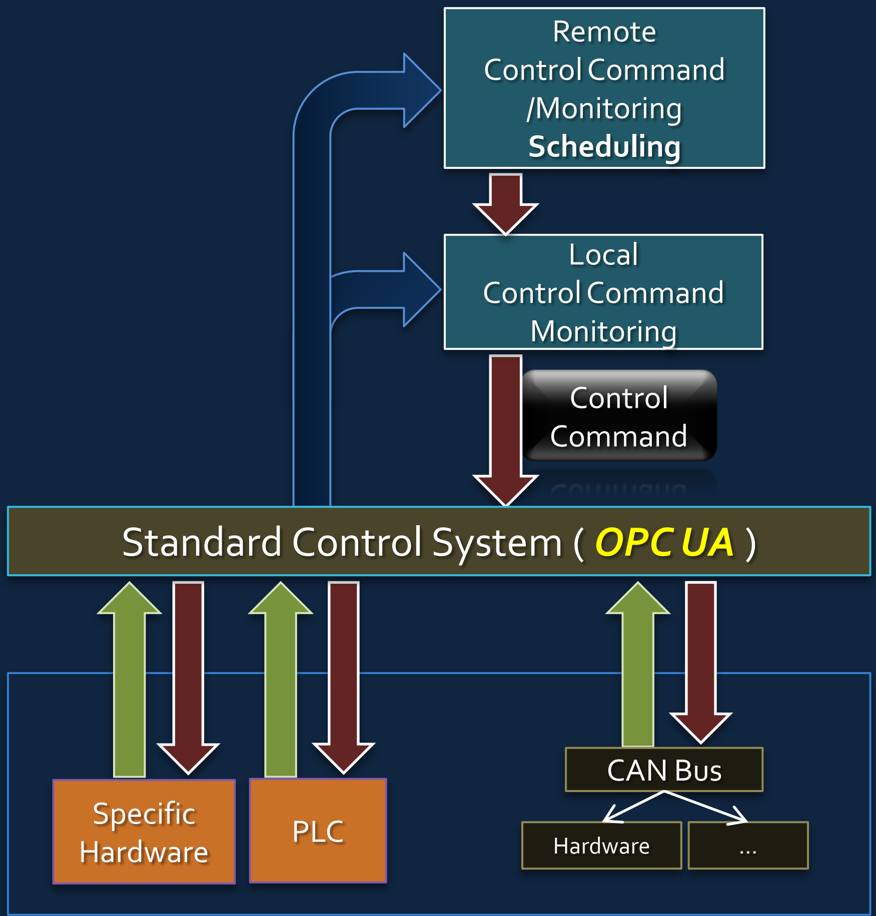
L.A.P.P
STATUS REPORT
(T. Le Flour, J.L Panazol)

Overview

- From **OPC** (OLE for Process Control) to **OPC-UA** (OPen Connectivity-Unified Architecture)
- **WORK AT LAPP WITH OPC UA AND ACS**
- **SHORT TERM DEVELOPMENTS**

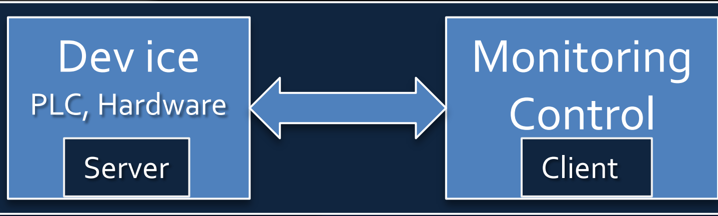
Slow Control/ Monitoring Schematic View



Annecy July 2010

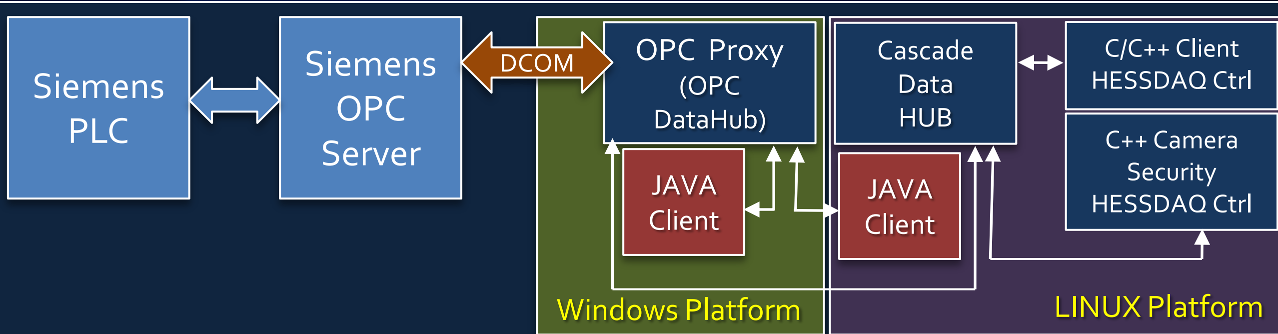
- Slow Control/ Monitoring
  - On site and remotely:
  - Checking specific devices
  - Knowing the current status of :
    - An array
    - A telescope
- Access information in a standard way independently of the context
  - standalone mode
  - DAQ components

Different possible configurations



1. Home made protocol
   1. Difficult to develop
   2. Difficult to maintain
   3. Difficult to upgrade

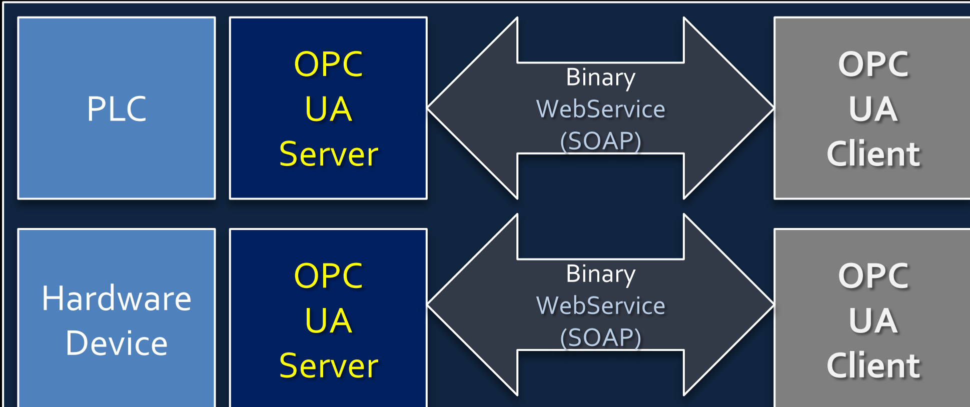
1



HESS 2 Camera Security & Camera Loading/Unloading Systems

1. Window platform is needed
2. All LINUX Platform should install the OPC communication software
3. The way of connecting OPC software is de-facto imposed
4. Difficult to connect other component implementing a different standard

2



1. Communication standard (WEB Services on binary transfer)
2. Platform independence & Interoperability
3. All languages possible for Client and Server side
4. Scalability
5. Evolution

3

OPC UA

- Fully specified by the OPC Foundation

- To have a standard way of accessing the devices information (independently of the device type : PLCs, Hardware devices)

- To be platform independent :
  - communication connections between OPC UA Clients and Server : Cross-Platforms interoperability

- To implement OPC/UA servers on embedded systems

- OPC Com could be also used with a UA architecture :
  - Via wrappers and proxies

- **RELIABILITY** : clients and servers can be monitored.

- Might help to implement Service Oriented Architecture (SOA) in CTA monitoring and slow control
  - reliability, performance, robustness, and security

OPC UA

- COM/DCOM no longer maintained by Microsoft (WEB Services)

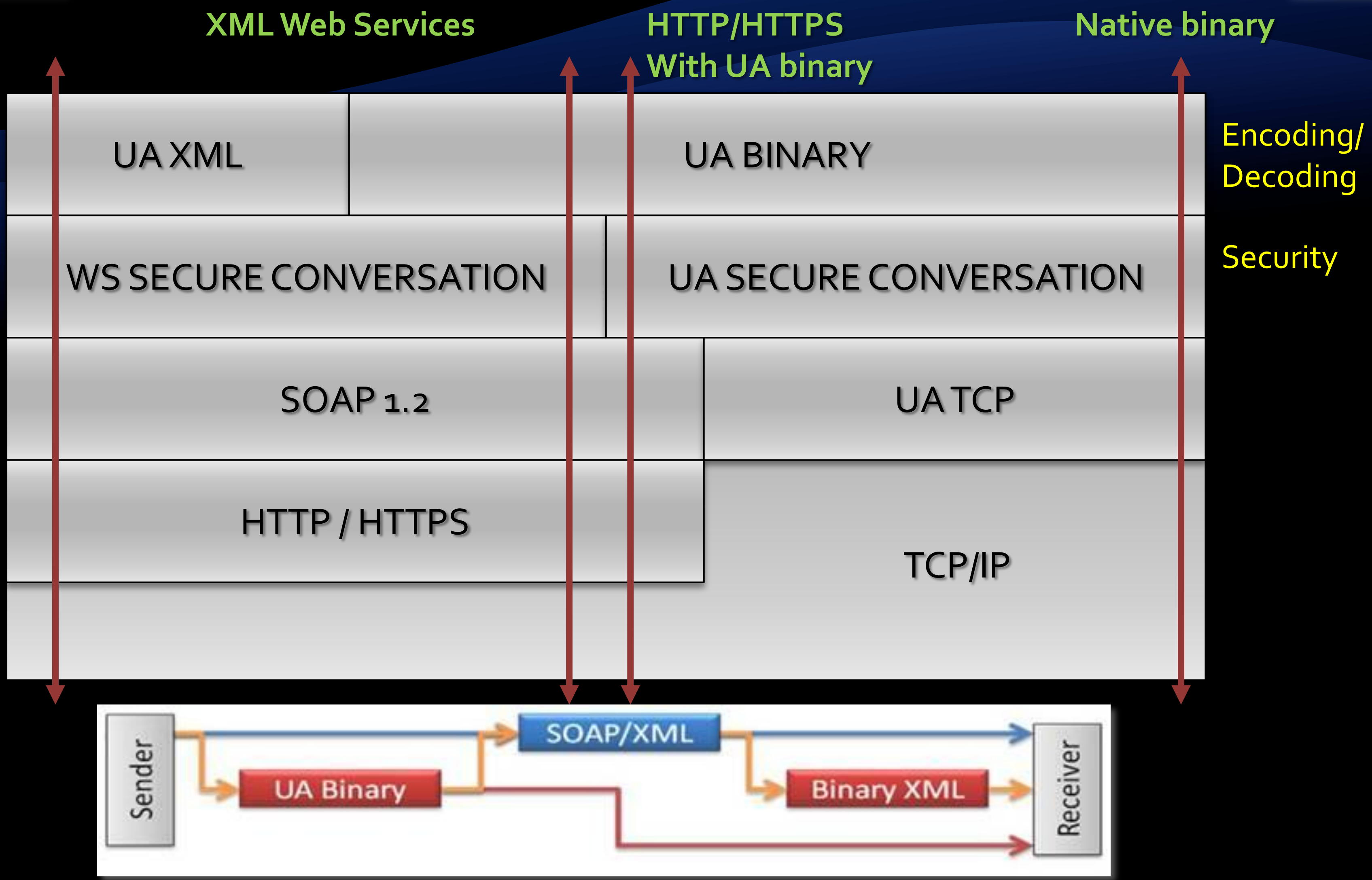
- From a DCOM to SOA(Services Oriented Architecture)
  - More efficient
  - Migration is easier from different platforms

- Migration from a representation data model to action model on objects
  - Based on 4 basic services :
    - Emitting a request, Reading a value, Writing a value, Subscribe to a variable(to follow its evolution)

OPC UA
Services

- Request/response *Services*
- Publisher *Services*
- *Server* to *Server* interactions
- Discovery Service Set
- *SecureChannel* Service Set
- *Session Service Set*
- *NodeManagement Service Set*
- *View Service Set*
- *Query Service Set*
- *Attribute Service Set*
- *Method Service Set*
- *MonitoredItem Service Set*
- *Subscription Service Set*

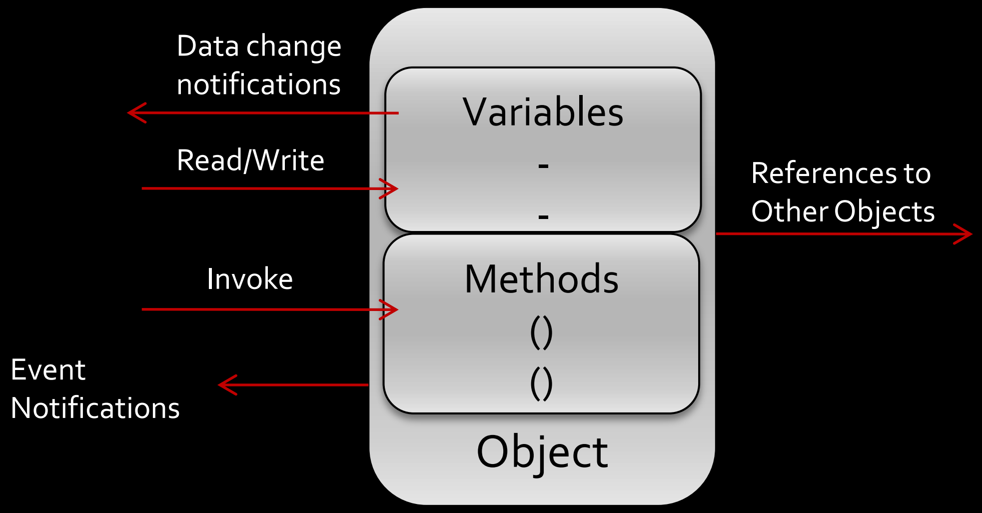
OPC/UA Messaging



OPC/UA : information model

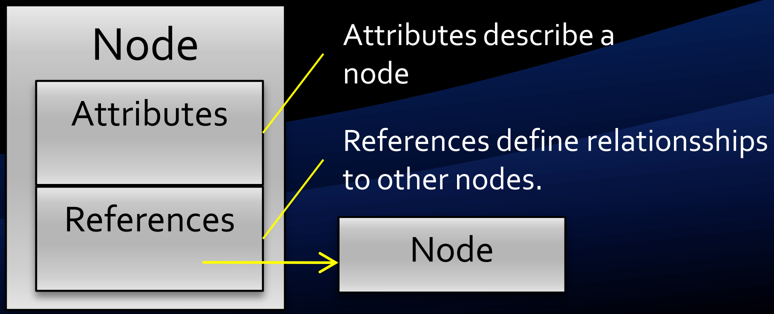
- All vendors of OPC UA servers should implement the "***unified***" information model

- Object Model :
  - OPC-UA is designed for exchanging information in an object-oriented manner
  - OPC/UA Address Space provide a standard way for servers to represent objects to client.

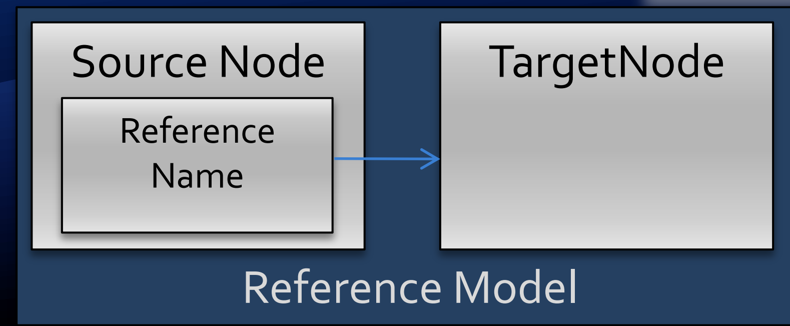


- The elements of this model are represented in the Address Space as Nodes.
- Each Node is assigned to a NodeClass and each NodeClass represents a different element of the Object Model

OPC/UA : information model

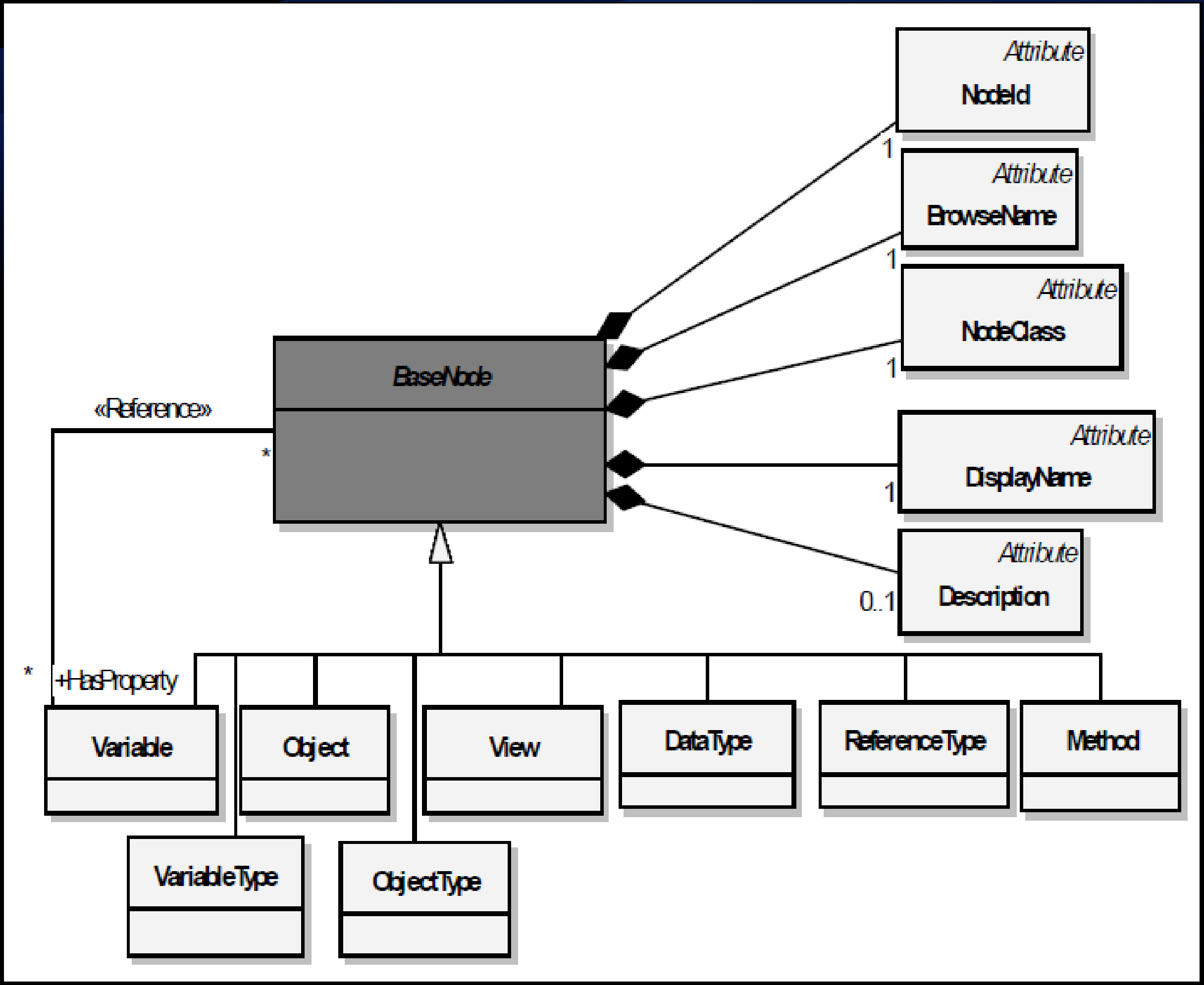


AddressSpace node model

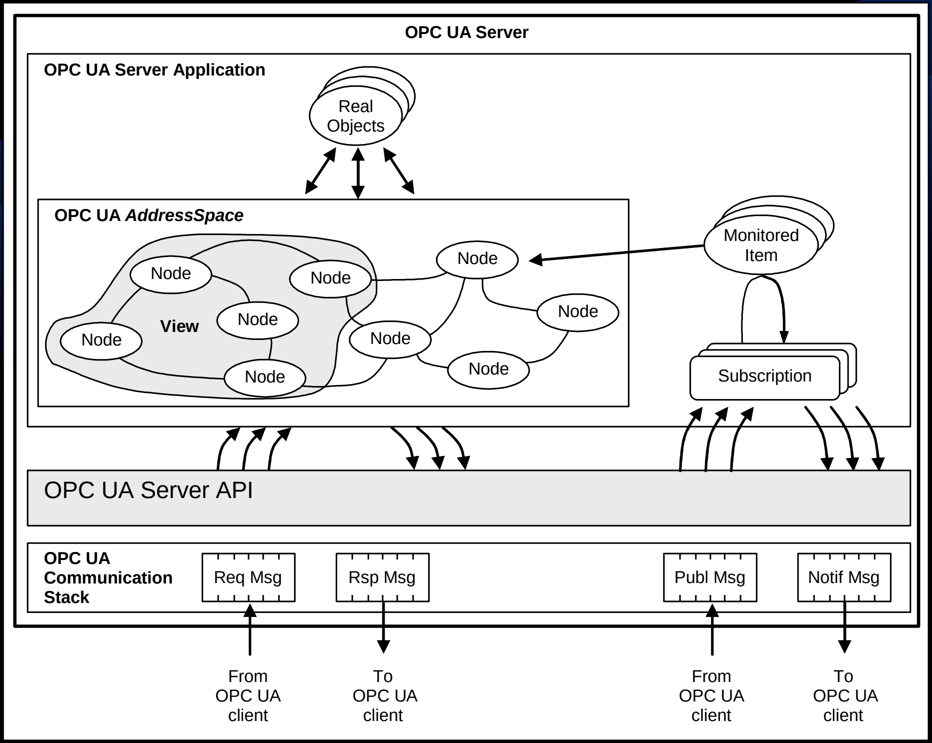


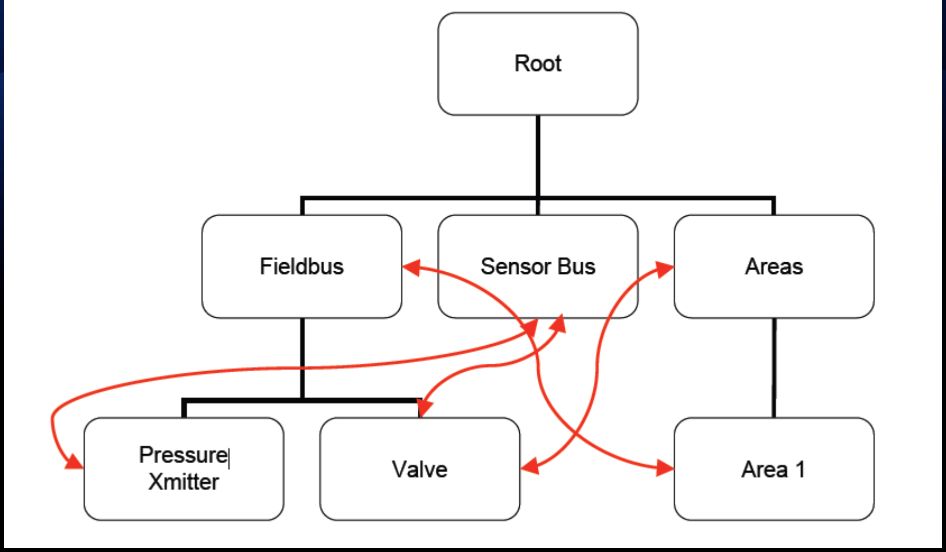
- Atrributes describes Nodes. Clients can access values using *Read,Write,Query* and *Subscription/MonitoredItem Services*
- References are :
  - used to relate Nodes to each other
  - Instance of *ReferenceType Nodes(visible in the Address Space)*
  - *TargetNode* may be in the same address space or in the address space of another OPC UA Server

OPC UA Information MetaModel



Information Model

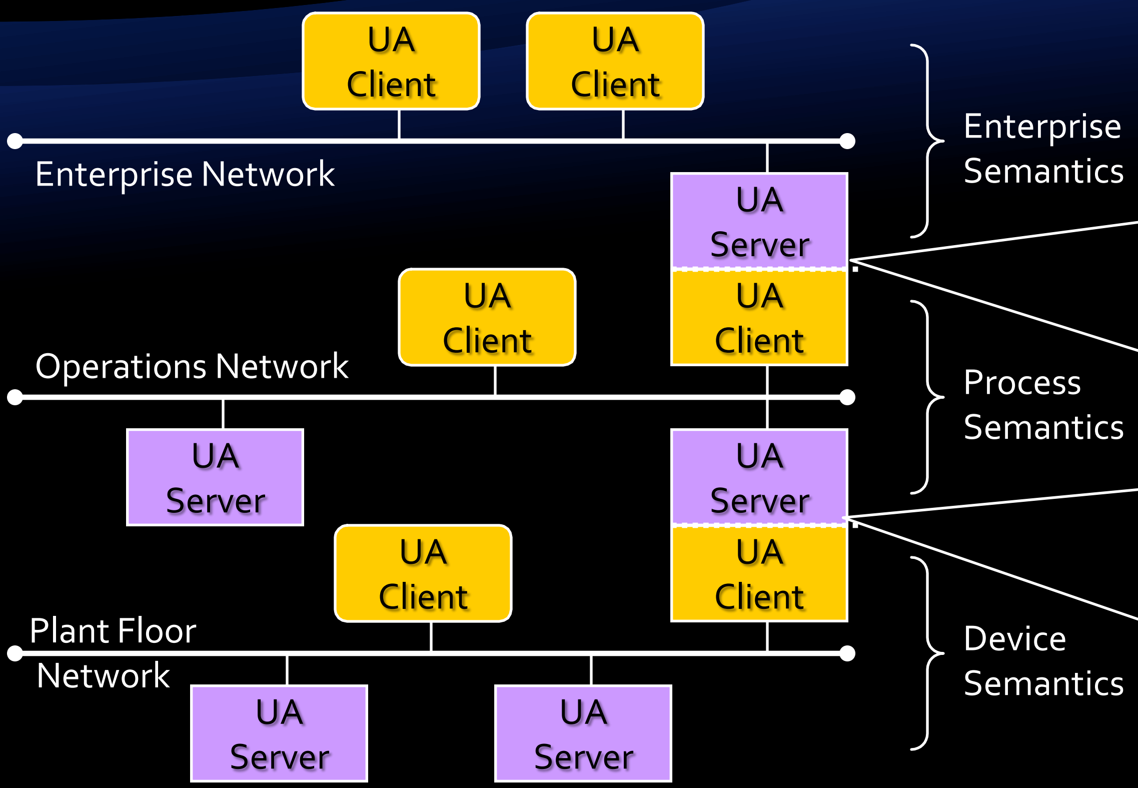




- Network Model
- Unlimited Named/Typed Relationships
- "Views" are used to present hierarchies
- *References* between *Nodes* permits *Servers* to organize the *AddressSpace* into hierarchies, a full mesh network of *Nodes*, or any possible mix.

- Real objects are physical or software objects
- Accessible by the OPC UA *Server* application
- A *View* is a subset of the *AddressSpace*.
  - *Views* are used to restrict the *Nodes* that the *Server* makes visible to the *Client*

UA Server Chaining

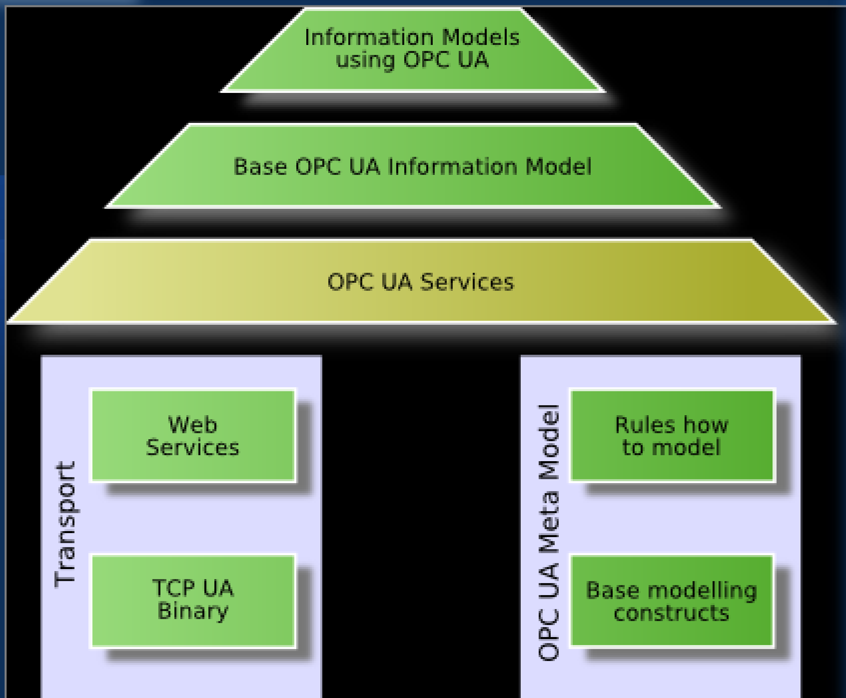


"Aggregating" UA Servers extract and process data from lower level "Device" UA Servers.

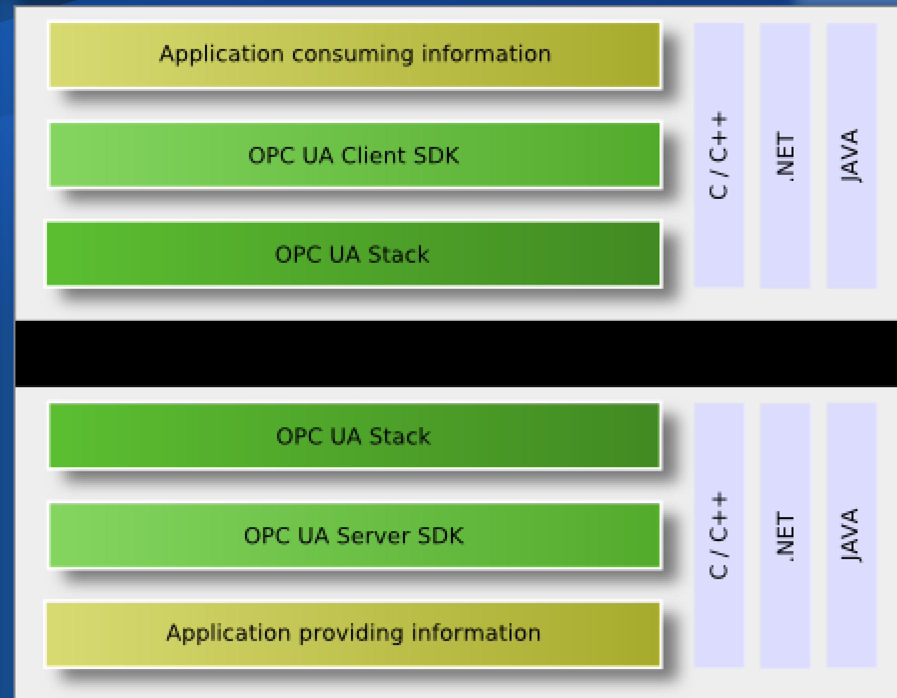
Data is recast using different information models appropriate for the clients at the higher level.

Figure from OSISoft

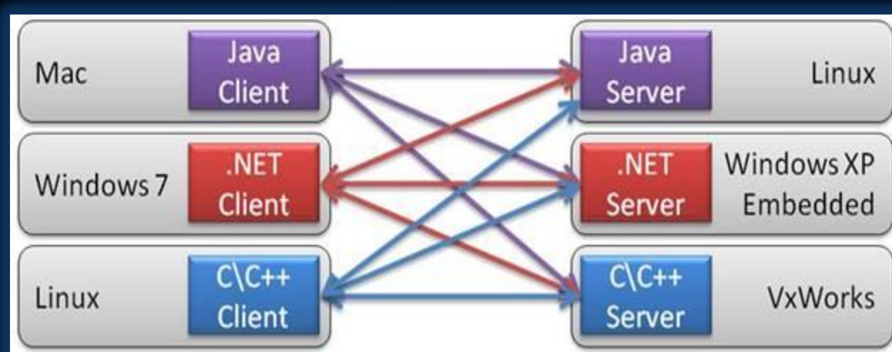
OPC/UA



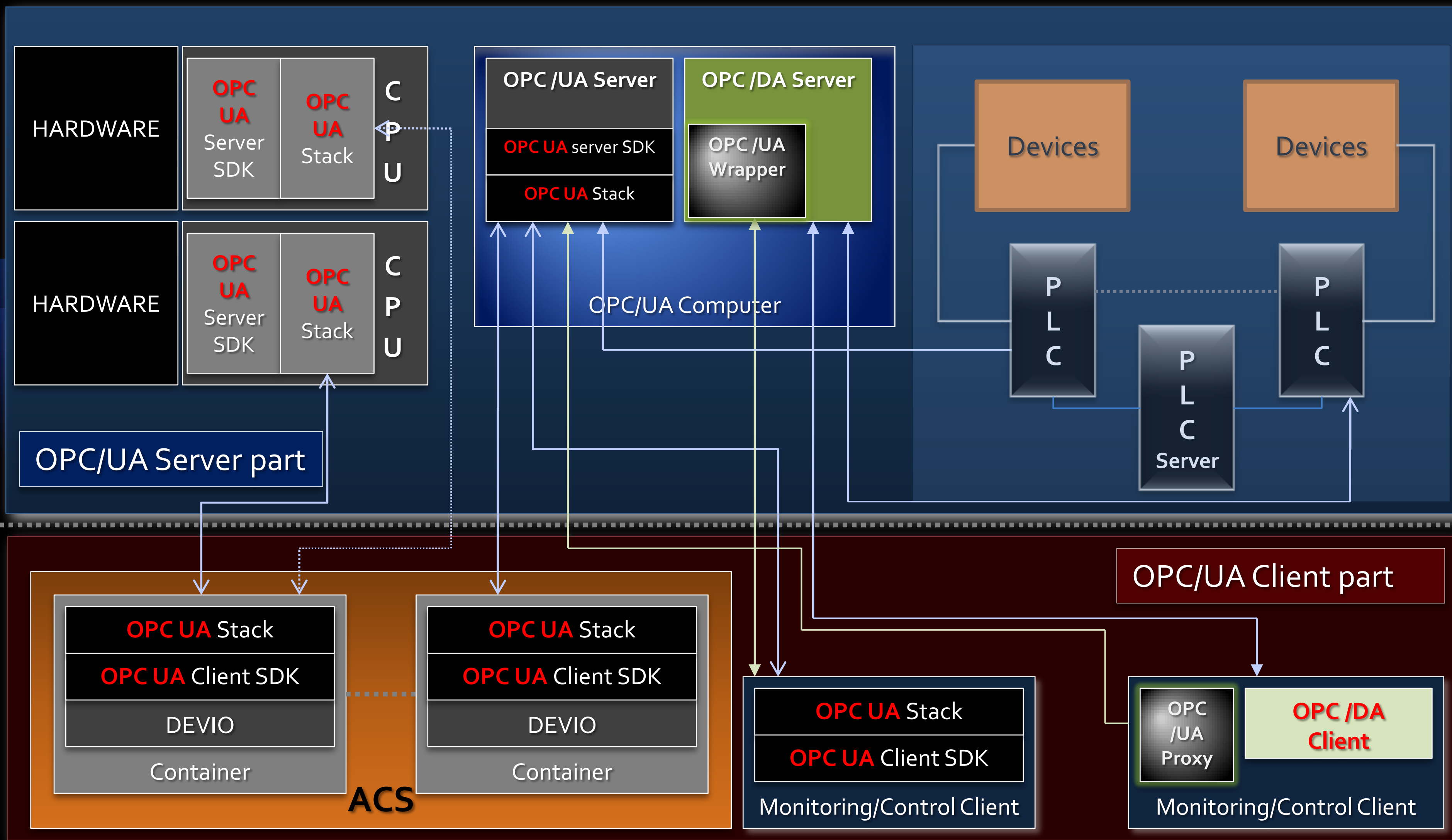
Information Models using OPC UA
Base OPC UA Information Model
OPC UA Services
Transport
Web Services
TCP UA Binary
OPC UA Meta Model
Rules how to model
Base modelling constructs



Application consuming information
OPC UA Client SDK
OPC UA Stack
C / C++
.NET
JAVA
OPC UA Stack
OPC UA Server SDK
Application providing information
C / C++
.NET
JAVA



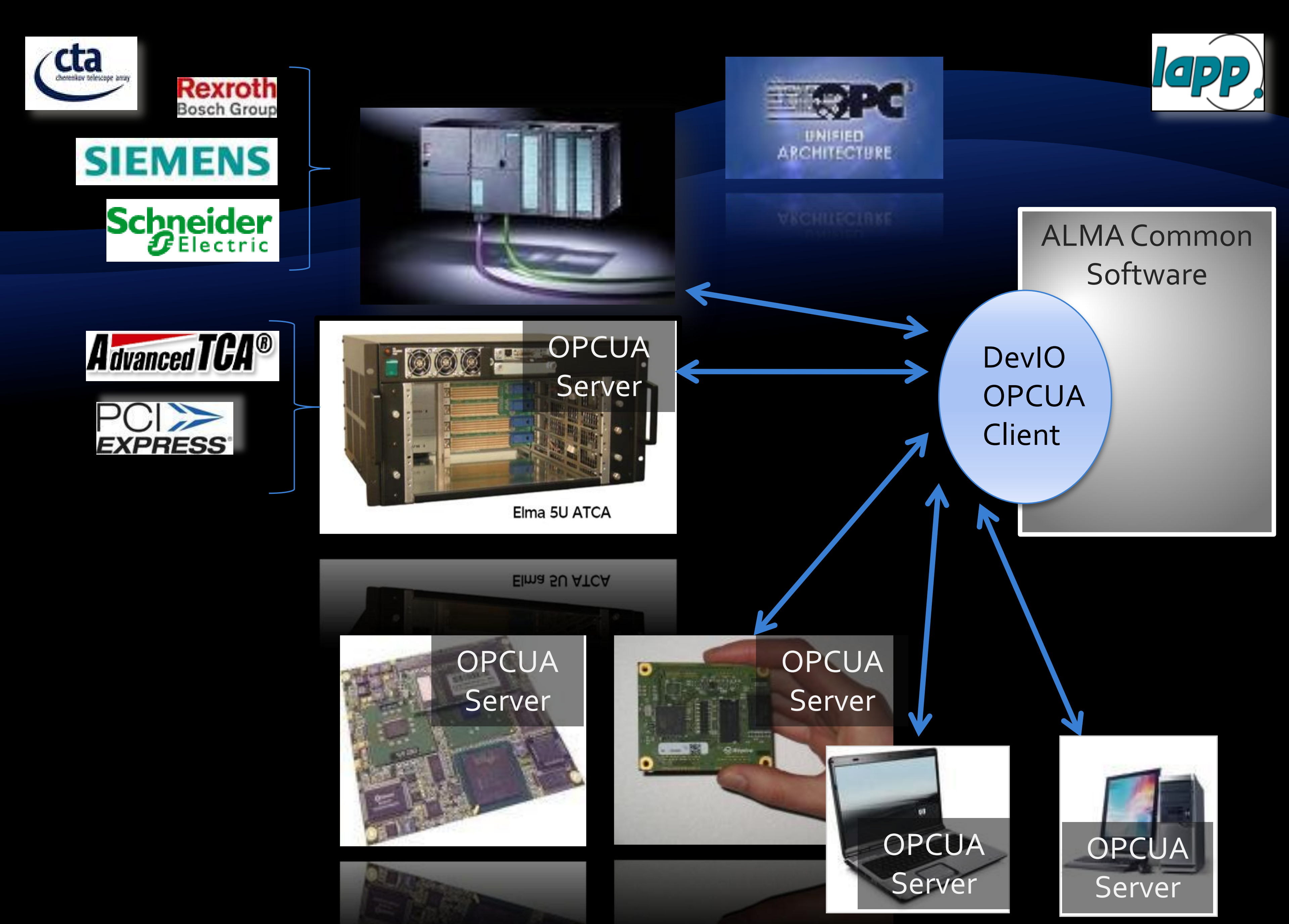
Mac
Java Client
Windows 7
.NET Client
Linux
C\C++ Client
Java Server
Linux
.NET Server
Windows XP Embedded
C\C++ Server
VxWorks



✓OPC UA server embedded in the PLC CPUs

✓Accessing the hardware ➔

✓Providing API and Libraries for client and server parts


cta
Rexroth
Bosch Group
SIEMENS
Schneider Electric
AdvancedTCA®
PCI EXPRESS
OPC
UNIFIED ARCHITECTURE
lapp
ALMA Common Software
DevIO OPCUA Client
OPCUA Server
Elma 5U ATCA
OPCUA Server
OPCUA Server
OPCUA Server
OPCUA Server

Summary

- Most of devices can be drive with OPC/UA Standard
- All software languages can be used :
  - .NET for Windows platform
  - C/C++ for embedded systems
  - JAVA for portable devices
- OPC/DA and OPC/UA can co-exists (Wrappers and Proxies)
- Homogeneous Environment
  - For portability :
    - CPU, Systems (Embedded or not)

Development@ LAPP

- Unified Automation GmbH (German company)
  - Kit Evaluation Version : C++ and ANSI C Server Software Development
    - precompiled libraries
  - OPC UA SDK for UA Client Development
- Environment
  - OPC-UA SDK Installed on Fedora 14.
    - openssl-0.9.8c - xml2.7.7 – gcc 4.5 required
  - ACS already installed on Red hat 5.3 distribution at LAPP and ported (easily) on Fedora 14
  - All software installed on a VMWare Fedora 14 image

Development@ LAPP

- OPCUA (client-Server C++)
  - Use a simple example of client/server program.
    - Compilation and link with OPC-UA framework
- ACS prototype :
  - CDB (simple configuration DB)
  - A OPCUA_devio for the connection with the OPCUA Server
  - A "container library" with some functions
    - Init() -> connect to OPCUA server
    - Read() -> describe all the structure of OPCUA server
    - Close() -> Disconnect to OPCUA server
- All components have been successfully compiled and run via the "ACS" Command Center.

Next@ LAPP

- Setting up a hardware platform including :
  - PLC, PLC Server, Security PLC
  - Some hardware (Motors,…)
- Understanding deeper the OPC UA Data model and Address Space.
- Monitoring (*HESS2 Security Crate*) via OPC/UA interfaces
  - UA Clients
  - UA Server (Siemens, …)

LAPP activities for LST Telescopes

Event

X

Alerts network

DAQ

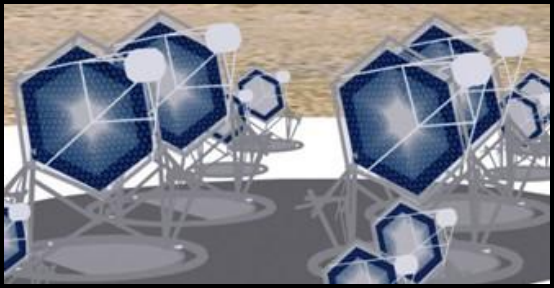
Communications with PLC

ie: T. Le Flour Annecy 2010

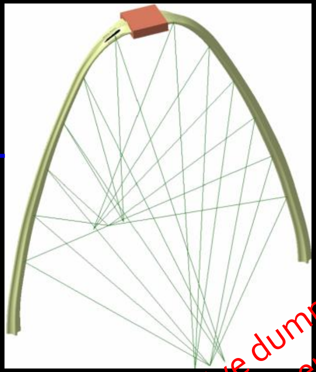
PLC

Telescope Automation

Critical Cycle in 20 s



Displacement of the telescopes



Stabilization of the camera

ie: G. Deleglise Oxford 2010

Active dumping System

Pointing of the target and data acquisition

21

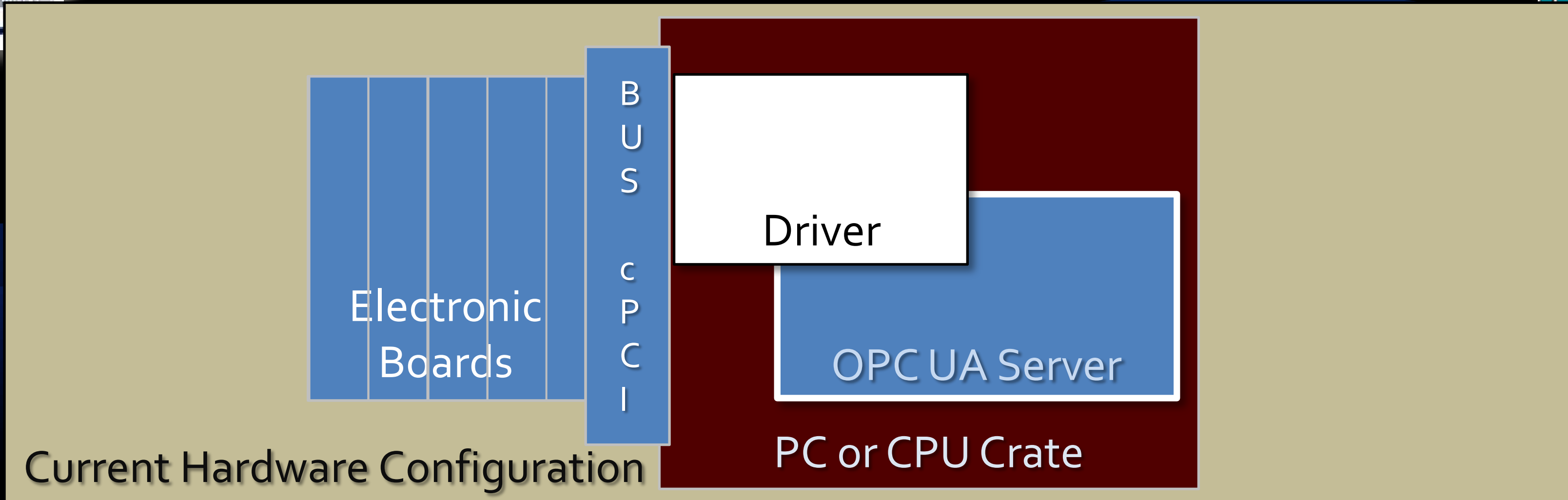
LAPP STATUS REPORT OPC UA / ACS

T. LE FLOUR/ J.L PANAZOL

reminders

- Objectives foreseen from 14/02/11 (last ACTL meeting)
  - OPC/UA study
    - Deeper understanding of this environment
  - Development and integration of an OPCUA environment in the HESS2 Camera Security Crate
    - Server parts
  - Development of OPCUA Clients (C++ and Java)

Current Status



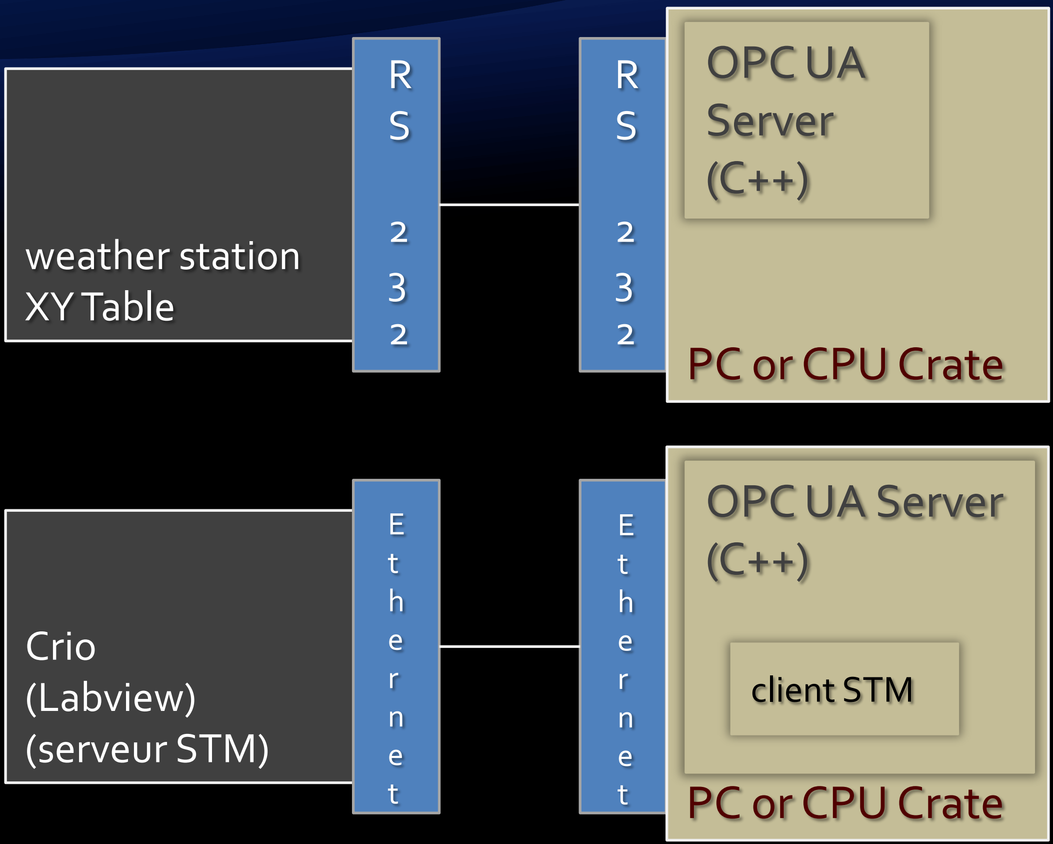
SERVER PART:

- PCI Driver has been ported on the Fedora14 Platform
- OPC UA Server implemented in C++ and currently runs on a Fedora14 platform
- All the possible actions on the OPCUA Nodes have been implemented
  - Standard calls on objects(Node) methods
  - Events
  - Callbacks

CLIENT PART :

- Server has been tested by using the generic client included in the *Unified Automation* distribution
- Server has accessed via a specific client written in C++
- An *ACS DevIO* encapsulating the OPC UA Server connection has been written, configured via Container description and tested successfully via the *ACS Command Center*
- A basic JAVA client also tested

Some other test platforms @LAPP



National Instrument CRIO Hardware used by the LST stabilization system driven by a LabView environment

Some news around OPC UA

- SIEMENS currently provide today an OPCUA environment (independently of the platform)
- BOSCH announces a OPCUA Environment before the end of this year
- National Instrument will probably go toward this OPC UA Environment (Questions on that topics on user forum has been asked)

Some feedbacks on an OPC UA Server Implementation

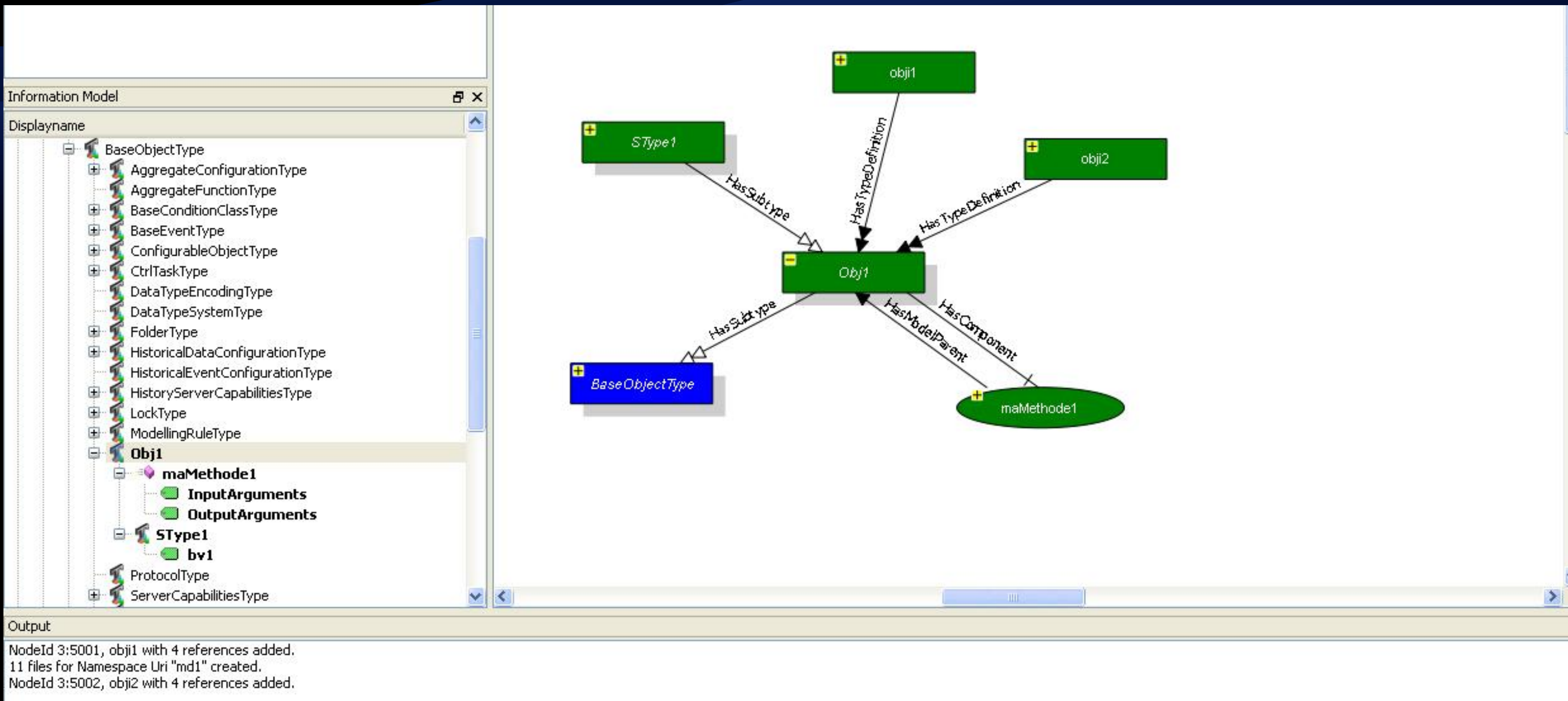
- For people in charge of implementing specific hardware access (without software development background) :
  - Coding could be complicated enough :
    - No description, many code lines to configure the server, the nodes, the node's properties, …
- This implies to put a layer on top of the existing software API.
- Tool boxes and expertise will be absolutely needed to make the implementation easier
- The client part is simpler to implement.

Some feedbacks on an OPC UA Server Implementation

- Some new features recently appears on the development kit of ***Unified Automation*** company.

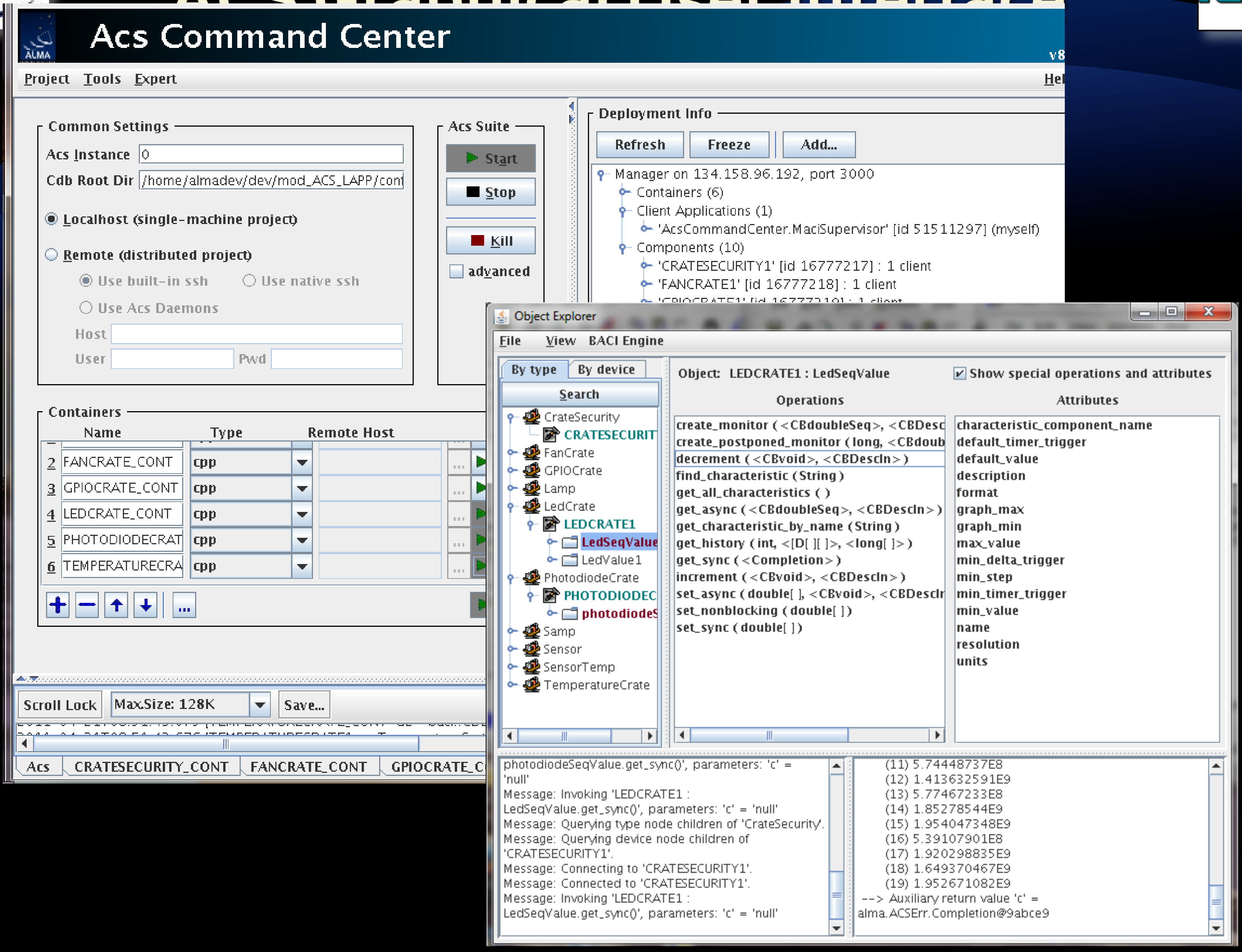
  - **UAModeler** : Graphical environment for the server description is proposed :
    - Full graphical node description :
      - Properties, specific node methods, …
    - Automatic code generation with code skeleton, xml server description , …

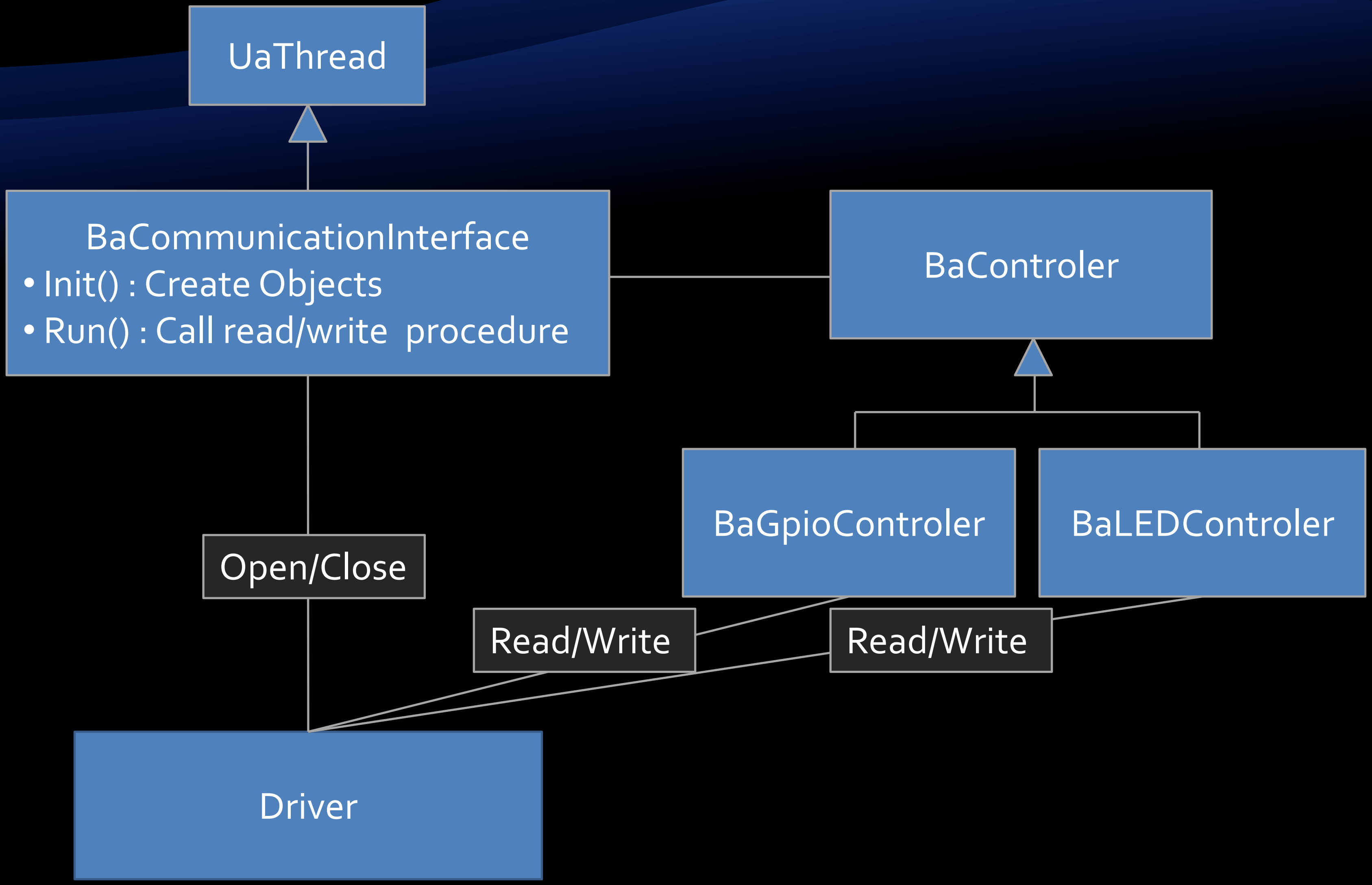
UA Modeler Graphical User Interface



UA Modeler Graphical user Interface
Nodes classes and instances views

ACS Graphical User Interface




UaThread
BaCommunicationInterface
• Init() : Create Objects
• Run() : Call read/write procedure
BaControler
BaGpioControler
BaLEDControler
Open/Close
Read/Write
Read/Write
Driver