OPC-UA Information Model for Large-Scale Process Control Applications

Pavel Trnka*, Petr Kodet*, Vladimír Havlena*,**
*Honeywell Prague Laboratory, V Parku 18, Prague, Czech Republic
**Department of Control Engineering, Czech Technical University in Prague, Karlovo náměstí 13, Prague, Czech Republic
{pavel.trnka,petr.kodet,vladimir.havlena}@honeywell.com

***Abstract* – The distributed and networked control of large-scale systems is typically designed as multi-layer control architecture. This brings advantage of a simplified design for complex control strategies; however, it complicates the information consistency between individual layers (PID controllers, advanced process controllers, real time optimizers) under changes in controlled plant topology and other events. Individual control layers require different representation of knowledge; however, the information solution has to guarantee the cross-layer integration, consistency and uniform representation of on-line and off-line process data, topology information and dynamics / performance models. The paper presents solution based on OPC Unified Architecture (OPC-UA) model and two levels control architecture. The solution unifies representation of data and topology in Service Oriented Architecture (SOA) and adopts Cloud Computing paradigm.**

I. INTRODUCTION

The implementation of a sophisticated control and monitoring system for a large-scale plant brings several challenges. The control system is typically constructed as a hierarchical architecture (*Fig. 2*) ranging from basic process control to advanced process control, real time optimization and high levels of scheduling and business planning. The main challenges are:

- Information model consistency on all hierarchy levels
- Cross-layer integration
- Event-driven consistent reconfiguration of all layers
- Support for flexible on-demand optimization and what-if analysis
- Data aggregation from heterogeneous sources

The solution to these challenges presented in this contribution is based on a two levels server architecture and OPC-UA information model, which builds on Service Oriented Architecture (SOA) and Cloud Computing paradigm. The factory floor is assumed to be controlled by a set of heterogeneous SCADA/DCS control systems. Raw data are aggregated and unified by L1 servers, which are bound and act as a single virtual server containing full OPC-UA information model with data, metadata and topology information. The virtual L1 server provide unified access point and event generation to a chain of L2 servers, which are specialized interfaces mapping information model to a cloud of shared services. The services are higher layers of control hierarchy: advanced process control, real time optimization, scheduling, business planning, etc.

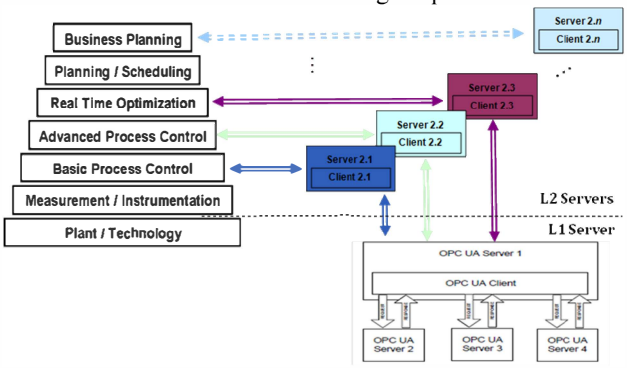

Fig. 2 Hierarchical control hierarchy with interconnection to Layer 1 and Layer 2 information servers

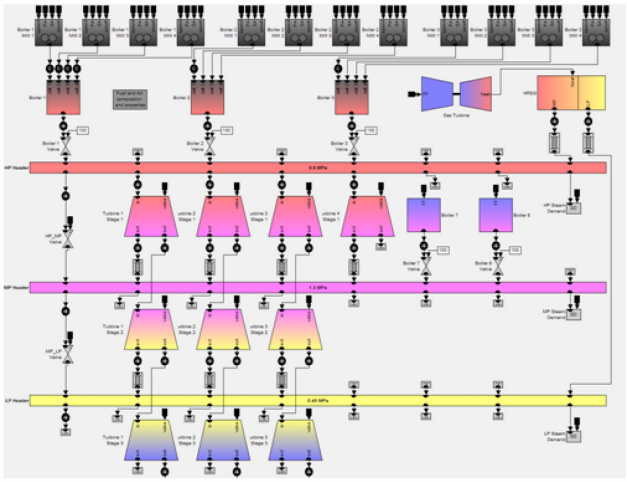

Fig. 1 Power plant simulator in Matlab Simulink.

The article presents design of information model for large-scale control systems, which is at the end demonstrated on a part of power plant (*Fig. 1*).

II. INFORMATION MODEL

Information model for process control has to consistently and uniformly represent on-line and off-line process information, which will be used by multiple users with different requirements and functionality. The users can be controllers on a different level of control hierarchy (PID controllers, advanced process controllers, real time optimizers), process operators using their operator screens, alarm management systems, process historian, etc.

The information model has two consistent levels. The levels differ in their information details according to their target use.

III. L1 Model

L1 model is a low level model containing detailed process information and topology information, which is used by L1 server. The process is represented as a set of devices, which have input and output ports and these ports are interconnected by streams. The devices, ports and streams create basic framework of L1 model. The additional details are usually directly associated with the objects of this basic framework.

A. Information Model Organization

OPC-UA information model topology is not limited to a tree. It allows "full mesh" topology; however, the backbone of this full mesh structure is usually tree of hierarchical references, which is used for nodes referencing.

The hierarchy tree is built by HasComponent references, which are automatically created by using *<Children> </Children>* directive. The tree structure can be further organized by folder objects, which are instances of FolderType type or its descendants. For the process control applications we introduce FolderAreaType, which derives from FolderType (*Fig. 4*).

B. Basic Objects

The basic objects of the proposed information model for process control applications are devices, ports and streams. These objects are instances of DeviceType, PortType, StreamType or their appropriate descendants.

The position of these types in their hierarchy is in Figure 4. ObjectType is a basic object type, ModelEntityType is an abstract ancestor of all objects used in our information model design and ModelElementType is an abstract ancestor of all physical model elements.

1. Devices

The device object aggregates information and objects related to a single device. The basic device object type DeviceType has the following structure:

DeviceType		
Name	**Type**	**Description**
`Details`	FolderType	folder aggregating device details and device parameters
`Ports`	FolderType	folder aggregating device ports
`Measurements`	FolderType	folder aggregating measurements in the device
`Actuators`	FolderType	folder aggregating actuators in the device

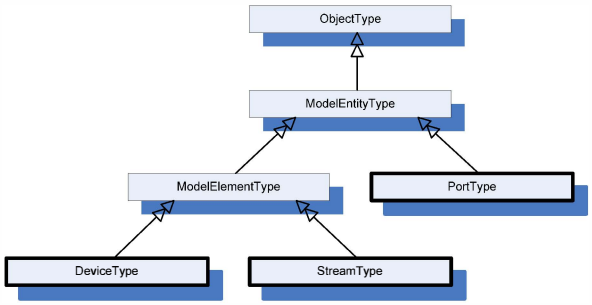

Fig. 3 Information model organization by FolderAreaType folders.

2. Ports

Device inputs and outputs are represented by ports, which are object instances of PortType ObjectType or its descendants.

Their orientation property determines the relation between oriented variables on the port and the signs of their values. It has enumeration type with possible values INPORT and OUTPORT. For OUTPORT the values directing out of the port have positive sign and vice versa.

It is possible to connect two ports of the same orientation (OUTPORT with OUTPORT).

Design constraints require all port object instances to be always created under Ports folder of the appropriate device object.

PortType		
Name	**Type**	**Description**

`Orientation`	Enum (optional)	Determines the relation between physical flow orientation (in device / out of device) and the sign of process value, limits, etc.
		`0 … INPORT`
		`1 … OUTPORT`

3. Streams

From OPC Unified Architecture book [1]: *"References do not contain any Attributes or Properties. In the case that some additional information should be added to the relation of two Nodes, a Proxy-Node must be added and connected by both Nodes instead of using a single Reference."*

In the L1 model this proxy-node is a stream object, representing interconnection between two devices. The stream is always connected by references (derived from the Flow reference – Section 2.4) to the ports of the devices it interconnects (never directly to the device objects).

The stream objects have object type StreamType or its descendants.

The property IsIsolating indicates if the stream has isolating capability and the property IsolationStatus indicates current isolation status. The IsolationStatus is important for L2 information model, as its change indicate topology change, which can initiates control strategy adjustment on multiple control layers.

The stream object instance should be created under the FolderAreaType object, which is shared by the two interconnected devices and which is the nearest in the objects organization hierarchy. The example is in Figure 9, where Stream12 interconnects devices Device1 and Device2. The Stream12 is created under Area2, which is shared by Device1 and Device2.

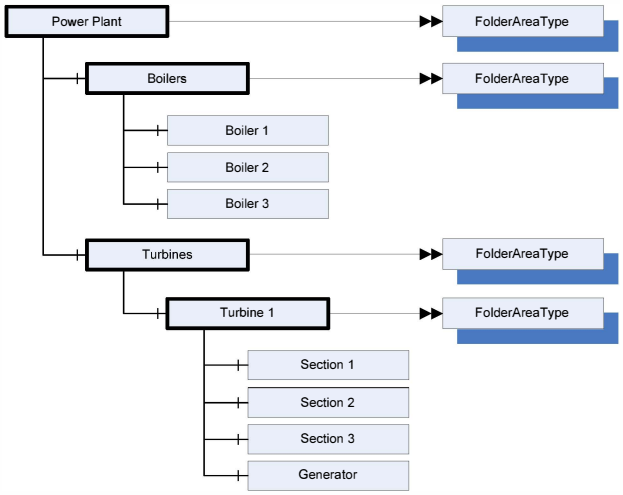

Fig. 4 Information model organization by FolderAreaType folders.

Design constraints require that the stream has to be connected to the ports of devices it interconnects. The connection is realized through OPC UA references, and these references must have ReferenceType derived from the Flow reference.

StreamType		
Name	**Type**	**Description**
`IsIsolating`	Boolean (mandatory)	Determines if stream has isolating capability (e.g. has steam isolating valve).

`IsolationStatus`	Enum (optional)	Current isolation status.
		`0 … NONAVAILABLE`
		`1 … OPEN`
		`2 … CLOSED`

4. *Variable with Engineering Units*

The specification of many variables in the information model does not have sense without clear association with engineering units. For these purpose we introduce new OPC UA VariableType which has DataType of Double type, and has mandatory engineering units information (implemented as OPC UA property).The name of this VariableType is DoubleEU.

DoubleEU		
Name	**Type**	**Description**
`EngineeringUnits`	EUInformation (mandatory)	Engineering units for the value.

C. *Physical Variables and Units Implementation*

The measured values or the variables for actuators should be represented by OPC-UA built-in objects of AnalogItemType type for continuous values or DiscreteItemType type for discrete variables.

Apart from the EURange the important property for AnalogItemType is EngineeringUnits. It is optional by OPC-UA standard; however, we would recommend to used this property as mandatory and always specify it.

AnalogItemType (OPC-UA standard type)		
Name	**Type**	**Description**
`Definition`	String (optional)	A vendor-specific, human readable string that specifies how the value is calculated.
`ValuePrecision`	Double (optional)	The maximum precision that the server can maintain for the item based on restrictions in the target environment.
`EURange`	Range (mandatory)	Defines the value range under normal operation.
`InstrumentRange`	Range (optional)	Defines the value range that can be returned by the instrument.
`EngineeringUnits`	EUInformation (optional)	Engineering units for the `value`.

1. *Engineering Units*

Important part of each value related to physical reality are engineering units. The OPC UA Data access specification [2] suggests using engineering units in accordance with United Nations CEFACT Codes for Units of Measurement [3].

The recommendation comes with an extensive table of engineering units. The table divides units to groups and specifies units conversion factors. The table also assigns unique 3 characters long alphanumeric code to each unit. A short excerpt from the table:

Name	**Symbol**	**Code**	**UnitId**
kilogram per second	kg/s	KGS	4933459
Pascal	Pa	PAL	5259596
Megapascal	MPa	MPA	5066817
degree Celsius	°C	CEL	4408652
Kelvin	K	KEL	4932940
Watt	W	WTT	5723220
Kilowatt	kW	KWT	4937556
Megawatt	MW	MAW	5062999
Percent	%	P1	20529

The alphanumeric code can be used to get numerical unit identifier (UnitId) and this identifier can be used by OPC-UA to uniquely identify individual units by the following code:

```
Int32 unitId = 0;
Int32 c;
for (i=0; i<=3;i++)
{
  c = CommonCode[i];
  if (c == 0) break;
  unitId = unitId << 8;
  unitId = unitId | c;
}
```

OPC-UA defines standard object type for engineering units representation. The structure of this object type together with recommended use in accordance with UB.CEFACT recommendation is in the following table.

EUInformation (OPC-UA standard type)		
Name	**Type**	**Description**
`NamespaceUri`	String (optional)	Identifies the organization that defines the EUInformation. Should be: http://www.opcfoundation.org/UA/units/un/cefact
`UnitId`	Int32 (optional)	Identifier for programmatic evaluation (-1 if not available). Should be UnitId computed from

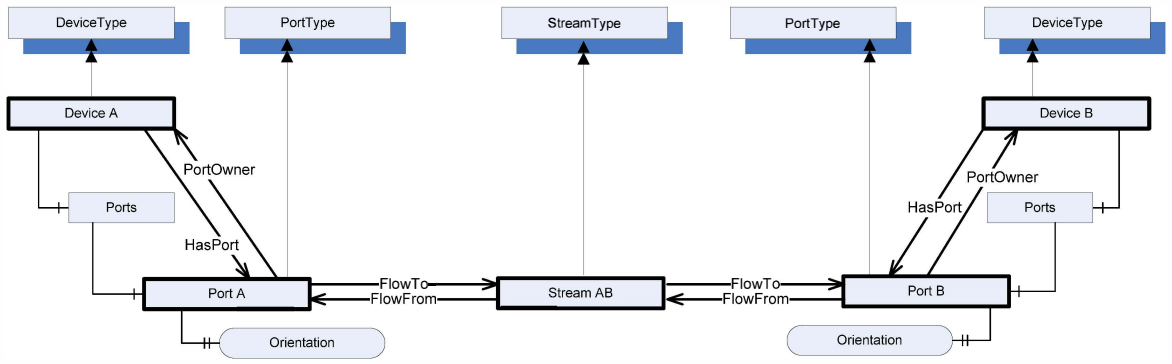

Fig. 6 Device-Port-Stream Example

		UN/CEFACT unit alphanumeric code (see code example above).
`DisplayName`	LocalizedText (optional)	Display abbreviation of the engineering unit (e.g. m/s). Should be copied from "Symbol" field of UN/CEFACT units table.
`Description`	LocalizedText	Full name of engineering unit (e.g. meter per second). Should be copied from "Name" field of UN/CEFACT units table.

2. *Physical Variable Types*

Physical variables (that are feasible for an association with ports or streams) can be divided into potential variables (also called across variables) and flux variables (also called through variables).

The flux variables can be measured in series with the device and they sum to zero in a point (current, force, mass flow, etc.). It is important for the flux variables that the sign of these variables value is related to their physical direction.

The potential variables can be measured in parallel with the device and the parallel measurement is usually done with respect to some reference point. The examples of potential variables are pressure, temperature, voltage, etc., where the respective references can be (atmospheric pressure, absolute zero, voltage ground level). The separation to flux and potential variables can be complicated by the fact that variables often appear in forms integrated or differentiated with respect to time.

The separation is important for proper interpretation of variable sign in a connection with ports and streams. The separation of analog variables to flux and potential is implemented by two subtypes of AnalogItemType to AnalogItemFluxType and AnalogItemPotentialType.

D. *Topology Information*

Topology information describes interconnections between devices. An interconnection can be represented by a simple reference from one device object to another. However, we chose more complex representation to include additional useful information.

The interconnection between two devices has always the following pattern of objects:

`Device → Port → Stream → Port → Device`

It requires an introduction of several new reference types, which are all derived from NonHierarchicalReference (admits loops), and are summarized in *Fig. 5*.

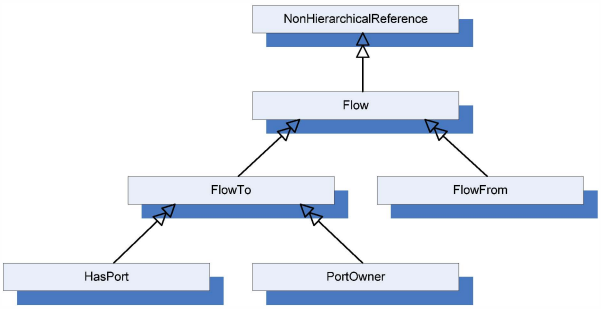

Fig. 5 Device-Port-Stream related references hierarchy.

Each device has HasPort references to its own input and output port objects. There is also back reference from the ports to their owing device. This reference has ReferenceType PortOwner. The design constraint is that each port can have only a single PortOwner reference.

The streams are oriented. The orientation is determined by the type of references between stream and the ports. The positive (flow) orientation is represented by FlowTo reference. The FlowTo reference is complemented by the FlowFrom reference, which is a back reference (against stream positive orientation). The design constraints are that a stream interconnects exactly two ports and that FlowTo and FlowFrom references from stream to both ports have to be consistent (*Fig. 7*).

Filtering the references from Flow type allows browsing from Device A to Device B: Device A → HasPort → Port A → FlowTo → Stream → FlowTo → Port B → PortOwner → Device B and back (against positive stream orientation) Device B → HasPort → Port B → FlowFrom → Stream → FlowFrom → Port A → PortOwner → Device A. Filtering the references from FlowTo type allows browsing in the positive stream orientation only: Device A → HasPort → Port A → FlowTo → Stream → FlowTo → Port B → PortOwner → Device B.

1. *Sensors and actuators binding to streams and ports*

Object instance of AnalogItemType can represent a process value measured by a sensor or a set point for an actuator. The object can be attached directly to a device (measurements or actuators folder) or to a port or a stream. An attachment of AnalogItem to a port or a stream is done by Signal or InvSignal reference. The correct choice of Signal or InvSignal is important for the flux type variables. Signal type reference indicates that the AnalogItem orientation complies with the orientation of the stream and the InvSignal type reference indicates that they are opposite. An example of the measurement attachment to the stream is in *Fig. 7*. Signal reference between Stream AB and Measurement A indicates that Measurement A has positive value for flux variables from Port A to Port B (same as Stream AB).

The idea is that AnalogItem can be attached to a port or a stream and the value of AnalogItem can be queried from an arbitrary object in the chain Port -> Stream -> Port. If the AnalogItem is the flux type variable (i.e. AnalogItemFluxType) then the queried value is returned with the correct sign according the orientation of port / stream / AnalogItem. Hereafter, we call this approach 'variables sharing'.

Assume an example in *Fig. 7*, where all AnalogItems are of flux type. Note that Port A and Port B are both output ports (OUTPORT).

Querying the Port A for the Measurement A value returns the value with unchanged sign, because the Port A orientation complies with the Stream AB orientation (Port A OUTPORT orientation vs. orientation of FlowTo reference) and the Measurement A orientation complies with the Stream AB orientation (referenced by Signal type reference).

Querying the Port A for the Measurement B value returns the value with inverted sign, because the Port A orientation complies with the Stream AB orientation, but the Measurement B orientation is against Stream AB orientation (referenced by InvSignal type reference).

Querying the Port B for the Measurement A value returns the value with inverted sign, because the Port B orientation is against the Stream AB orientation (Port B OUTPORT orientation vs. orientation of FlowTo reference), and the Measurement A orientation complies with the Stream AB orientation.

And similarly for other ports and measurements:

	Sensor A	Sensor B	Sensor C
Port A	+	-	-
Stream AB	+	-	-
Port B	-	+	+

Note that the flux variables sharing makes sense even when Stream AB IsolationStatus is set to CLOSED. This is in contrary with the potential variables sharing.

Now assume a similar *Fig. 7*, where all AnalogItems are of potential type (i.e. AnalogItemPotentialType).

The mutual orientation of ports / streams and AnalogItems is ignored for potential type variables. The values sign remains unchanged. Now, there is an important design decision of measurements sharing for potential variables across Port -> Stream -> Port chain. Interpreting streams in the framework of bond graphs gives sense to share for example pressure measurement on Port A with Port B (unless Stream AB is closed).

On the other hand if the Stream AB represents a pipe of a significant length, then the pressure drop cannot be ignored and the sharing of potential variables is wrong.

We chose the second option, where the potential variables are not shared along the stream:

	Sensor D	Sensor E	Sensor F
Port A	**N/A**	**N/A**	**N/A**
Stream AB	+	+	**N/A**
Port B	**N/A**	**N/A**	+

E. Example

Fig. 8 shows an example of an interconnection between devices in the plant information model.

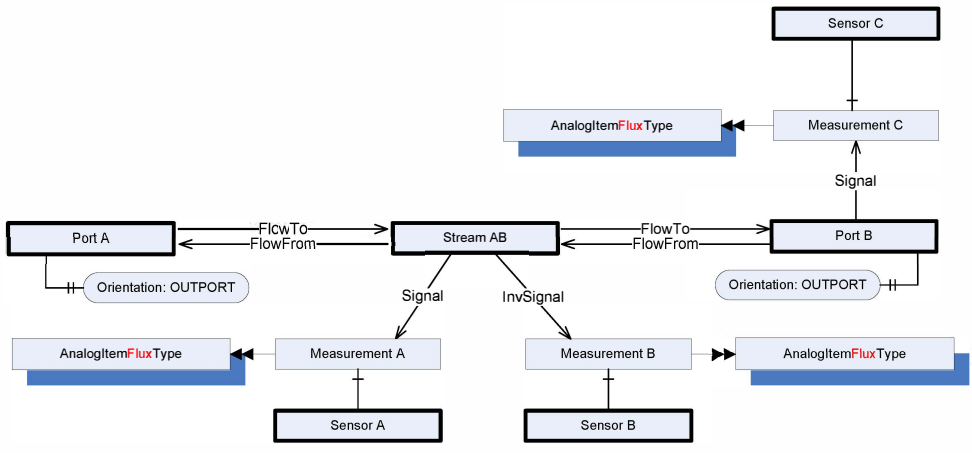

Fig. 7 Measurements attached to the stream. Measured variables are of Flux type.

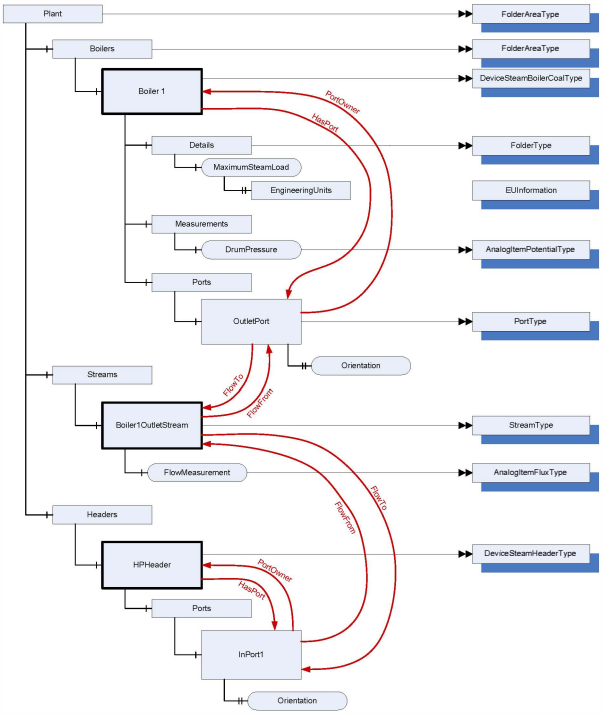

Fig. 8 Power plant example.

IV. L2 Model

L2 model is a high level model for advanced process control and other higher control layers (APC, RTO, and MES). It contains information necessary for retrieving dynamic or static behavior models needed by higher control layers.

The L2 model has dynamic and / or static model of individual devices (examples of model are state space linear model, transfer function matrix, nonlinear static model, etc.). The L2 model includes description of ports role in controller design (MV – Manipulated Variable, DV – Disturbance Variable, CV – Controlled Variable, etc.) and it includes devices topology description.

The L2 model uses object types of L1 model and extends them with additional functionality where needed.

F. L2 Device

The DeviceL2Type extends L1 DeviceType type by Models folder for device models. The folder can include multiple device models of a different type (*Fig. 9*).

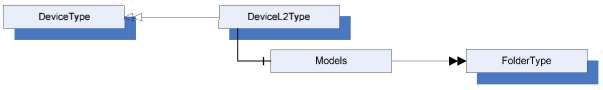

Fig. 9 DeviceL2Type

1. Device Models

The basic model types together with their hierarchy are in *Fig. 10*. The first separation is to static and dynamic models. The dynamic models are then specialized to linear models, where subtype ModeDynamicLinearSSType is linear state space model and ModeDynamicLinearTFType is transfer function model.

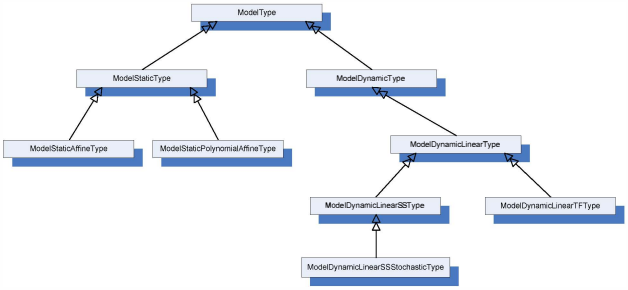

Fig. 10 Basic model type hierarchy.

G. L2 Port

An extension of PortType for L2 model is PortL2Type. It adds ModelIndex, which is port index in the model input or output vector. The second new property is Role, which indicates the role of the port in the control strategy (manipulated variable, controlled variable, measured disturbance variable or unmeasured disturbance variable).

PortL2Type		
Name	**Type**	**Description**
`Role`	Enum	port role in controller design: MV ..... Manipulated Variable CV ..... Controlled Variable MDV ..... Measured Disturbance Variable UDV ..... Unmeasured Disturbance Variable
`ModelIndex`	Enum	port index in the model input or output vector

H. L2 Stream

L2 information model uses the same stream type as L1 information model. It also uses the same references derived from Flow reference type.

V. Conclusion

The paper presented one of possible OPC-UA information models for large-scale control applications. The solution builds on service oriented architecture and is ready for cloud based implementation.

Acknowledgment

The authors gratefully acknowledge the financial support of the European Commission FP7 project ”AESOP” (project no. 258682),

References

[1] Wolfgang Mahnke, Stefan-Helmut Leitner, and Matthias Damm, OPC Unified Architecture.: Springer, 2009.

[2] OPC Foundation. (2011) OPC UA Specification Part 8 - Data Access (RC 1.02).

[3] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT). (2011, November) Codes for Units of Measurement (Recommendation No. 20). http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_Rev7e_2010.zip.