OPC UA Information Modelling (in a Nutshell)

Object

References Objects

Object

Object

Object

Browse

Call

ObjectType

BaseType

Type Information

SubType

Variable

Generates Events

Subscribe to

Read/Write

OPC UA Extensibility

ObjectType | Structure (Variables and Methods) and Semantic of Objects
VariableType | Structure (Sub-Variables and DataType) and Semantic of Variables
DataType | Structure and Semantic of DataTypes
ReferenceType | Semantic of References, ReferenceType Hierarchy
EventTypes | Structure (Event Fields) and Semantic of Events

Examples:
- File Transfer
- State Machines
Let’s model a Device – Valve

> Base Functionality
  > Value can be opened or closed

Type Definition of Valve

Variable representing Status

Instance Declaration: No Value, defines structure of Instances

Same structure as Type

Each Type derived from BaseObjectType

FolderType: Valves

Organized in address space

ValveType: Valve_1

BaseDataVariableType: Status
Let’s model a Device – Valve

> **DataType of Status?**
  > Opened / Closed -> Boolean?
  > Semantic Matters -> User Defined Enumeration
Let’s model a Device – Valve

> Identification
> Each Device has a SerialNumber

Valve: <DisplayName>
SN: <SerialNumber>

Simple UI Element based on TypeDefinition

TypeDefinition of Valve

Property containing Serial Number (String)

BaseObject
type

ValveType

BaseDataVariableType: Status

PropertyType: SerialNumber

Open
Let’s model a Device – Valve

> Identification – Better Approach
> Use Part 100 OPC UA Devices

Derive from DeviceType
Already defines SerialNumber
As defined in DeviceType
Allows Part 100 aware clients (like UAExpert) to deal with Device
Let’s model a Device – Valve

> **Controlling the Device**
>  > Pure Opened / Closed could be realized by writing the Status Variable
>  > Let’s assume you want to
>  > Open the value for a specific duration
>  > Only open it to a certain degree
>  > **Methods with Input Arguments**

```c
ControlDevice (  
[in] Byte DegreeOpenness  
in] Duration TimeOpened  
);
```

**Additional Variable**
Let’s model a Device – Valve

> Controlling the Device
> Updated UI

Valve: <Display\Name>
SN: <Serial\Number>
Open
Degree \text{Openness} \quad \text{Control}

Control
Degree \text{Openness} \quad \text{Time Opened} \quad 0.0
Cancel \quad \text{OK}
Event Fields

Represents a discrete Condition that is considered to be not normal
Let’s model a Device – Valve

> Add an Alarm

Alarm can be represented as Object

Updated UI

Valve: <DisplayName>
SN: <SerialNumber>

Open

Degree Openness

Control

Alarm

Ack
Let's model a Device – Valve

> Add an Alarm - Advanced

- **ValveType**
  - **BaseDataVariableType:** Status
  - **ControlDevice**
    - **PropertyType:** InputArguments
    - **BaseDataVariableType:** DegreeOpenness
    - **ValveAlarmType:** ValveFault
  - **ValveAlarmType:** ValveFault
  - **OffNormalAlarmType**
    - **PropertyType:** SerialNumber

**Event List**

<table>
<thead>
<tr>
<th>EventId</th>
<th>Message</th>
<th>Severity</th>
<th>Time</th>
<th>Ack</th>
<th>Active</th>
<th>SerialNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Valve broken</td>
<td>90</td>
<td>.....</td>
<td>NO</td>
<td>YES</td>
<td>#1234567890</td>
</tr>
<tr>
<td>125</td>
<td>Valve OK</td>
<td>5</td>
<td>.....</td>
<td>NO</td>
<td>NO</td>
<td>#0987654321</td>
</tr>
<tr>
<td>129</td>
<td>Valve broken</td>
<td>80</td>
<td>.....</td>
<td>YES</td>
<td>YES</td>
<td>#1234567890</td>
</tr>
</tbody>
</table>
Modelling is supported by tooling

UAModeler
- Define Information Models
- Export NodeSet-File
  - Standard format
  - Can be used for configuration (e.g. UaGateway)
- Code Generation

© Unified Automation GmbH – All rights reserved.
Demo

Information Model in UAModeler

Code Generation

Compile and Run

OPC UA

OPC UA Server

No Values, need to be implemented

Instances

OPC UA Instances

No Values, need to be implemented
Questions?

Dr. Wolfgang Mahnke
ascolab GmbH
wolfgang.mahnke@ascolab.com
OPC Day Seminar: 13:00-17:30 Meeting room 203, 2nd floor