Intégration du matériel

Homogénéisation des accès

OPCUA

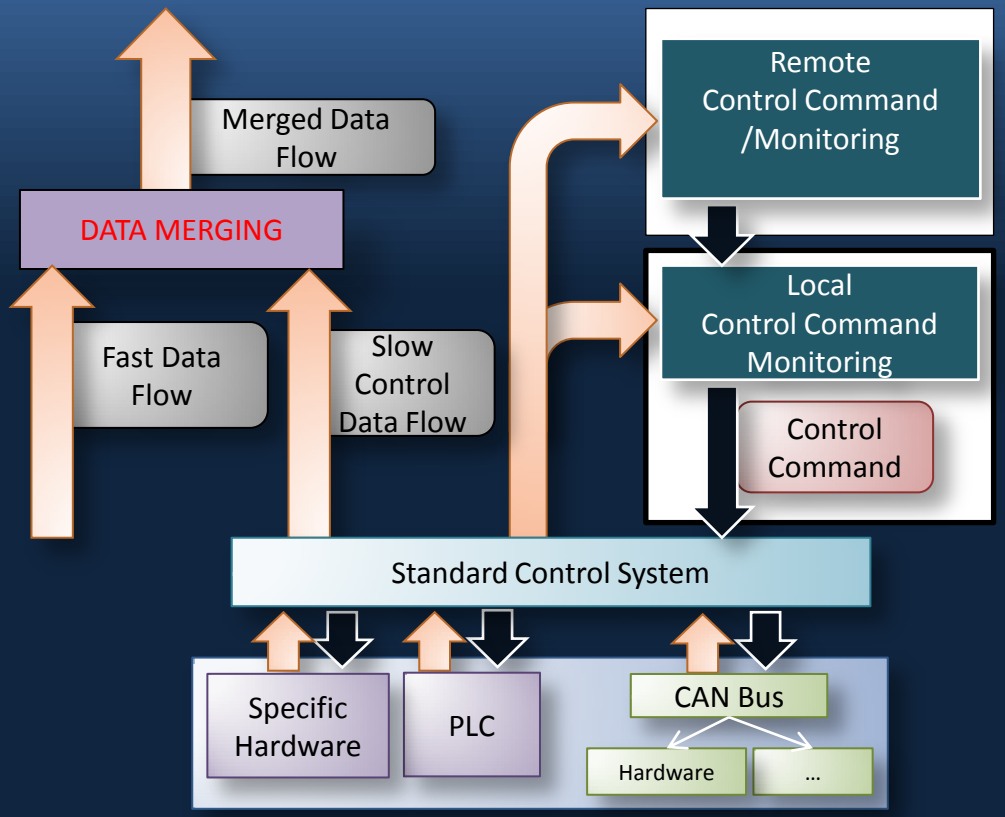
L.A.P.P :

T. Le Flour, J.L Panazol

Plan

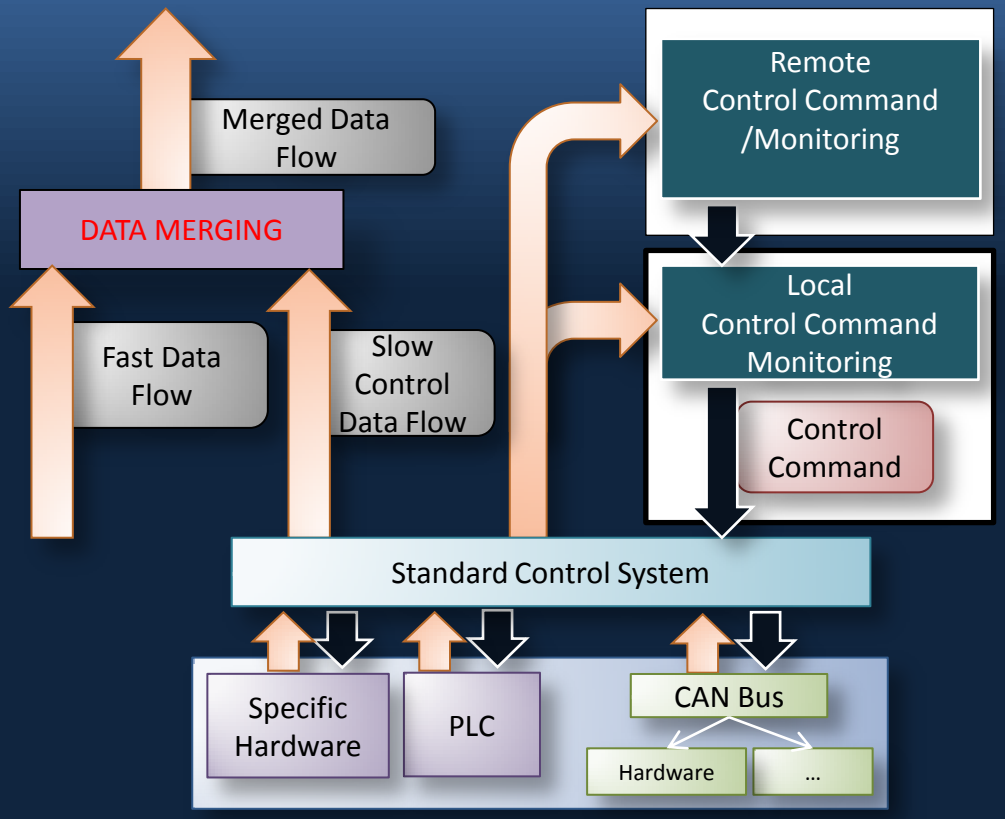
- Le “Slow Control”
- De l’OPC a l’OPCUA
- CTA
- Vers une description générique
- Conclusion

Le « Slow Control »



- Control de l'environnement
  - Alarmes, Seuils, …
- Configuration du matériel
- Monitoring du matériel
- Pilotage de matériels hétérogènes
  - Sécurité, …
- Qualité des données
- Maintenance
  - Statistiques d'utilisation
  - …

Le « Slow Control »



- D'un point de vue « Utilisateur »
  - Accéder a l'information sur site et a distance
  - Connaitre l' état global des systèmes , sous/systèmes
- Accès a l'information indépendant du contexte et de façon homogène
  - Depuis les composants DAQ

•*Objectifs* :

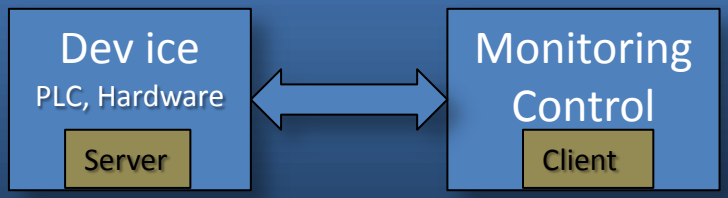
•Homogénéisation des accès aux matériels.

• Standardisation et Hiérarchisation du monitoring/contrôle.

Quelques remarques générales

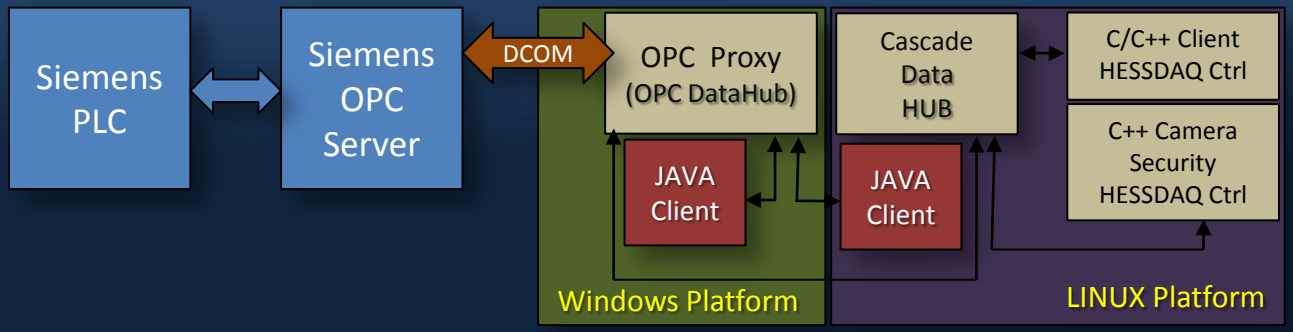
- En fonction du niveau d'expertise, un utilisateur doit connaitre l'état courant du matériel(localement et a distance)
  - ➢Hiérarchisation de l'information
- L'accès a ce type d'informations doit être considéré :
  - ➢ Réseau local et réseau étendu (firewall)
- Plusieurs type de données « Slow Control » coexistent :
  - ➢ Celui lié a la qualité de la données.
  - ➢ Celui lié a l'information sur le matériel (Taux d'utilisation , Fréquence de panne)
    - •Utilisable directement par des outils de diagnostiques.

L'intégration de matériel



1. Protocole « maison »
    1. Difficile a développer
    2. Difficile a maintenir
    3. Difficile a faire évoluer

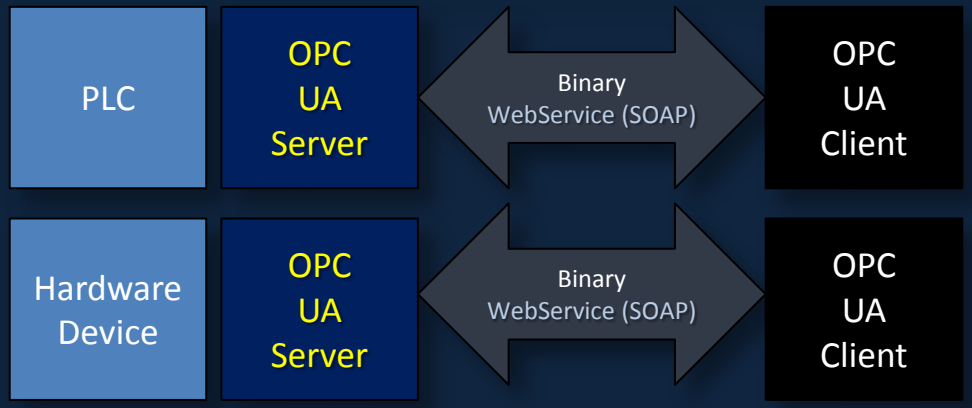
1



HESS 2 Camera Securite & 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 ou transfert binaire)
2. Independence de la plate-forme et interopérabilité
3. Tout langage possible(Cote client et serveur)
4. Evolution

3

De l'OPC a l'OPCUA

- Traditionnellement, chaque fois qu'un programme nécessitait l'accès aux données d'un périphérique, une interface personnalisée, un pilote ou driver, devait être écrit.
- L'objectif de l'OPC est de définir une interface commune écrite une fois puis réutilisées par n'importe quel logiciel d'entreprise.
- Un serveur OPC dédié a un périphérique particulier peut être réutilisé par n'importe quelle application agissant en tant que client OPC.
- Un serveur OPC utilise la technique Microsoft OLE (aussi connu sous le nom de Component Object Model ou COM) pour communiquer avec les clients.
- OPC est une marque déposée de la Fondation OPC.

De l'OPC a l'OPCUA (2)

- OPC a été conçu pour relier les applications Windows et les matériels et logiciels du contrôle de processus.

- La norme définit une méthode cohérente pour accéder aux données de terrain de dispositifs d'usine.
  - Reste la même quel que soit le type et la source de données.

- Les serveurs OPC fournissent une méthode permettant à différents logiciels d'accéder aux données de dispositifs de contrôle de processus, (exemple : un automate)

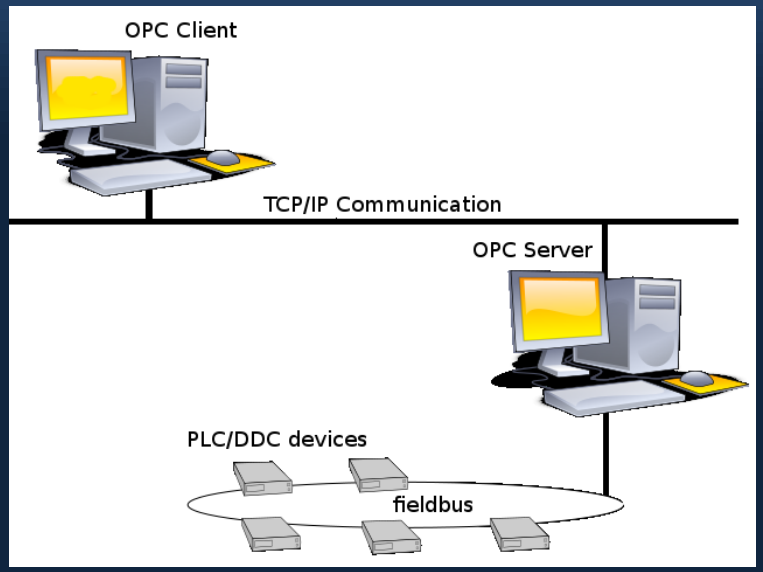
De l'OPC a l'OPCUA (3)

- Un serveur OPC :
  - Se comporte comme une API(Application Programming Interface) ou un convertisseur de protocole
  - Se connecte a un module « Hard » comme un PLC (Programmable Logic Controller)
    - Rafraichit les donnes avec un temps de cycle interne ~ 100ms.
  - Traduit les données dans un format standard OPC

- Les applications OPC (clients OPC) se connectent au serveur OPC pour :
  - Lire et écrire de données

- Plusieurs clients peuvent se connecter a un même serveur OPC

- Un Client OPC peut se connecter a plusieurs serveurs OPC simultanément.



Spécifications OPC

Séries de spécifications

OPC DA

Communication des données réelles a partir du matériel (PLCs,…)

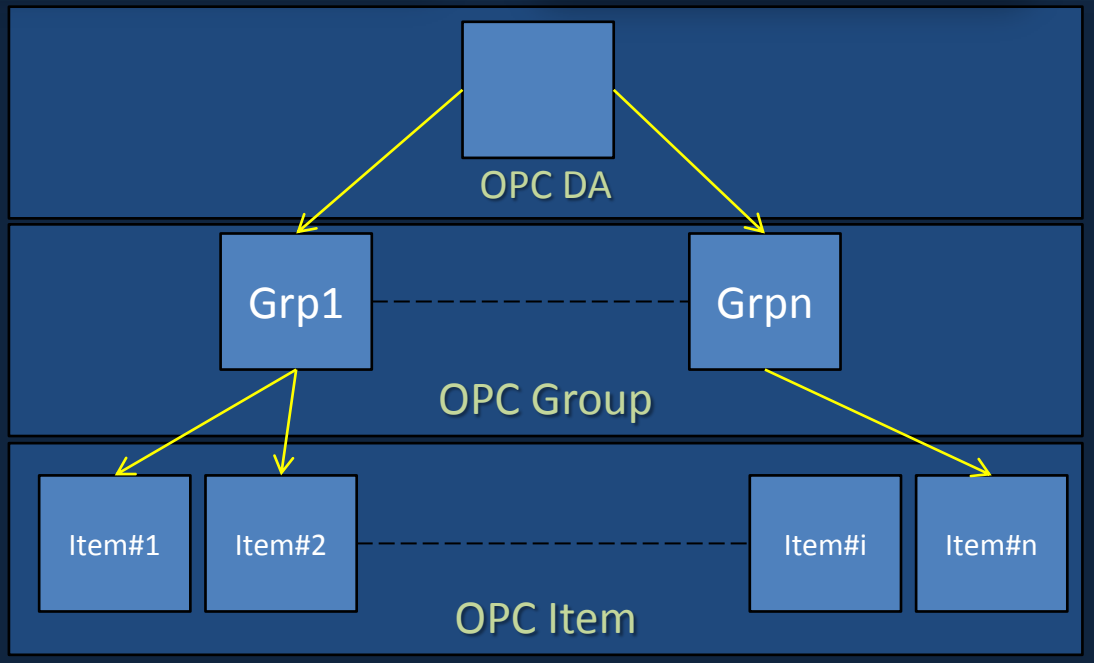
OPC A/E

Alarmes et Notification d' événements sur demande

OPC HDA

Communication des données « histoire » (matériel, applications, BD, …)

- Item OPC DA
  - Caractérisé par 3 attributs:
    - La valeur (la dernière sauvegardée)
    - La qualité de la donnée (masque permettant de caractériser la valeur)
    - Une date représentant la dernière mise a jour de la valeur et de la qualité.



OPC DA : Modèle de données (simplifie)

Quelques conclusions

- sur l'OPC « Classique » :
  - OPC classique impose une dépendance a Window qui n'est pas souhaitable.
  - Si l'on souhaite réaliser du monitoring distant, la configuration peut devenir très contraignantes.
  - Si l'on souhaite réaliser du monitoring/contrôle sur du matériel hétérogène(autres que des PLCs), l'OPC ne peut pas être utilise dans tous les cas de figures.
- Sur l'environnement de développement
  - Préserver une uniformité sur l'environnement de développement
    - CPU, Système & développement logiciel
  - Préserver une homogénéité dans les différents couches logicielles et dans les accès aux matériels

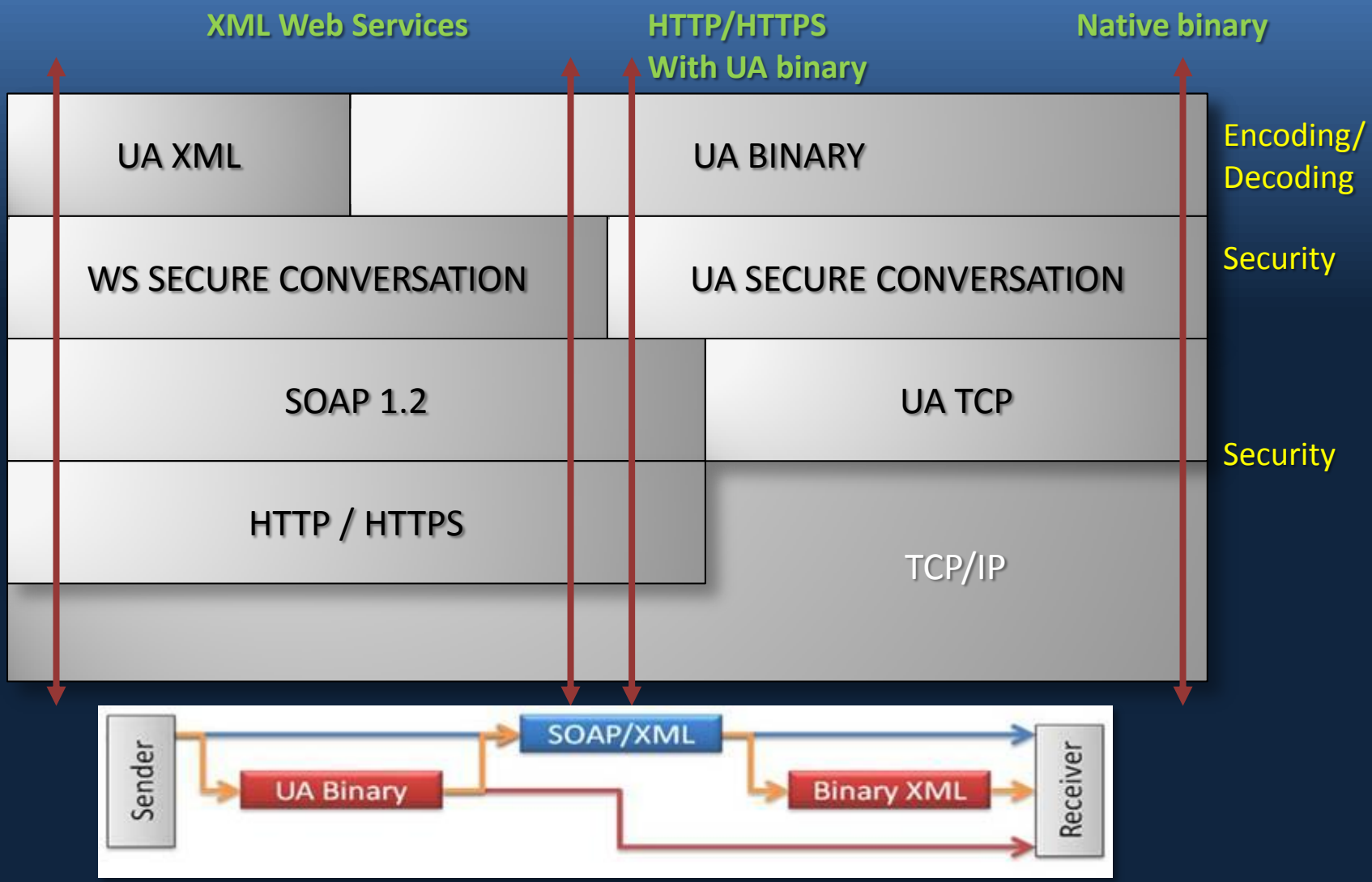
OPC UA

- Permet l' accès a tout type de matériel
- Indépendant de la plateforme
  - La communication entre les Clients OPC UA et Serveur permet l' interopérabilité entre toutes les plateformes
- L' implémentation de serveur OPCUA sur des systèmes embarques(Processeur ARM)
- Préserve l'environnement existant base sur l'OPC classique
  - Via wrappers et proxies
- **Fiabilité** : clients et serveurs peuvent facilement surveilles
- Architecture orientée Service (SOA) :
  - Fiabilité , performance, Robustesse et Sécurité

Protocoles OPC UA

- 2 protocoles sont supportes :
  - Le protocole « binaire » : opc.tcp://Server
    - Offre la meilleure performance
    - Utilise le minimum de ressources(Pas de XML,HTTP et SOAP)
    - Offre une meilleure interopérabilité
    - Utilise un seul port TCP(4840) pour la communication (facilitant le tunneling ou la gestion des pare-feux).
  - Le protocole HTTP http://Server pour les Web Service.
    - SOAP pour l' interopérabilité avec les applications type « ERP » existants.
    - Gestion des pare-feux également (Port 80, 8080, 443)

OPC/UA Messaging

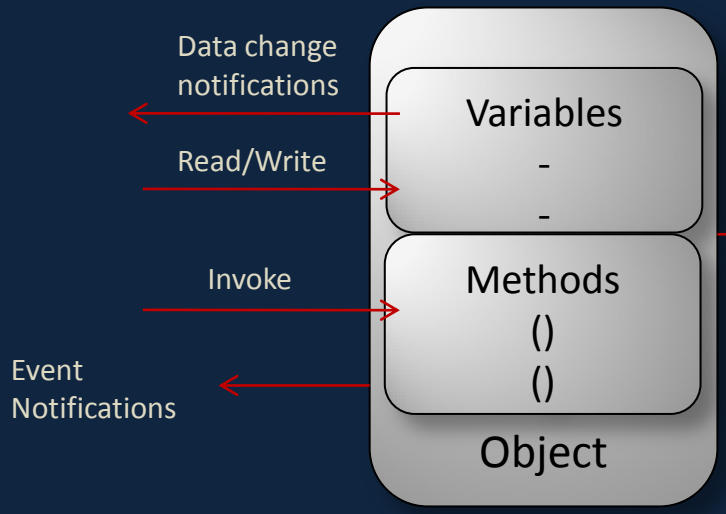


OPC UA Services

- OPCUA = Architecture orientée Service (SOA)
- Les 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*

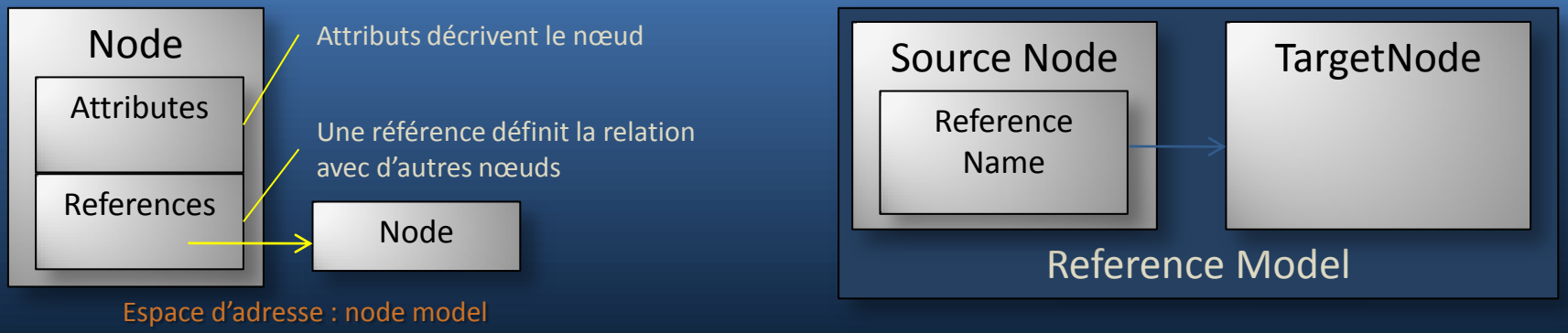
OPC/UA : Le modèle d'information

- Tout serveur OPCUA doit implémenter le modèle d'information **unifié**.
- OPCUA : Le Model "Objet"
  - OPC-UA est implémenté afin d' échanger de l'information de façon "orientée objet"
  - « OPC/UA Address Space » fournit un moyen standard pour les serveurs de présenter les objets aux clients.



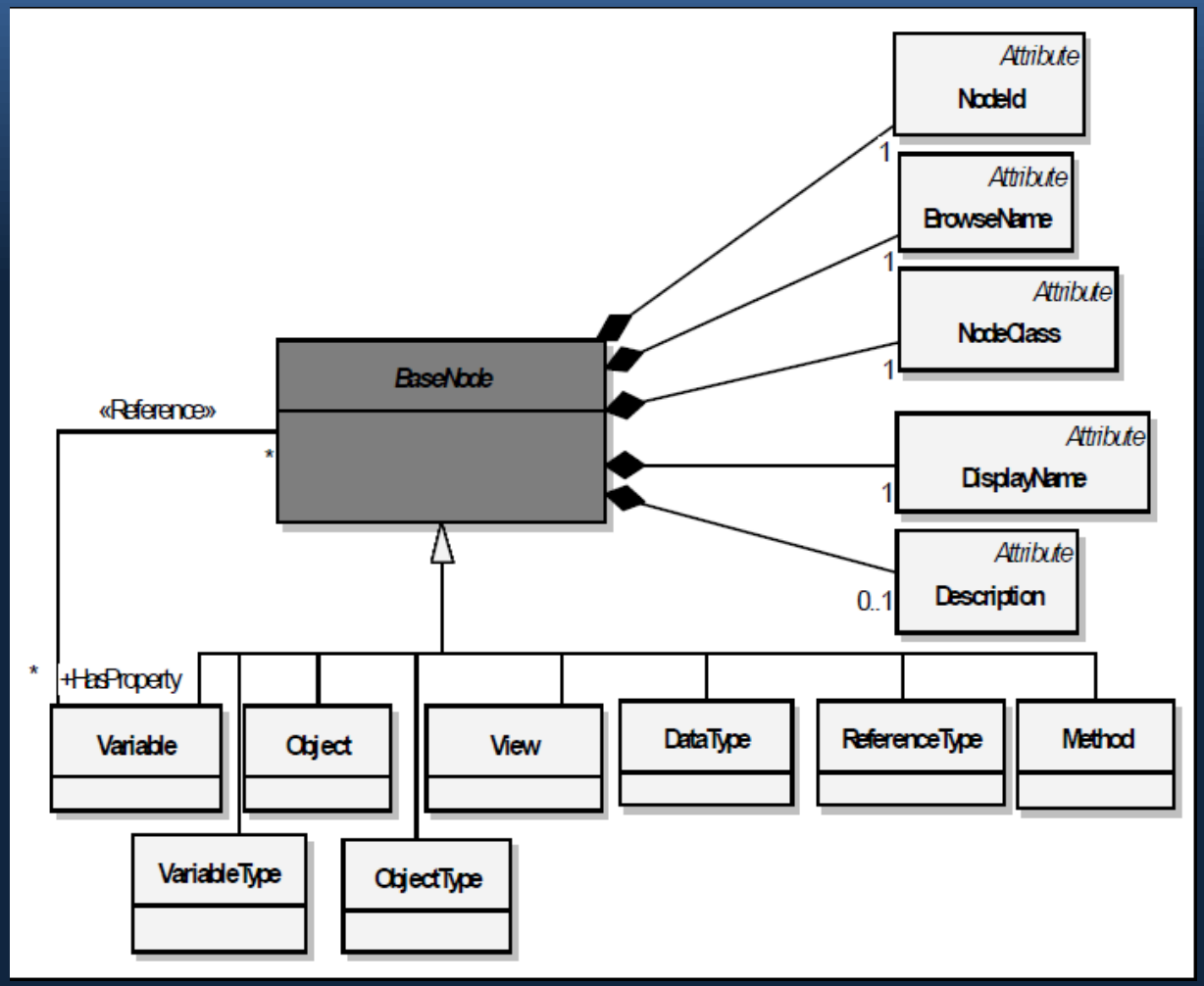
- Les éléments du modèle sont représentés dans l'espace d' adresse comme des "Noeuds"
- Chaque nœud est associe a une classe NodeClass

OPC/UA : information model

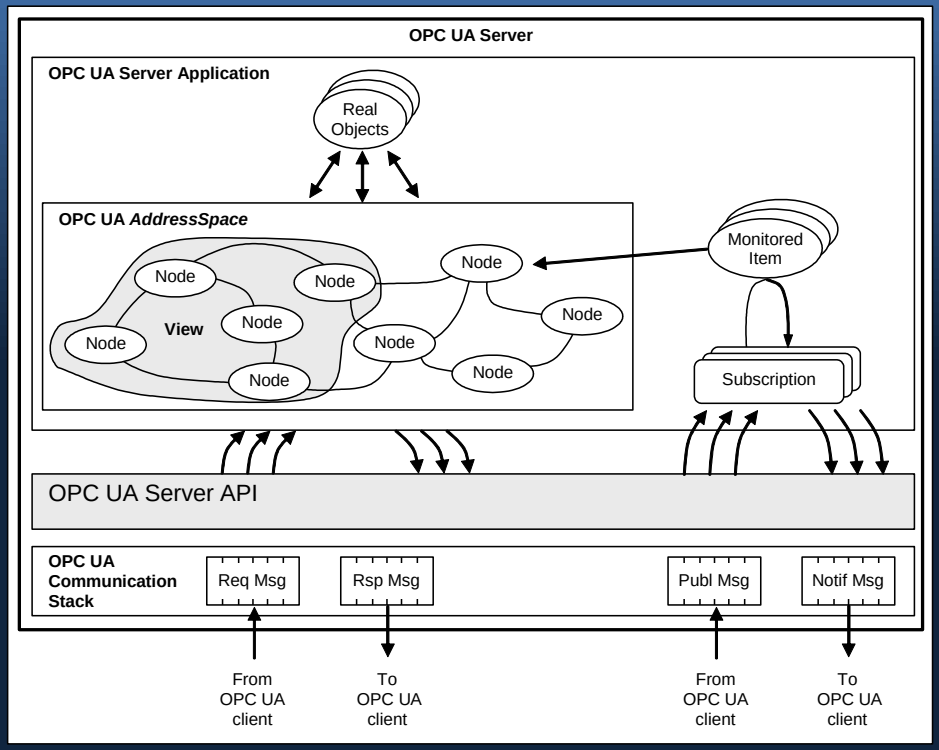


- Les attributs décrivent des Nœuds.
- Les Clients accèdent aux valeurs des attributs de nœuds en utilisant les services :
  - *Read,Write,Query* et *Subscription/MonitoredItem*
- Les références sont:
  - Utilisées pour relier les nœuds entre eux.
  - Les instances de type « **ReferenceType** » sont visibles dans l' espace d'adresses
  - Un nœud « cible » peut se situer dans le meme espace d'adresse ou dans l'espace d'adresse d'un autre serveur

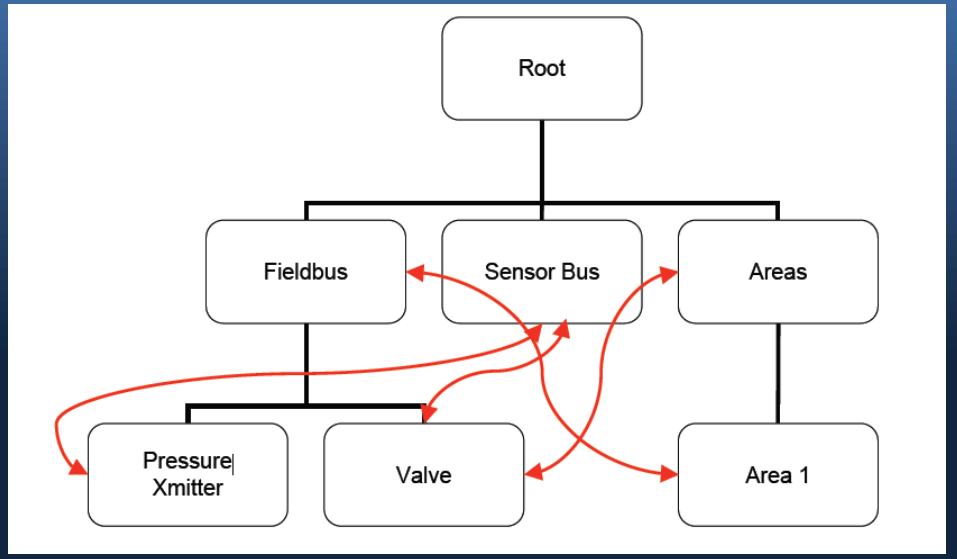
OPC UA Information MetaModel



Information Model

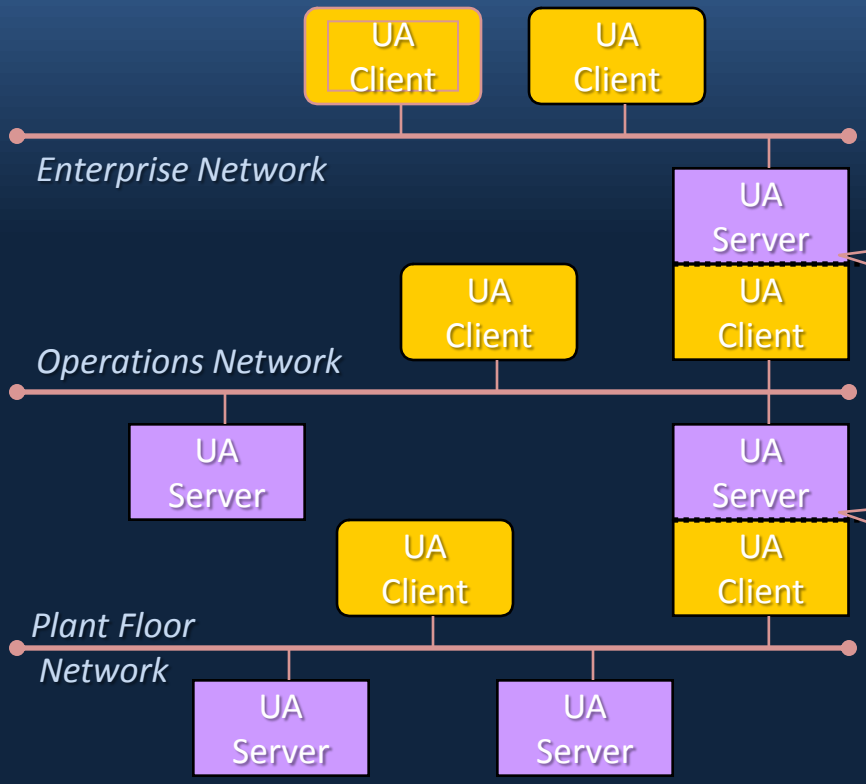


- Les objets sont “physiques” ou logiciel
- Une « Vue » est un sous-ensemble de l’espace d’adresse.
- Une « vue » est utilisée pour réduire la visibilité des nœuds aux différents type de clients.



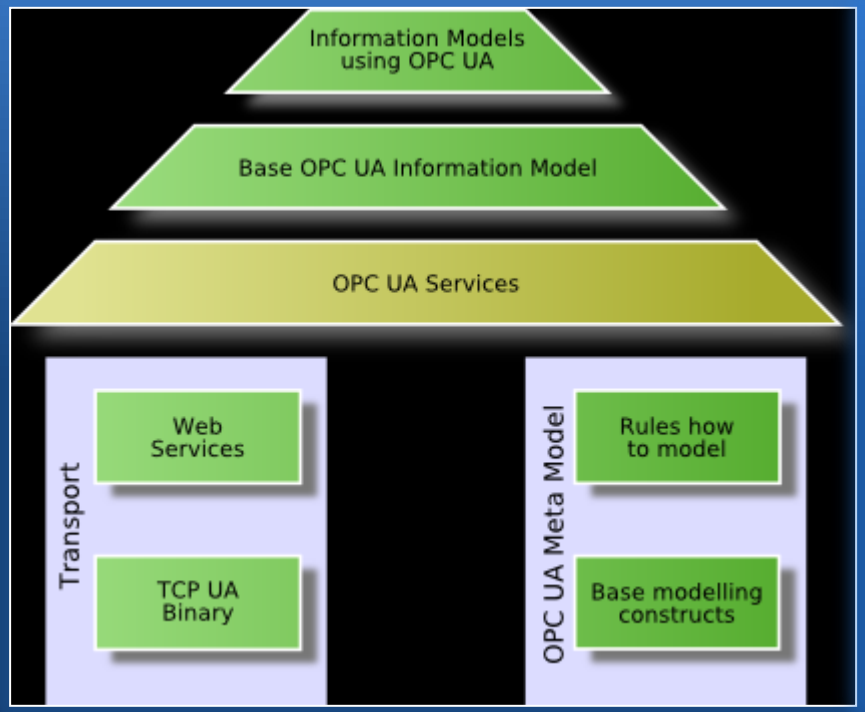
- Une modèle « réseau »
- Le nombre de relation entre objets est illimitée
- Les « vues » sont utilisées pour présenter des hiérarchies
- Les ***Références*** entre les nœuds permettent aux serveurs d’ organiser l’espace d’adresse en hiérarchies (réseau maillée de nœuds)

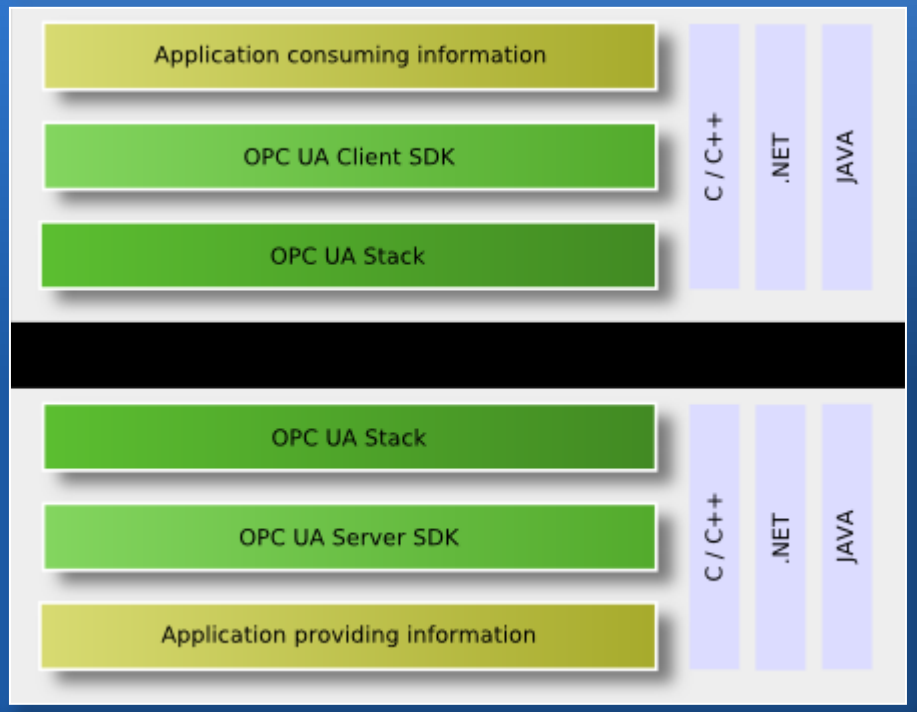
Serveur OPCUA : Le Chainage

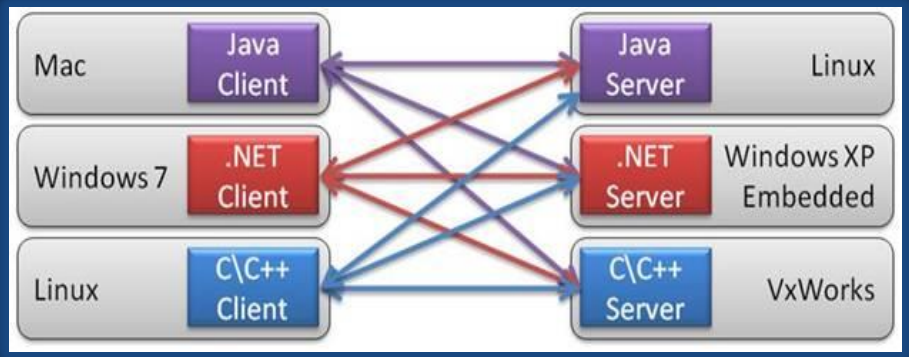


- L' agrégation de serveurs OPCUA permet l'extraction et le traitement de données depuis les couches basses(matérielle) vers les données « Entreprises »
- Les données sont « typées » en utilisant un modèle d'information différent approprié pour les clients du niveau supérieur

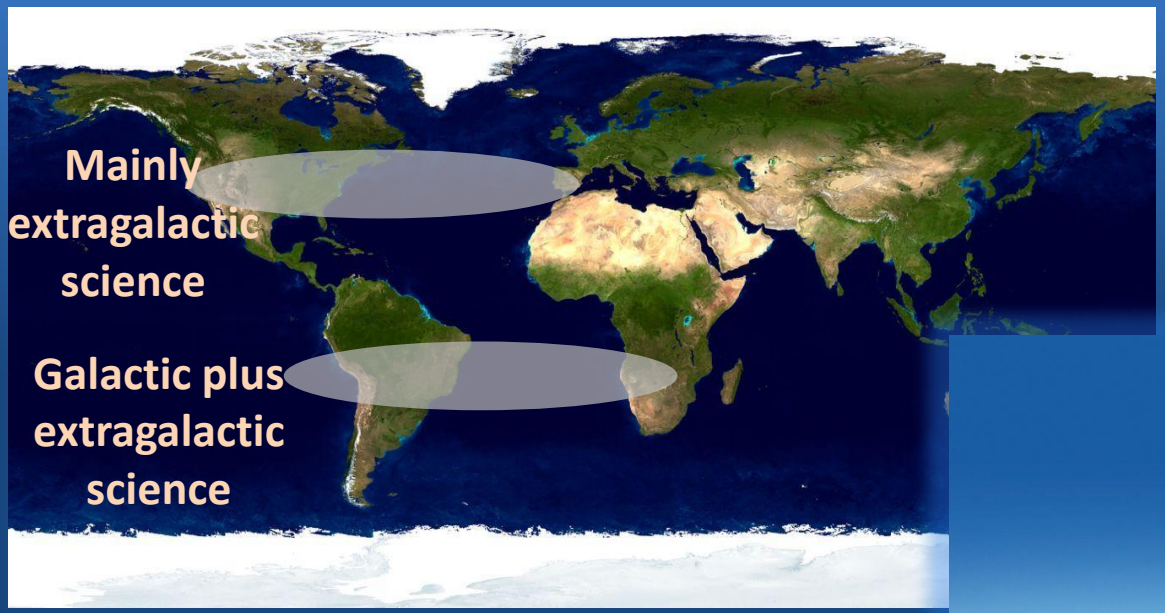
OPCUA

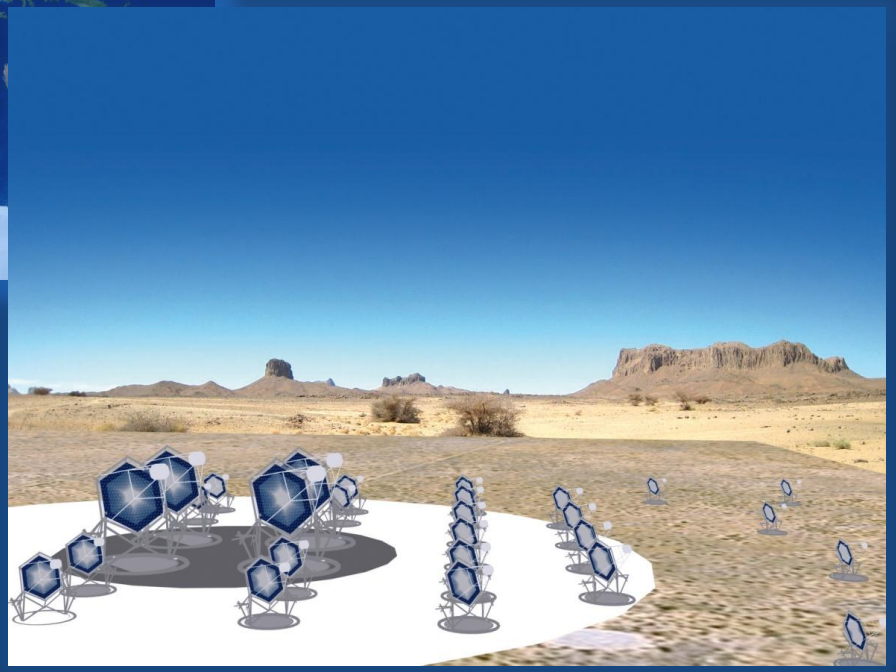

Information Models using OPC UA
Base OPC UA Information Model
OPC UA Services
Transport
Web Services
TCP UA Binary
OPC UA Meta Model
Rules how to model
Base modelling constructs


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

CTA :Cherenkov Telescope Array





Projet européen de grand réseau de télescopes Cherenkov de nouvelle génération en astronomie gamma de très hautes énergies.

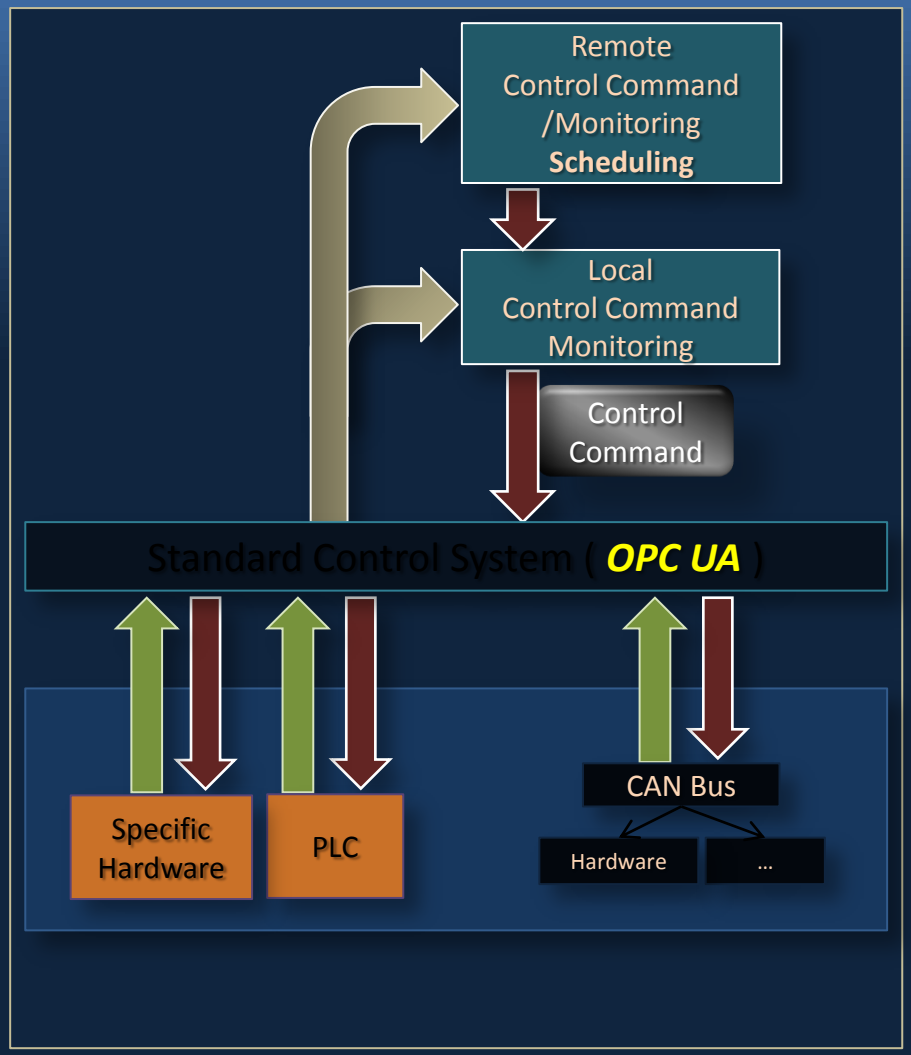
3 tailles sont prévues:

- ***6m, 12m et 23m de diamètre***

CTA : Les projets techniques au LAPP

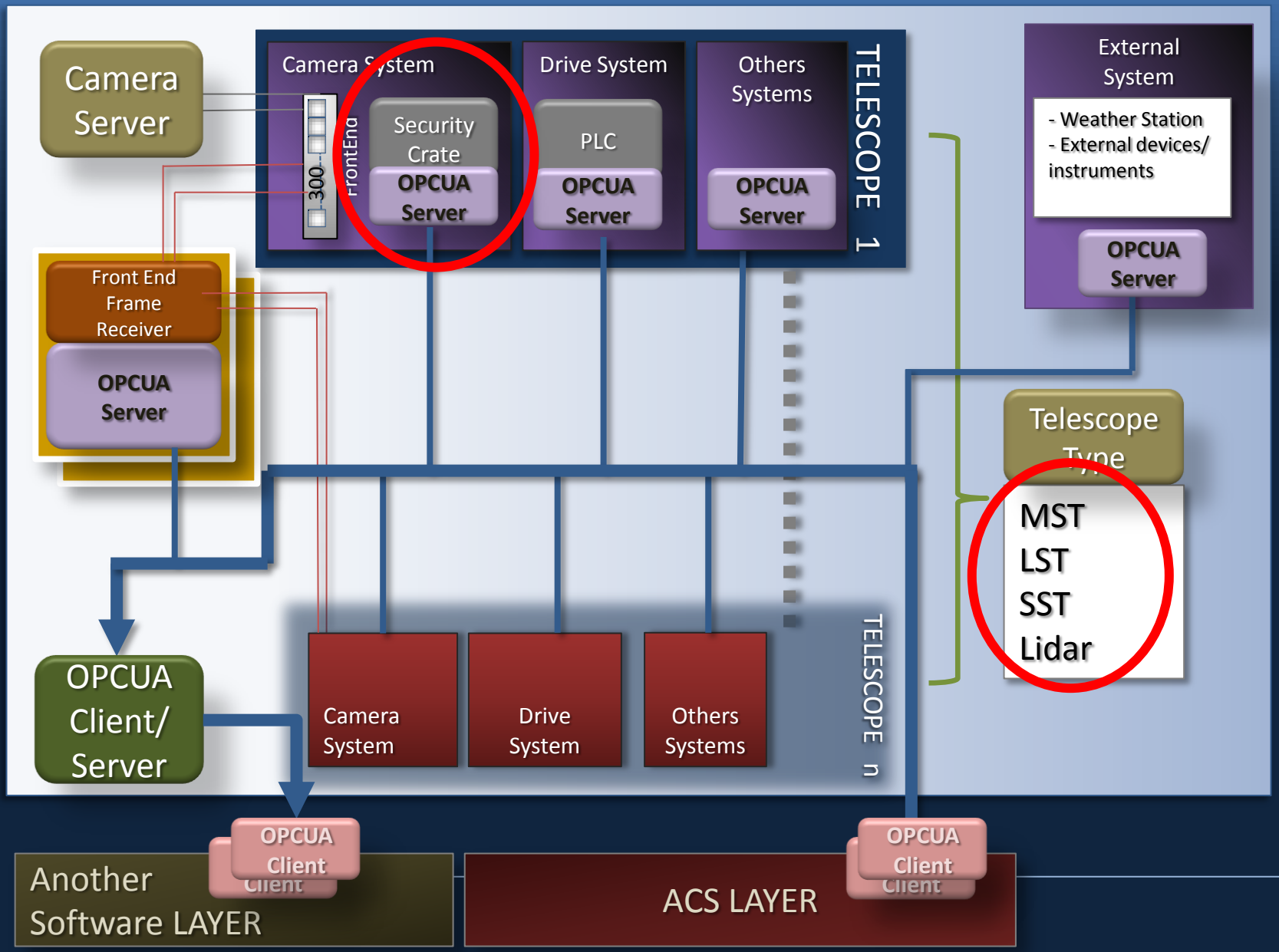
- Mécanique :
  - Conception d'une structure extrêmement légère et élancée pour le design du télescope de plus grande taille (diamètre du miroir de 23m, et focale de 28m).
- Automatisme :
  - Automatisation du contrôle de positionnement du télescope de grande taille : motorisation, « Drive System », Supervision,…
- Electronique
  - Développement du système de sécurité de la camera MST(Middle Size Télescope)
- Informatique
  - Data Management : Modèle de calcul distribué, Archive, systèmes de DB et MetaDB, « Cloud », portail scientifique
  - MST , LST(?) « Monitoring & Slow Control » , LST Supervision, …

CTA Slow Control



- Surveiller la situation d'un réseau, d'un télescope, d'un élément de télescope, …
- Récupérer les informations issues du matériel et les remonter dans les couches logicielles (DAQ)
- Fournir les informations liées au matériel localement et a distance.
- L' intégration du matériel et son accès d'un point vue logiciel doit être homogène et indépendant du type de matériel ➔ **OPCUA**

Slow Control : Vue logique



Slow Control : La mise en œuvre

- Plusieurs prototypes OPCUA étudiés et/ou mis en œuvre :
  - Carte « FrontEnd » (NECTAR Cam)
  - Banc de tests de la carte « FrontEnd »
  - Châssis sécurité de HESS2
  - Contrôle des actionneurs de miroirs.

Nos constats

- Pour un « non » expert :
  - L'écriture d'un serveur OPCUA peut être fastidieux.
  - Le « développeur » n'est pas nécessairement l'intégrateur du matériel
- Il faut rendre l' intégration de matériel plus simple :
  - En cachant le plus possible la "machinerie" OPCUA
  - En limitant le code a écrire
  - En fournissant des moyens de décrire le matériel a intégrer
    - Les points de contrôle(set, get, …)
    - Le type de connexion du matériel(USB , RS232, TCPIP, …)
    - Le type de transport (UDP, TCP, …)
    - Le type de transfert (PUSH, PULL)
    - Les conditions de démarrage et d' arrêt du matériel :
      - Aspects de sécurités.
    - Les dépendances et hiérarchies entre différents matériels a intégrer
- La description doit être partagée par le serveur et les/les clients

OPCUA : La Description du matériel

- Basée sur XML et XSD
  - Le fichier « xsd » servant de dictionnaire et permettant la validation des fichiers XML.

- Les fichiers XML sont utilisés comme « entrée » du serveur OPCUA pour:
  - Décrire la connexion au matériel
  - Alimenter les « points de données » dans le serveur OPCUA
  - Décrire le flot de données issu du matériel
    - Afin d'analyser le contenu du flot
    - Afin de localiser dans le flot le nom et les valeurs des points de données.

Le dictionnaire XSD : Quelques exemples

```
- <xs:element name="Object">
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="Multiplicity"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="Datapoint"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="Method"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="Object"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
- <xs:element name="Device">
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="Type"/>
        <xs:element ref="Name"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="Attribut"/>
        <xs:element ref="Serial"/>
        <xs:element ref="Interface"/>
        <xs:element minOccurs="0" ref="TCP_UDP_Config"/>
        <xs:element minOccurs="0" ref="Address"/>
        <xs:element minOccurs="0" ref="Talk_prototype"/>
        <xs:element minOccurs="0" ref="Instruction_set"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="Object"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
- <xs:element name="TCP_UDP_Config">
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="Port"/>
        <xs:element ref="Address"/>
        <xs:element ref="TimeOut_sec"/>
        <xs:element ref="TimeOut_usec"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <xs:element name="Interface">
  - <xs:simpleType>
    - <xs:restriction base="xs:string">
        <xs:enumeration value="serial"/>
        <xs:enumeration value="gpib"/>
        <xs:enumeration value="udp"/>
        <xs:enumeration value="tcp"/>
        <xs:enumeration value="usb"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
```

Quelques exemples: XML files

- description matérielle: Nectar.xml

```
– <Device>
    <Type>ServeurOCPUA slowControlNectar</Type>
    <Name>Device_Nectar</Name>
  – <Attribut>
      <Name>Author</Name>
      <Value>Panazol jean luc</Value>
    </Attribut>
  – <Attribut>
      <Name>Serial</Name>
      <Value>0000000</Value>
    </Attribut>
    <Serial>000000</Serial>
    <Interface>udp</Interface>
  – <TCP_UDP_Config>
      <Port>50800</Port>
      <Address>134.158.96.192</A
      <TimeOut_sec>0</TimeOut_
      <TimeOut_usec>5000</Time
    </TCP_UDP_Config>
    <Address>2.0.0.0</Address>
  </Device>
```

Extrait de la description du matériel

```
– <Compound>
    <Name>FrontEnd</Name>
  – <Attribut>
      <Name>Version</Name>
      <Value>version1.1</Value>
    </Attribut>
    <Multiplicity>1</Multiplicity
  – <Object>
      <Name>CntrlSlc</Name>
      <Multiplicity>1</Multiplic
    – <Datapoint>
        <Name>Trame</Name>
        <Type>OpcUaId_Analog
        <Multiplicity>1</Multip
      </Datapoint>
    – <Datapoint>
        <Name>VMC</Name>
        <Type>OpcUaId_AnalogItemType</Type>
        <Multiplicity>1</Multiplicity>
      </Datapoint>
```

Description d'un objet « composé »

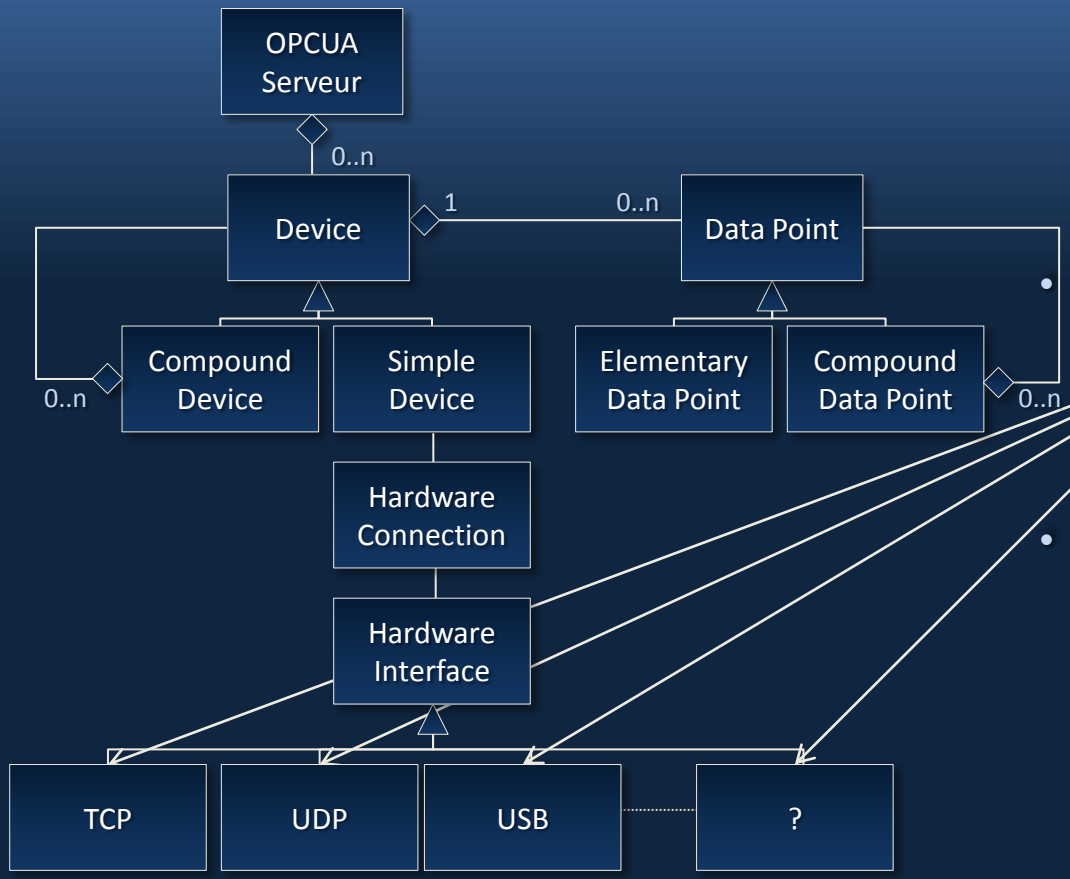
```
    – <Method>
        <Name>fonction1</Name>
      </Method>
    – <Method>
        <Name>call</Name>
      – <Argument>
          <Name>test</Name>
          <Type>integer</Type>
        </Argument>
      – <Argument>
          <Name>test2</Name>
          <Type>integer</Type>
        </Argument>
      </Method>
  </Object>
– <Object>
    <Name>CntrlNECTARReg</Name>
    <Multiplicity>1</Multiplicity>
  – <Datapoint>
      <Name>Trame</Name>
      <Type>OpcUaId_AnalogItemType<
      <Multiplicity>1</Multiplicity>
```

description d'une méthode d'accès

```
<Frame_definition>
  <Name>CNTRLDAQ</Name>
– <Frame_element>
    <Pos>0</Pos>
    <Nb>1</Nb>
    <Name>DAQOnOffb</Name>
  </Frame_element>
– <Frame_element>
    <Pos>1</Pos>
    <Nb>1</Nb>
    <Name>Nf</Name>
  </Frame_element>
– <Frame_element>
    <Pos>2</Pos>
    <Nb>1</Nb>
    <Name>ParRun</Name>
  </Frame_element>
– <Frame_element>
    <Pos>3</Pos>
    <Nb>1</Nb>
    <Name>Q_Sampls</Name>
  </Frame_element>
```

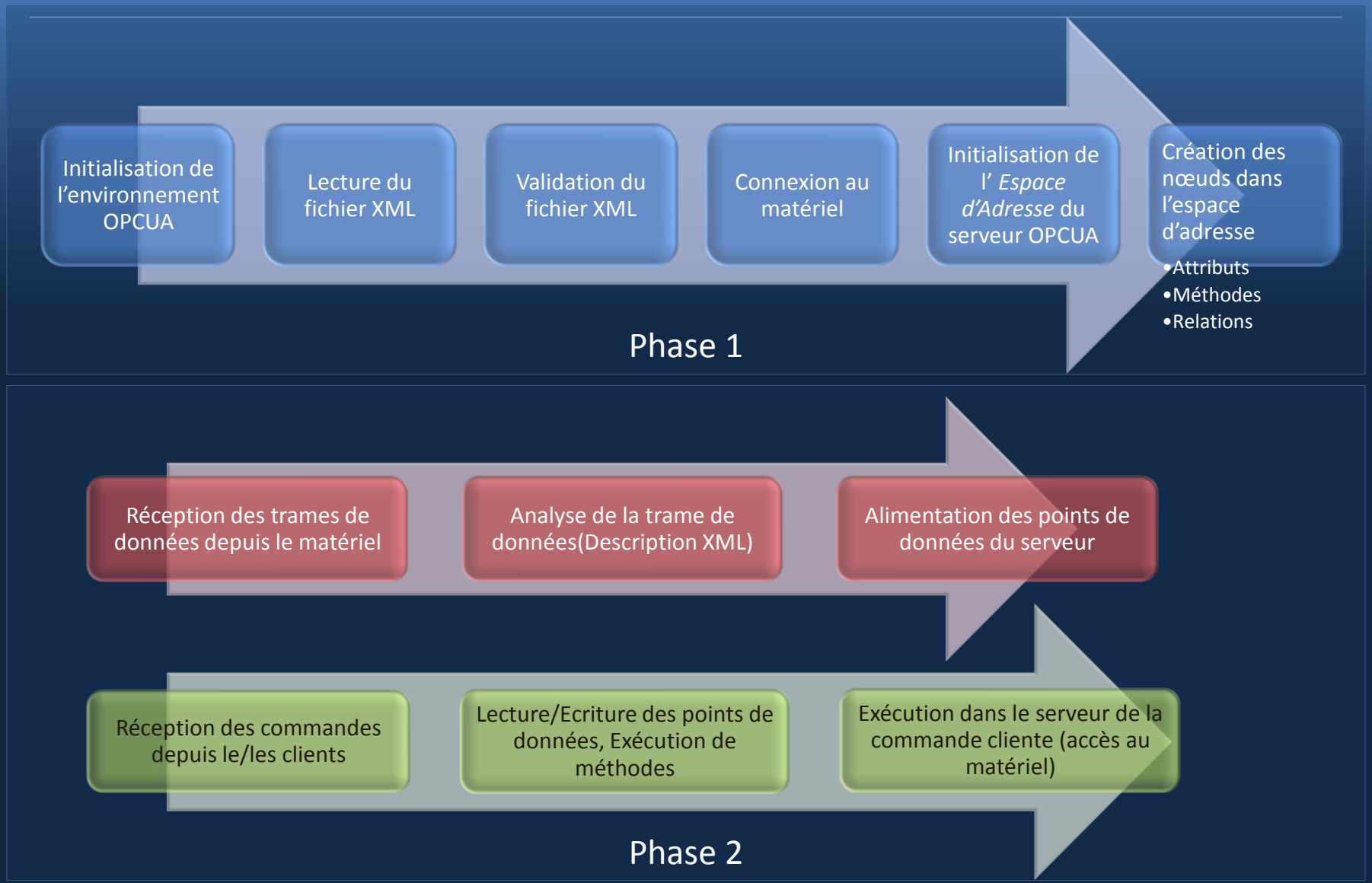
Extrait de la description de la trame UDP

OPCUA Server : Vers un Modèle d' implémentation



- La partie "abstraite"(Hardware Interface) sera constituée d'un jeu de méthodes « standards »
  - connect(…), read(…), write(…), close(…)
  - Le flux d' entrée/sortie sera également décrit pour permettre son analyse.
- L' intégration d'un matériel sera réalisée par héritage de cette partie abstraite.
  - Les méthodes connect(…), read(…), write(…), close(…),… seront implémentées a ce niveau
- La partie "CTA OPCUA Layer" s'adressera toujours aux même classes d'objets

Le serveur OPCUA : les grandes phases



OPCUA Server : modele UML

- Définition d'un modèle UML.

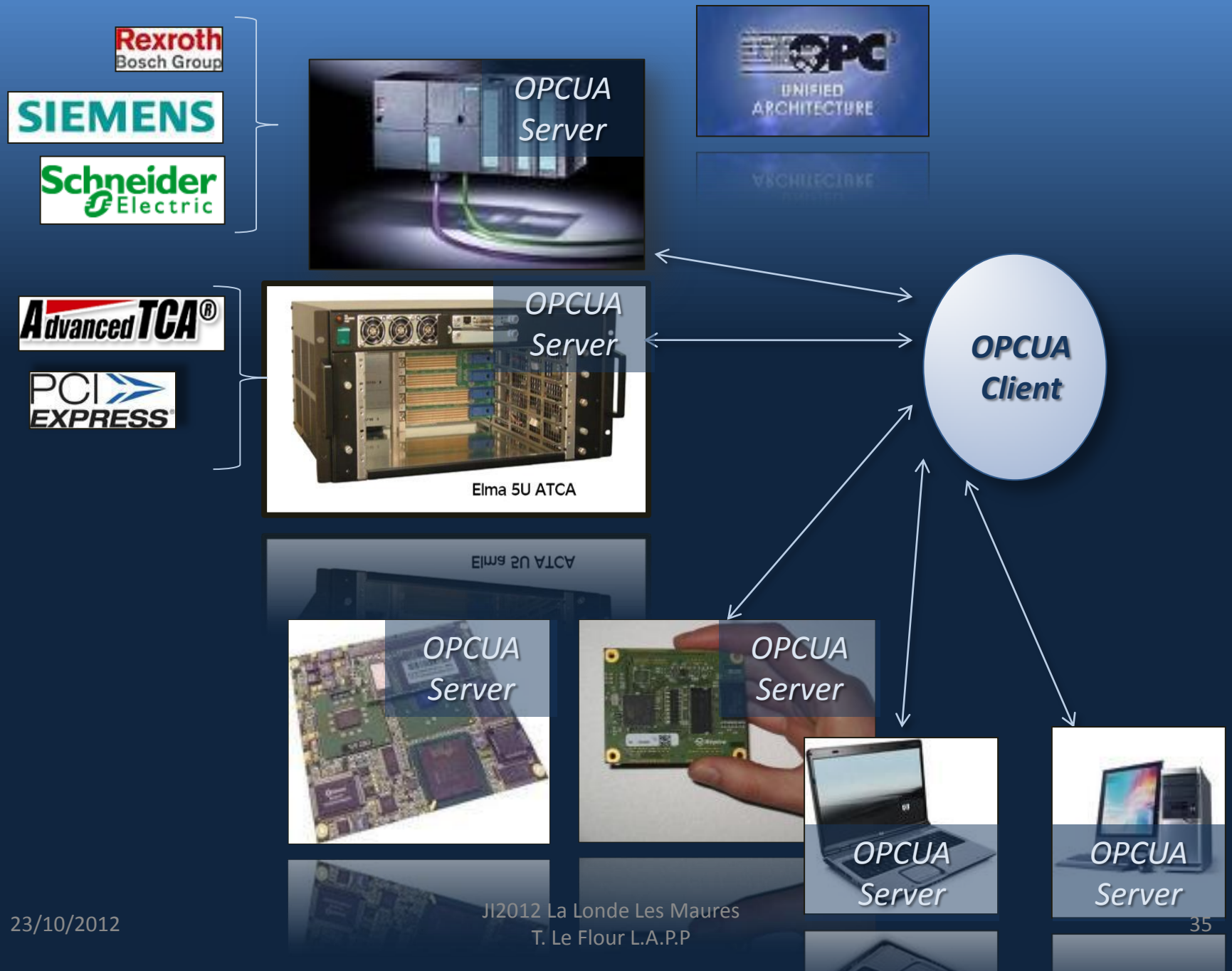
- A partir d'un tel modèle :
  - Fournir un serveur OPCUA Server cachant la partie fastidieuse de l' implémentation
  - Basé sur une description matérielle (XML/XSD, …), il sera possible de construire la complète représentation d'un matériel dans un serveur OPCUA :
    - Construction de l'espace d'adresse (nœuds, relations, …).
  - Il sera possible d' écrire l' implémentation multi-langage nécessaire dans CTA (Java , C++, …) de manière cohérente en respectant un modèle d' implémentation cohérent :
    - description de classes : contenu et relations

Conclusions

- OPCUA : Evolution naturelle de la connexion de matériel aux systèmes de supervision ou autres applicatifs.

- L'approche « générique » doit pouvoir s'appliquer a des environnements variés et donc indépendante d'un contexte

- Limite(s) actuelle(s) de OPCUA :
  - Pas « Open Source » actuellement :
    - Cependant, OPCDay a informe de l' existence d'un groupe de travail sur un OpenOPCUA.
  - licences and prix
  - Les documentations et/ou les sources sont disponibles aux membres de l'OPC Foundation :
    - Plusieurs types de membre avec des cotisations différentes pour les droits d' accès.
  - Gestion de licence centralisée IN2P3 (comme Labview, Cadence, …)


Rexroth
Bosch Group
SIEMENS
Schneider
Electric
AdvancedTCA®
PCI
EXPRESS
OPCUA
Server
OPC
UNIFIED
ARCHITECTURE
OPCUA
Server
Elma 5U ATCA
OPCUA
Client
OPCUA
Server
OPCUA
Server
OPCUA
Server
OPCUA
Server