Open Platform Communications Unified Architecture
OPCUA

(released in 2008)

Majid Aminian

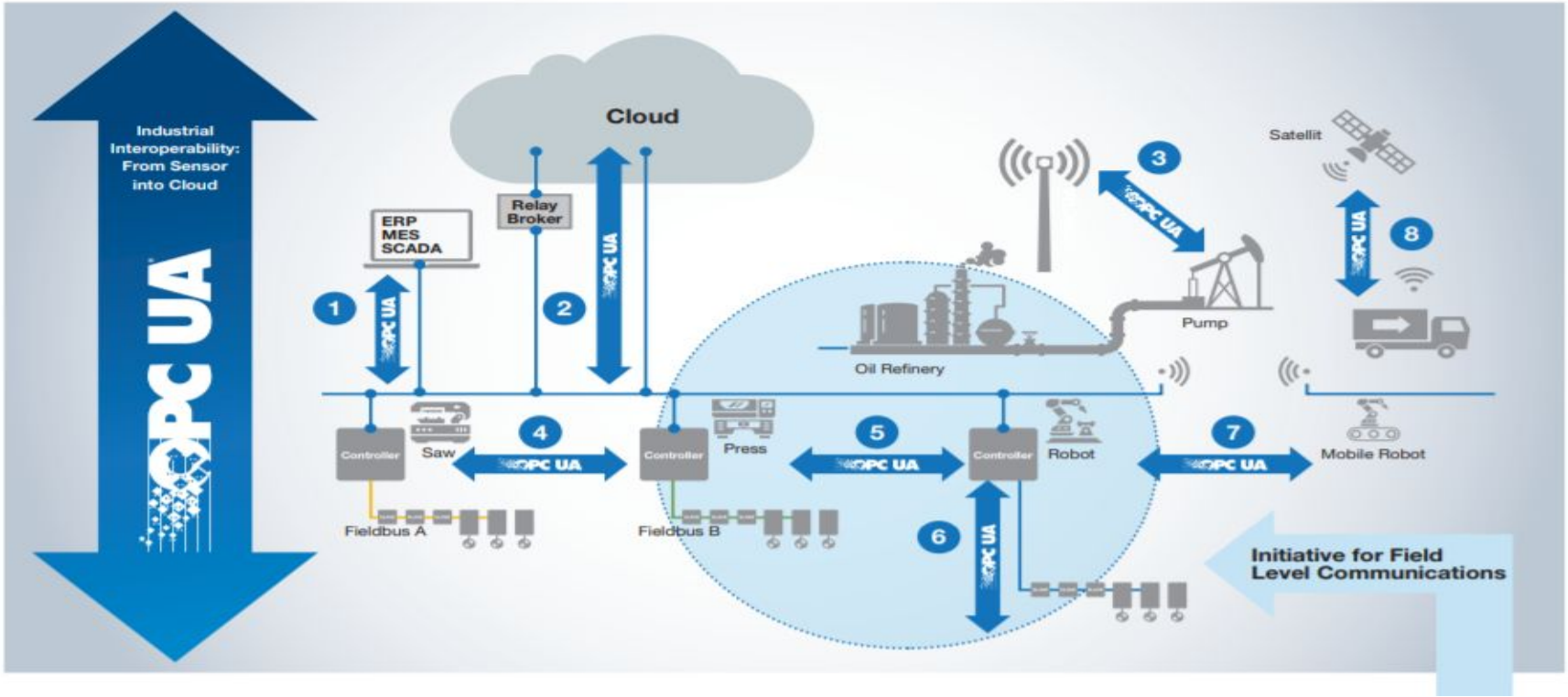
Physikalisch-Technische Bundesanstalt
Braunschweig und Berlin

05/08/2019

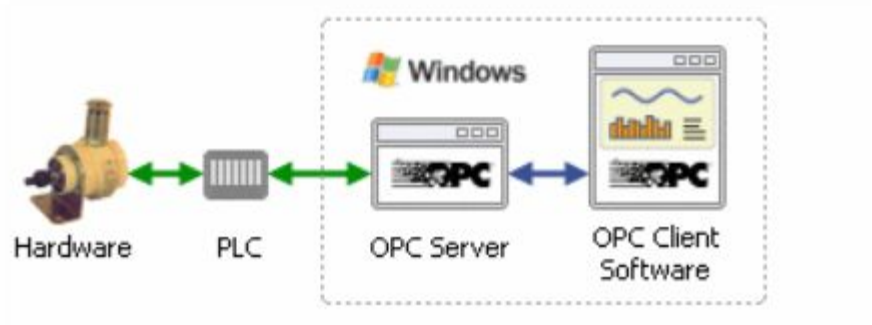
OPC UA information modeling framework turns data into information and transport of machine-to-machine data and interfaces and the semantics of data

OPC UA standardizes industrial connectivity and ensures interoperability between products from different manufacturers

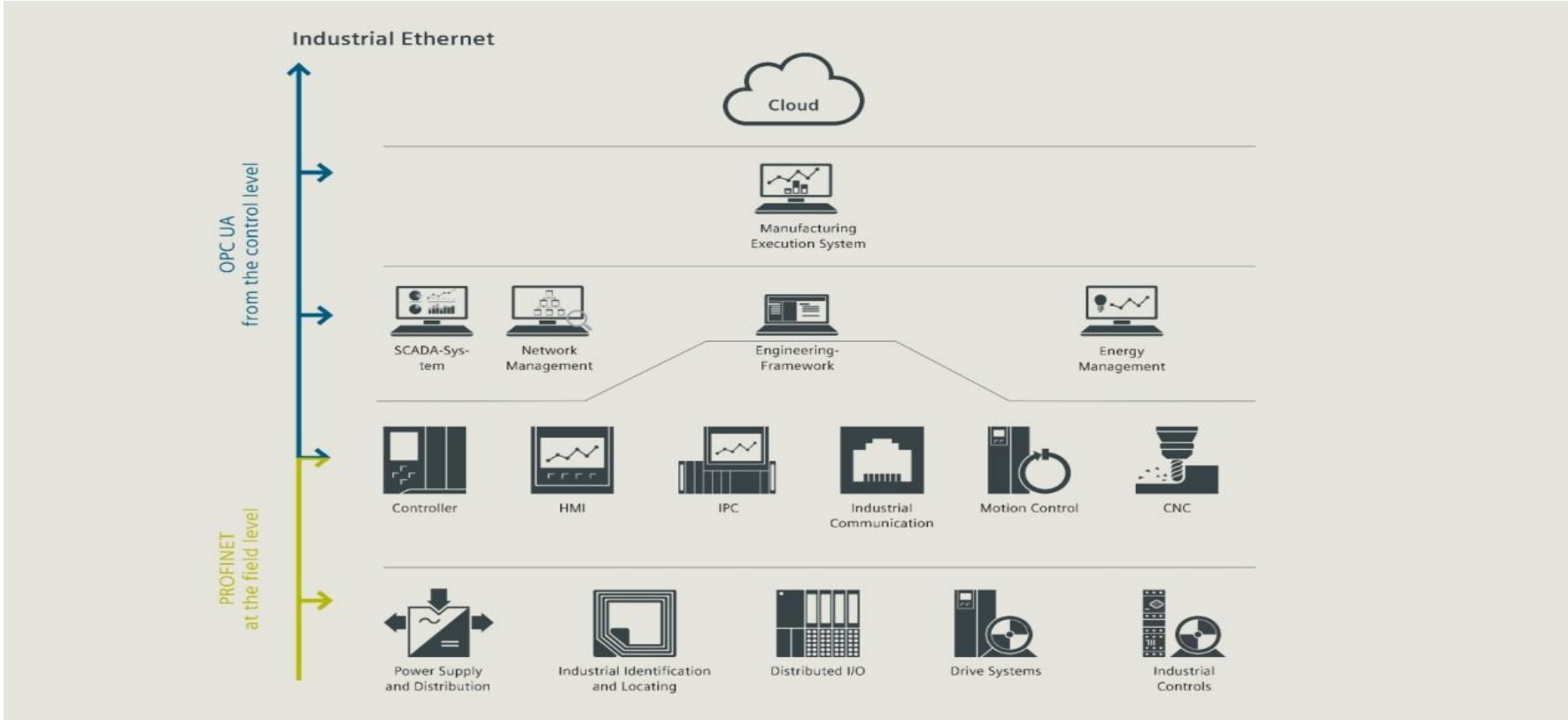
The Industrial Interoperability Standard OPC UA from Sensor to Cloud



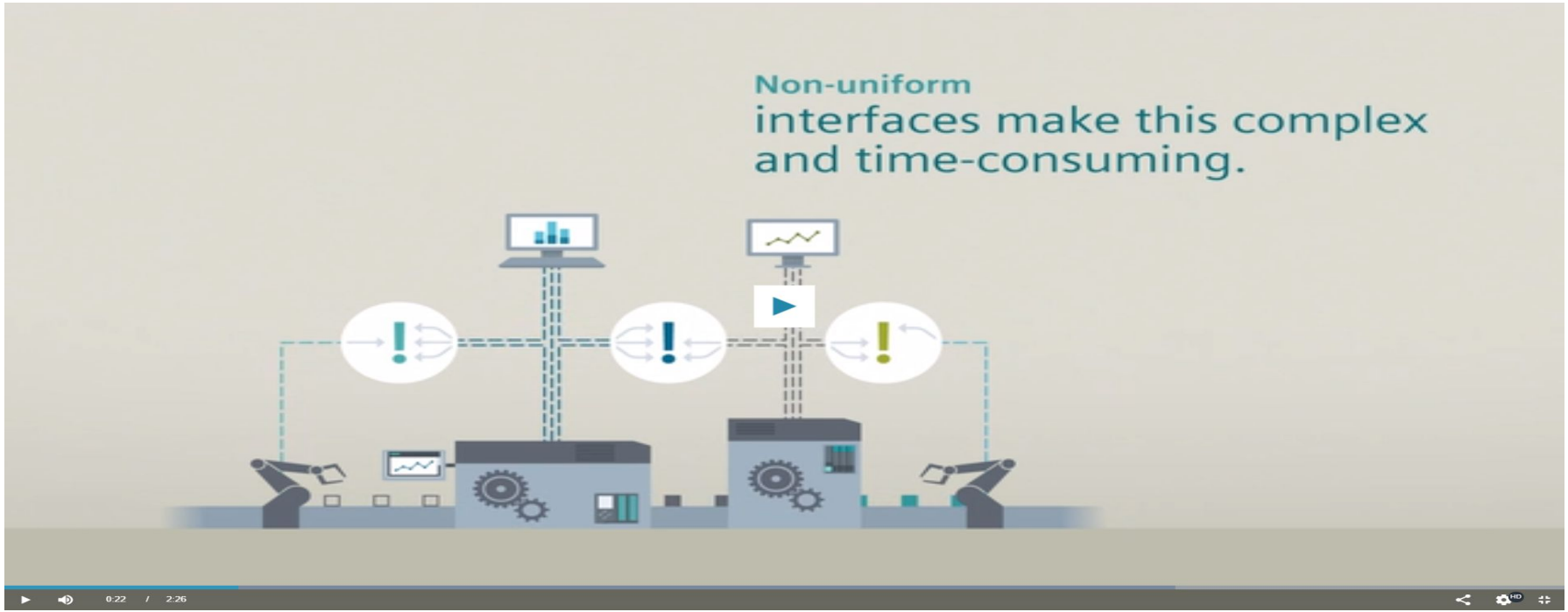
OPCUA Process



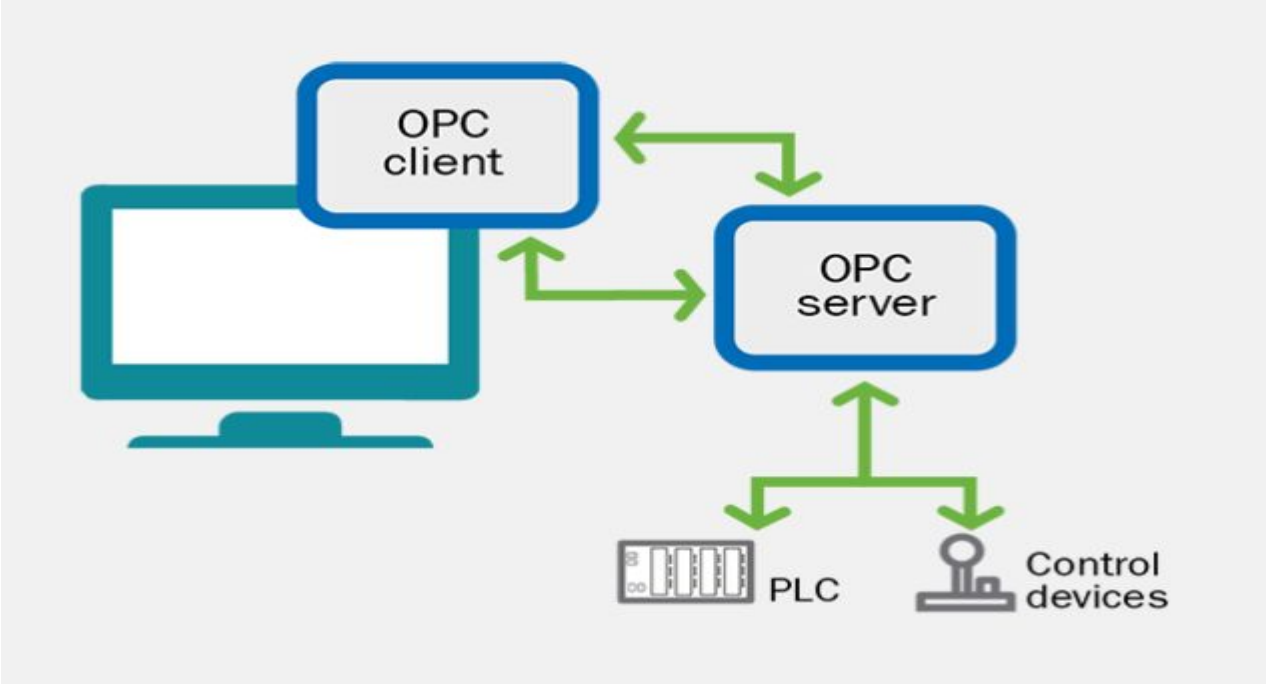
OPC UA provides a very flexible and adaptable mechanism for moving data between enterprise-type systems and the kinds of controls, monitoring devices and sensors that interact with real world data



A Robust Set of Services –OPC UA provides a full suite of services for eventing, alarming, reading, writing, discovery and more.

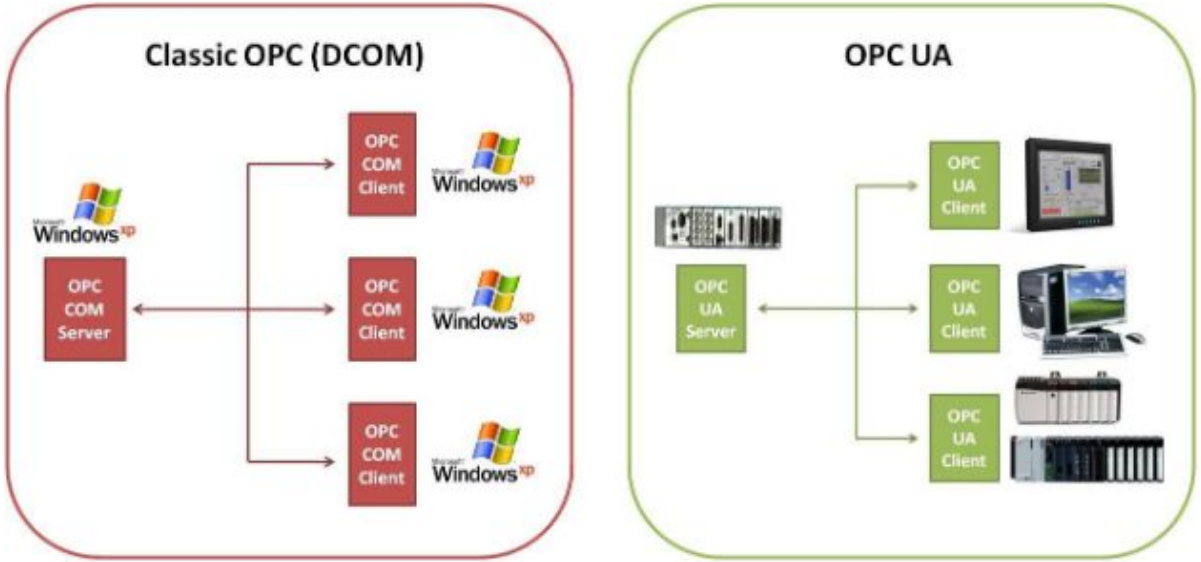


Internet Capability –OPC UA is fully capable of moving data over the Internet



available hardware platforms and operating systems:

- **Hardware platforms: traditional PC hardware, cloud-based servers, PLCs, micro-controllers (ARM etc.)**
- **Operating Systems: Microsoft Windows, Apple OSX, Android, or any distribution of Linux, etc.**



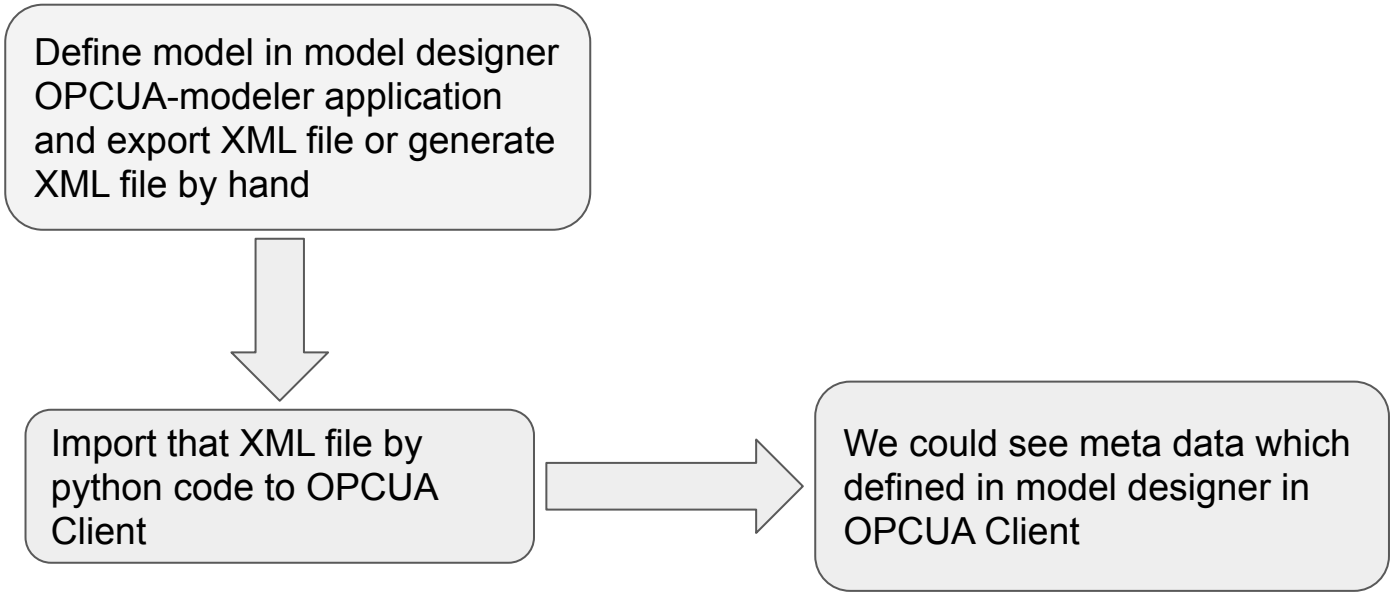
OPCUA client application:

In this application we can manage data that was sent by our codes(python codes) and watch streaming data .

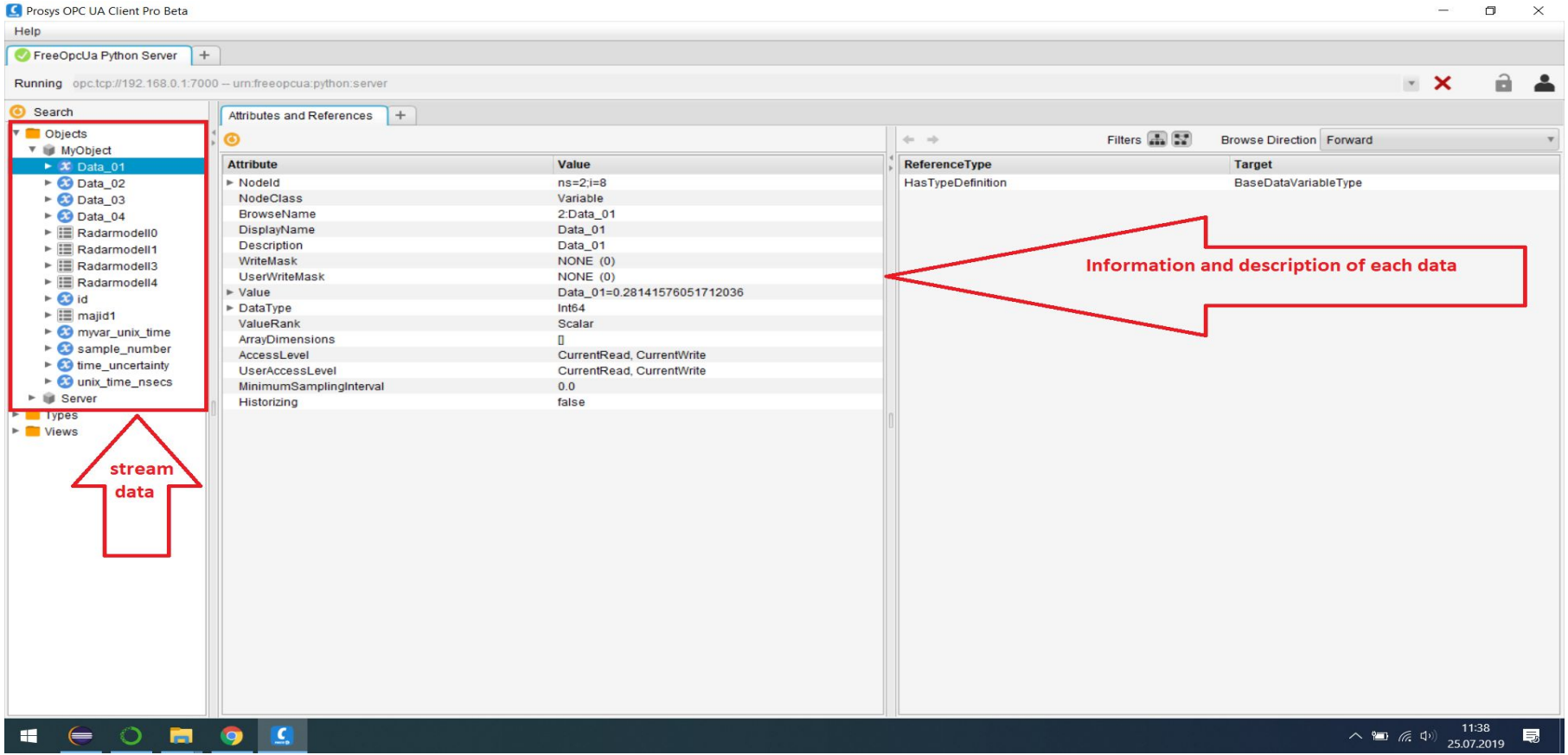
Note: we could use different kind of free applications for OPCUA client and server.

In my project i used ‘Prosys OPC UA Client’ .

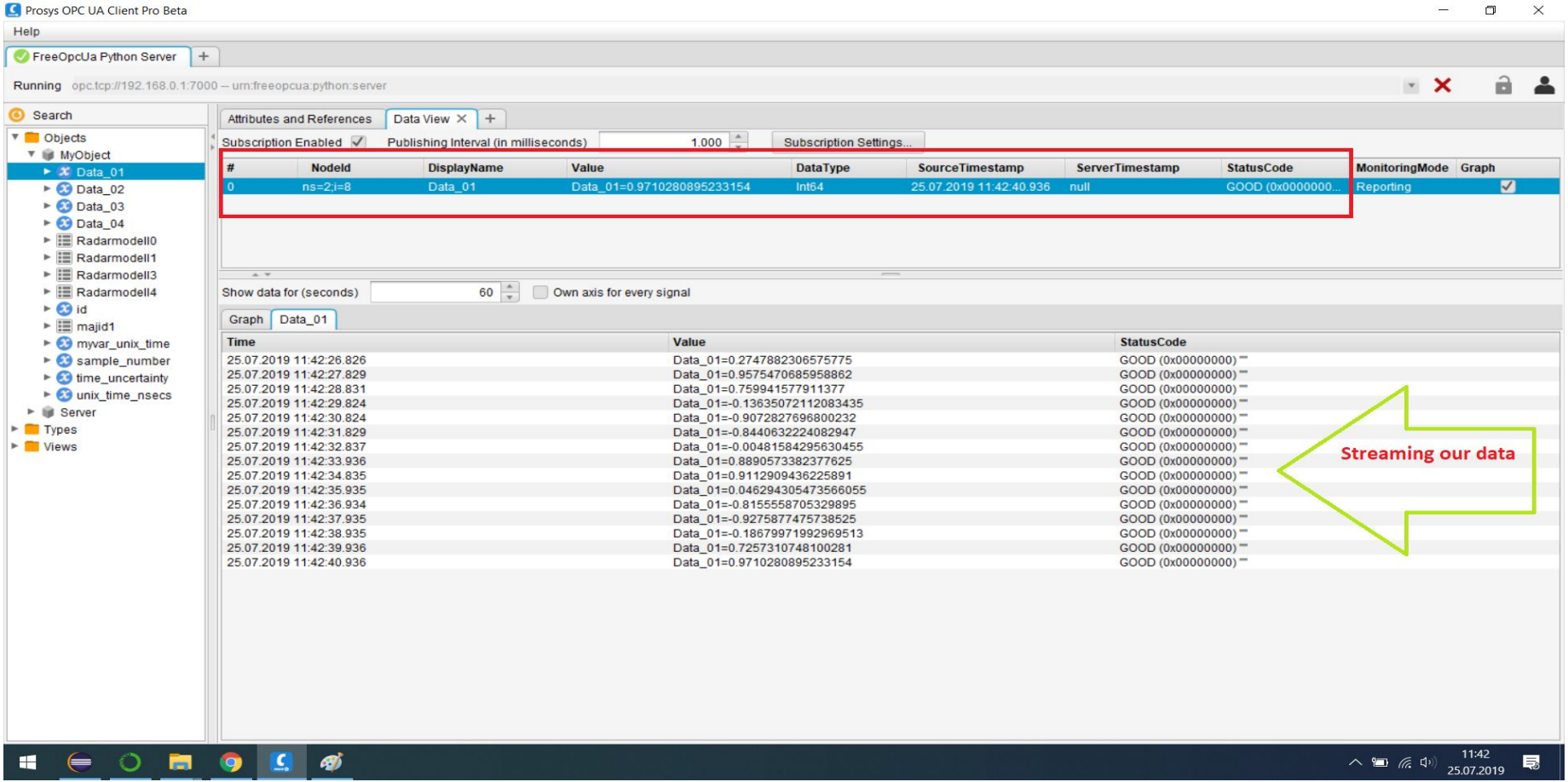
Flowchart of Model generation:



OPCUA Client Environment:

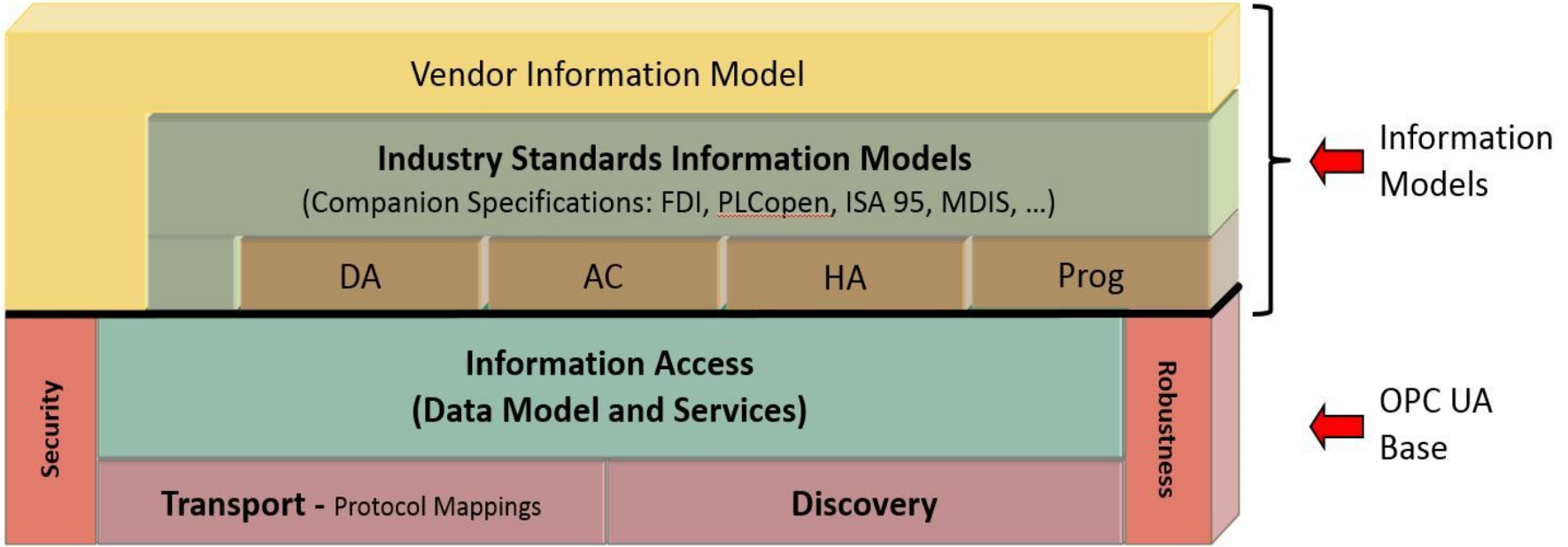


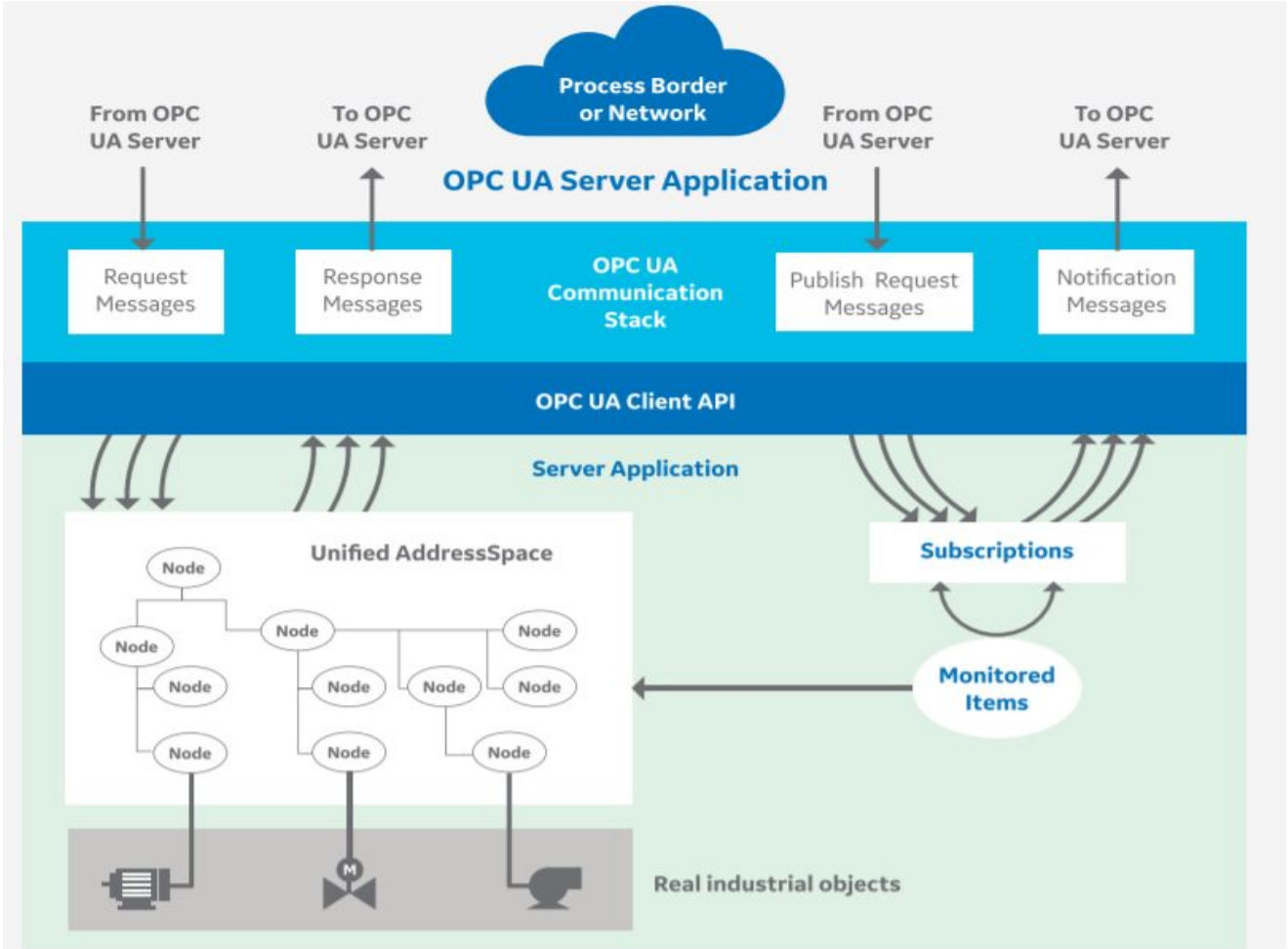
OPCUA Client Environment:

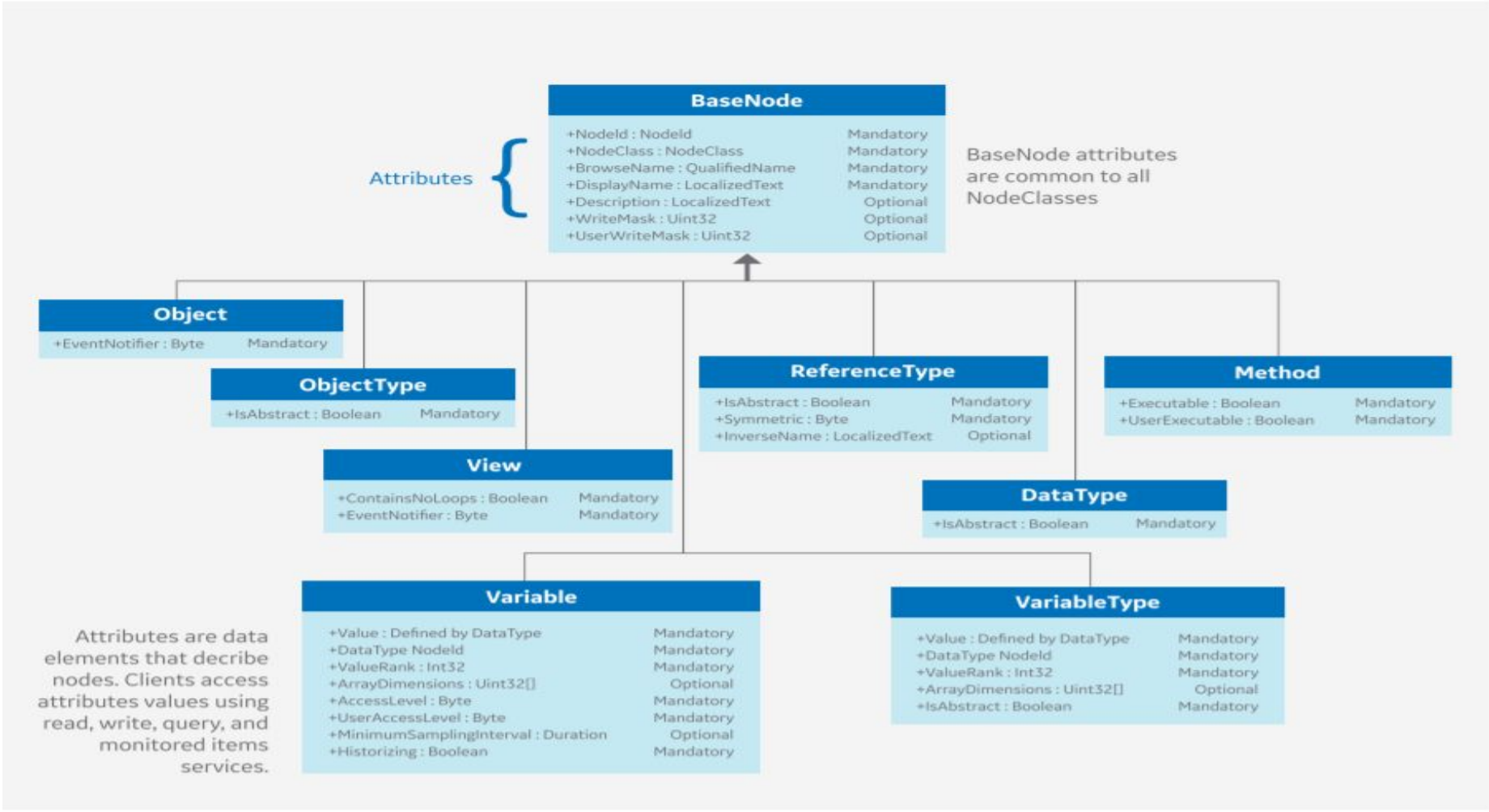


Information Modeling:

The OPC UA information Modeling framework turns data into information. With complete object-oriented capabilities, even the most complex multi-level structures can be modeled and extended. Data-types and structures are defined in profiles. For example, the exiting OPC Classic specification were modeled into UA profiles which can also be extended by other organizations:




Process Border or Network
From OPC UA Server
To OPC UA Server
From OPC UA Server
To OPC UA Server
OPC UA Server Application
Request Messages
Response Messages
OPC UA Communication Stack
Publish Request Messages
Notification Messages
OPC UA Client API
Server Application
Unified AddressSpace
Node
Subscriptions
Monitored Items
Real industrial objects


BaseNode
Attributes
+NodeId : NodeId Mandatory
+NodeClass : NodeClass Mandatory
+BrowseName : QualifiedName Mandatory
+DisplayName : LocalizedText Mandatory
+Description : LocalizedText Optional
+WriteMask : Uint32 Optional
+UserWriteMask : Uint32 Optional
BaseNode attributes are common to all NodeClasses
Object
+EventNotifier : Byte Mandatory
ObjectType
+IsAbstract : Boolean Mandatory
View
+ContainsNoLoops : Boolean Mandatory
+EventNotifier : Byte Mandatory
ReferenceType
+IsAbstract : Boolean Mandatory
+Symmetric : Byte Mandatory
+InverseName : LocalizedText Optional
Method
+Executable : Boolean Mandatory
+UserExecutable : Boolean Mandatory
DataType
+IsAbstract : Boolean Mandatory
Variable
+Value : Defined by DataType Mandatory
+DataType NodeId Mandatory
+ValueRank : Int32 Mandatory
+ArrayDimensions : Uint32[] Optional
+AccessLevel : Byte Mandatory
+UserAccessLevel : Byte Mandatory
+MinimumSamplingInterval : Duration Optional
+Historizing : Boolean Mandatory
VariableType
+Value : Defined by DataType Mandatory
+DataType NodeId Mandatory
+ValueRank : Int32 Mandatory
+ArrayDimensions : Uint32[] Optional
+IsAbstract : Boolean Mandatory
Attributes are data elements that decribe nodes. Clients access attributes values using read, write, query, and monitored items services.

Meta data of data(Information Model)

```
            <Reference ReferenceType="HasComponent">ns=1;i=6005</Reference>
            <Reference ReferenceType="HasComponent">ns=1;i=6004</Reference>
        </References>
    </UAObject>
    <UAVariable DataType="LocalizedText" NodeId="ns=1;i=6006" BrowseName="1:Data_01" AccessLevel="3">
        <DisplayName>Data_01</DisplayName>
        <References>
            <Reference ReferenceType="HasProperty" IsForward="false">ns=1;i=5002</Reference>
            <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
        </References>
    </UAVariable>
    <UAVariable DataType="Float" NodeId="ns=1;i=6007" BrowseName="1:Data_2" AccessLevel="3">
        <DisplayName>Data_2</DisplayName>
        <References>
            <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
            <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5002</Reference>
        </References>
    </UAVariable>
    <UAVariable DataType="Float" NodeId="ns=1;i=6008" BrowseName="1:Data_3" AccessLevel="3">
        <DisplayName>Data_3</DisplayName>
        <References>
            <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
            <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5002</Reference>
        </References>
    </UAVariable>
    <UAVariable DataType="Float" NodeId="ns=1;i=6009" BrowseName="1:Data_4" AccessLevel="3">
        <DisplayName>Data_4</DisplayName>
        <References>
            <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
            <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5002</Reference>
        </References>
    </UAVariable>
    <UAVariable DataType="Int64" NodeId="ns=1;i=6001" BrowseName="1:id" AccessLevel="3">
        <DisplayName>id</DisplayName>
        <References>
            <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
            <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5002</Reference>
        </References>
        <Value>
            <uax:Int64>0</uax:Int64>
        </Value>
    </UAVariable>
    <UAVariable DataType="UtcTime" NodeId="ns=1;i=6003" BrowseName="1:myvar_unix_time" AccessLevel="3">
```

python libraries which we are working with them:

1- google protobuf:

For buffering and getting data on the internet

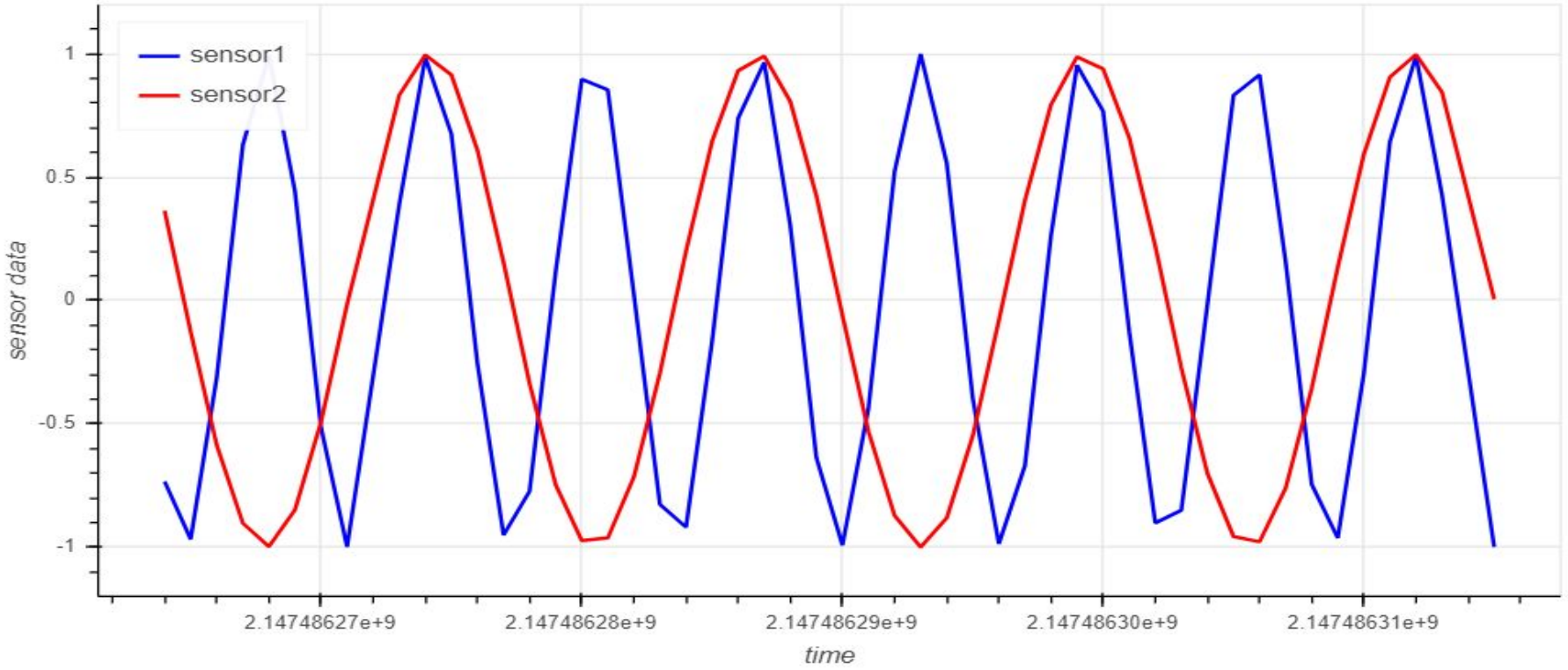
2-OPCUA client and server:

For communicating with opc application

3- Streamz and Holoviews:

for streaming data and plotting them

A sample of streaming plot:



Resources for reading more about Information Model:

- https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model/
- http://www.commsvr.com/UAModelDesigner/Index.aspx
- http://www.cas.internetdsl.pl/commserver/P_DowloadCenter/P_Publications/P-15010101-AddressSpaceInterchangeXML.pdf
- https://commsvr.gitbook.io/ooi/
- https://documentation.unified-automation.com/uamodeler/1.6.2/html/modeling.html

Additional source for generating information model XML file

- http://www.commsvr.com/UAModelDesigner/Index.aspx
- http://documentation.unified-automation.com/uasdkhp/1.0.0/html/_doc__opc_ua__opc_ua_information_models.html
- https://commsvr.gitbook.io/ooi/semantic-data-processing/addressspaceaddressspacemodel
- https://documentation.unified-automation.com/uamodeler/1.6.2/html/index.html
- https://documentation.unified-automation.com/uasdkhp/1.0.0/html/_l2_ua_node_ids.html
- https://opcua.rocks/custom-information-models/