Laboratoire d'Annecy-le-Vieux de Physique des Particules

Comparaison et retour d'expériences des implémentations OPCUA

- E. Chabanne (LAPP)
- D. Hoffmann(CPPM)
- P. Gauron (LAL)
- T. Le Flour (LAPP)
- P. Nikiel (CERN/ATLAS)
- J.L Panazol (LAPP)
- P. Sizun(CEA/IRFU)
- V. Voisin(LPNHE)

irfu cea saclay

CERN

CENTRE DE PHYSIQUE DES PARTICULES DE MARSEILLE CPPM

LABORATOIRE DE L'ACCÉLÉRATEUR LINÉAIRE

LPNHE PARIS

LAPP

cnrs

UNIVERSITÉ SAVOIE MONT BLANC

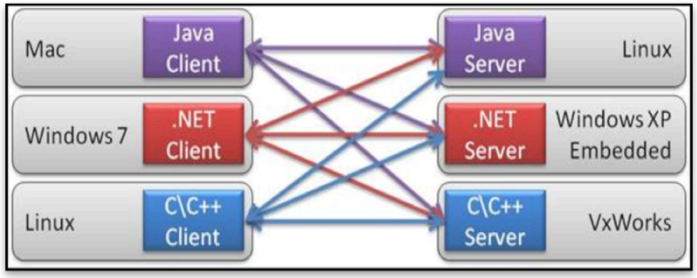
SOMMAIRE

- **Rappels sur l'OPCUA**
- **Les différentes piles OPCUA Client/ Serveur**
- **Exemples de contextes d'utilisation**
- **Quelques outils associés**
  - QUASAR
  - MOS
  - Client Matlab
- **Tendances et Orientations.**

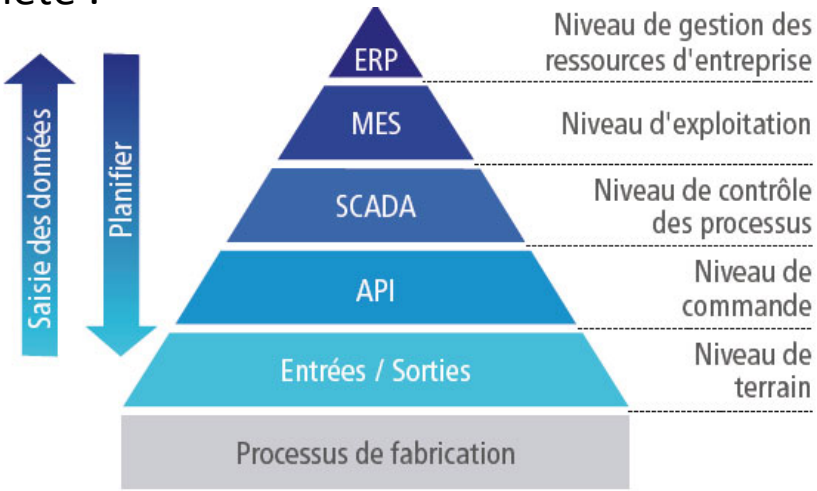
OPCUA #1

- **Présentation détaillée au JI2012 (https://indico.in2p3.fr/event/6514/session/4/contribution/16/material/slides/1.pdf)**
- **OPC et OPCUA sont des standards industriels permettant la communication de matériels ou logiciels hétérogènes à l'aide d'un interface commune. (M2M : Machine to Machine ou M2H : machine to Human)**
- **Architecture unifiée (UA), pour une meilleure interopérabilité**
  - Regroupement des fonctionnalités de l'OPC :
    - Unification des différents systèmes « OPC Classic » : OPC DA, OPC A&E, OPC HAD
    - Evolution du modèle de communication :
      - COM/DCOM vers 2 protocoles :
        - » Protocole binaire **UA TCP** : opc.tcp://Server offrant la meilleure performance et interopérabilité
        - » Protocole **SOAP/HTTP** : http:/Server pour les services Web et assurant l'interopérabilité avec les applications type « ERP ».
    - une architecture orientée service (SOA) multiplateforme
  - Evolution du modèle d'information => modèle d'information unifié OO
  - Renforcement de la sécurité. (Certificats X509)

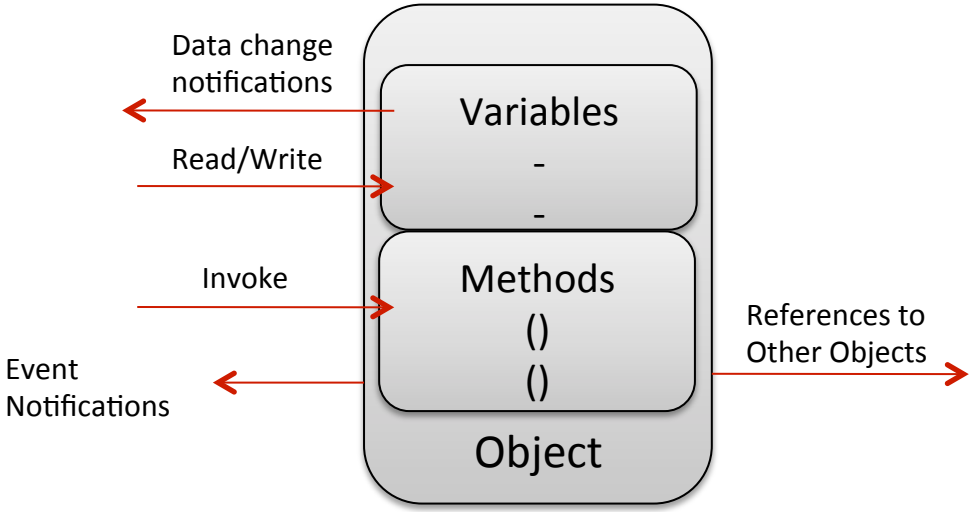
- **OPCUA : multiplateforme et multi-langage**



- Possibilité évolutive pour la mise en réseau complète :

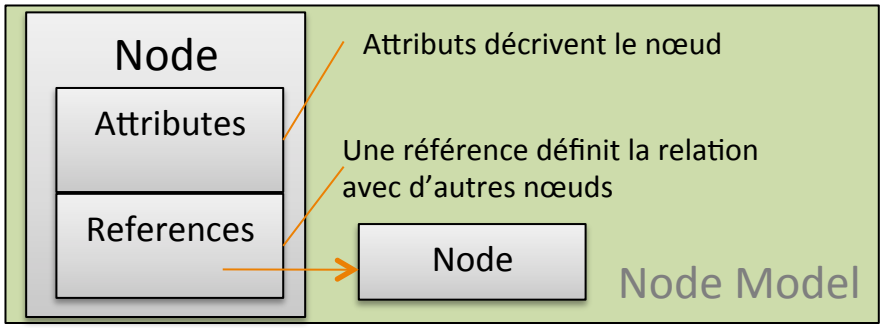


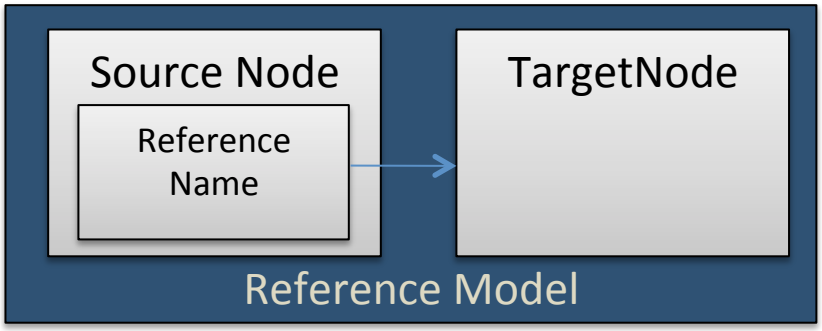
- **Le modèle d'information :**



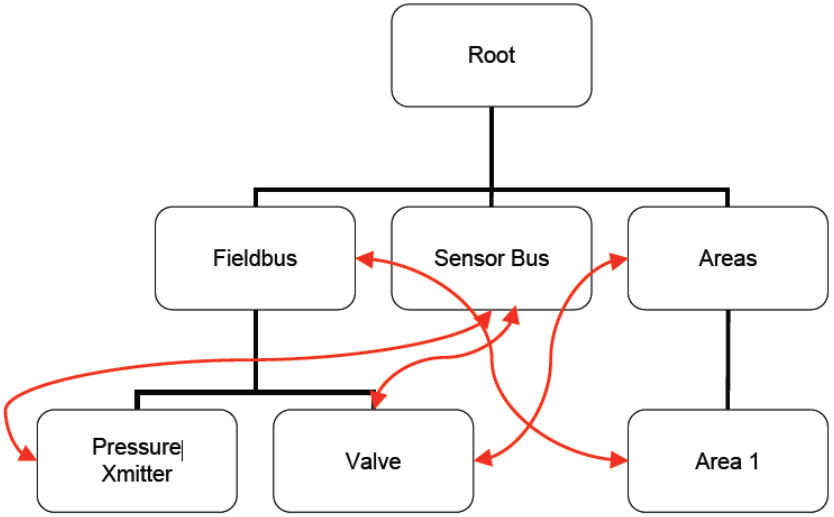
- Les éléments du modèle sont représentés dans l'espace d' adresse comme des "Noeuds"
- Chaque nœud est associe a une classe NodeClass

OPCUA #3





- **Les attributs décrivent des Nœuds.**
- **Les références sont :**
  - utilisées pour relier les nœuds entre eux
  - un nœud « cible » peut se situer dans le même espace d'adresse ou dans l'espace d'adresse d'un autre serveur



- Un modèle « réseau »
- Le nombre de relation entre objets est illimité
- Les « vues » sont utilisées pour présenter des hiérarchies
- Les ***Références*** entre les nœuds permettent aux serveurs d'organiser l'espace d'adresse en hiérarchies (réseau maillée de nœuds)

Piles logicielles Client/Serveur

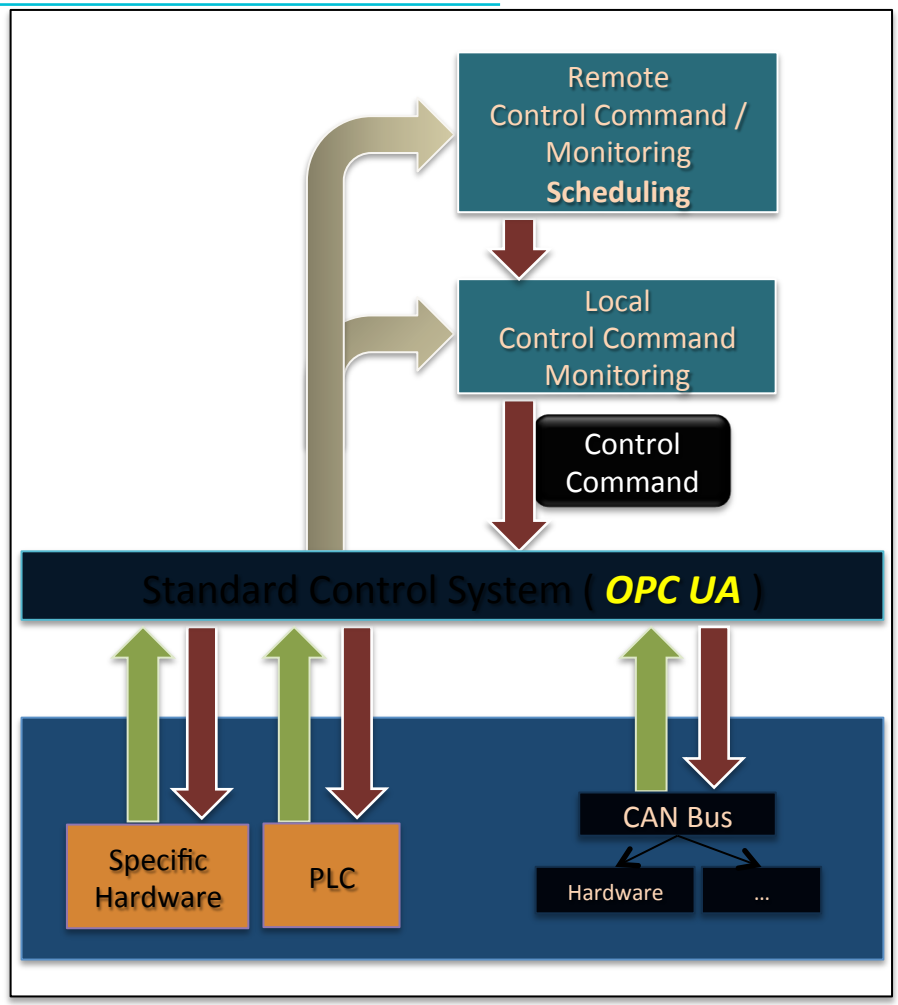
- **Marché commercial :**
  - Complètes et certifiées vis-à-vis des spécifications OPCUA (OPC Fundation)
  - Unified Automation (C++) :
    - supporté pour un nombre limité de plateforme
    - license « source » disponible pour une compilation vers des plateformes non supportées : SL, CentOS, ARM, …
  - Prosys (Java)
  - d'autres existent …

- **Logiciel libre :**
  - Niveau d'implémentation n'est pas toujours au même niveau :
    - les fonctionnalités ne sont pas toutes disponibles dans chacune des piles logicielles
  - « open62541 » : (C)
    - disponible sur les principales plateformes (Windows, GNU Linux, Raspbian OS)
  - **FreeOPCUA** (C++/python)
    - bibliothèque C++ et implémentation python !
  - **OpenOPCUA** (C/C++)
    - Debian/Windows

- **Entre deux mondes :**
  - pyUAF (Python wrapper)
    - Enveloppe une librairie binaire (commerciale ou libre) dans du code « utilisateur » Python

Les différentes piles OPCUA Client/Serveur

- **Retour d'expérience (CTA)**
  - Unified Automation et Prosys sont les 2 piles (commerciales) retenues pour les implémentations OPCUA coté client et serveur.
  - Le prix des licences a pu/peut limiter l'intérêt de l'OPC UA .
  - Le **logiciel libre** doit (devra) être considéré comme une alternative.
  - Cependant, il est important de :
    - **harmoniser** le choix en définissant des critères de sélection et d'études.
    - Eviter la libre utilisation de différentes piles :
      - Maintenance logicielle augmentée
      - Rythme variable d'implémentations des nouvelles spécifications de l'OPC UA pouvant provoquer des incompatibilités



Les différentes implémentations OPCUA serveur et cliente doivent être testées par l'« **OPC Foundation Certification Lab** » pour valider la compatibilité, l'interopérabilité, la robustesse. l'efficacité

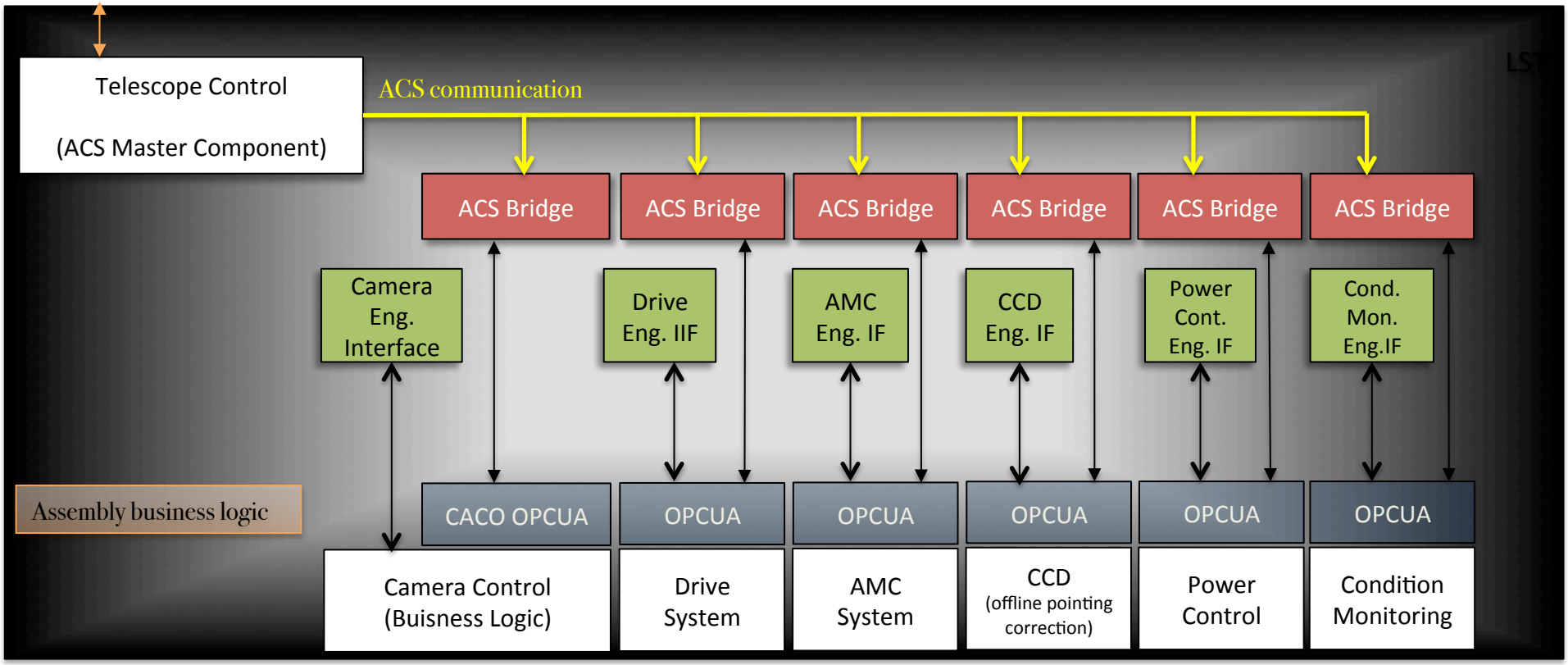
CONTEXTES D'UTILISATION

- **Contexte général : On-Line – Slow Control (intégration de matériel pour control et monitoring)**
- **CTA : Cherenkov Telescope Area**
  - OPCUA est le « middleware » prescrit pour le contrôle et le monitoring du matériel et son intégration dans le framework CTA
  - **MST** : Camera NectarCam , **LST :** Système « Drive », Pilotage des miroirs.
- **SuperNemo :**
  - Monitoring et Contrôle du matériel :
    Gaz Factory, Control Boards, Control Coil, Light Calibration, Source Calibration, Power Supply, Environnement de caverne de Modane etc….
- **ATLAS :**
  - Le système de contrôle d'expérience ATLAS :
    7 types de serveurs, 91 instances mises en production, 50'000+ canaux analogiques, contrôle de blocs d'alimentation(Wiener, ISEG)
  - *Interface aux matériels « custom » du CERN.*
  - *Accès client OPCUA : WinCCOA (Scada), PyUaf(Tests), UAExpert(diagnostic), applications spécialisées basées sur open62541.*

CTA : Environnent de développement

2 middlewares recommandés par CTA

- ACS (*ALMA Control Software*) au niveau du réseau (*Array ConTroL*)
- OPC UA au niveau des télescopes
  - « ***Passerelles*** »: ACS DevIO basés sur le SDK OPC UA en Java de Prosys

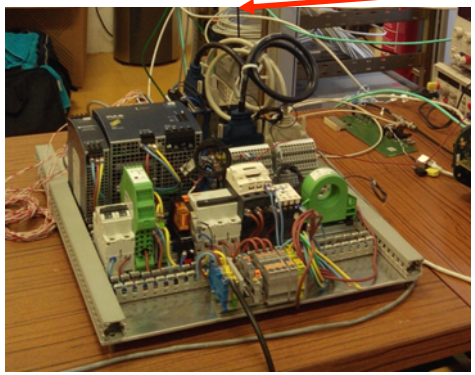


OPC UA dans la caméra NectarCAM pour les MST (*Medium Size Telescope*)

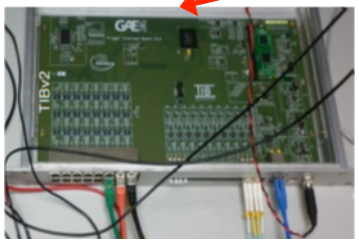
Array ConTroL

Remote Camera Controller

OPC UA
ACS
UDP
Telnet



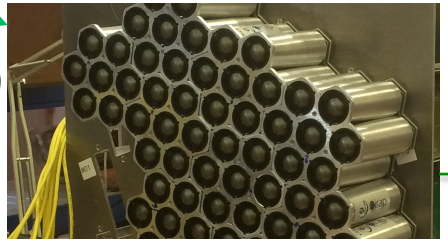
Embedded Camera Controller
(NI Compact RIO, MOS)



Trigger Interface Board
(Raspberry Pi, MOS)

- **Nectar Module Controller** (UA SDK)
- **Calibration Box** server (MOS)
  (Scientific Linux)

Camera Event Builder
(Scientific Linux, UA SDK)



Camera Modules(FPGA)

Unified Automation C++ server SDK :

- Nectar Module Controller (Scientific Linux 7): 265 modules x 200 variables, Qt GUI
- Event Builder (Scientific Linux 7)

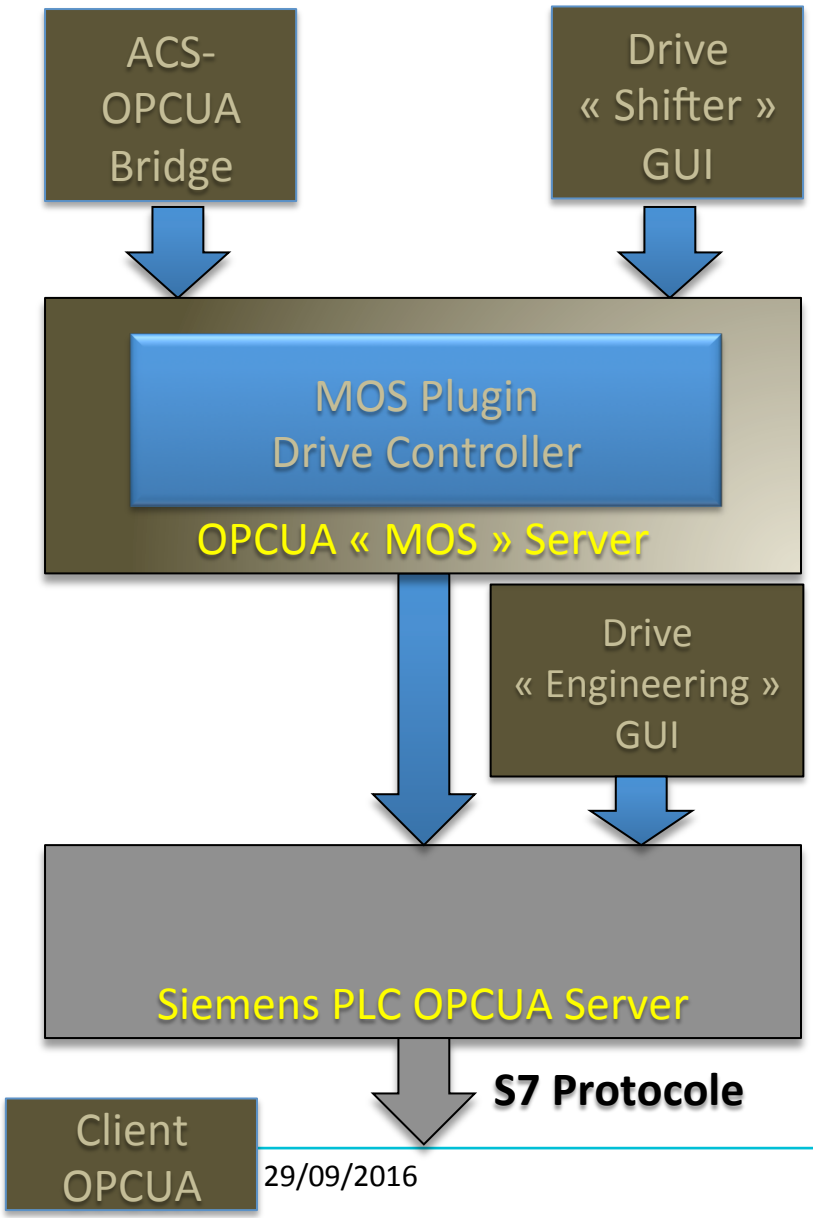
MOS

- Embedded Camera Controller (NI Compact RIO 9068)
- Trigger Interface Board (Raspberry Pi Compute Module 2)
- Calibration box, LED flasher (Scientific Linux 7)
- Equipements de test (switch optique, …)

PyUAF (API Python pour le client SDK d'Unified Automation)

- Scripts de calibration, de test, mise au point d'algorithmes
- Interfaces graphiques de contrôle en PyQt

CTA LST Drive



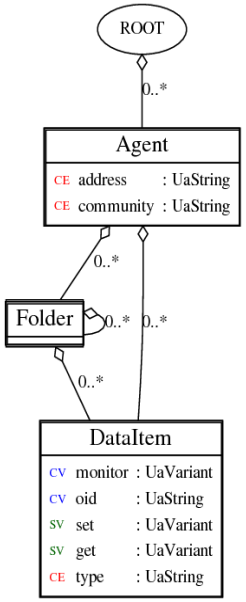
- **Le serveur OPCUA Siemens contient la table d'échange de données générée par le compilateur Siemens.**
  - aucun appel de méthode
  - procédé d'exécution à la *mode* « PLC »
- **Le serveur OPCUA « MOS » permet :**
  - l'implémentation de méthode **OPCUA** , conformément a un ICD
  - assure l'intégration du contrôle du « Drive » pour les couches supérieures.
  - en tant que **client OPCUA**, la communication avec le PLC (OPCUA server) a travers l 'implémentation d'un « *plugin* » MOS

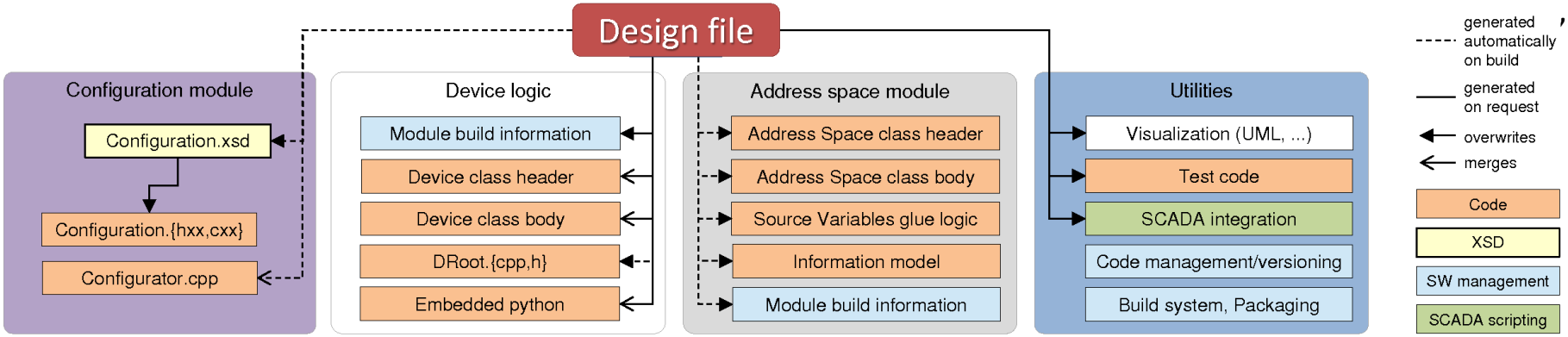
Les outils OPCUA

- **Pourquoi développer des outils autour d'OPCUA ?**
  - Les intégrateurs de matériel ne sont pas « obligatoirement » des développeurs
    - Les outils doivent faciliter l'intégration
  - Les outils doivent « cacher » la machinerie OPCUA
  - Les outils doivent limiter le nombre de ligne de code a écrire.
  - Les outils doivent fournir les moyens de décrire « simplement » le matériel : contrôle, surveillance, configuration, …
  - Les serveurs et les clients peuvent partager les memes descriptions.

Quelques outils associés : QUASAR

- Quick OPC-UA Server Generation Framework
- Création des serveurs OPC-UA (en C++) utilisant modèle semblable à l'UML
- L'utilisateur n'écrit rien à la main sauf un code spécifique à chaque application (nommé "device logic")
- Fonctionne avec les piles: Unified Automation et opServeurs fonctionne sous Linux, Windows, Linux embarqué (ARM, FPGA softcores etc.)
- Utilisés par les entreprises ISEG, CAEN, pour OPC-UA pour leur blocs d'alimentation
- Utilisés par 21 types de serveurs (5 en production)
- Beaucoup de « modules optionnels » (CAN, Python device logic, …)
- https://github.com/quasar-team/quasar





Quelques outils associés : MOS *Multipurpose OPCUA Server*

1 Application Multipurpose OPCUA Server

1 fichier de description pour adapter le serveur à votre projet

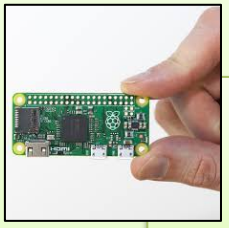
1 « Plugin » :
Librairie spécifique pour accéder à votre matériel ou autre …

Application => Pas d'implémentation de code
Binaire => Pas de compilation :
Pas besoin de licence.

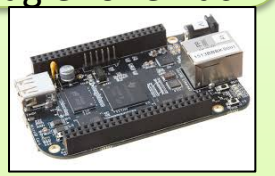
Description XML pouvant être extraite d'un document type « *ICD : Interface Control Document* »

OPTIONNEL
Si besoin : implémentation de la *« logique métier »* avec ou sans bibliothèque externe

PC bureau
PC Industriel
PC « Rackable »
Laptop PC



ARM
Raspberry Pi
Raspberry Pi Zero
BeagleBoneBlack

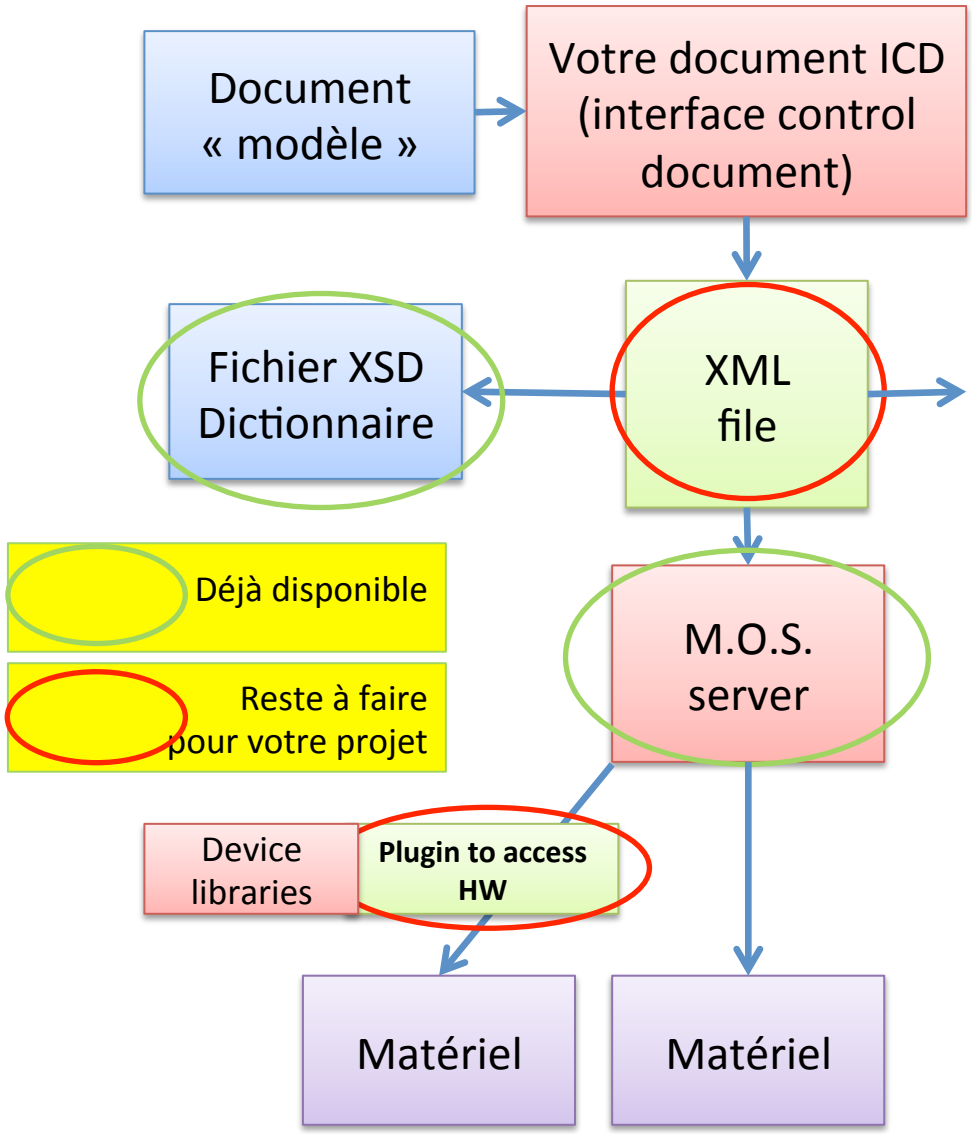


Multipurpose-OPCUA server

Other Platform
Altera SocKit eval board
(ARM + FPGA SoC)

National Instrument
cRIO
(RT linux)
ARM

Quelques outils associés : MOS




```
88
89      <CompoundDatapoint>
90        <Name>Mirror_</Name>
91        <Multiplicity>10</Multiplicity>
92        <CompoundDatapoint>
93          <Name>Actuator_</Name>
94          <Multiplicity>2</Multiplicity>
95          <SimpleDatapoint>
96            <AccesLevel>3</AccesLevel>
97            <Name>Turn</Name>
98            <Type>int32</Type>
99            <Method>
100             <Name>turn_left</Name>
101             <DeviceInstruction>
102               <Name>TurnLeft</Name>
103               <Value>TurnLeft</Value>
104             </DeviceInstruction>
105           </Method>
106           <Method>
107             <Name>turn_right</Name>
108             <DeviceInstruction>
109               <Name>TurnRight</Name>
110               <Value>TurnRight</Value>
111             </DeviceInstruction>
112           </Method>
113           <Method>
114             <Name>goto</Name>
115             <Argument>
116               <Name>value XX</Name>
117               <Type>int32</Type>
118               <Access>Input</Access>
119               <Description>xx mm</Description>
120             </Argument>
121             <Argument>
122               <Name>value YY</Name>
123               <Type>int32</Type>
124               <Access>Input</Access>
125               <Description>yy graycode</Description>
```

Exemple de fichier de description

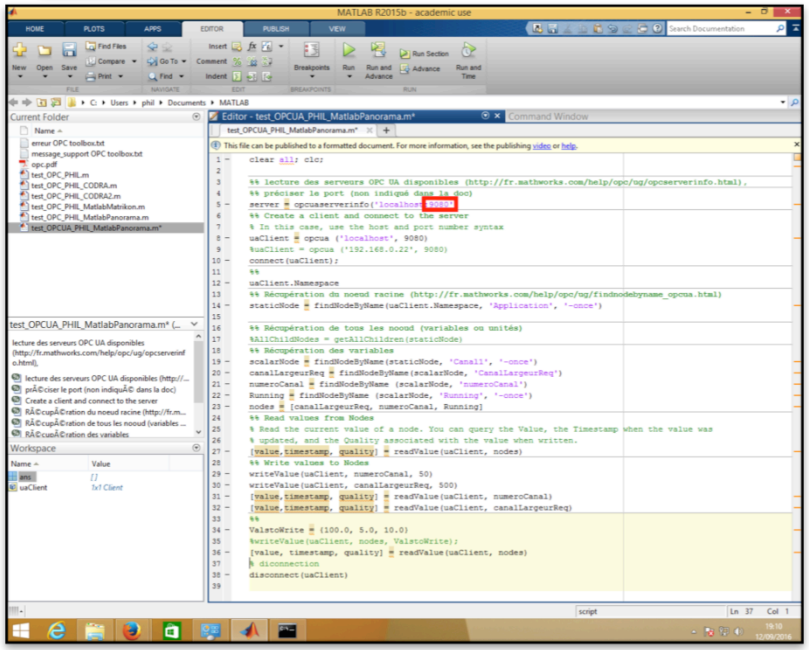
Quelques outils associés : Matlab

- Essais pour le démonstrateur ThomX (projet Idex) : un client Matlab fait par un utilisateur s'interface avec la supervision Panorama, point de connexion unique, qui relaie/traduit au contrôle-commande

- Client Matlab
- Supervision Panorama
- Serveur OPC-UA, client TANGO

CC ≠ serveurs TANGO

- module «OPC toolbox» inclus à la licence IN2P3 pour utiliser Matlab comme client OPC-UA (version > 2015b)
- essais effectués avec le serveur OPC-UA de Panorama (dans le client Matlab, préciser le port : «localhost***:9000***»)



Tendances et Orientations

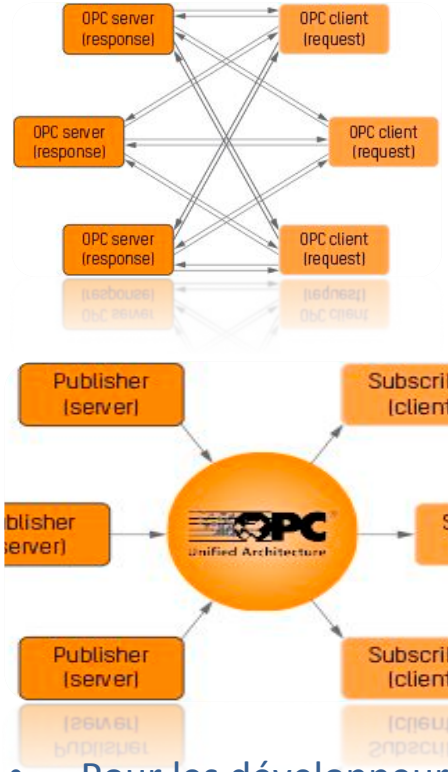
- **La fondation OPC a récemment ouvert l'accès a son « dépôt » de code : http://github.com/opcfoundation**

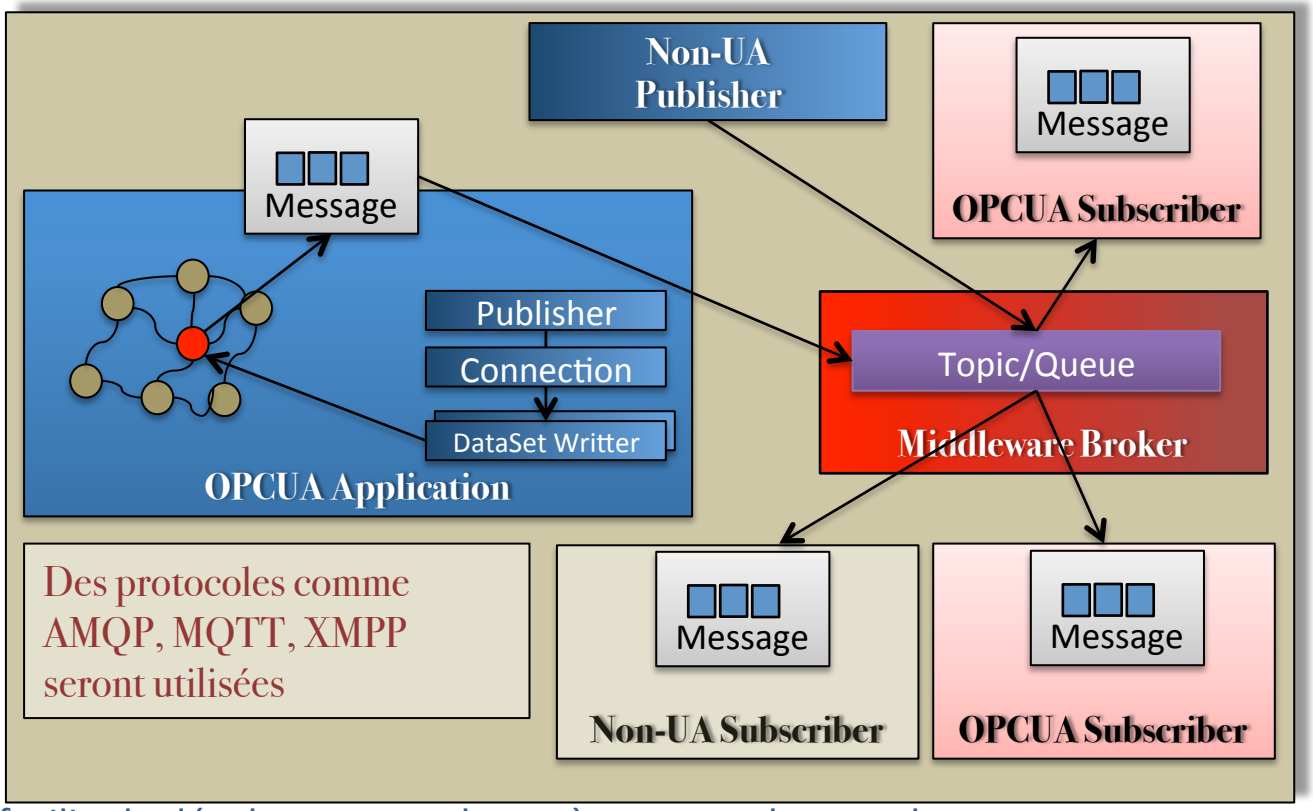
- **La politique du logiciel libre se concrétise pour favoriser l'adoption de la technologie.**

- **Fondation OPC and OMG annonce une collaboration pour l'étude de la connectivité entre OPCUA et DDS (Data Distribution Service)**
  - Travail concret sur la mise en œuvre d'une passerelle entre l'OPCUA (plant-floor data)

Tendances et Orientations

- **La fondation OPC a annoncé l'ajout des mécanismes de «publication/ souscription» a la spécifications OPCUA**





- Pour les développeurs, cela facilite le développement de systèmes complexes et hyper communicants : la spécification Pub/Sub définit un format pour les messages permettant leur utilisation par des « souscripteurs » sans connaissance OPCUA.
- La mise en œuvre de solutions d'automatisation flexibles et modulaires s'en trouve également simplifiée.

Tendances et Orientations

- ***IOT*** **: Internet of Things and** ***IIOT*** **: Industrial Internet of Things**
  - Chaque niveau devient « smart » : SmartSensor, SmartMachine, SmartProduct, SmartFactory
  - Besoin d'analyse rapide de ces lots de données (Big Data)
  - Pour une analyse des problèmes, une recherche d'optimisation, de prévisions, des prises de décisions.
- **Industrie 4.0 :**
  - 4e révolution industrielle : ***numérisation*** après la mécanisation, l'industrialisation et l'automatisation pour une meilleure gestion de la production et de la logistique.
  - Modèle d'architecture RAMI4.0 développé par l'industrie allemande où ***l'OPCUA*** est sélectionné comme la technologie de communication.

Conclusions

- OPC UA, successeur d'OPC
  - pisté comme « techno d'intérêt » en 2012
  - devenu un candidat *solide* en 2016
- Nombreux choix d'implémentation autour d'un standard (IEC 62541)
- Multitude de fournisseurs garantissant la pérennité
- Ouverture vers la communauté « Logiciel Libre »
- Nombreux exemples d'utilisation dans les communautés HEP accélérateur (CERN) + astro
- Compétences consolidées dans IN2P3/IRFU : au moins 4 labos connus à ce jour

Laboratoire d'Annecy-le-Vieux
de Physique des Particules

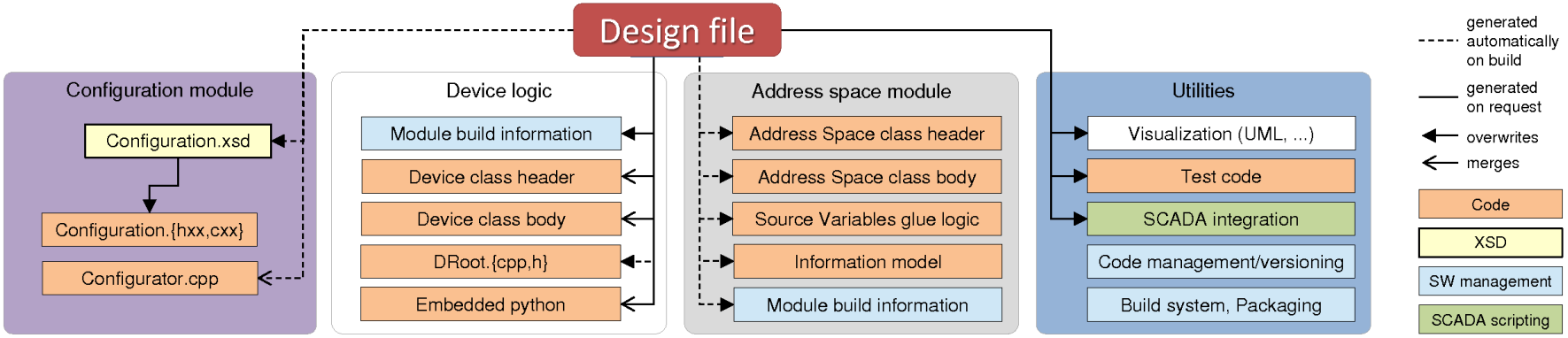
BACKUP SLIDES

cnrs

UNIVERSITÉ
SAVOIE
MONT BLANC

QUASAR en détail

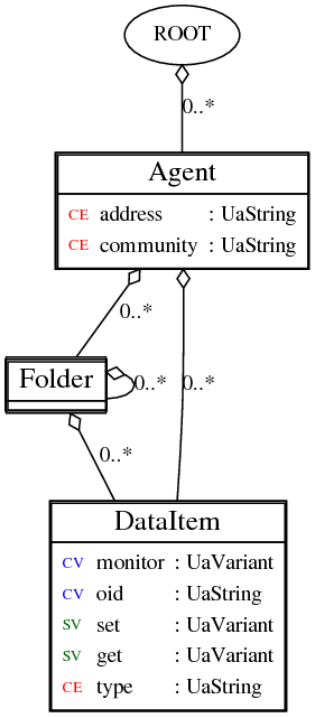
- L'entrée: modèle d'information en XML
- Génération du code C++, du XSD de configuration et d'autres:



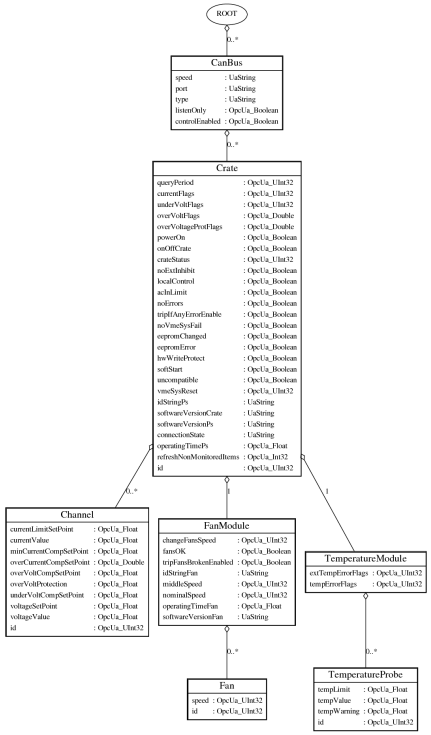
- 1 design par type du serveur, 1 fichier config (XML) par instance du serveur

- Plus d'informations:
  - A generic framework for rapid development of OPC-UA servers, CHEP2015, Okinawa, Japan
  - QUASAR – A generic framework for rapid development of OPC-UA servers, ICALEPCS 2015, Melbourne, Australia

- **Exemples des modèles « UML » en Quasar**



Serveur generique
(ici: SNMP)



Serveur specifique
(ici: bloc d'alimentation Wiener)

Open62541 en QUASAR et ATLAS

- En QUASAR, on offre une couche d'abstraction adaptant l'API d'open62541 à l'API d'UA-SDK
  - nommé open62541-compat
  - https://github.com/quasar-team/open62541-compat
- Bien adaptée pour création de serveurs
  - Peut être utilise dehors QUASAR
- Encore quelques limitations d' open62541-compat
  - Quelques types de donnes pas encore ajoute
  - « no-sampling » pas encore fait en open62541… etc…
- Open62541 utilisé aussi pour les clients
  - Par exemple, un client OPC-UA spécialise pour WinCC OA
- L'API souvent plus simple que l'API de UA-SDK
- Pas des problèmes de licences!

- MOS is an OPCUA server written in C++ with Unified Automation environment toolkit. Documentation and tutorials available : https://forge.in2p3.fr/projects/mos/wiki/MOS_Tutorial
- MOS runs on any Linux platforms , ARM and NI CPU.
- Each device or composed device is described by using an XML file.
- The description concerns :
  - The monitoring points
  - The configuration points
  - The control points
  - The methods
  - Device commands.
- The description is used to provide an well organized OPCUA name space accordingly to the device organization (XML description)
- Complete presentation in Indico (1rst OPCUA workshop in Milano October 2014) https://www.cta-observatory.org/indico/conferenceDisplay.py?confId=749

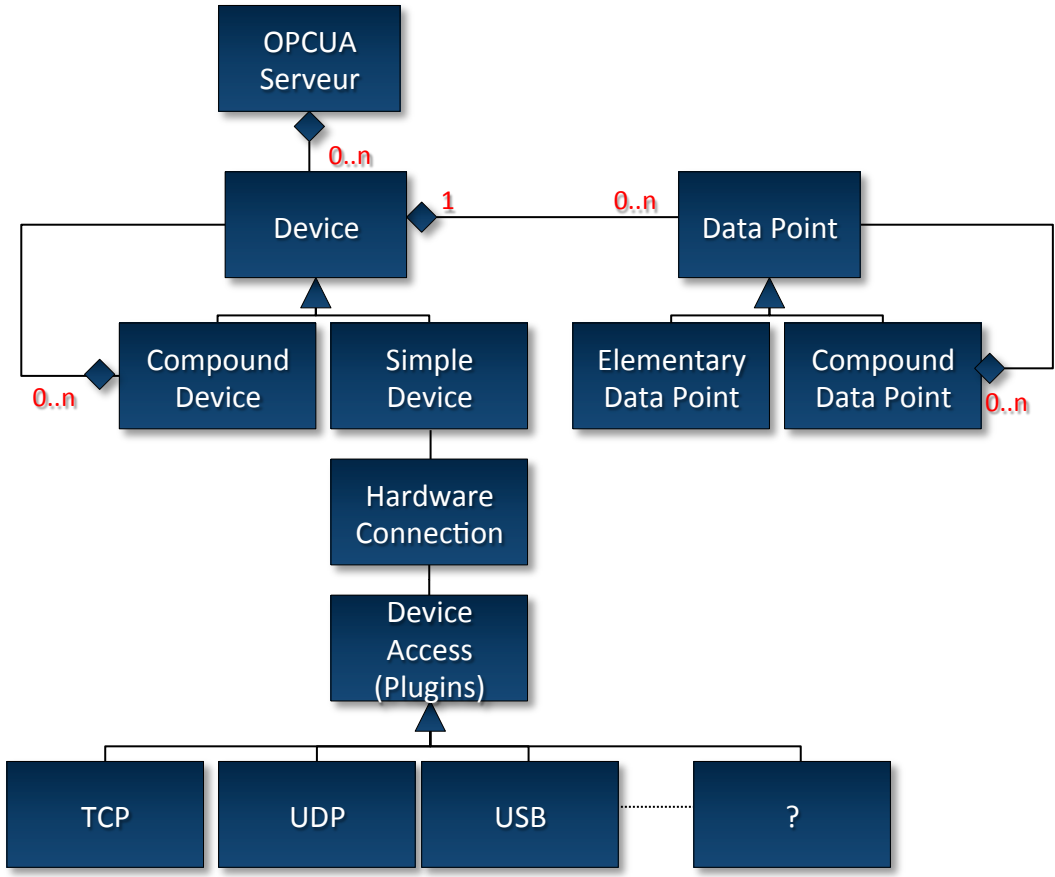
MOS :Device communication

- The MOS OPCUA Server is responsible to send the order to the devices and collect the information coming out the device (monitoring)

- Depending on the device communication complexity , 2 cases :
  - For a basic communication : all the commands can be easily described in the XML file and send to the device by MOS.
  - For a complex communication : all the commands can be described in the MOS but the command management is done via a user plugin (implemented by the user/integrator)
    - MOS send information to the plugin.
- Plugin :
  - Server plugin : written by the developer without any dependencies to the UA libraries ➔ no licence.
  - Client plugin : a server plugin can send acces (as a client) a distant OPCUA server .
    - No dependency with the UA librar ➔ No licence
  - Plugin access are described in the MOS XML File (location , ...)
  - C++ Dynamic loading mechanism used for.
  - Built-in plugin are available : TCP, UDP , ...

MOS : XML description

- XML is used to describe a device or a composed device and all the associated characteristics.
- XML is used to generate a well organized OPCUA name space(naming convention) following the device hierarchy.
- MOS can be used :
  - to communicate with a device
  - a simulator :
    - Depending on how you describe the device or the data point/ methods in a device :
      - MOS can be used as PLC simulator (no method call)
      - MOS can be used as top layer of a device.



Method calls

- 2 types of method calls :
  - Synchronous method call :
    - Standard in OPCUA :
      - Method call returns when the execution is finished
      - Client is hanged to the end of method execution.
    - Well adapted for short execution
    - Default setting in MOS
  - Asynchronous call :
    - Well adapted for long execution
    - MOS launches a thread , creates a OPCUA structure in the name space for the return parameters.
    - The OPCUA method call returns immediately
    - At the end of the method thread, MOS fills the output parameters area and sends a event associated to the corresponding method.
    - Client is responsible to listen to the end of method call (OPCUA event management) and read the output parameters area.