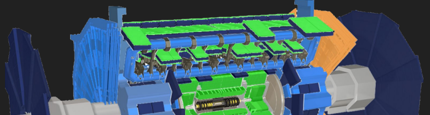
Common Tools for

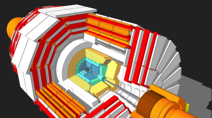
Electronics Integration in Detector Controls

Stefan Schlenker, CERN

With material and input from:
P. Moschovakos, P. Nikiel

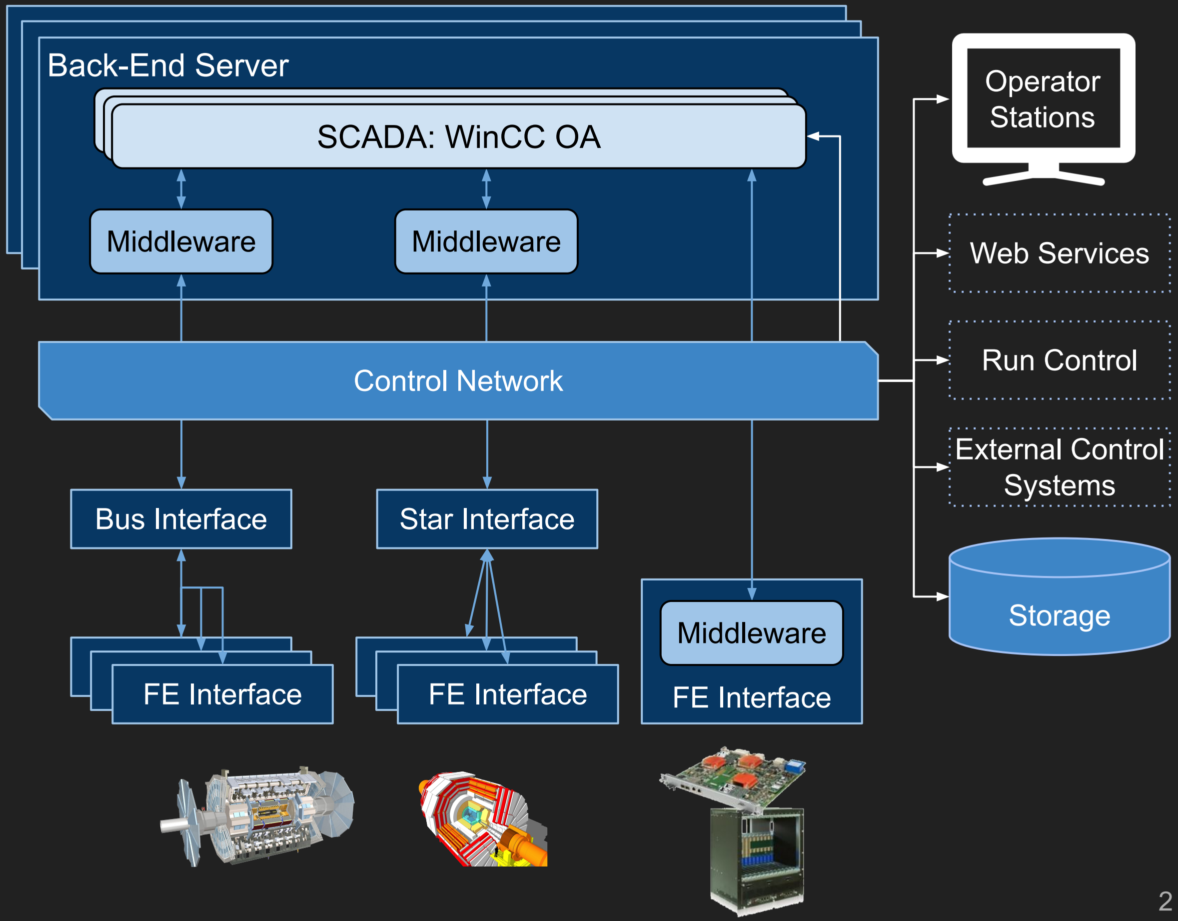
ACES 2020, 26-28 May, CERN



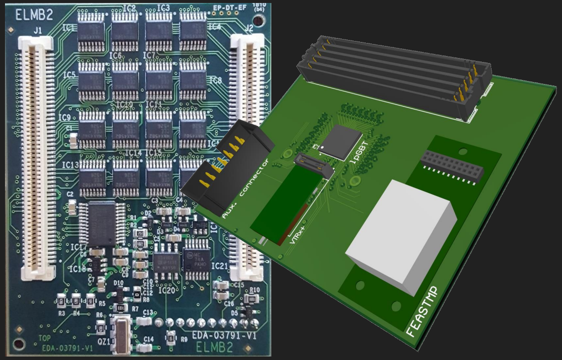


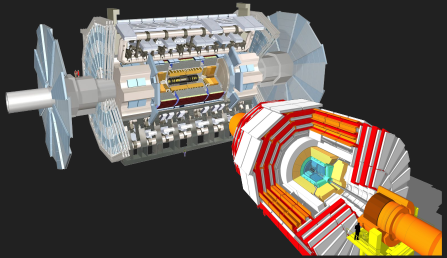
DCS Architecture

- Hardware interface components for detector front-end or back-end electronics with different field topologies
- Middleware as component glue and hardware abstraction layer
- Distributed hierarchy of SCADA applications on back-end servers
- High level layer for operation, storage and link to other systems



Detector Front-End Interface Hardware





Currently Used

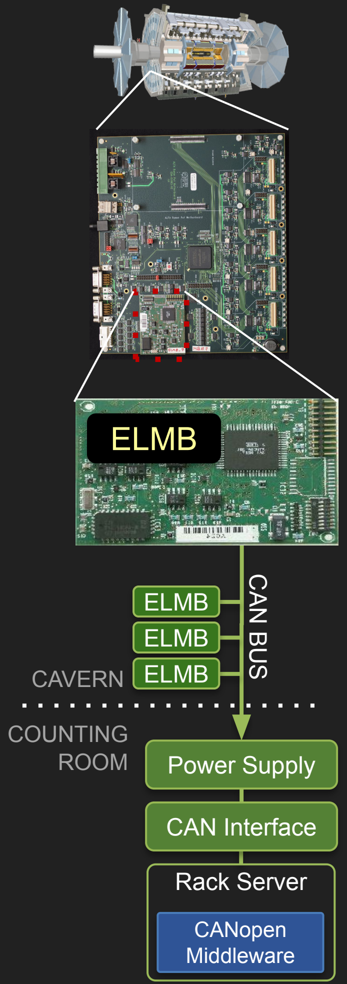
Radiation tolerant **on-detector I/O concentrators**

- **Commercial**, e.g. CAEN
- Custom-built: **Embedded Local Monitoring Board (ELMB)**, previously workhorse for LHC experiments, not fully suited for HL-LHC though

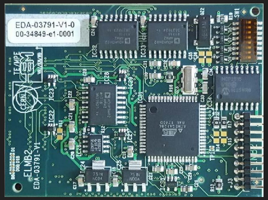
ELMB reminder

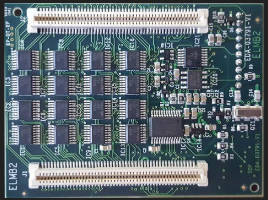
- Specs:
  - MCU, 16-bit ADC, 64 analog inputs (differential, 25mV-5V), GPIOs, SPI
  - Magnetic field tolerance
  - Radiation qualified up to: NIEL - 10^{12} n/cm^2, TID - 14 krad, SEU - 1×10^{-13} cm^2/bit
- Connection with back-end:
  - Isolated CAN interface, bus with up to 64 nodes, CANopen compliant
  - CAN power supply (2x 12V A/D and CAN), custom

⇒ **Address** component **obsolescence and** HL-LHC **radiation**



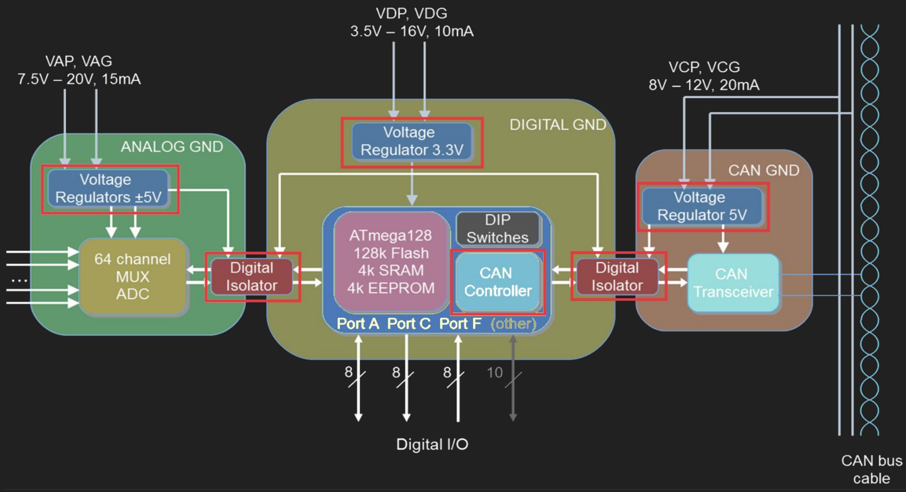
ELMB2



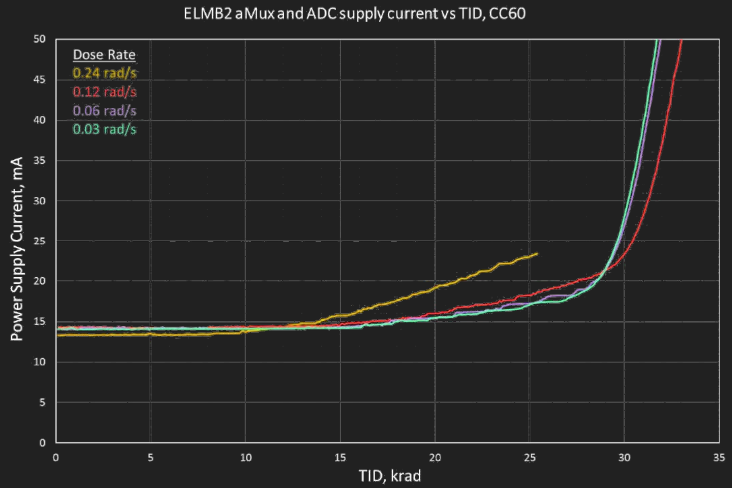


Drop-In replacement for ELMB

- Mainly intended for legacy replacements
- Replaced components (rad sensitive and obsolete ones)
- Improved ADC conversion rate (~doubled)
- Prototype radiation tests:
  - NIEL - 2×10^{13} n/cm^2
  - TID - 32 krad, limited by ADC
- Status:
  - Pre-series done
  - Production being prepared for 3k boards (requests collected by electronics coordinators)
  - Planning to redo radiation and magnetic field tests with production boards



For details, see [D. Blasco et al. (Poster)]

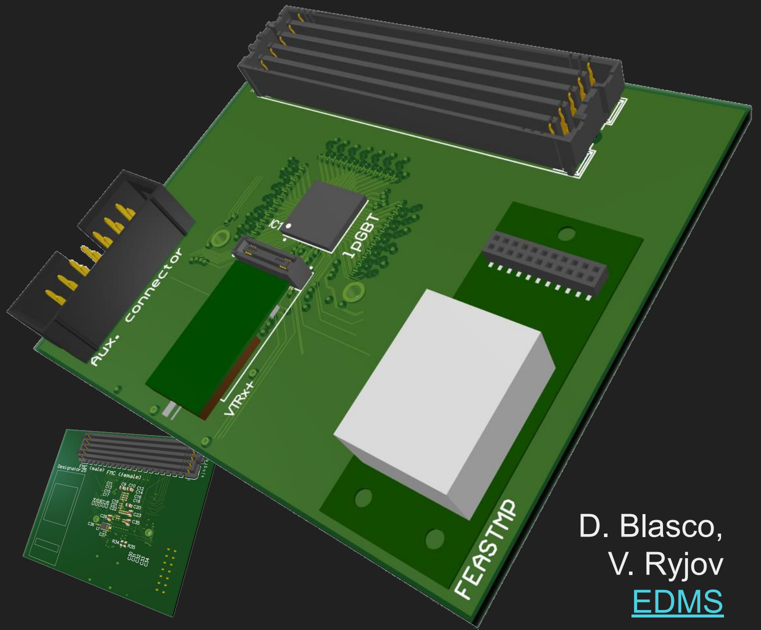


Embedded Monitoring and Control Interface (EMCI)

Move the DCS I/O concentrator concept to the HL-LHC era:

- Based on lpGBT ASIC
- Independent communication interface for FE electronics devices (via eLinks)
- Additional analog and digital monitoring/control (lpGBT-SCA: ADC/DAC, I2C, GPIO)
- Connection with counting rooms via optical link (star topology)
- Radiation-hard components (lpGBT, VTRx+, FEASTMP)
- Use as mezzanine or stand-alone

Status: Layout of prototype ongoing



D. Blasco,
V. Ryjov
[EDMS]

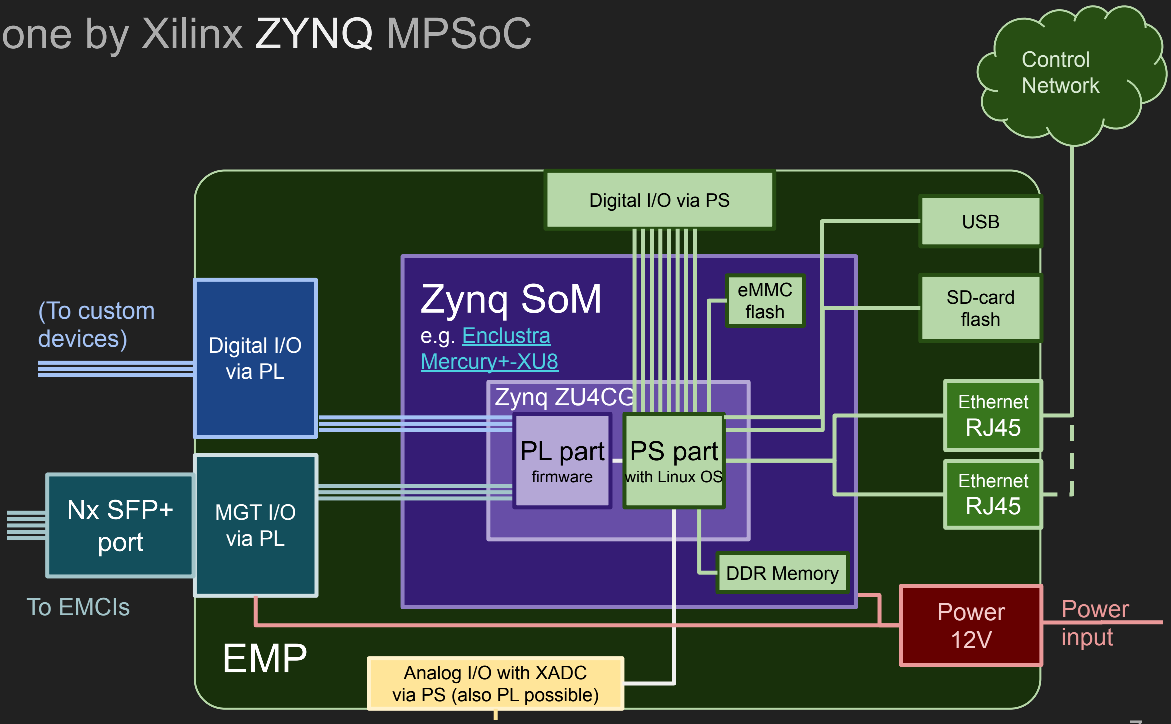
For details, see [D. Blasco et al. (Poster)]

Embedded Monitoring Processor (EMP)

Off-detector receiver for EMCIs

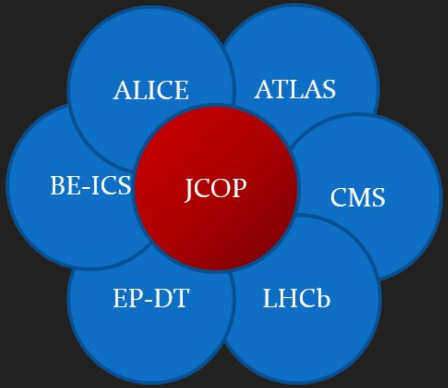
- Several optical links to EMCIs, Ethernet towards back-end
- Processing and I/O interfacing done by Xilinx ZYNQ MPSoC
- COTS components:
  - ZYNQ SoM on custom carrier
  - FireFly or SFP+
- Additional digital and analog I/O (PL/PS)
- Embedding of DCS middleware possible (OPC UA)
- Status: SoM candidate selected, proof-of-concept with fw/sw ongoing

See also D. Blasco et al. (Poster)



Integration Software Toolbox

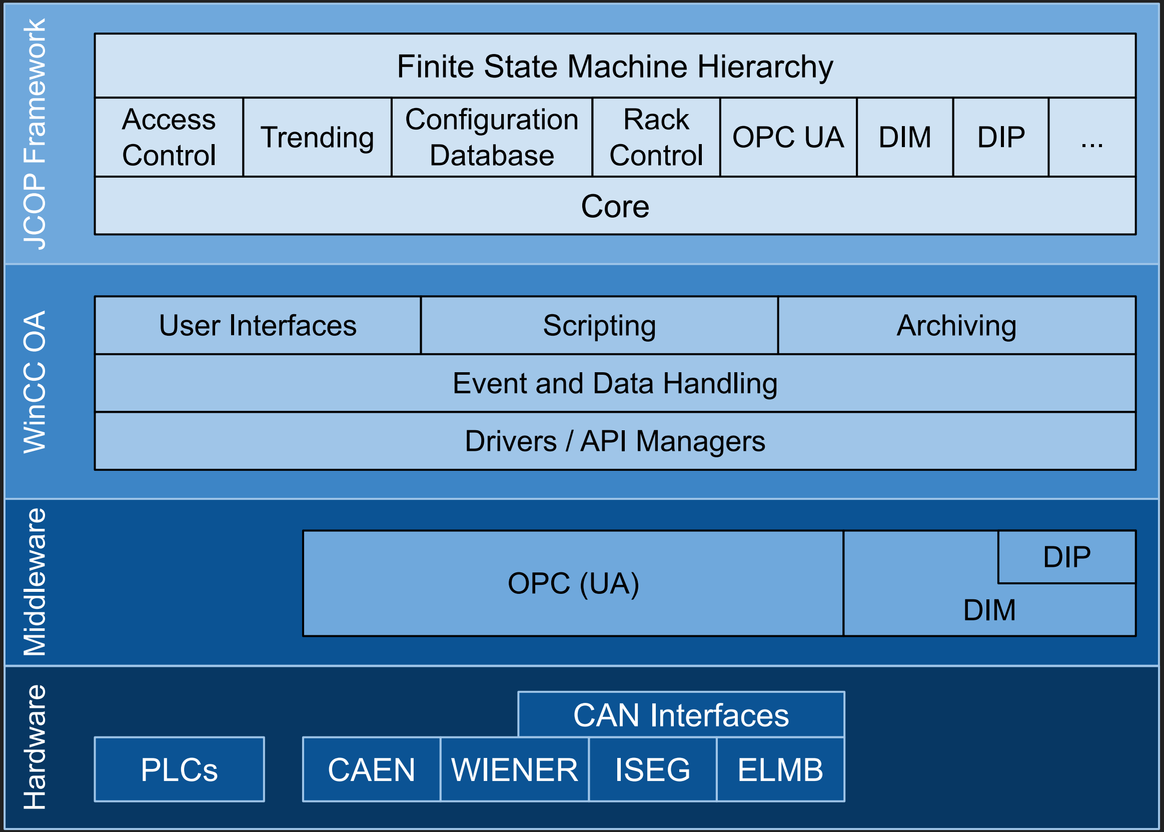
Joint COntrols Project



What is JCOP?

- Collaboration of LHC Experiments, CERN EP/DT and BE/ICS for providing common detector control solutions
- Hosts also common control system projects from EP/DT:
  - Detector Safety Systems
  - Gas and Magnet Control Systems
- Tool set for detector controls:
  - Front-end interface support (e.g. CAN interfaces)
  - Middleware applications for common hardware such as power supplies or I/O concentrators
  - SCADA application WinCC Open Architecture (Siemens)
  - Layer of software applications and tools for WinCC OA: JCOP Framework

JCOP Stack, Recent Developments



(Selected) Developments

- NextGenArchiver (CERN BE/ICS), adds InfluxDB and Custom back-ends in addition to Oracle
- New Alarm Screen (CERN BE/ICS), complete re-write: performance, new functionality and usability
- Migration of legacy OPC middleware to OPC UA

HL-LHC activities require lots of new device integrations such as:

- ATCA shelves and blades
- New power supplies
- FE via VL+
- Custom hardware

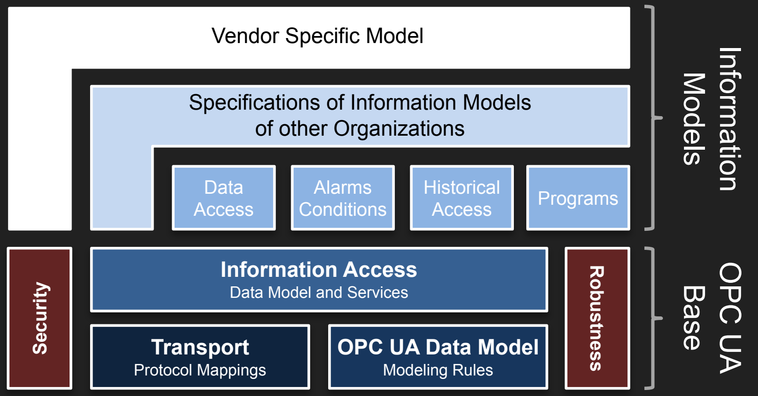
and thus middleware applications

Building Middleware Made Easier

OPC Unified Architecture

Industrial machine-to-machine communication protocol for interoperability

- Originally developed by OPC Foundation for IoT applications (keyword Industry 4.0)
- OO Information modeling capabilities
- Enhanced security, performance and scalability
- Supports buffering, session mgmt, pub-sub, per-connection heartbeats/timeouts, discovery
- Multi-platform implementation, lightweight ➲ embedding possible
- Commercial SDKs available with stack from OPC foundation or open source stack implementations (C, C++, Java, Python, ...) for servers and clients



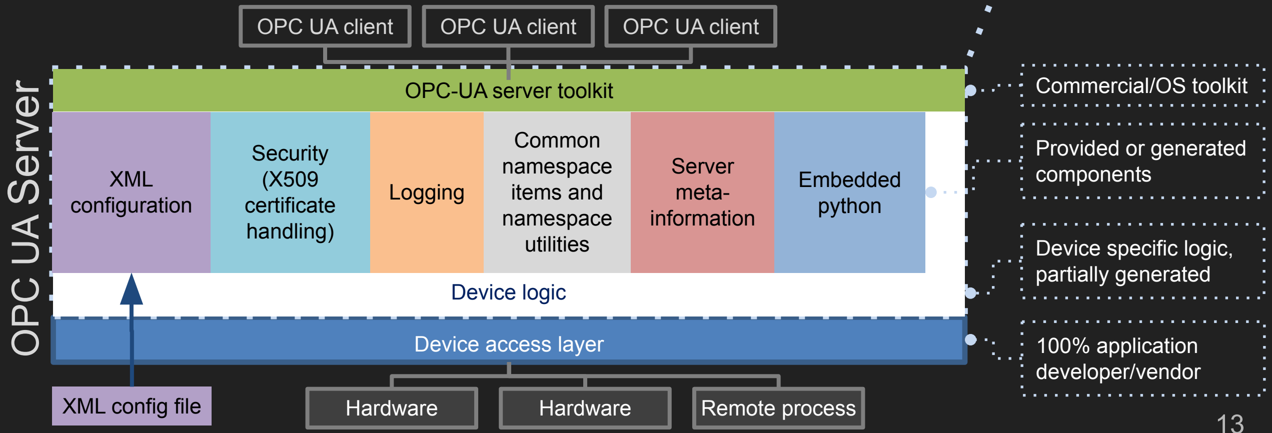
- Excellent experience since 2012
- Fully supported by JCOP
- Created a development environment for middleware applications which generates OPC UA related code...

quasar – Quick opcUA Server generAtion fRamework

A tool for rapid C++ OPC UA server development and more

- CERN-made (currently ATLAS DCS, BE-ICS, alumni) framework for model-driven creation of OPC UA software components
  - generates servers, clients, SCADA integration layer, etc...
- Made with effort efficiency in mind (design, development, testing, deployment)
- quasar-based software used in JCOP as well as beyond CERN
- quasar can build 100% free and open source OPC UA servers and clients
- Validated on different platforms, operating systems, software deployment strategies etc...
- Choice of OPC UA stack used:
  - UA SDK (paid license)
  - open62541 (free & open-source)
- Dependencies:

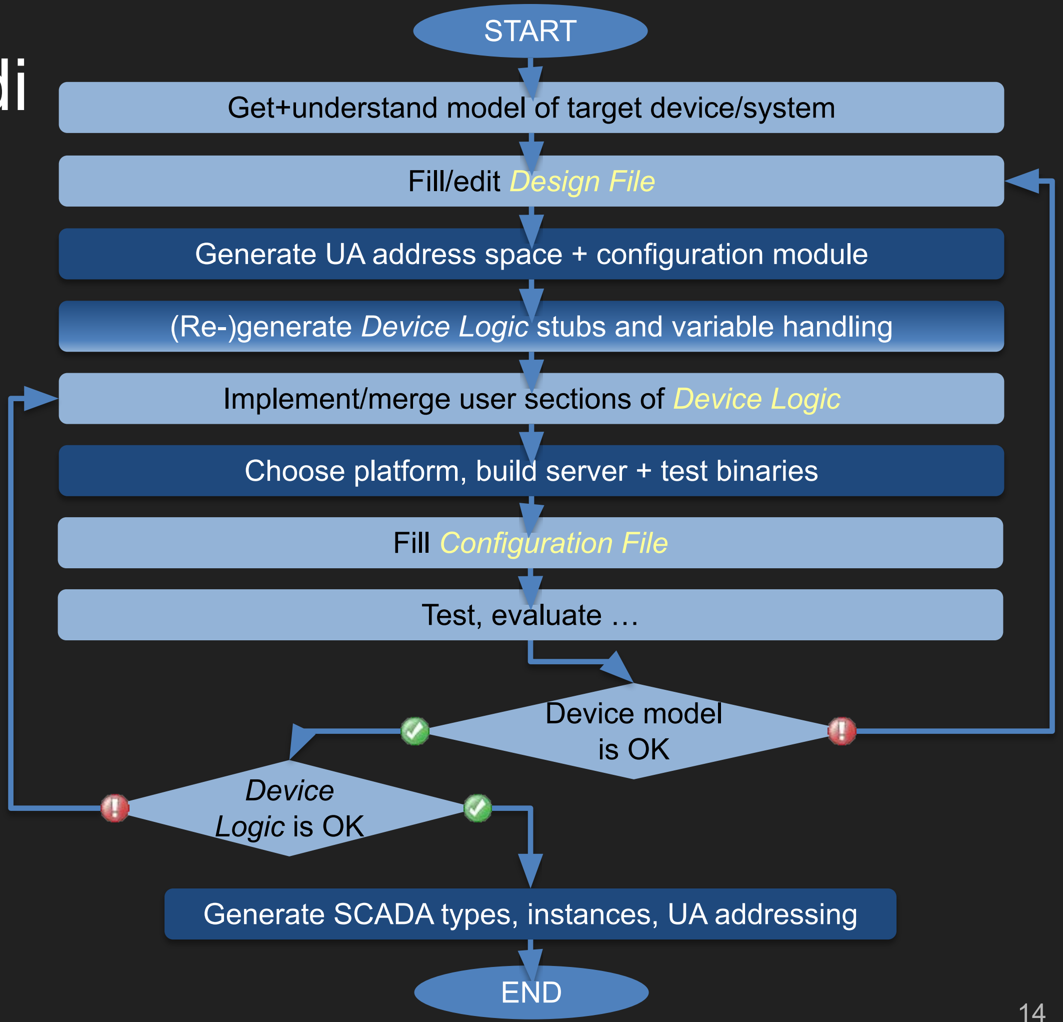
C++ compiler (gcc ...), boost, cmake, xsd, python, xerces-c, libxml2, openssl



quasar – Modus Operandi

Developer benefits:

- *Design file* can be created using provided XML schema
- Roughly 50-90% of code can be generated
- User sections of *Device Logic* stubs are well separated, merging tool simplifies re-generation after design changes or quasar upgrades
- CMake based build system with pre-built toolchains for several platforms
- Can generate RPMs or perform INSTALL (Yocto+PetaLinux)
- *Configuration file* can be created from a generated XML schema (XSD)



quasar – Components & Tools

Archiving

SQL and **NoSQL** archivers

XML configuration

Generated schema ➲ simple creation

Validation tool ➲ verify design constraints

Generated loader for object instantiation and runtime access to configuration

python support

Embed python scripts inside server device logic

Embed server project **in** an **existing** python **application** (no C++ coding)

Python clients (UaoForPython)

Tools

Design visualization: UML generator

Platform toolchains: Linux x86_64, i686, ARM (Raspbian, YOCTO, PetaLinux, CentOS), Windows 32/64

Easy **RPM generator**

Generated program to **test full address space**

Documentation: in-design doc to generate auto-documentation in config schema and address space

Software management: consistency checker helps using versioning system

Logging

Provides **API** and **exchangeable back-end, component** based

Client Generation

Generate client code from quasar-based server project (UaoForQuasar), enables server chains, no OPC-UA specific code for users!

CalculatedVariables

attaching synthetic variables to existing hardware-gathered data (without writing any code)

WinCC OA Integration

fwQuasar - Generates WinCCOA data types, instances and addresses

Server meta-information

Items, build info, thread pool size, run time …

More to come…

quasar – Hardware Integration Modules

Device-related business logic

- Remains under the full responsibility of the server developer
- Some JCOP-supported hardware integration modules are provided along with quasar

CanModule (M. Ludwig CERN BE/ICS, V. Filimonov PNPI, P. Nikiel CERN EP/ADO)

- Abstracts CAN bus device access for a set of commercial CAN interface devices
- Multi-platform (Windows, Linux), dynamic loading of device-specific code
- Supports:
  - Systec CANmodul (USB)
  - Peak PCAN (USB)
  - KVaser (USB, PCI)
  - Anagate (Ethernet)
  - Linux: any other SocketCAN device

SnmpModule (P. Moschovakos CERN EP/ADO)

- SNMP device integration
- Generates code from SNMP MIBs
- Based on the development in AtcaOpcUa server
- Uses internally Net-SNMP API
- Work-In-Progress, in collaboration with BE-ICS to ensure long term maintenance

quasar – WinCC OA Integration

P. Nikiel, CERN EP/ADO

fwQuasar component

Generates WinCC OA types and their instances from server design and configuration

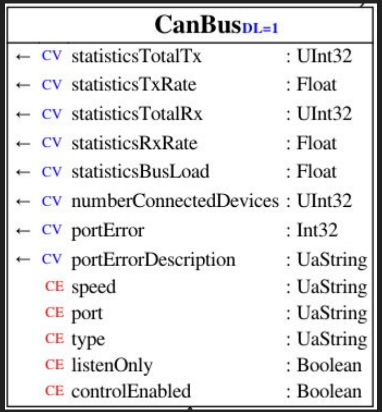
- Generates OPC UA addresses for all datapoint elements
- Uses internally quasar module Cacophony

⇒ Facilitates creation of dedicated JCOP framework components for a quasar-modeled devices

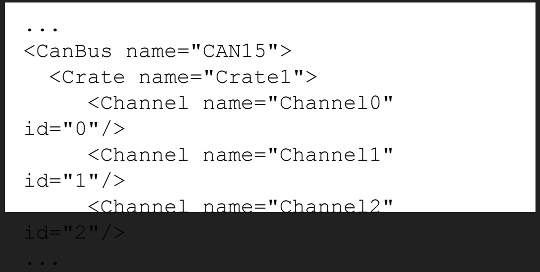
input:

output:

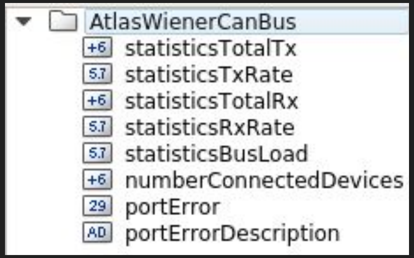
server design xml



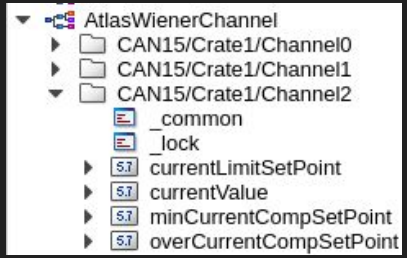
server configuration xml



WinCC OA Types

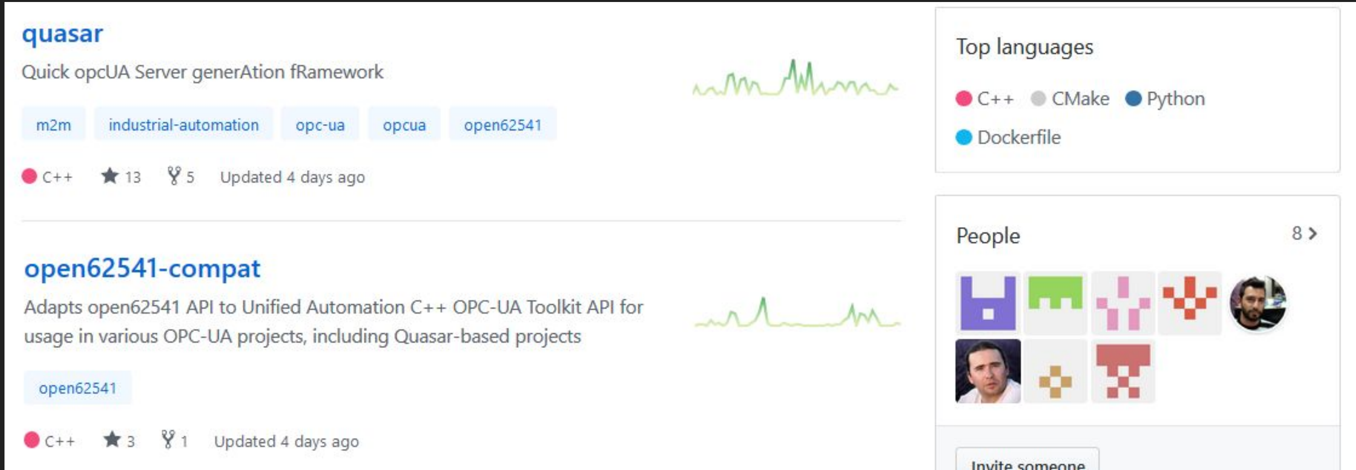


WinCC OA Datapoints

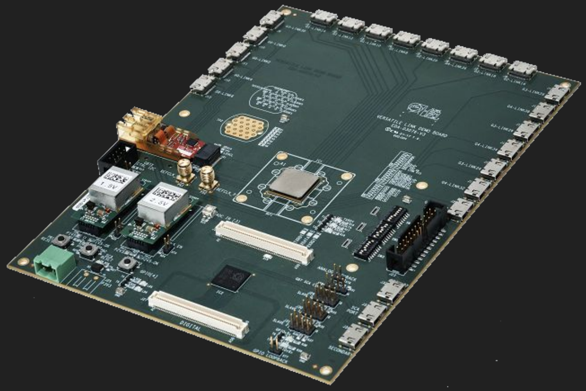


quasar – Pointers

- Project page: https://github.com/quasar-team/quasar, distributed under LGPL
- ecosystem (optional modules), WinCC OA integration, C++ OPC-UA client generation facility
- Current developers, quasar-developers@cern.ch:
    - CERN EP/ADO: P. Nikiel (development lead), P. Moschovakos, S. Schlenker
    - CERN BE/ICS: B. Farnham, M. Ludwig
- Documentation, video tutorials, papers etc.: https://github.com/quasar-team/quasar/wiki
- Support: community effort, JCOP support for DCS systems



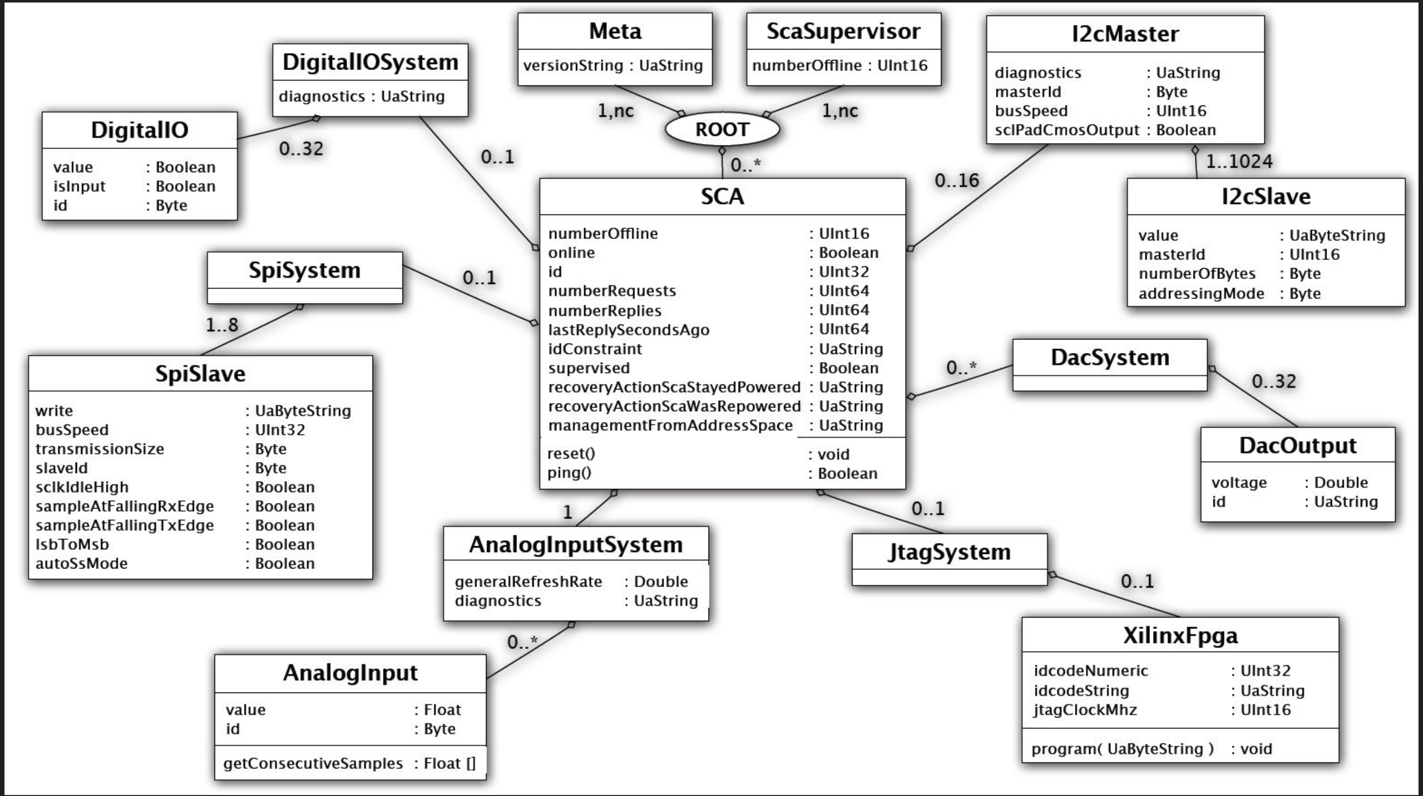
Example: quasar-based Integration of the GBT-SCA ASIC

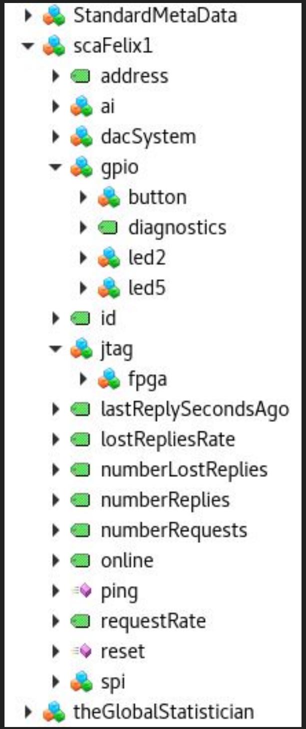


quasar – Server Example (GBT Slow Control Adapter)

Generated class diagram of an advanced quasar design for GBT-SCA

Authors: SCA-SW team (P. Nikiel + P.Moschovakos CERN, H. Boterenbrood NIKHEF)





Running quasar example server seen by OPC UA client UI

quasar – Server Example (GBT Slow Control Adapter)

- GBT-SCA communication via software requires complex interface with hardware/firmware though (ATLAS specific in this case), server business logic is only part of the job
- OPC UA client can used by DCS (monitoring) and DAQ (configuration) applications

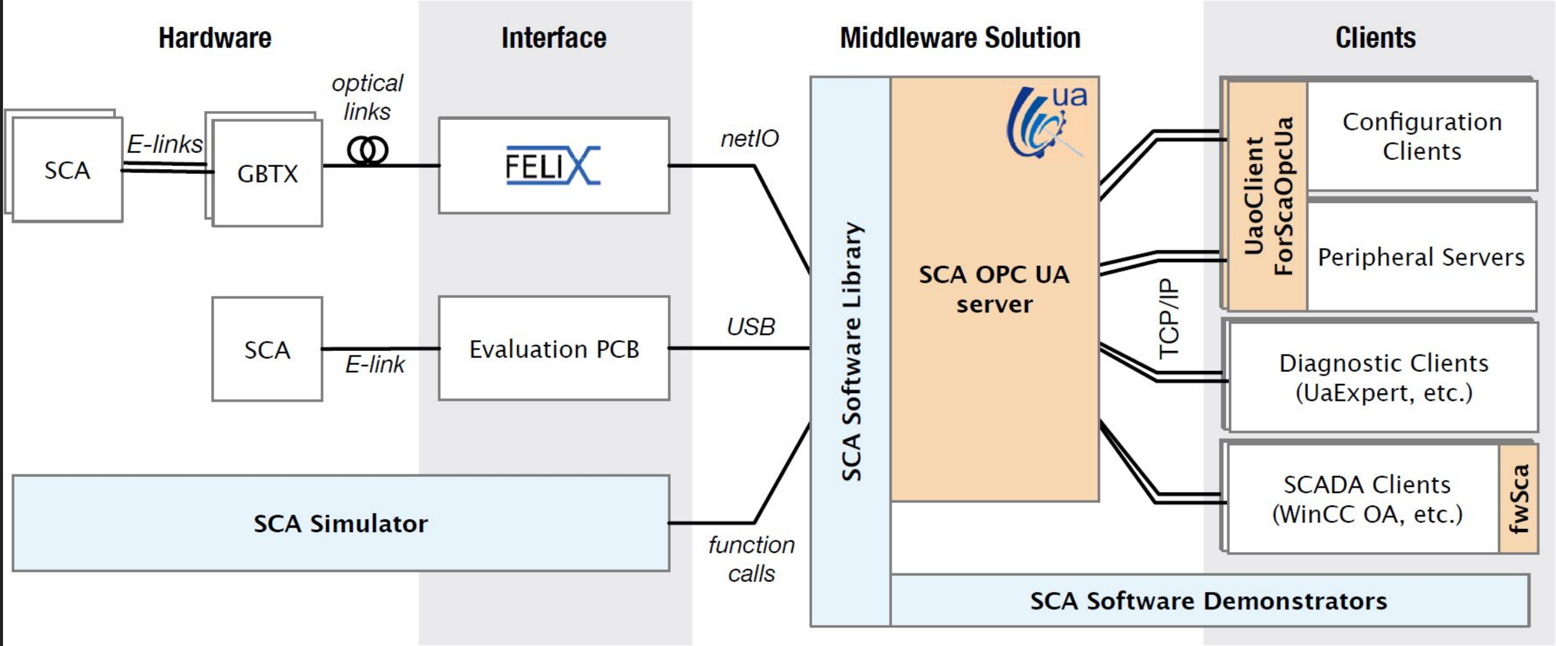


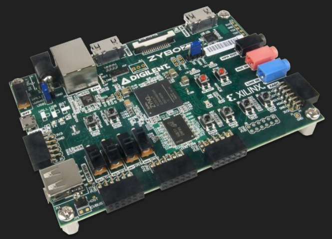
Figure by P. Moschovakos

Embedding quasar

quasar on SoC

Embedding of quasar OPC UA into your FE interface

- No custom device-specific protocols, provides directly industry-standard communication interface
- Allows for sophisticated abstraction layer on device if desired
- Integration into DCS back-end facilitated



Experience so far:

- See also CERN SoC Workshop 2019 for overview and tutorial
- Builds for Yocto / PetaLinux (G. Stark UCSC, P. Nikiel CERN)
  - Tested on a variety of platforms/devices (ZYNQ 7020, ZYNQ Ultrascale+ ZU19, RaspberryPi, qemu with Yocto)
  - However, Yocto learning curve is steep! Recommending to go another way...
- … Native builds on CentOS for ARM
  - Software handling much easier, need to take care of device tree though
  - Tests with CentOS 7 on ZCU102 in ATLAS MuCTPi project (R. Spiwoks, P. Papageorgiou, P. Nikiel CERN)
    - Built demo quasar server for I2C sensor monitoring
    - Reached publishing frequency of 1.8 kHz, limit imposed by I2C readout rate
    - Marginal CPU usage and ~few 10 MB of memory footprint
- Cross-compiling tested for multiple platforms

See also next presentation from R. Spiwoks

JCOP-Supported OPC UA Servers

OPC UA Servers for FE-Interface Hardware

CanOpen Server (V. Filimonov, PNPI)

- First OPC UA server within JCOP, in production since 2013
- Implements CanOpen standard communication
- Primarily used for interfacing ELMB / ELMB2 boards
- Uses quasar CanModule for CAN interface hardware access
- Corresponding JCOP framework component: fwElmb

Server example for EMCI/EMP (P. Nikiel, CERN EP/ADO)

- Very early stage of development (proof of concept)
- Target: Embedded on ZYNQ Ultrascale+ of EMP
- Aiming for providing an interface to common lpGBT functionality
- Communication to EMP firmware via AXI interface

JCOP quasar-based OPC UA Servers

Middleware for supported hardware vendors (B. Farnham, CERN BE/ICS)

- Collaboration with CAEN, Wiener, Iseg (power systems and VME crates)
- quasar-based OPC UA servers
  - HAL libraries provided (mostly) by vendors (incl. sources), rest of server device logic by CERN
  - HAL implementations for CAN equipment use quasar CanModule
  - Hardware discovery mode ⇒ server creates its own config file
  - Supported platforms Windows and Linux
- Status:
  - Deployment ongoing in production systems of experiments, to be finished during LS2
  - Integration of new module types of known vendors ongoing
- WinCC OA integration with well-established components (fwCaen, fwWiener, fwIseg)
- For HL-LHC upgrades: when planning procurement of new equipment types need to require always support in respective HAL libraries by vendors
- ATCA power supplies: JCOP will provide corresponding OPC UA server

ATCA: Shelf, Board and Power Supply Integration

Based on material from P. Moschovakos, 15th xTCA IG meeting

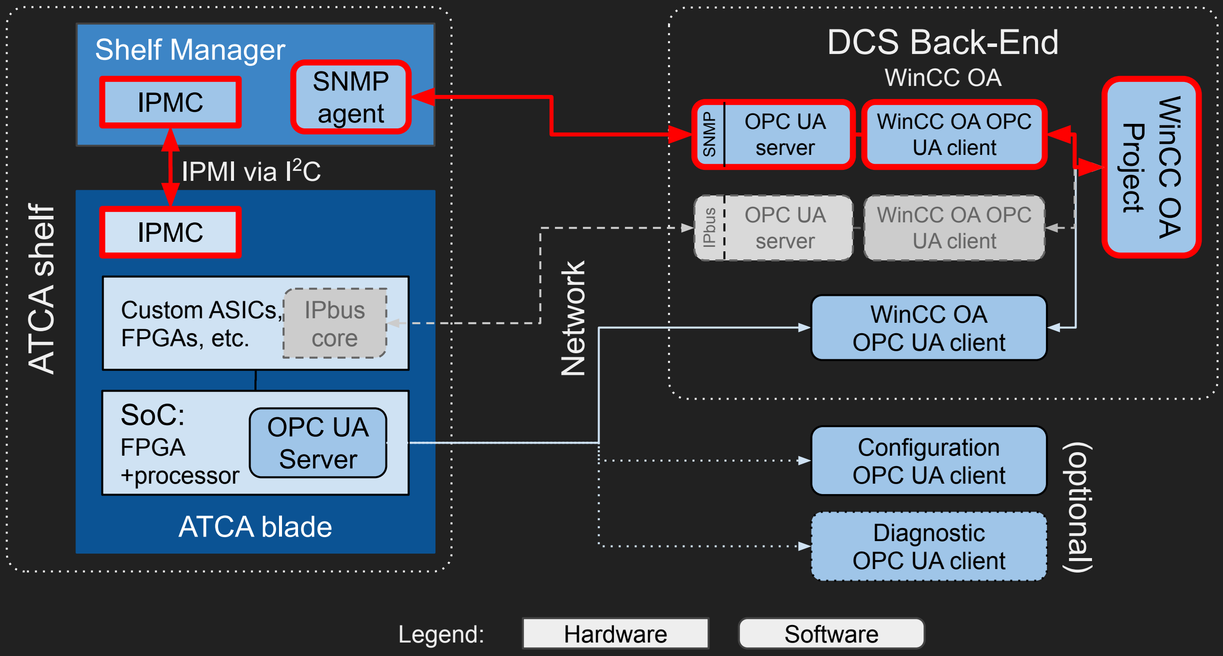
ATCA Monitoring/Control Options

Several Communication Paths

- Via **shelf manager** (ShM), overall shelf control and blades via IPMC
- Direct communication with **blade** components
  - FPGA cores
  - SoCs
  - ...

⇒ Concentrating on ShM path

ATCA Integration Example

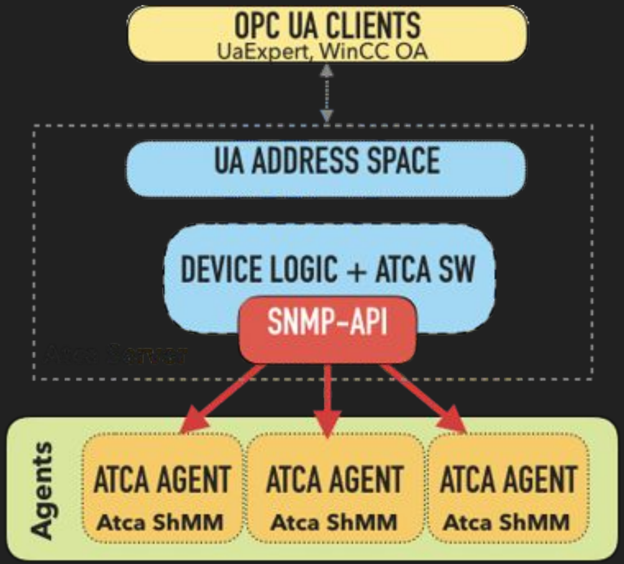


AtcaOpcUa for Shelf Management

- quasar-based OPC UA server for managing ATCA shelves via ShM, monitor/control shelf and blade IPMCs
- Designed for PICMG ATCA platforms that use Pigeon Point ShMs
- Uses SNMP-based external interface of ShM via Net-SNMP
- Validated on CERN supported nVent Schroff shelves with
  - ShMM 500
  - ShMM 700R
- Provides automatic hardware discovery (shelf, blades, IPMCs, sensors etc.), only existing entries are populated

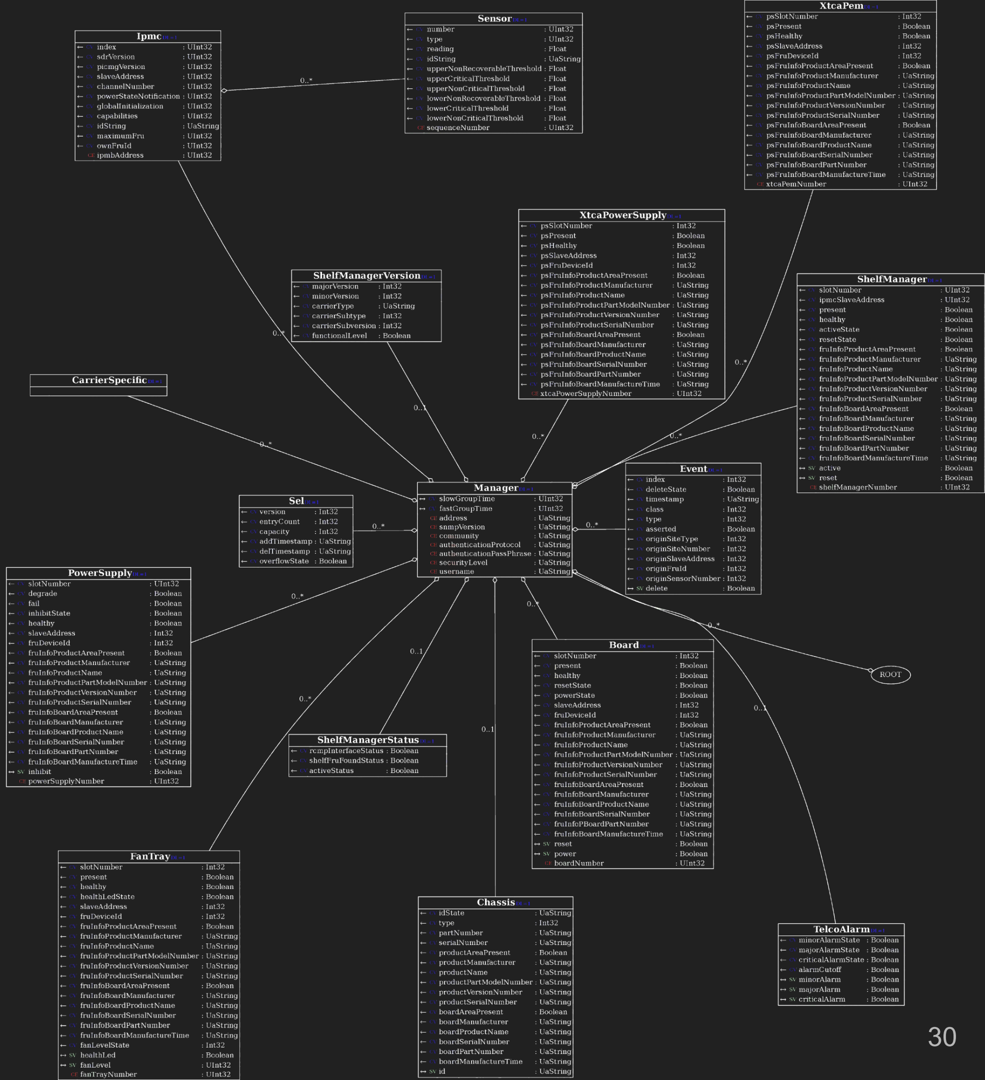
⇒ Interest egroup (releases, feedback ...): opc-ua-atca

Potential to become fully supported JCOP solution

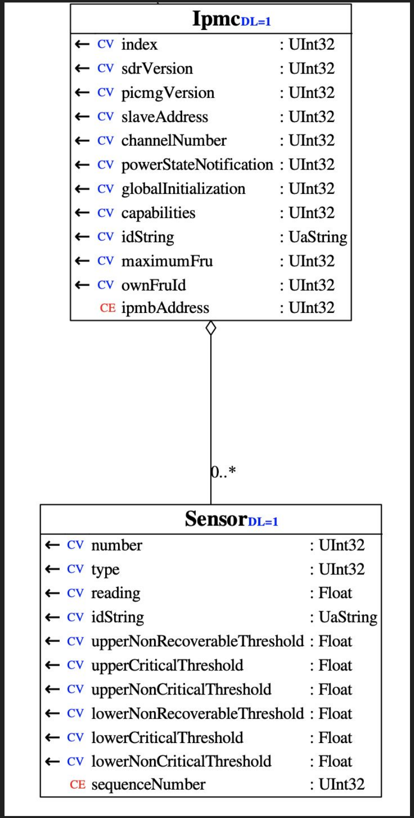


AtcaOpcUa Design

- Template-based and **generated from** the Pigeon Point ATCA **MIB** using Jinja2
  - Fully covers *basic* **group variables** (blades, fan trays, power supplies, shelf managers, chassis …)
  - Selected *advanced* **variables** covered
  - Support for *TELCO* **alarms**
- Hardware **discovery** walking SNMP tree
- **Synchronous** SNMP **communication** with ShM (**poll**)
  - For reliability and **ShM resource protection** (~1k parameters polling $10s^{-1}$ max per ShM)
  - Distinguish between **dynamic and static parameters** to control poll interval (e.g. temperature vs. serial no.)



IPMC and Blade-Specific Hardware



- Any IPMC that conforms to the standard can be monitored
- IPMC and sensors are automatically discovered and populated into the server
- Representation of IPMC and sensors in design follows IPMB hardware representation
- Automatic distinction between types of sensors developed
  - Supported: temperatures, voltages, fans speed etc.
  - Mechanism to facilitate addition of non-supported types was introduced
- Connected sensors are addressable using their IPMC address and sequence number in configuration

AtcaOpcUa in Action

```
2020-03-02 09:34.40.373938 [QuasarServer.cpp:101, INF] Logging initialized.
2020-03-02 09:34.40.387774 [BaseQuasarServer.cpp:346, INF] Configuration Initializer Handler
2020-03-02 09:34.40.387848 [QuasarServer.cpp:125, INF] Server running in regular mode, address space will be built from contents of x
2020-03-02 09:34.40.416489 [MetaAmalgamate.cpp:3281, INF] meta configuration found in the configuration file, configuring StandardMet
2020-03-02 09:34.40.416591 [MetaAmalgamate.cpp:3200, INF] StandardMetaData.Log configuration found in the configuration file, configu
2020-03-02 09:34.40.416661 [MetaAmalgamate.cpp:3133, INF] general non-component log level will be [INF]
2020-03-02 09:34.40.416743 [MetaAmalgamate.cpp:2578, INF] setting log level to [INF]
2020-03-02 09:34.40.416813 [MetaAmalgamate.cpp:3118, INF] no StandardMetaData.Log.ComponentLogLevels configuration found in the confi
2020-03-02 09:34.40.416900 [MetaAmalgamate.cpp:3069, INF] configuration for logging component  handle [0] name [CalcVars] using value
2020-03-02 09:34.40.417015 [MetaAmalgamate.cpp:2516, INF] setting component [name:CalcVars id:0] to level [INF]
2020-03-02 09:34.40.417101 [MetaAmalgamate.cpp:3219, INF] no StandardMetaData.SourceVariableThreadPool configuration found in the cor
2020-03-02 09:34.40.417763 [MetaAmalgamate.cpp:3236, INF] no StandardMetaData.Quasar configuration found in the configuration file, c
2020-03-02 09:34.40.417852 [MetaAmalgamate.cpp:3250, INF] no StandardMetaData.Server configuration found in the configuration file, c
2020-03-02 09:34.40.418049 [SnmpBackend.cpp:64, INF] [asml1c-stf0.cern.ch] Using SNMP version 2c
2020-03-02 09:34.40.438717 [CalculatedVariablesEngine.cpp:262, INF, CalcVars]  #ParserVariables: 1181 #CalculatedVariables: 0 #Synchr
2020-03-02 09:34.40.439041 [CalculatedVariablesEngine.cpp:297, INF, CalcVars] Optimized(suppresed) 1181 ParserVariables not used in c
2020-03-02 09:34.40.439104 [CalculatedVariablesEngine.cpp:262, INF, CalcVars]  #ParserVariables: 0 #CalculatedVariables: 0 #Synchroni
2020-03-02 09:34.40.439149 [QuasarServer.cpp:78, INF] Initializing Quasar server.
2020-03-02 09:34.40.441714 [opcserver.cpp:157, INF] Opened endpoint: opc.tcp://pcaticstest08.dyndns.cern.ch:48050
2020-03-02 09:34.40.441777 [QuasarServer.cpp:48, INF] Server main loop started!
```

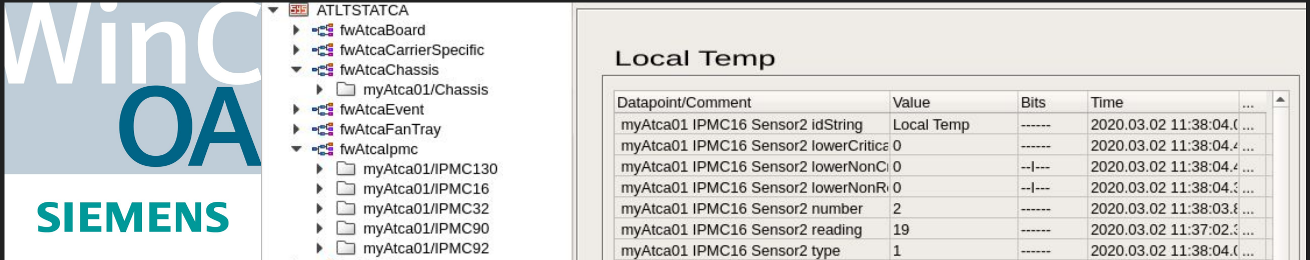
Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	
NS2|String|myAtca01.IPMC92.idString	idString	Upper Fan Tray	String	9:38:28.522 AM	9:38:53.592 AM	Good
NS2|String|myAtca01.IPMC92.Sensor10.idString	idString	Fan Tach 2	String	9:38:38.244 AM	9:39:25.453 AM	Good
NS2|String|myAtca01.IPMC92.Sensor10.reading	reading	5400	Float	9:41:16.578 AM	9:41:16.578 AM	Good
NS2|String|myAtca01.IPMC92.Sensor11.idString	idString	Fan Tach 3	String	9:38:38.738 AM	9:39:46.154 AM	Good
NS2|String|myAtca01.IPMC92.Sensor11.reading	reading	5460	Float	9:40:17.874 AM	9:40:17.874 AM	Good
NS2|String|myAtca01.IPMC92.Sensor9.idString	idString	Fan Tach 1	String	9:38:37.442 AM	9:40:10.623 AM	Good
NS2|String|myAtca01.IPMC92.Sensor9.reading	reading	5460	Float	9:41:16.504 AM	9:41:16.504 AM	Good
NS2|String|myAtca01.IPMC92.Sensor8.idString	idString	Temp Out Right	String	9:38:36.907 AM	9:40:18.490 AM	Good
NS2|String|myAtca01.IPMC92.Sensor8.reading	reading	19	Float	9:40:17.303 AM	9:40:22.683 AM	Good
NS2|String|myAtca01.IPMC92.Sensor7.idString	idString	Temp Out Center	String	9:38:35.835 AM	9:40:26.703 AM	Good
NS2|String|myAtca01.IPMC92.Sensor7.reading	reading	22	Float	9:41:16.301 AM	9:41:16.301 AM	Good

- Server
- StandardMetaData
- myAtca01
  - Board1
  - Board10
  - Board11
  - Board12
  - Board13
  - Board14
  - Board2
  - Board3
  - Board4
  - Board5
  - Board6
  - Board7
  - Board8
  - Board9
  - Chassis
  - FanTray1
  - FanTray2
  - IPMC130
    - Sensor0
      - idString
      - lowerCriticalThreshold
      - lowerNonCriticalThreshold
      - lowerNonRecoverableThreshold
      - number
      - reading
      - type
      - upperCriticalThreshold
      - upperNonCriticalThreshold
      - upperNonRecoverableThreshold
    - Sensor1
    - Sensor10

ATCA WinCC OA Integration

fwAtca Component

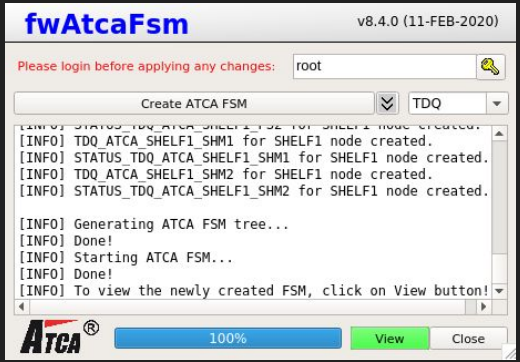
- Integration of discovered hardware based on (discovered) config file of AtcaOpcUa server
- Automatic creation of datapoints, addresses, descriptions, archiving parametrization possible
- Provides library with commonly used functionality
- Possibility to apply alarm thresholds defined in IPMC or ShM on corresponding alerts for created datapoints

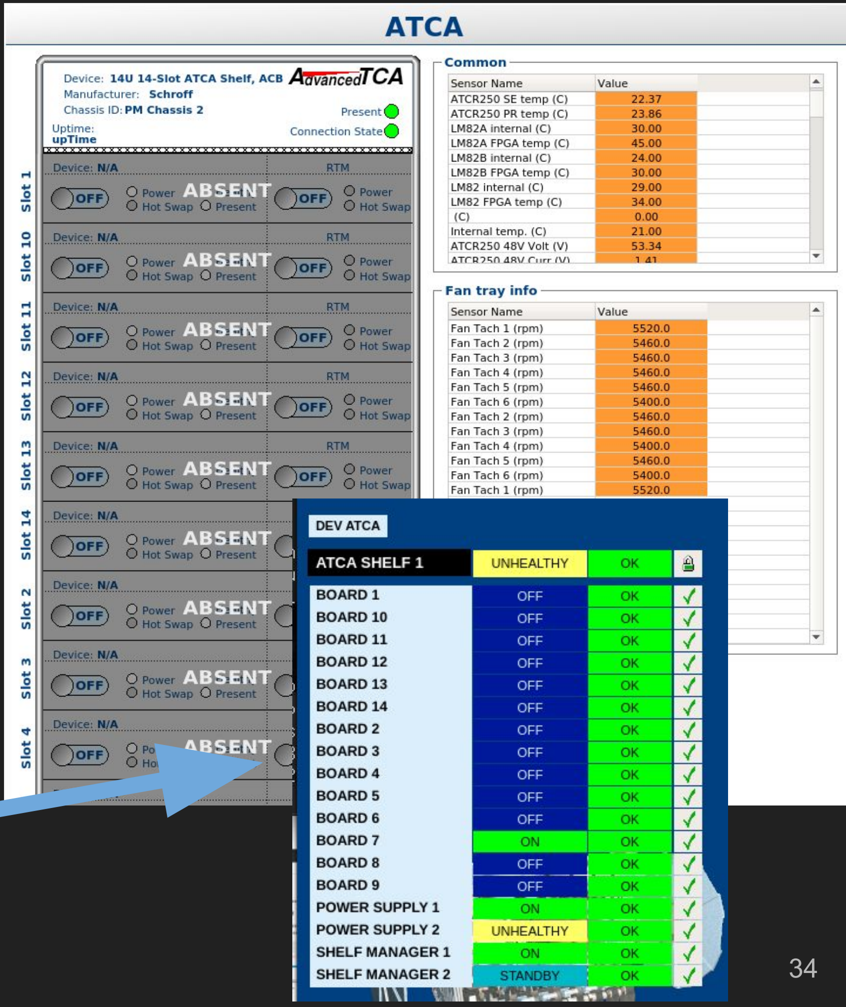


Generation of ATCA Finite State Machine Hierarchy

fwAtcaFsm component

- Automatic **creation of tree of FSM objects** using the **JCOP FSM tool**, including generic UI panels
- Based on **discovered hardware**
- Foresees **extension of structure with user-defined objects**, e.g. for integration of direct blade monitoring path
- Currently, full plug-and-play for ATLAS FSM only



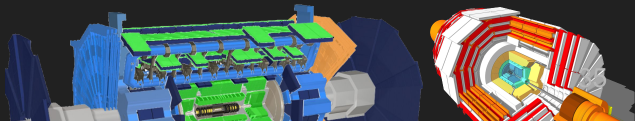


Conclusions

HL-LHC stimulates activity in development of integration tools for detector control

- New or upgraded front-end hardware components for DCS
- New technologies in front-end devices and interfaces (SoCs, FE ASICs, VL+ receivers) open up possibilities for detector control but also require novel middleware approaches
- quasar provides device integrators with a powerful toolset to create middleware and link with the JCOP-based DCS back-end
- JCOP readily provides DCS tools for the key hardware components for detector upgrades - power supplies - and may include ATCA equipment

Thank you!



Backup

Some quasar-based projects

Name	Description	Status	Notes
LArPurity	ATLAS Liquid Argon calorimeter purity analyzer	Production since 2015	
IpBus	Generic IpBus	Production since 2018	will become deprecated
ATLAS Wiener	Wiener VME crates interfaced with CAN	Production since 2015	
TileHvMicro	HV Micro, ATLAS Tile calorimeter	Production since 2015	
CAEN	CAEN power supplies, JCOP	Production since 2018	
ISEG	ISEG power supplies, JCOP	Production since 2017	
JCOP Wiener	SNMP and CAN Wiener devices	Development	
FtkSbc	SBC monitoring, ATLAS FTK	Production since 2017	
SCA	GBT-SCA for ATLAS NSW, LAr and BIS	In test	+ Uao
HvSys	HVSys monitoring and controls over RS232, ATLAS TRT	Production since 2017	
Generic SNMP	Generic SNMP	Development	
LAr LTDB	LTDB+LATOME monitoring and controls	Development	+ Uao, + peripheral
gFEX	ATLAS L1 TDQ gFEX	Development	+ embedded (ZYNQ)
eFEX	ATLAS L1 TDQ eFEX	Development	+ used from Python
ATCA Shelf Manager	nVent Schroff (aka Pigeon Point) PPS MIB	Deployed in test systems	
EMP	Embedded Monitoring Processor (EMCI Receiver)	Early Development	+ embedded (ZYNQ)

quasar system-wide example

based on existing project

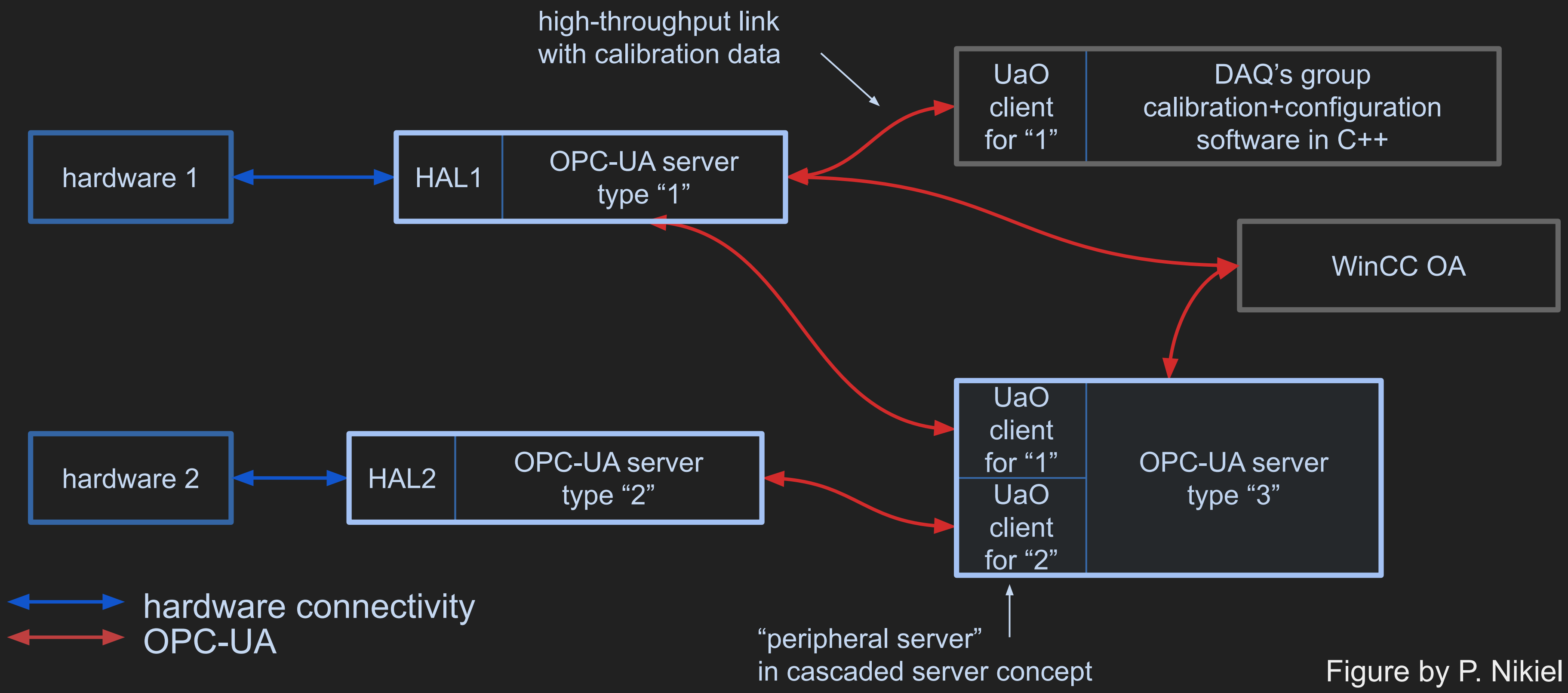
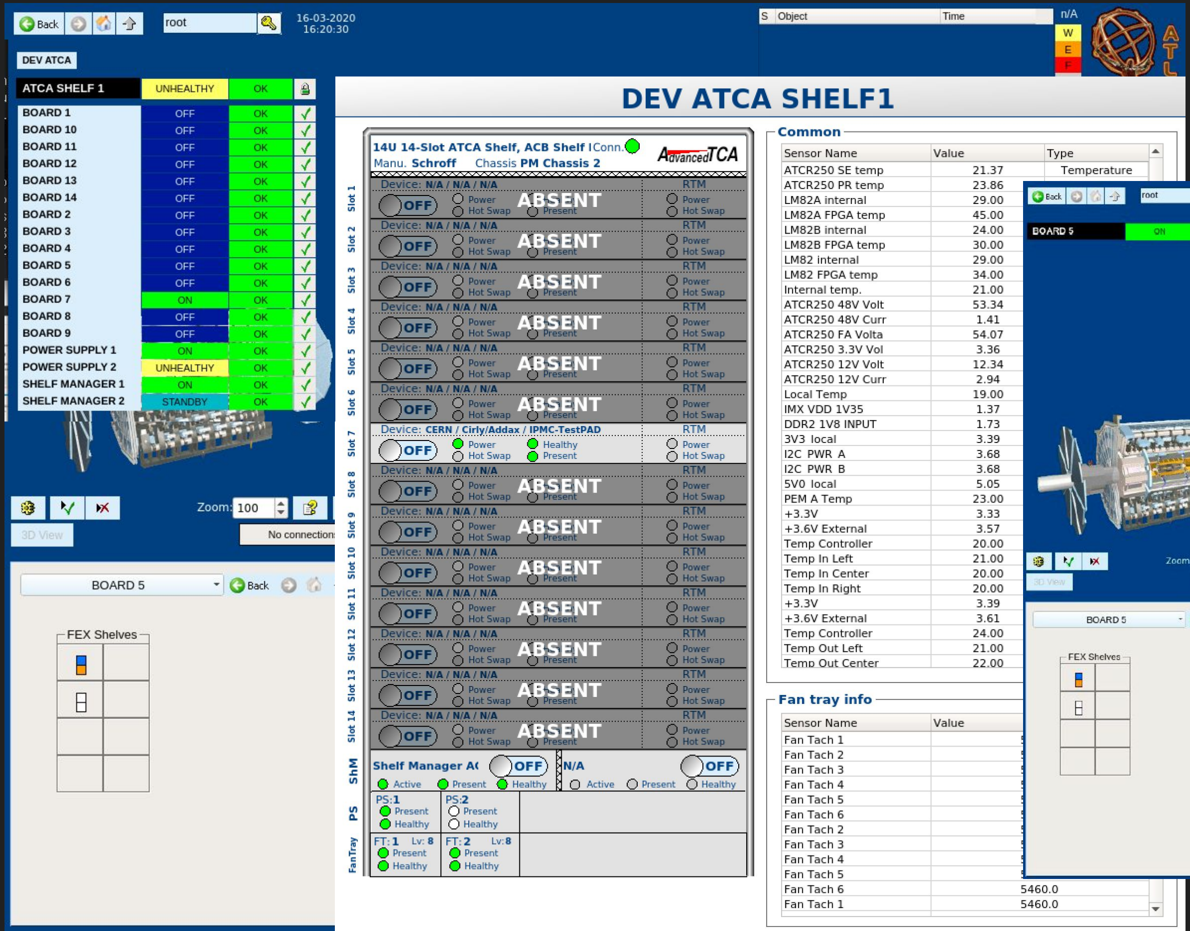
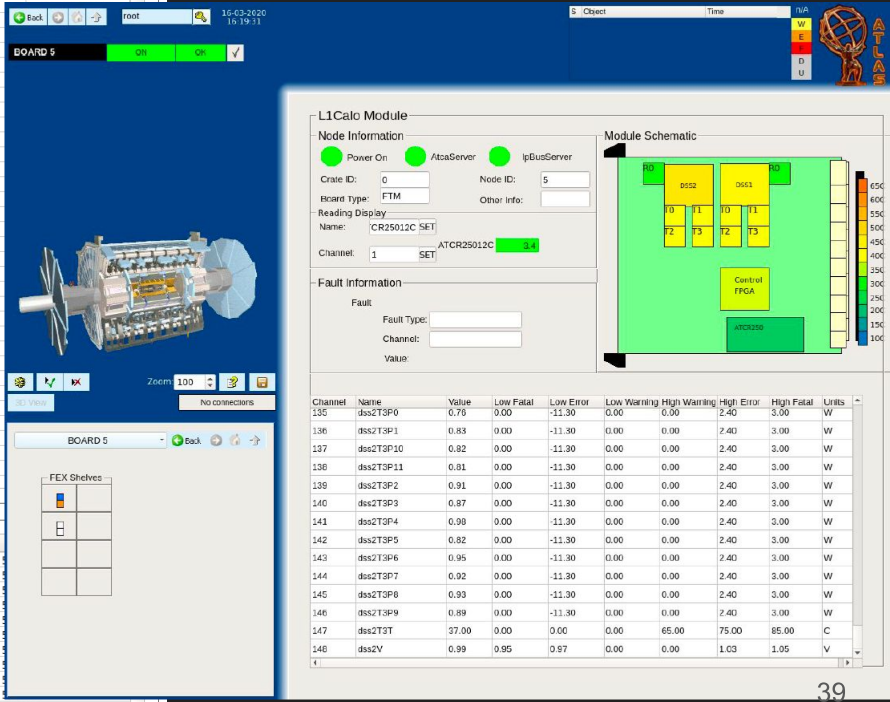


Figure by P. Nikiel

STF L1Calo ATCA integrated in ATLAS FSM



Example in L1Calo, left tree generated with fwAtcaFsm, right custom path for additional board information



P. Thompson, R. Turner