INDIN' 2011
9th IEEE International Conference on
Industrial Informatics

OPC Unified Architecture (OPC UA)

New Opportunities of System Integration and Information Modelling in Automation Systems

Dirk van der Linden
Artesis University College of Antwerp
Department of Applied Engineering & Technology
Belgium

Wolfgang Granzer, Wolfgang Kastner
Vienna University of Technology
Automation Systems Group
Austria

artesis

AUTOMATION SYSTEMS GROUP

The speakers

Dirk van der Linden
Artesis University College of Antwerp
Department of Applied Engineering & Technology
Belgium
dirk.vanderlinden@artesis.be

Wolfgang Granzer
Vienna University of Technology
Automation Systems Group
Austria
w@auto.tuwien.ac.at

Wolfgang Kastner
Vienna University of Technology
Automation Systems Group
Austria
k@auto.tuwien.ac.at

Antwerp – Culture and economy (port)

Artesis – Association - University

artesis

auha

OPC UA Research Group

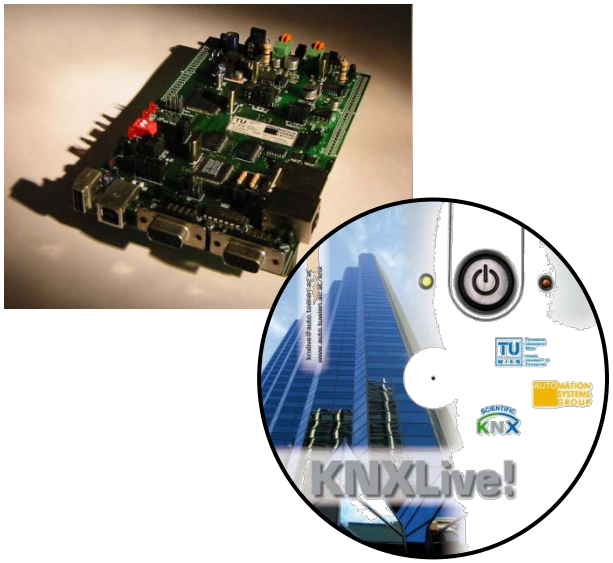
Dirk van der Linden

Maarten Reekmans

Vincent Vanderputten

Automation Systems Group

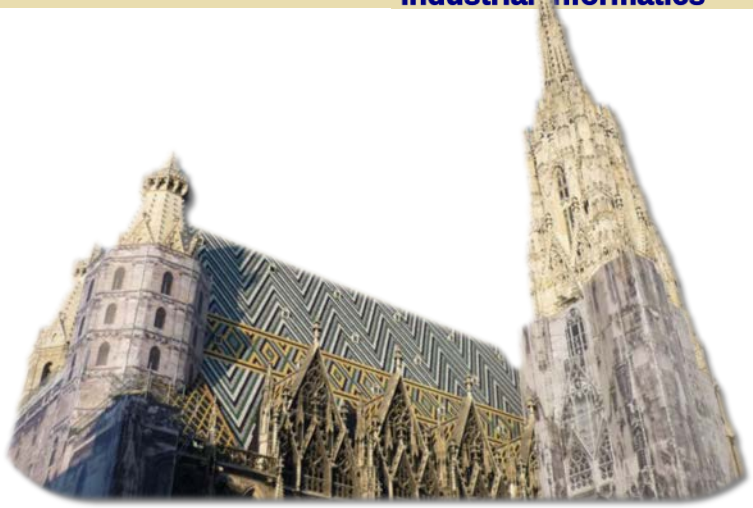
- Computer Engineering/Software Engineering
  - System analysis and system design
  - Knowledge engineering and information modelling
  - (Enterprise) System interfaces
  - Advanced control strategies
  - Internet of Things
- Distributed Automation
  - Building automation (wired/wireless)
  - Home automation/Smart homes
  - Industrial automation and control
  - Networked embedded systems



TU WIEN ! AUTOMATION SYSTEMS GROUP

Current research topics

- Deep integration
  - QoS in embedded networks
  - Security/Safety
- Management integration
  - Information modelling & profiles
  - Web Services
  - Ontologies
- Domains of interest
  - Energy and performance optimization
  - Surveillance, supervision & monitoring
  - Smart grids/Smart cities
  - Ambient assisted living



Outline

- Introduction and motivation to OPC UA
- Address space and information modelling
- Coffee break and discussion
- Communication services in OPC UA
- Advanced concepts in OPC UA
- Profiles and conformance units
- OPC UA software and development kits
- Demonstration
- Discussion

- A note to this tutorial:
  - Discussions are welcome! → ask questions at any time

INDIN' 2011
9th IEEE International Conference on
Industrial Informatics

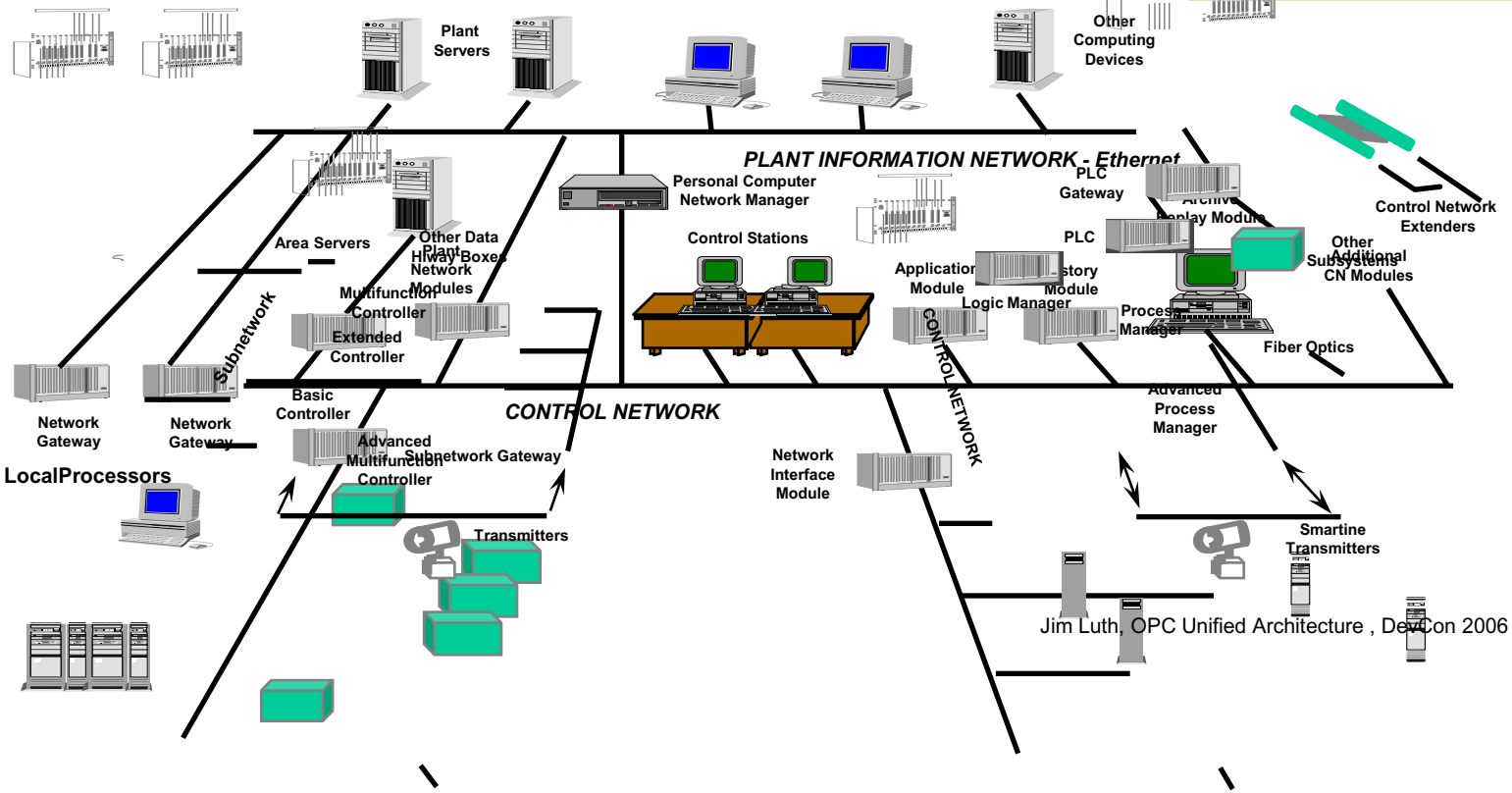
Introduction and motivation to OPC UA

Introduction and motivation to OPC UA

- Introduction and motivation
- Why is OPC in general useful, history
- OPC Foundation
- Comparison with classical OPC
- Overview of the OPC UA standards
  - Different parts of OPC UA specifications
  - Two pillars:
    - (Cross-platform) Transport mechanisms
    - (Meta-)Data modelling

Industrial communication

hours
min
sec
msec
µsec



- Vertical + Horizontal communication
- Complex information flows
- Multi-Vendor devices + Proprietary protocols
- Multiple interfaces

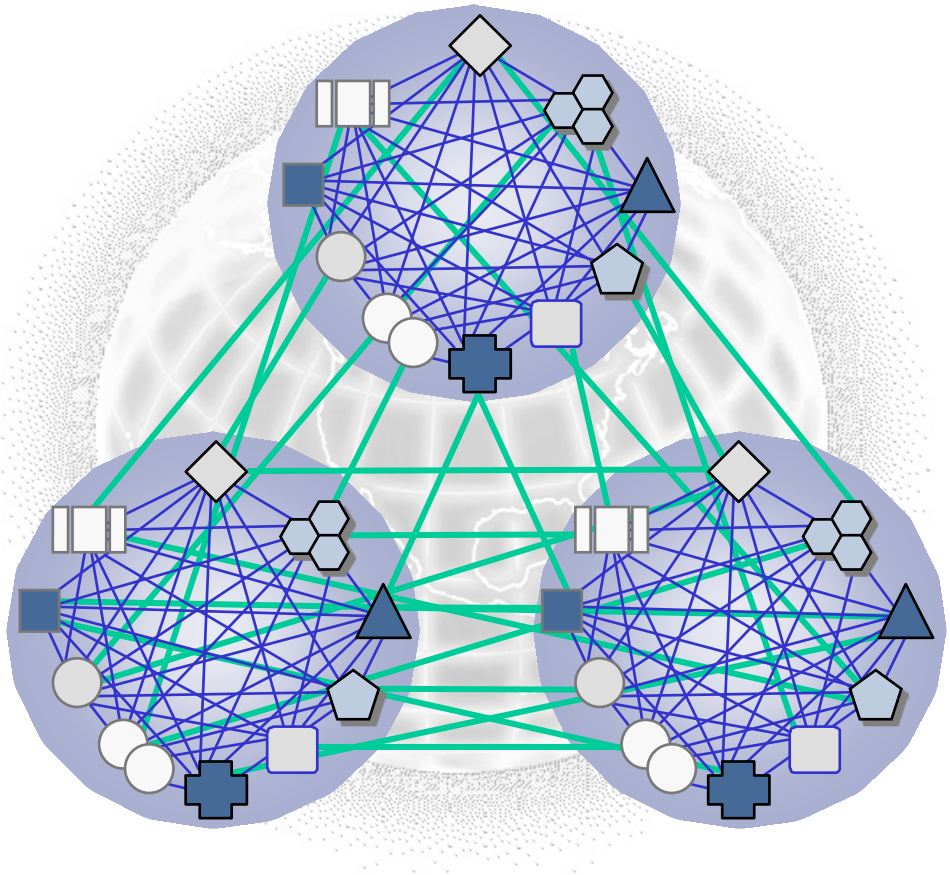
The Inter-enterprise nightmare

- Different suppliers & manufacturers

→ Proprietary technologies and protocols

→ ”Point-to-point” integration

→ Maintenance complex and expensive



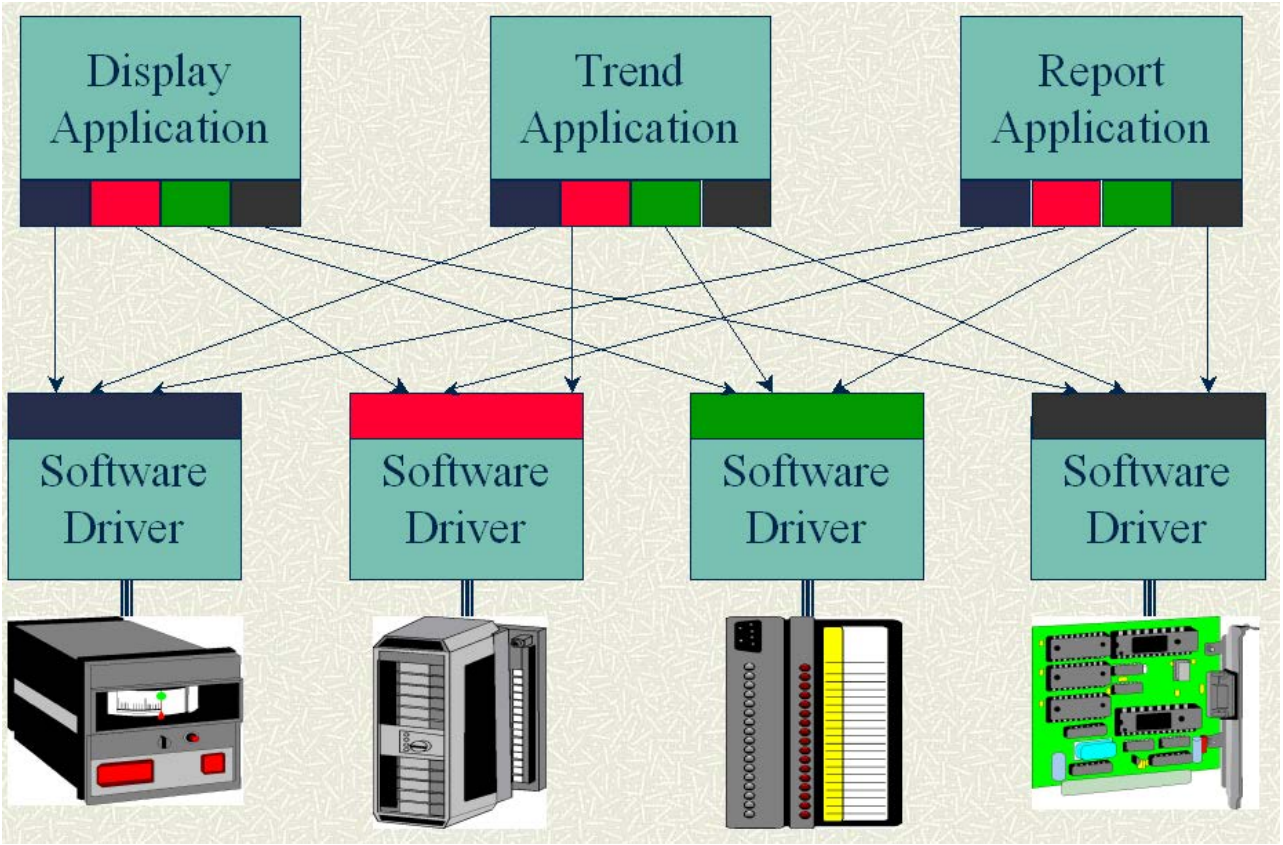
OPC Foundation (www.opcfoundation.org)

Many incompatible protocols

Jim Luth, OPC Unified Architecture , DevCon 2006

Interconnecting devices & software

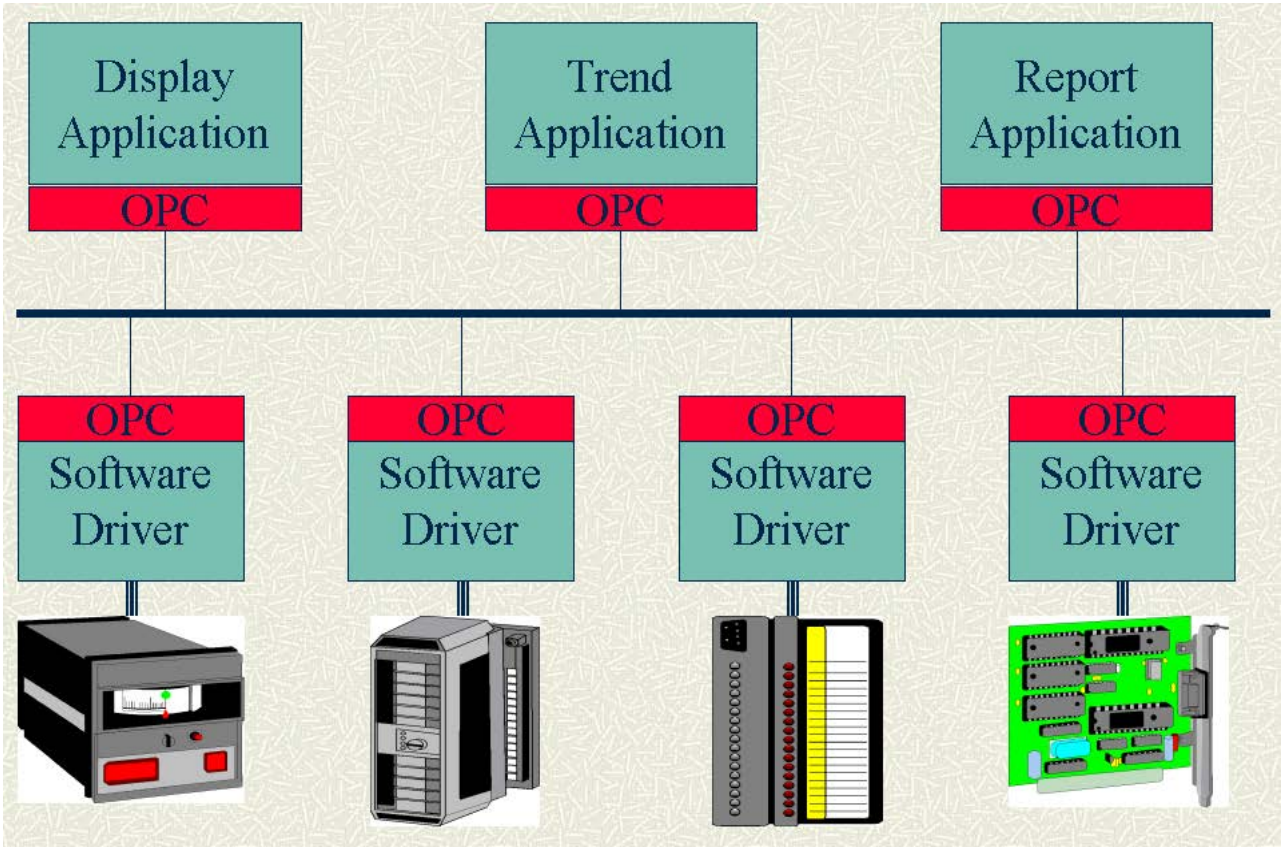
- No standard interface → custom expensive solutions
- No reusability, no flexibility



Proprietary protocols & custom solutions

Interconnecting devices & software

- 1996: OPC as a solution



Advantages of OPC

- Hardware vendors:
  - Optimized communication of their devices
- Software developers:
  - Less development and maintenance of drivers
- System integrators:
  - Integrated solutions possible with different brands
- End users:
  - Broader range of choice of hard- and software

OPC Foundation

- Creation of OPC Foundation 1994 (task force)
  - Then: OPC → **O**LE for **P**rocess **C**ontrol
  - Now: OPC → **O**pen / inter**O**perability
    Portability / **P**erformance
    Connectivity / **C**ollaboration
- Creates and manages standards:
  - 1996 → OPC DA
  - Later → OPC A&E, OPC HDA, OPC XML-DA, …
  - Most recent → OPC UA

OPC® FOUNDATION

OPC Foundation: OPC Europe

OPC Europe Board of Directors

Dr. Reinhold Achatz	OPC Foundation Vice President and Board Member	*Siemens*
Thomas J. Burke	OPC Foundation President	*OPC Foundation*

OPC Europe Officers

Stefan Hoppe	OPC Europe President	*Beckhoff*	stefan.hoppe_AT_opcfoundation.org
Tino Hildebrand	OPC Europe Vice President	*Siemens*	tino.hildebrand_AT_siemens.com
Yvonne Neumann	Marketing	*MatrikonOPC*	yvonne.neumann_AT_matrikonopc.com
Matthias Damm	Technology	*ascolab*	matthias.damm_AT_ascolab.com
Juergen Lange	Marketing-	*Softing*	juergen.lange_AT_softing.com
Manny Mandrusiak	VP Marketing	*OPC Foundation*	manny.mandrusiak_AT_opcfoundation.com

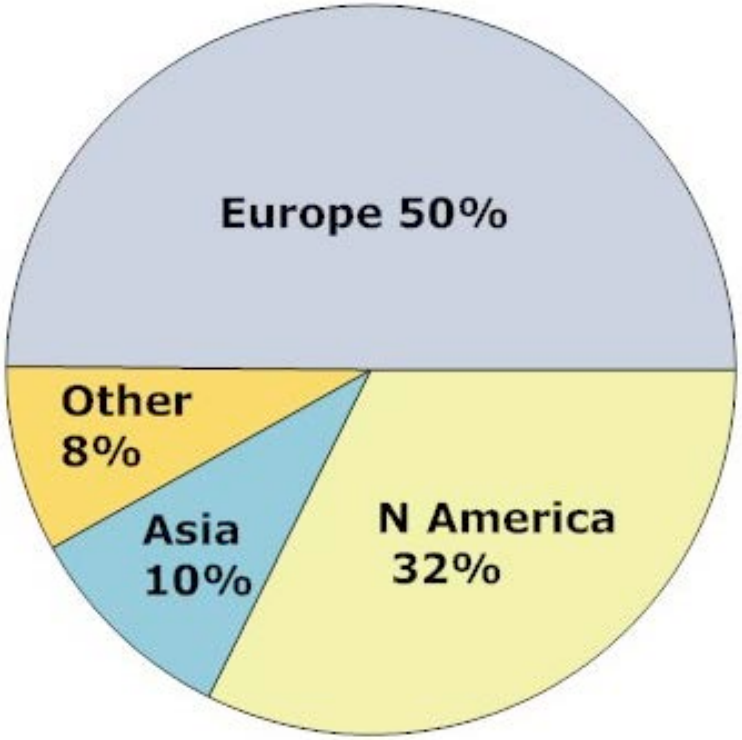
OPC Europe Country Representatives

Germany	Stefan Hoppe	*Beckhoff*	stefan.hoppe_AT_opcfoundation.org	Germany Events
Austria	Michael Haas	*Certec*	michael.haas_AT_certec.at	Austria Events
Belgium	Dirk van der Linden	*Artesis University College of Antwerp*	dirk.vanderlinden_AT_artesis.be	www.webcom-eu.org & http://code.google.com/p/opc-ua/
Czech Republic	Zbynek Zahradnik	*OPC Labs*	zbynekz_AT_opclabs.com	Czech Events
Finland	Jouni Aro	*Prosys*	jouni.aro_AT_prosys.fi	Finland Events
France	Michel Condemine	*4CE Industry*	michel.condemine_AT_opcfoundation.org	OPC France Events
Ireland	Liam Power	*Embedded Labs*	liam.power_AT_embeddedlabs.com	Ireland Events
Italy	Claudio Fiorani	*Progea srl*	cfiorani_AT_progea.com	Italy Events
Poland	Dr. Marius Postol	*CAS*	mailto:mpostol_AT_cas.eu	Poland Events
Spain	Nacho Armesto	*University of Vigo*	nacho.armesto_AT_gmail.com	Spain Events

OPC Foundation (www.opcfoundation.org)

OPC Foundation: some facts …

- More than 400 members
  - About 150 vendors
  - About 40 end users



- Worldwide mailing list for surveys: 30 000+
- Companies building OPC products: 3500+
- OPC products: 22 000+

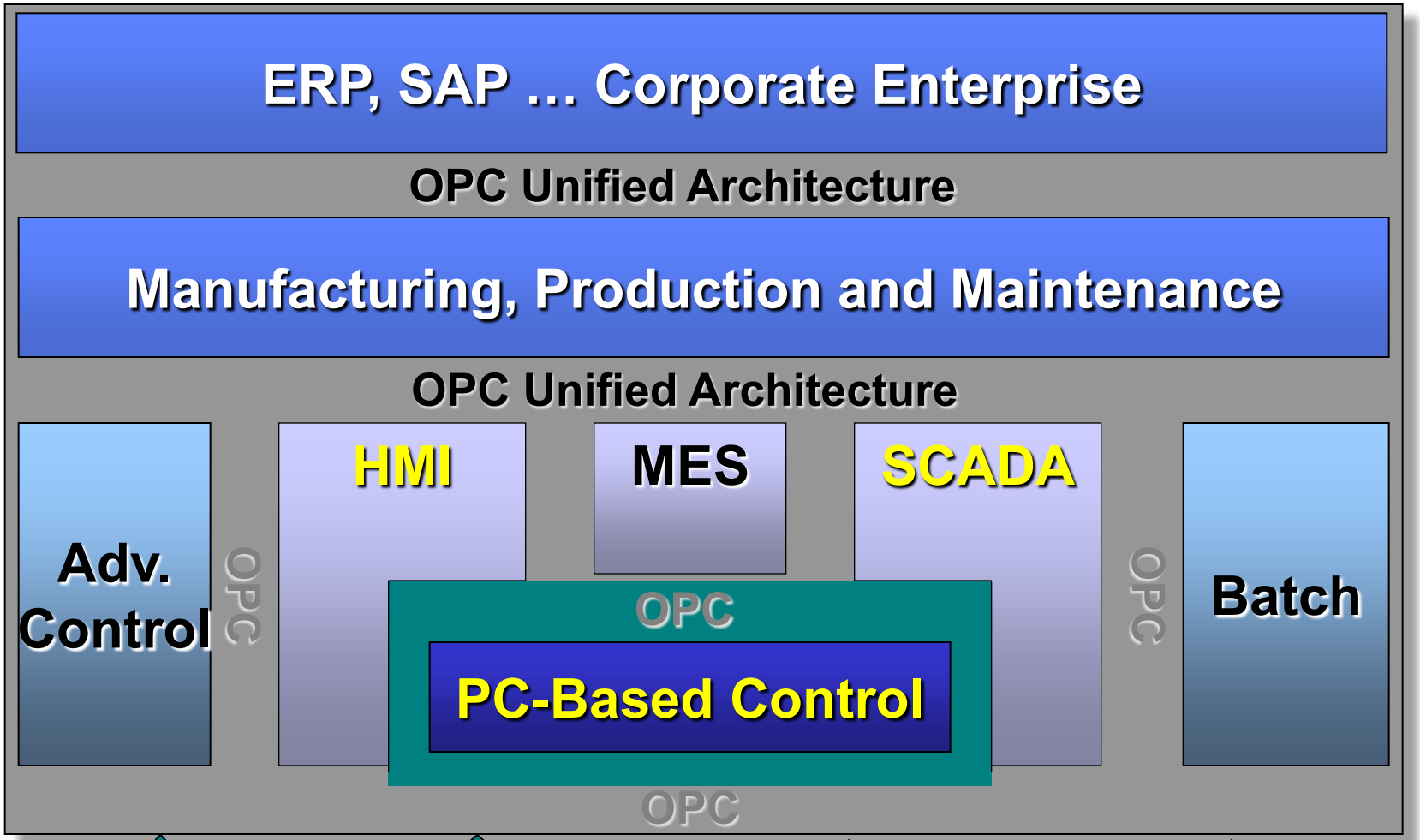
Evaluation of “classical” OPC

- Particularly applications in control networks
- DCOM fast but with limitations (Windows platform)
- Data Access (online data collection) frequently implemented
- Other specs rather rare
  - Alarms & Events (A&E)
  - Historical Data Access (HDA)
  - OPC Data eXchange (DX)
  - OPC Security
  - OPC DA XML
  - OPC Commands
  - OPC Complex Data

OPC Unified Architecture: Web-based

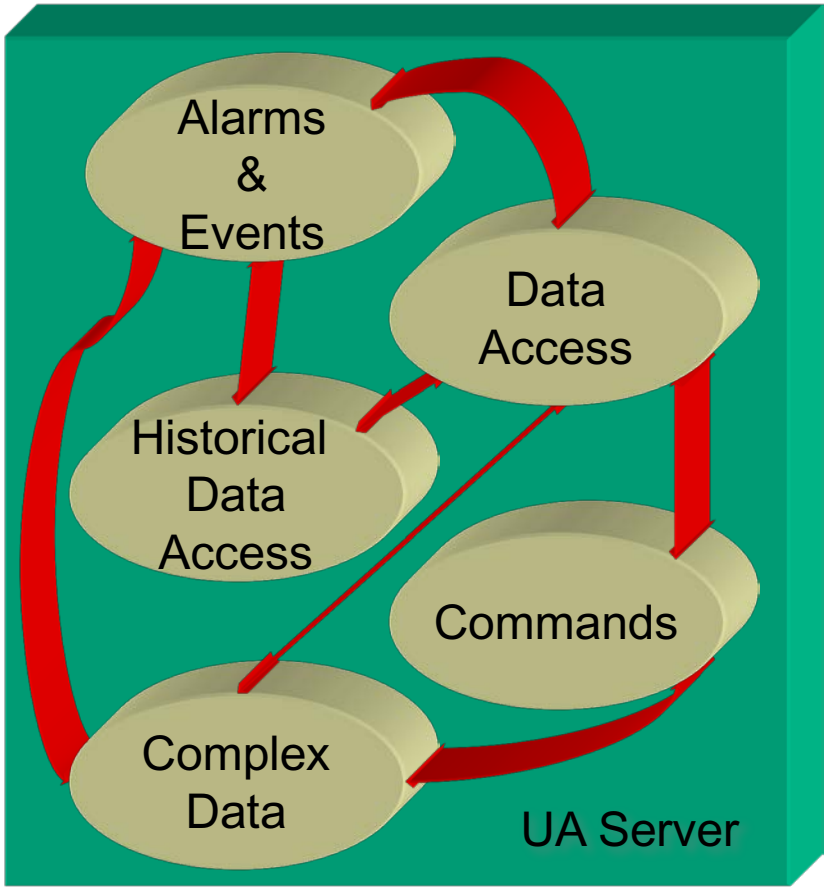
- Platform independent
- Technology independent
- More application domains accessible
- Metadata- information modelling
- No loss of success points
  - Interoperability
  - Multi-vendor
  - Performance

Extension: More application domains



Jim Luth, OPC Unified Architecture, DevCon 2006

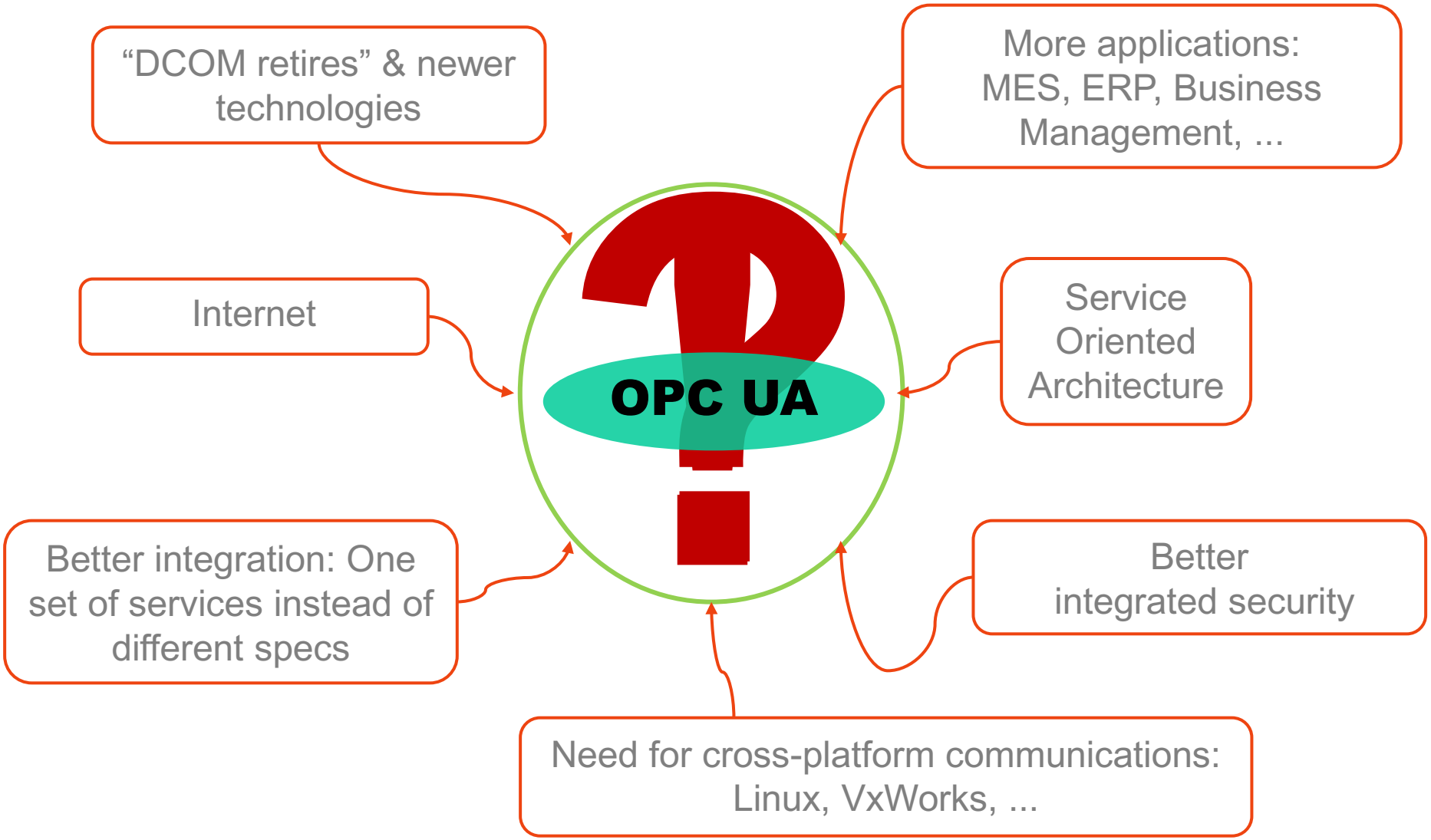
OPC interface unification



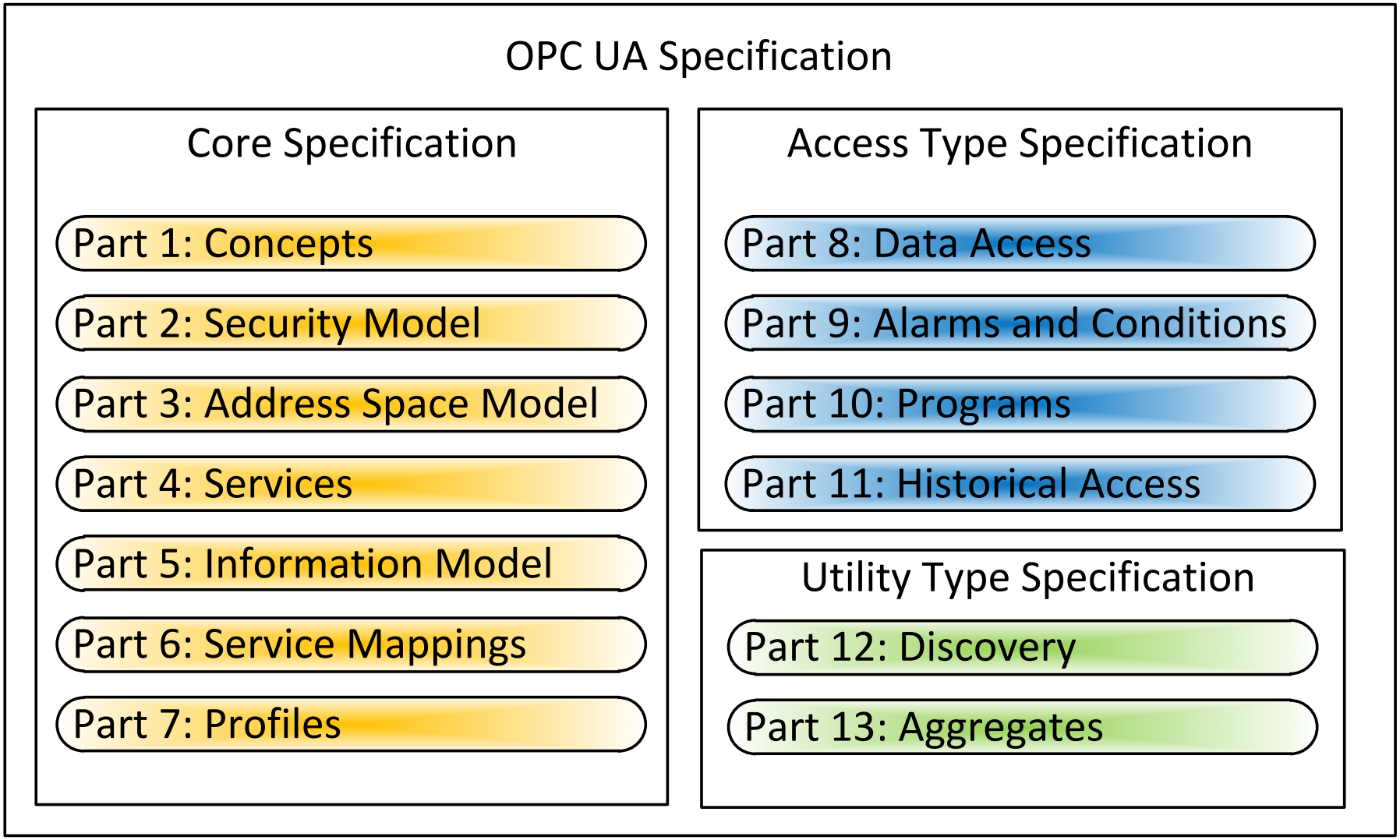
- Service Oriented Architecture
- Single set of services
  - Query, Read, Write, Subscribe
- Named/Typed relationship between nodes
- OPC UA Server
  - Functionality of existing OPC
  - Single set of services
  - More functionality

Jim Luth, OPC Unified Architecture, DevCon 2006

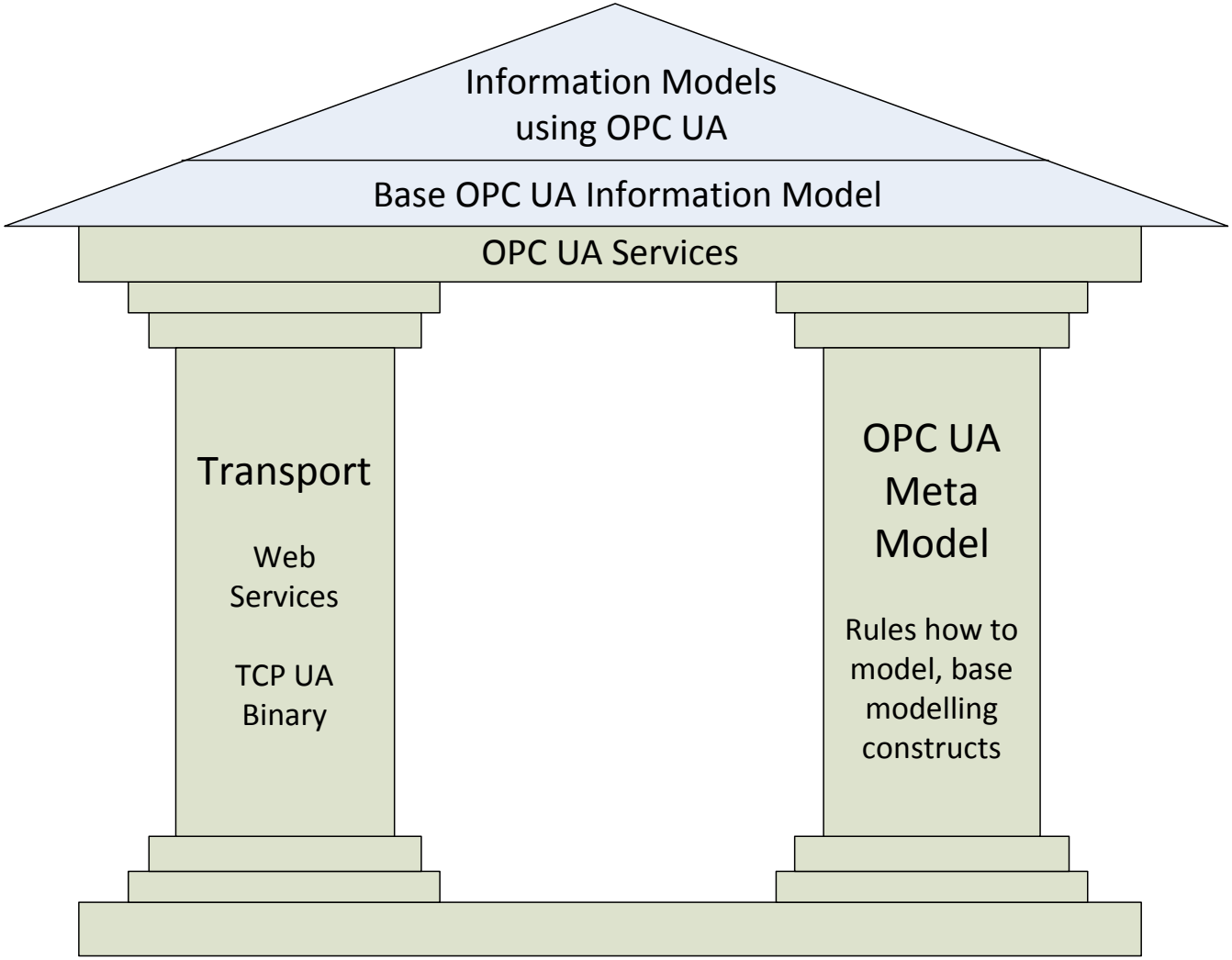
From OPC to OPC UA



OPC UA multi-part specification



OPC UA: Two pillars



Mahnke et. al., OPC Unified Architecture, 2009

INDIN' 2011
9th IEEE International Conference on
Industrial Informatics

OPC UA address space and information modelling

Overview

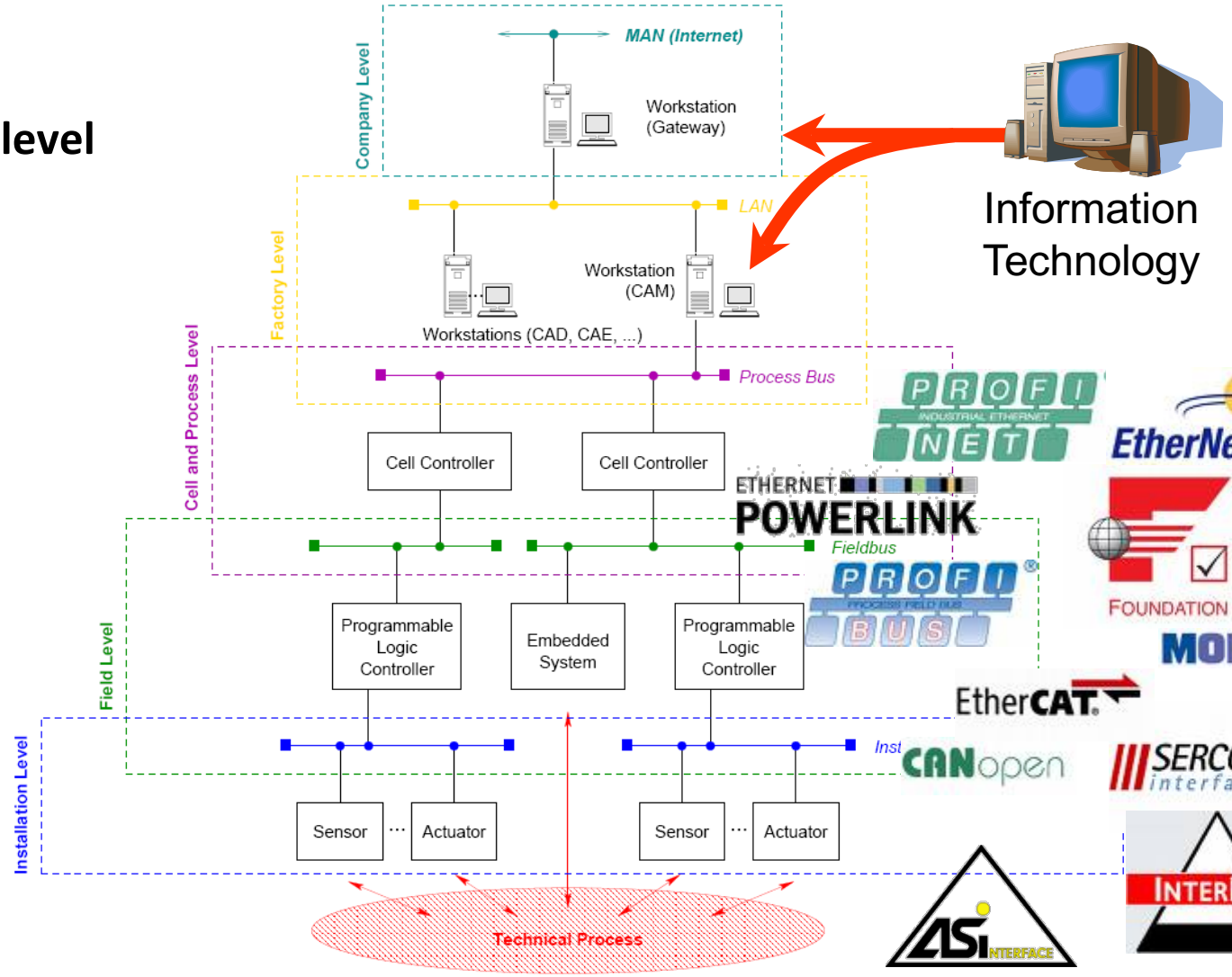
- Why information modelling?
- Information modelling in OPC UA
- OPC UA Address space
- Standard OPC UA information models
- Specifying user-defined OPC UA information models
- Use case: Building automation:
  - Information models for KNX
  - Information models for BACnet

Industrial automation

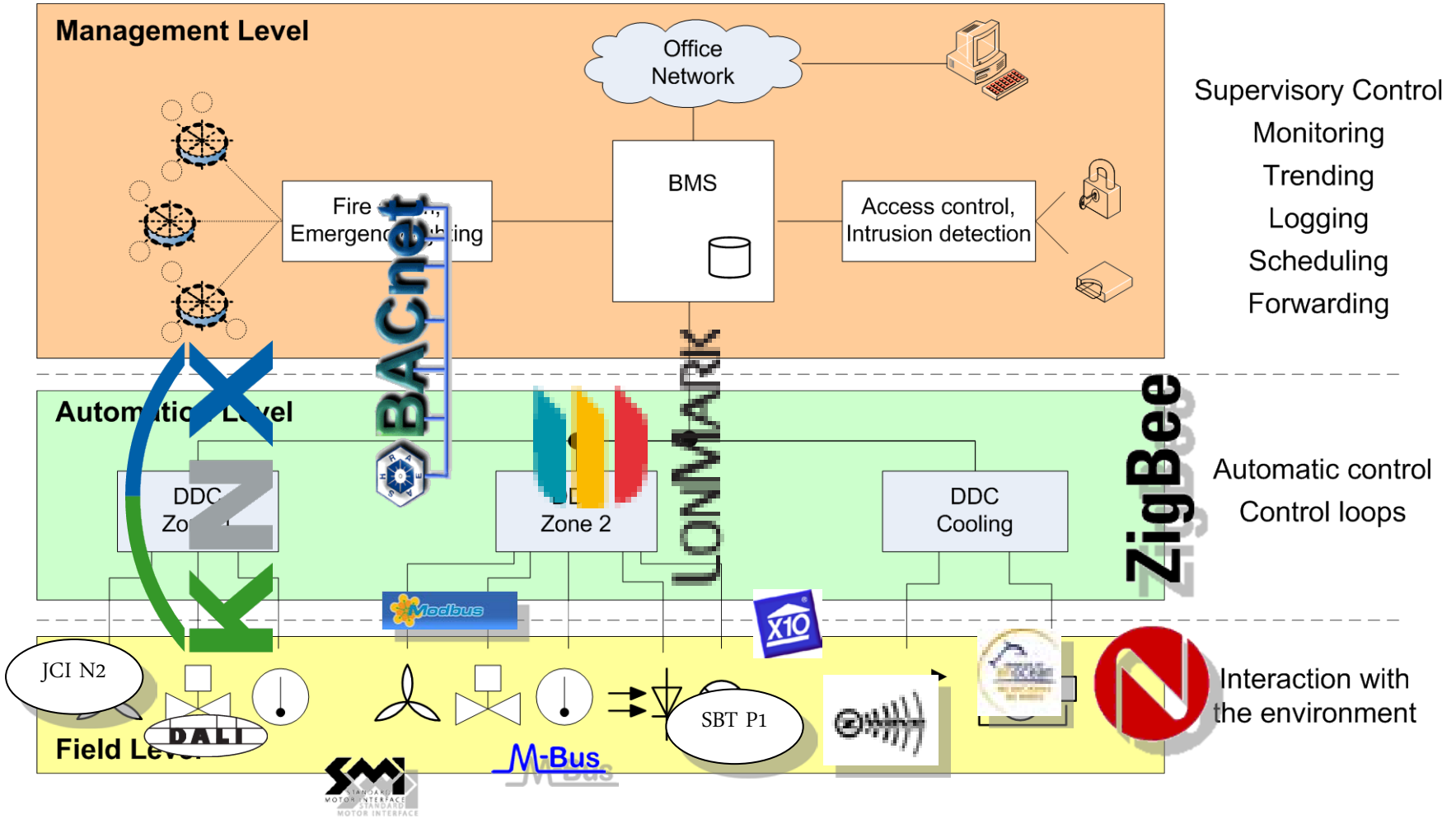
Company level

Cell level

Field level



Building automation



Why information modelling?

- Automation systems are very heterogeneous
  - → Different technologies/standards are used
- Each technology has its own concept to model/represent process data
- At the company/management level a uniform view of the entire system is required
  - Management applications require a live process image

→ Information i.e., process data has to be modelled in a technologically independent way

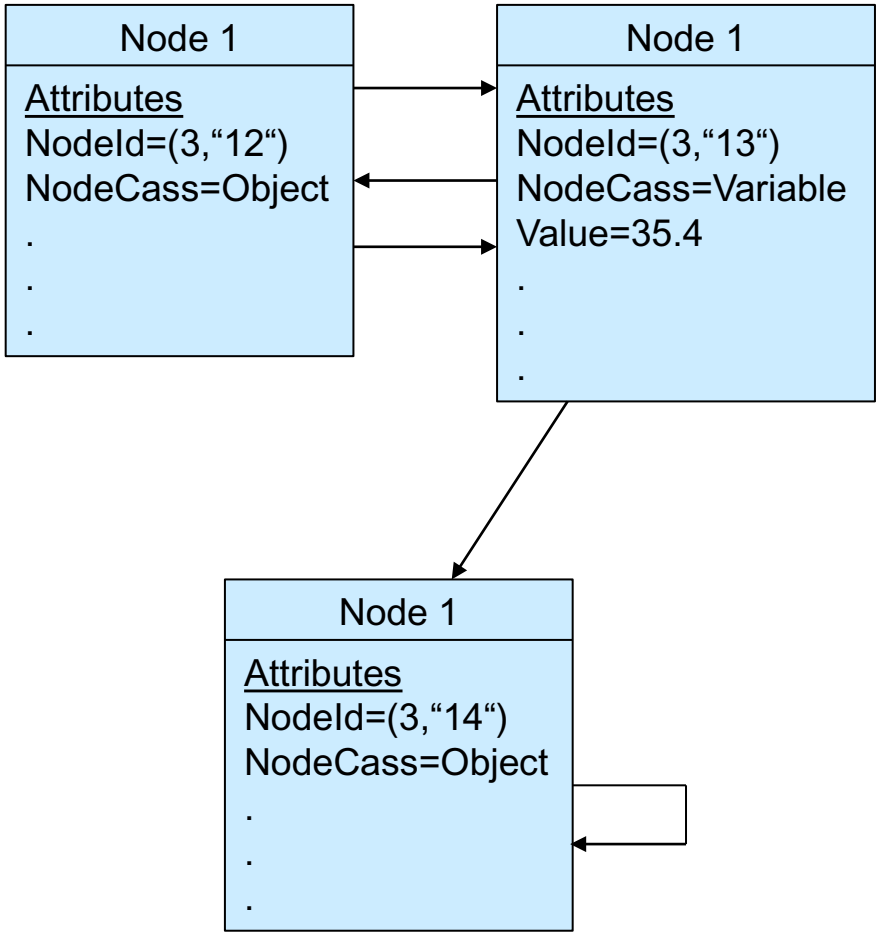
Information modelling
is a key concept for providing interoperability

Information modelling in OPC UA

- Compared to classical OPC, OPC UA provides an advanced information modelling concept
- Key features:
  - Object-oriented modelling mechanism
    - Type hierarchies and inheritance
    - A model consists of full-meshed nodes
  - The basic OPC UA information model(s) can be extended
    - Vendors and developers can define their own models on top of the built-in ones
  - Information models (including type information and hierarchies) are defined server-side
    - Clients can retrieve these models

OPC UA information modelling concept

- Information is represented as nodes
- Nodes consists of
  - Attributes
  - References
- (Process) data and its semantic (i.e., meta-data) is encapsulated by the attributes and the references of a node



Node classes

- Base NodeClass → Abstract node class
- Object NodeClass
- Variable NodeClass
- Method NodeClass

  (Object, Variable, Method NodeClass → Representing data)

- ReferenceType NodeClass
- ObjectType NodeClass
- VariableType NodeClass
- DataType NodeClass

  (ReferenceType, ObjectType, VariableType, DataType NodeClass → Type definition)

- View NodeClass → Definition of client views

Base NodeClass

- Abstract node class where all other classes are derived
- Standard attributes:

Name	Use	Data type	Description
NodeId	M	NodeId	Uniquely identifies the node
NodeClass	M	NodeClass	Specifies the node class
BrowseName	M	QualifiedName	Identifies the node during browsing
DisplayName	M	LocalizedText	Display name within user interface
Description	O	LocalizedText	Textual description
WriteMask	O	UInt32	Attributes that are writable by an OPC UA client
UserWriteMask	O	UInt32	Attributes that are writable by a user currently connected to an OPC UA server

References

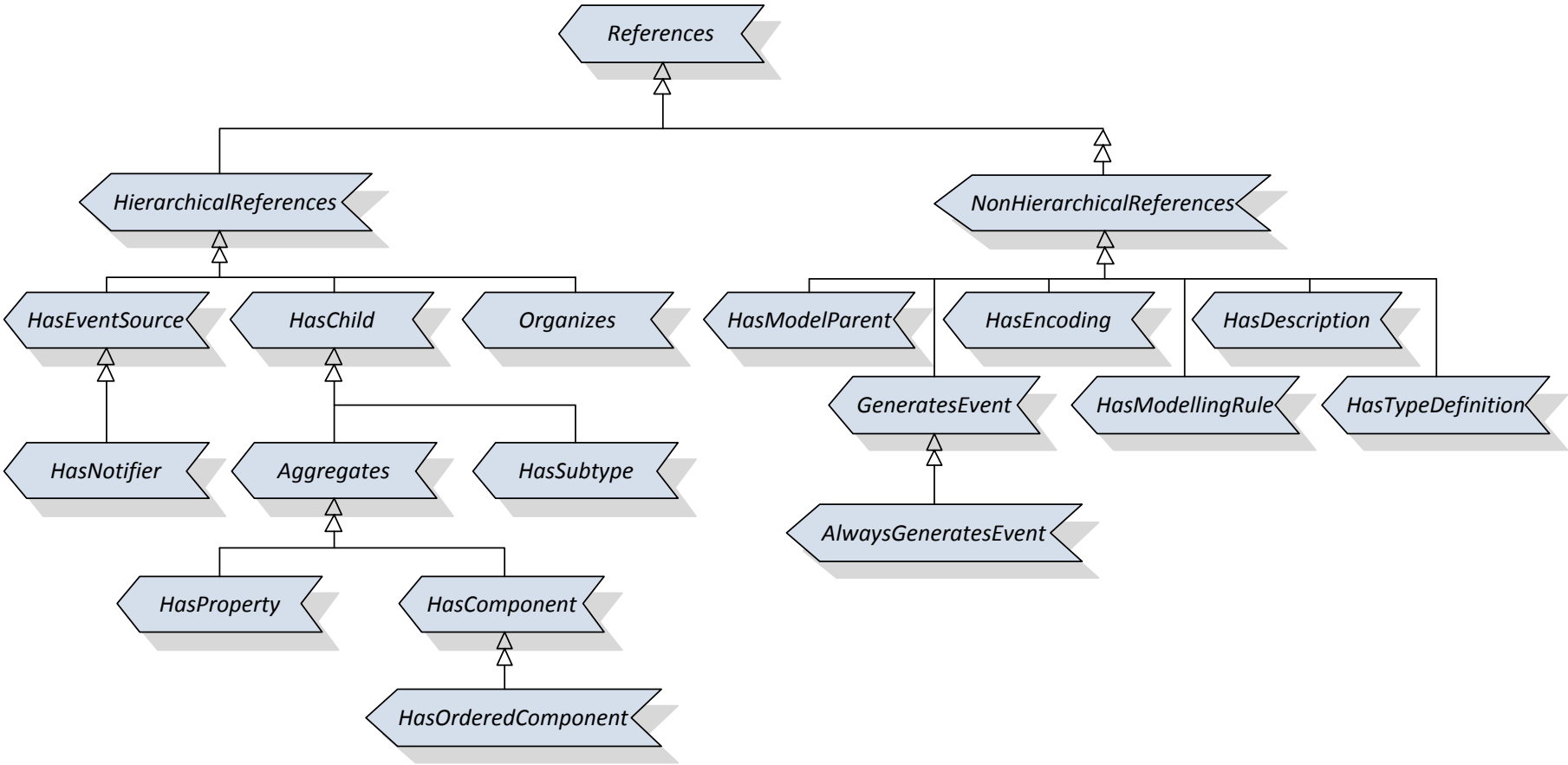
- References are used to link nodes together
- References are defined by:
  - Source node
  - Reference type
  - Target node
- Target node may be located at remote server
- References can be symmetric or asymmetric
- References can be hierarchical or non-hierarchical

ReferenceType NodeClass

- Each reference is of a certain type
- Name of reference type adds semantic to nodes and their relations
- Additional attributes:

Name	Use	Data type	Description
IsAbstract	M	Boolean	Indicates whether type can be used directly
Symmetric	M	Boolean	Specifies whether reference is symmetric
InverseName	O	LocalizedText	The name of the inverse reference if it is asymmetric

Standard reference types



Objects

- Objects are used to structure the address space
- Similar to other object-oriented concepts
- Objects may contain
  - Variables
  - Methods
  - Other objects
- Embedded nodes are linked to the object by using *HasComponent* (or a subtype of it) or *HasProperty* references
- Objects do not contain real process data
- Additional attributes:

Name	Use	Data type	Description
EventNotifier	M	Byte	Identifies whether it can be subscribed to events and whether the history of events is accessible

Variables

- Variables are used to model data (i.e., data points like process data, set points, configuration parameters, …)
- OPC UA clients can read, write these data and subscribe to changes of the data
- Additional attributes:

Name	Use	Data type	Description
Value	M	NodeId	Actual value
DataType	M	NodeId	NodeId of the DataType node of the value
ValueRank	M	Int32	Array dimension
ArrayDimensions	O	UInt32[]	Size of each array dimension
AccessLevel	M	Byte	Standard access rights
UserAccessLevel	M	Byte	User access rights
MinimumSamplingInterval	O	Duration	Minimum sampling of server
Historizing	M	Boolean	Indicates whether history is collected

Data variables and properties

- OPC UA defines two kinds of variables
  - Data variables
    - Represent the data of an object
    - Changes are frequent (e.g., provided by a sensor)
    - Can be complex (may contain sub-variables)
  - Properties
    - Represent meta-data
    - Do not change often (e.g., configuration parameter)
    - Cannot be complex
- Separation is not always clear
- Both kinds are of type Variable *NodeClass*

Methods

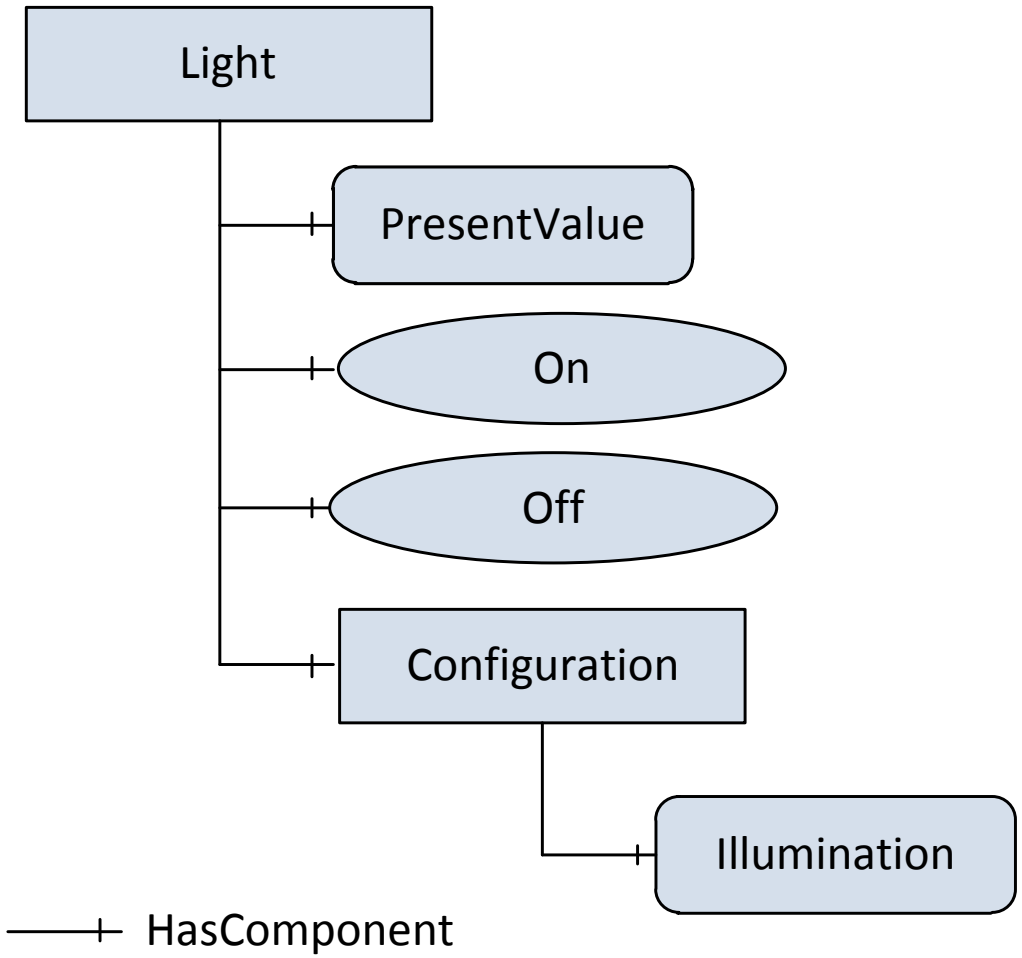
- Used to invoke (fast) actions within an OPC UA server
- Additional attributes:

Name	Use	Data type	Description
Executable	M	Byte	Uniquely identifies the node
UserExecutable	M	Byte	Specifies the node class

- Input and output arguments are passed as properties

Name	Use	Data type	Description
InputArguments	O	Argument[]	Array of input arguments
OutputArguments	O	Argument[]	Array of output arguments

Objects, variables, methods: An example



Type information and hierarchy

- Main feature of OPC UA‘s modelling concept is the definition of types
- Inheritance can be used to derive subtypes of already existing types
- Types can be defined for:
  - Objects
  - Variables
  - Data types
  - References

ObjectType NodeClass

- Used to define the structure and semantics of objects
- Object types can be simple (contain no sub nodes) or complex (contain further objects, variables, methods)
- Additional attributes:

Name	Use	Data type	Description
IsAbstract	M	Boolean	Indicates whether type can directly be used

- Inheritances of object types
  - Attributes are inherited but may be overridden
  - Derivation is done by using *HasSubtype* reference

VariableType NodeClass

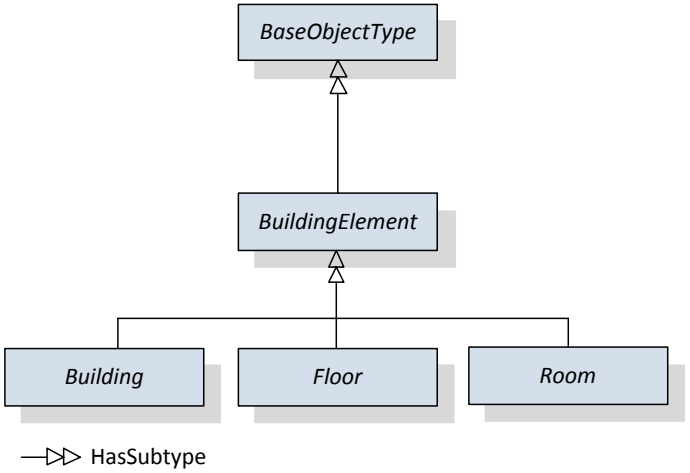
- Defines structure and semantics of variables
- Object types can be simple (contain no sub variables) or complex (contain further variables)
- Additional attributes:

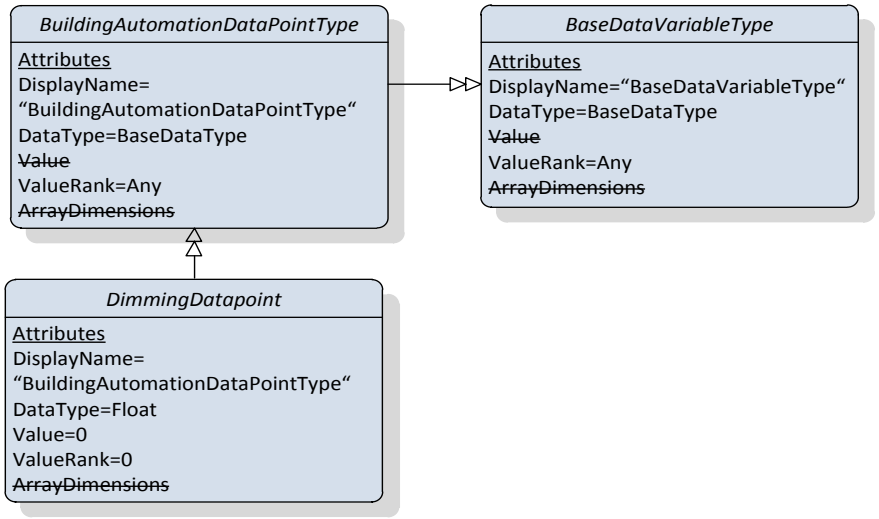
Name	Use	Data type	Description
Value	O	Variable	Default value for instance declarations
DataType	M	NodeId	NodeId of data type of Value
ValueRank	M	Int32	Array dimension
ArrayDimensions	O	UInt32[]	Size of each array dimension
IsAbstract	M	Boolean	Indicates whether type can be used directly

- Inheritances of variable types
  - Attributes are inherited but may be overridden (special rules apply to *DataType*, *ValueRank*, and *ArrayDimensions*)
  - Derivation is done by using *HasSubtype* reference

Simple object and variables types

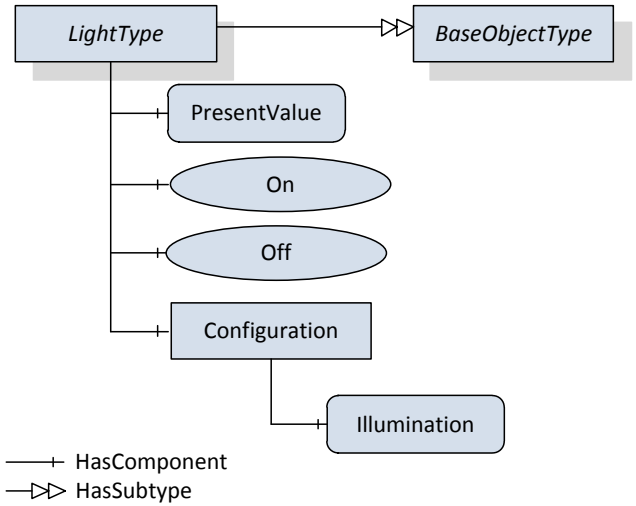
- Must not embed sub-objects (for object types only), variables or methods
- Default values for *Description* and *DisplayName* can be provided
- Semantics is added by the name of the type
- Examples:

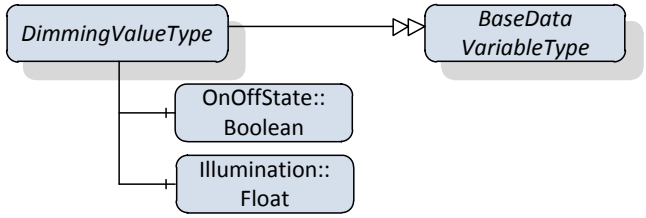




Complex object and variable types

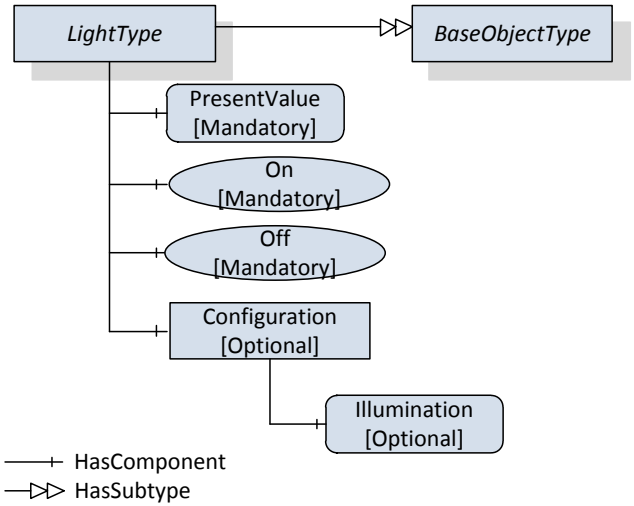
- Embed sub-objects (object types only), variables or methods and thus define a complex data structure
- Default values can also be provided for the sub-nodes
- Semantics is added by the name of the type AND by the structure of the type
- Examples:





Modelling rules

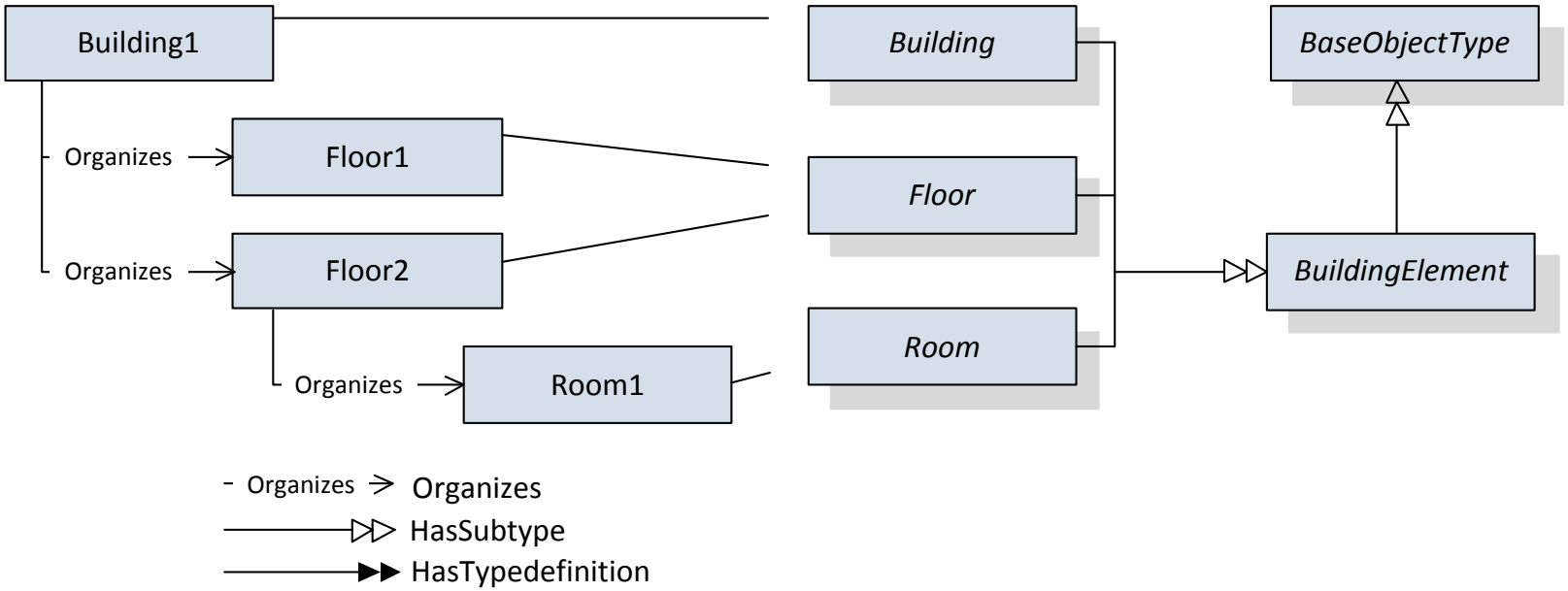
- Modelling rules define constraints on instances of complex objects and variables
- Standard modelling rules:
  - *Mandatory*: sub-node must be present in instances
  - *Optional*: sub-node does not need to be present
  - *Constraint*: other constraints apply to instances
    - E.g.: *ExposesItsArrays* – each array entry must be exposed as sub-variable



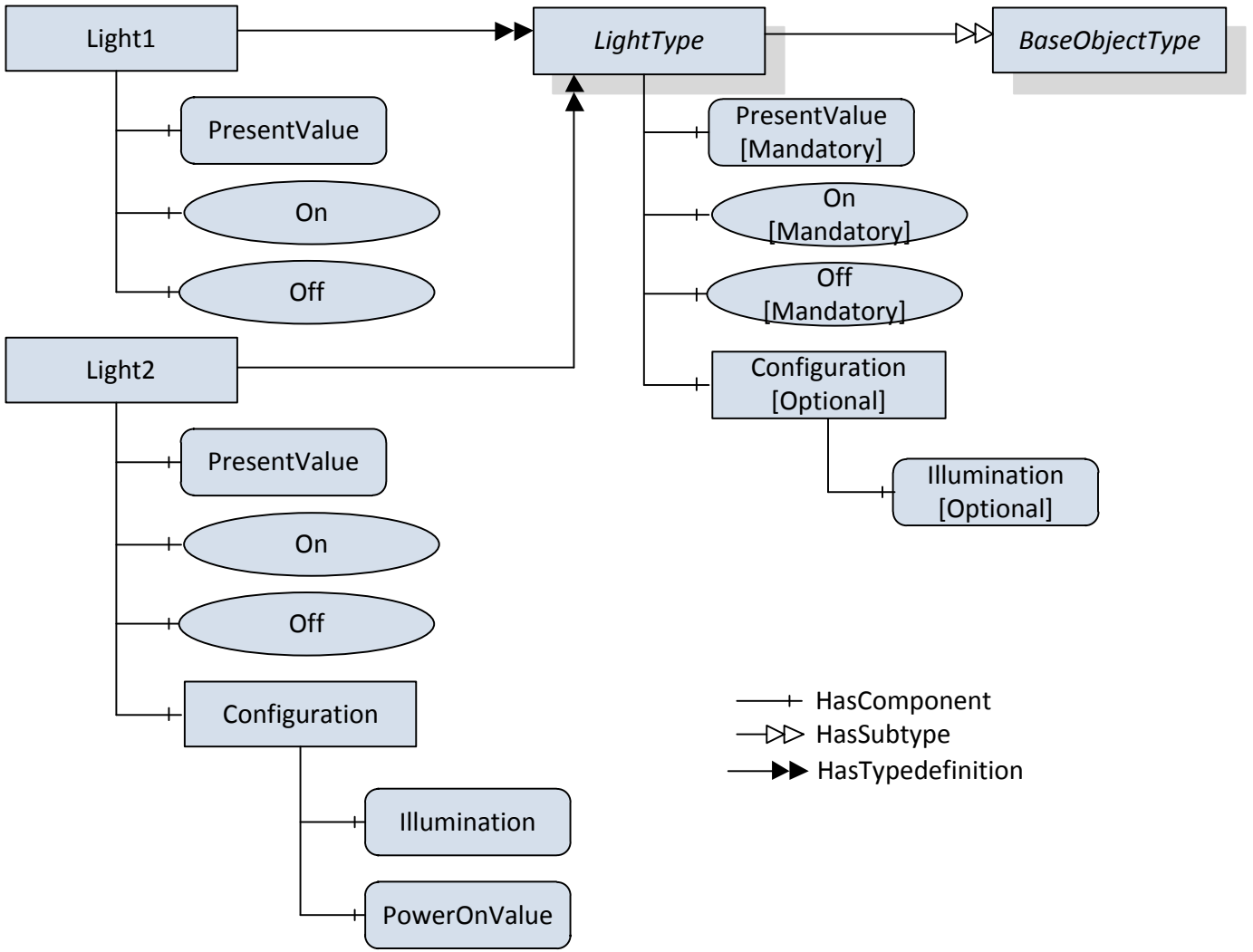
Instances of object and variables types

- Instances inherit all default values of the attributes of the type definition (except *NodeId* and *NodeClass*)
- In case of complex types the structure is also inherited according to the defined modelling rules
  - Sub-nodes without modelling rule are NOT inherited
  - Sub-nodes of super-types are also inherited
- Type definition is specified by *HasTypedefinition* references
- Multiple inheritance is not allowed

Instances of simple types



Instances of complex types

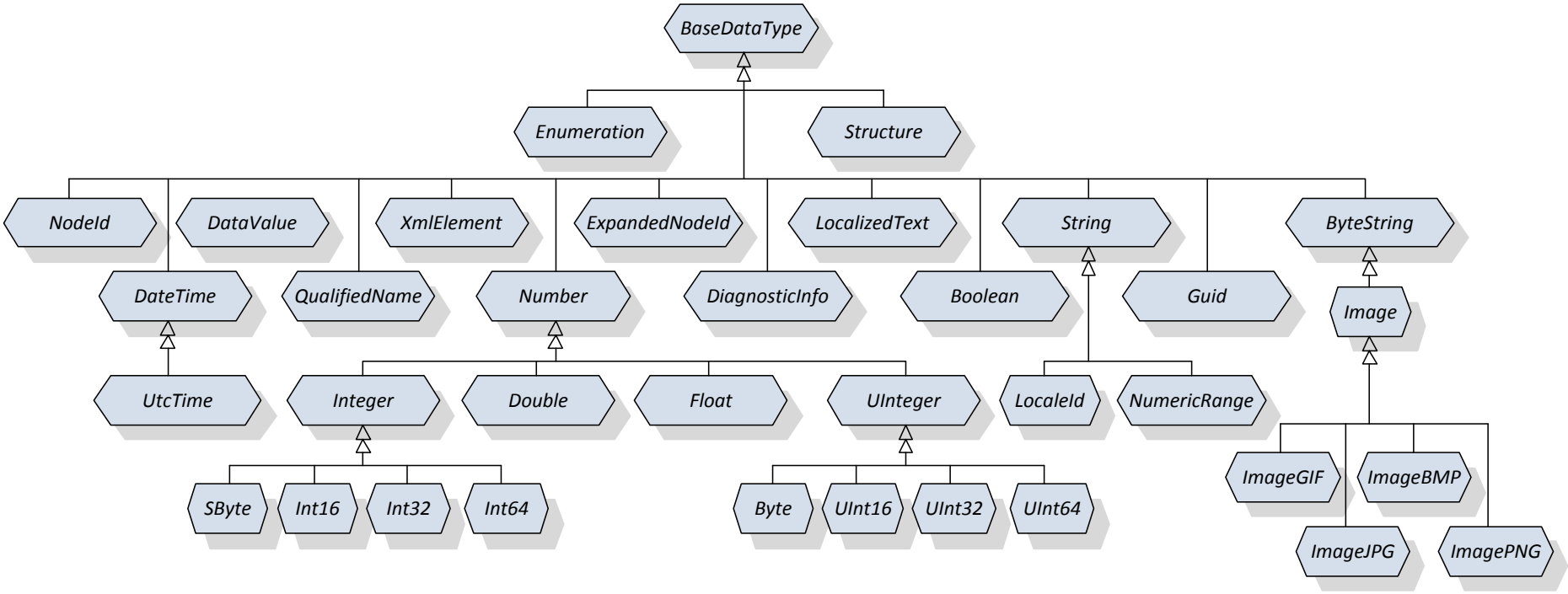


DataType NodeClass

- Value attribute of variables need a dedicated data type
  - All other attributes have a fixed data type
- Data types are represented as nodes of type *DataType NodeClass*
- Different kinds of data types
  - Built-in data types
  - Simple data types
  - Enumeration data types
  - Structured data types
- Additional attributes:

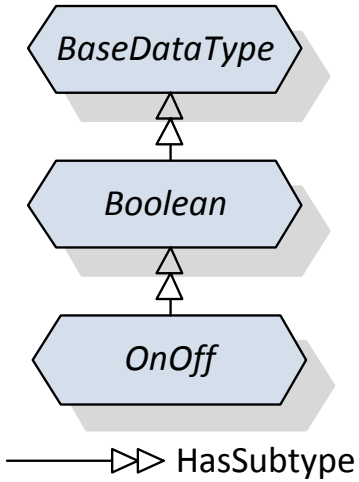
Name	Use	Data type	Description
IsAbstract	M	Boolean	Indicates whether type can directly be used

Built-in data and simple data types



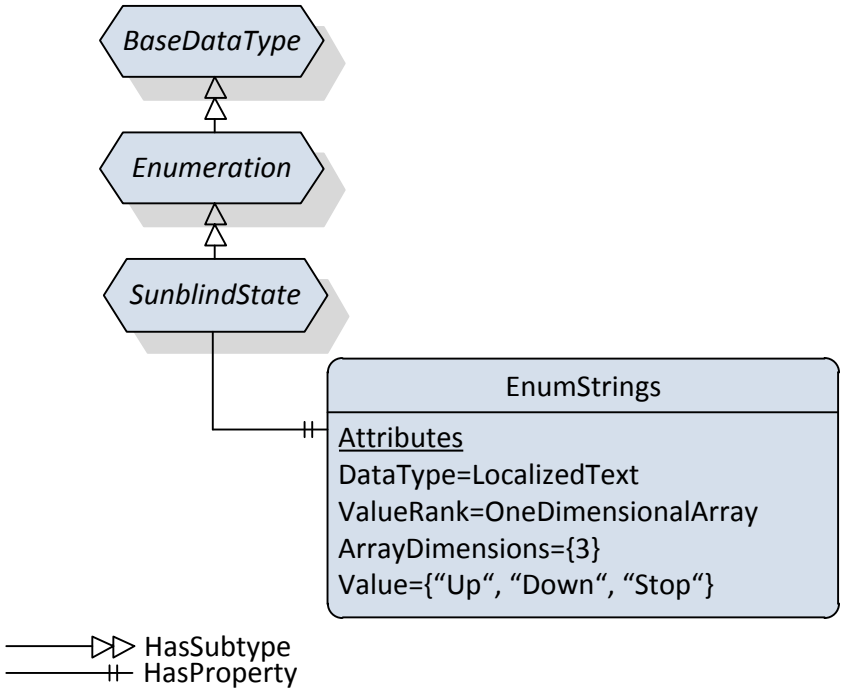
Simple data types

- Simple data types are derived from built-in ones
- Semantics is only added by the given name
- Example:



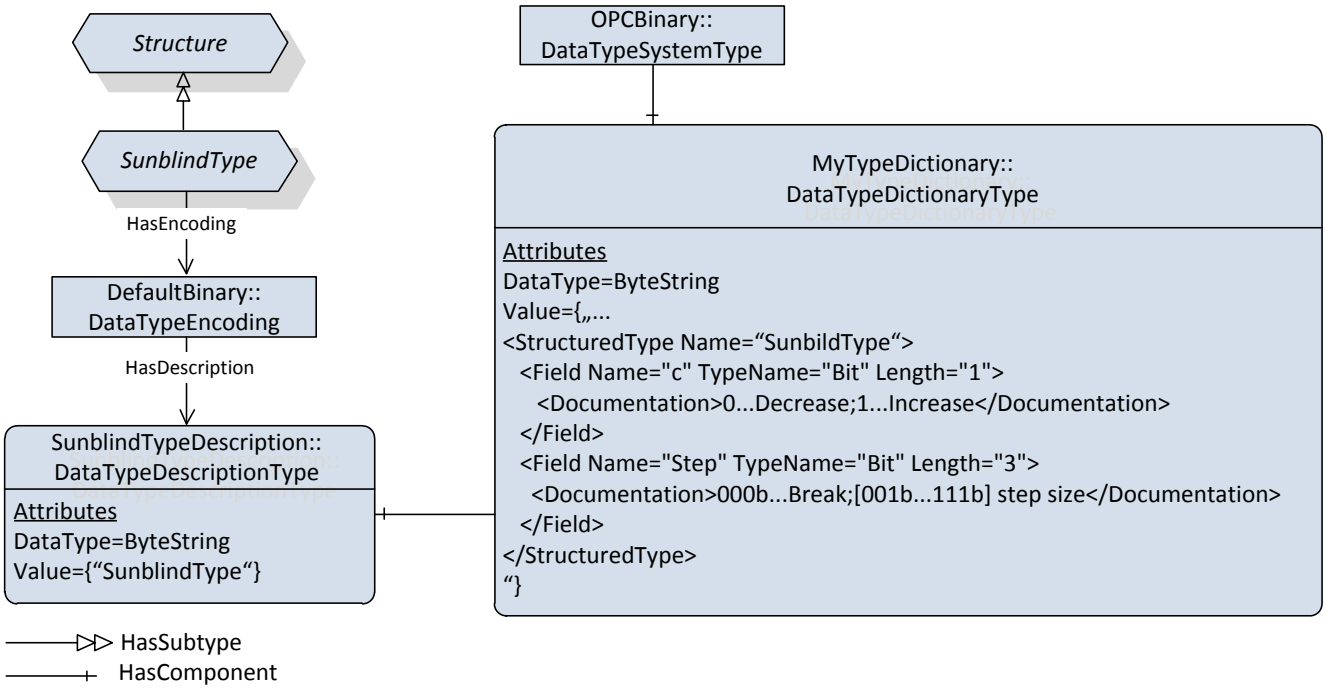
Enumeration

- Enumerations are derived from the abstract data type Enumeration
- Enumeration values are specified by property “EnumStrings“



Structured data types

- Can be used to define any form of data structure (e.g., concatenated data types)
- Encoding (XML or binary) has to be specified within the address space of server
  - Client is able to request the encoding
- Structure itself is specified as XML



View NodeClass

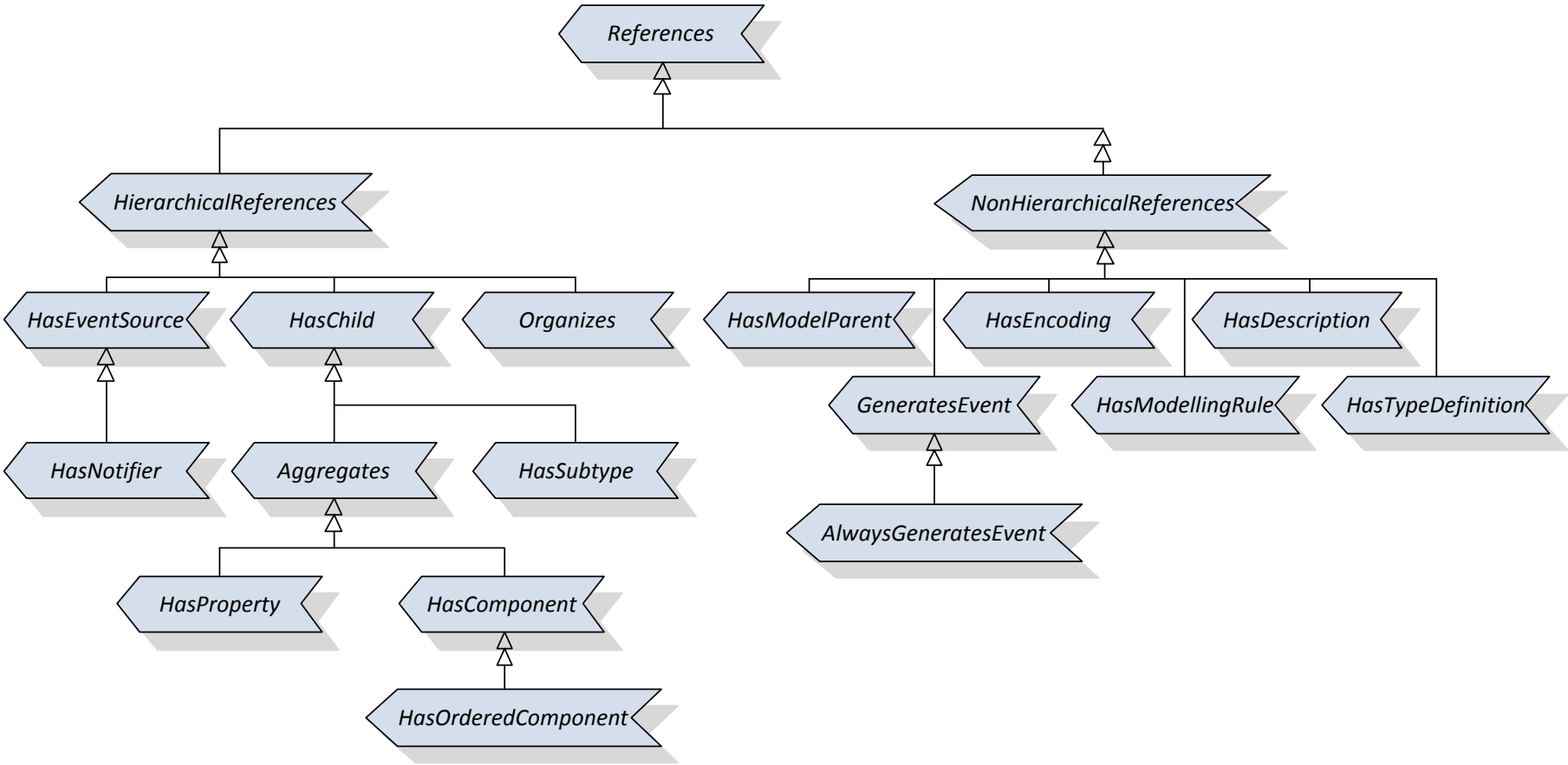
- Views are used to restrict the address space of a server for dedicated users/applications
- Useful for large address spaces
- Additional attributes:

Name	Use	Data type	Description
ContainsNoLoops	O	Boolean	View does not contain loops
EventNotifier	O	Byte	Identifies whether it can be subscribed to events and whether the history of events is accessible

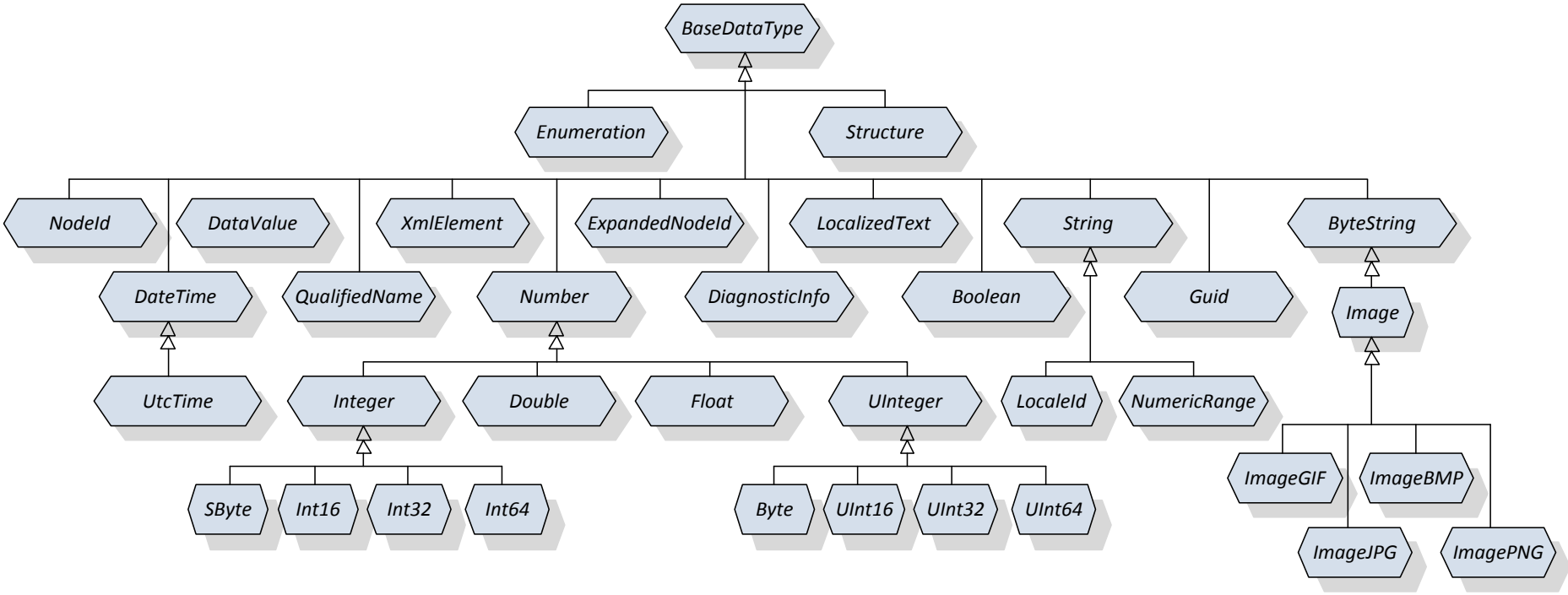
Base OPC UA information model

- Part 5 of OPC UA specification defines a standard information model for general use
- Consists of:
  - Standard reference types
  - Standard data types
  - Standard object and variable types
  - Standard objects and variables

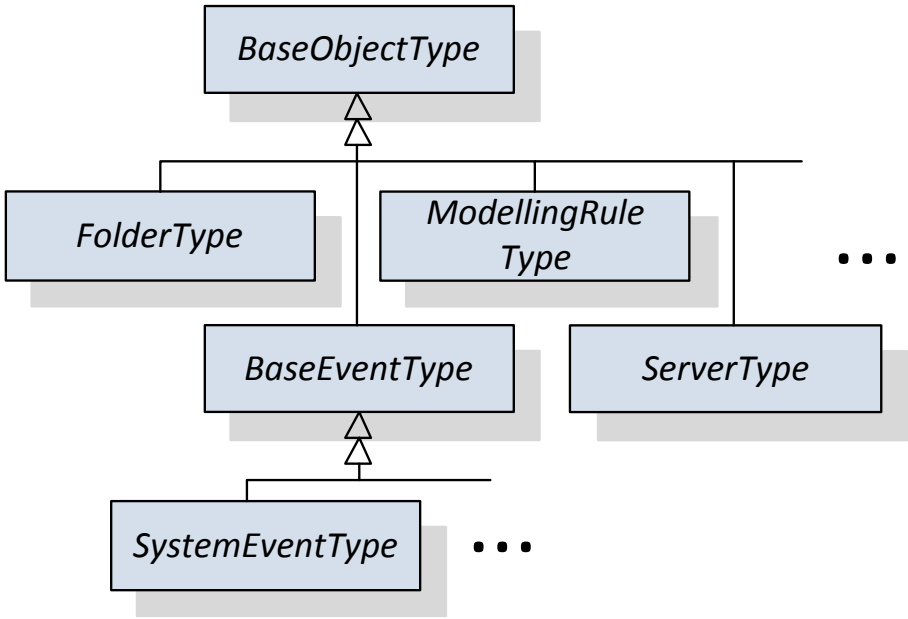
Standard reference types

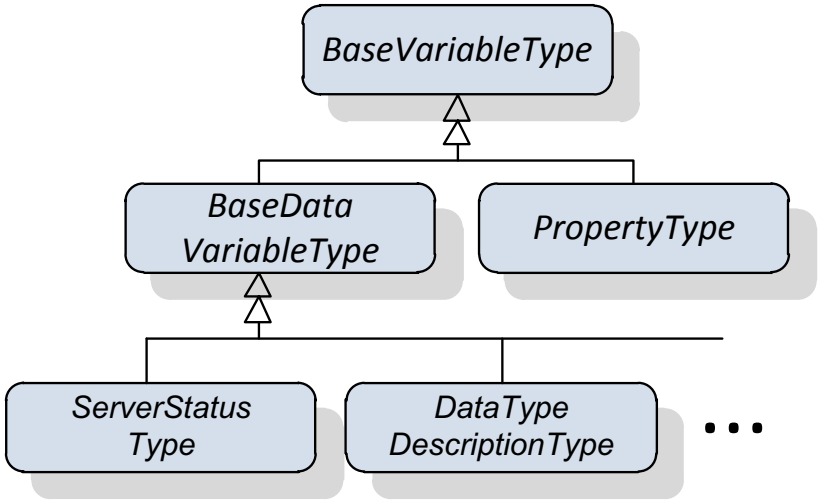


Standard data types

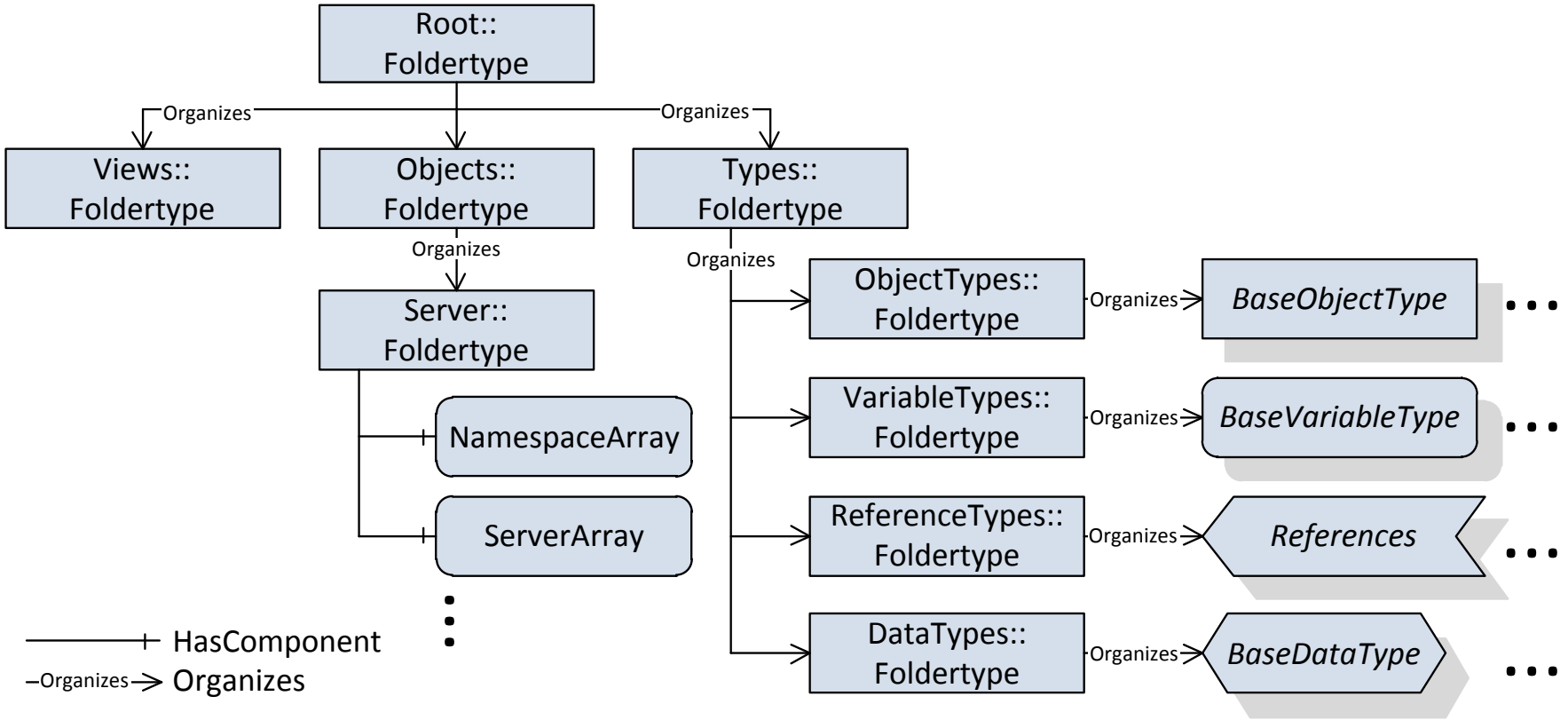


Standard object and variable types

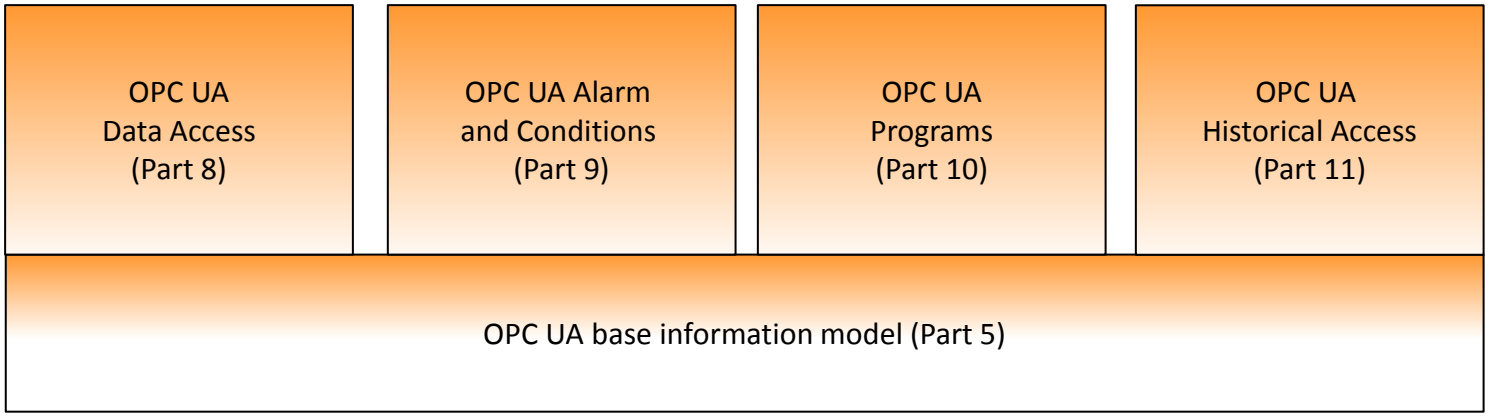




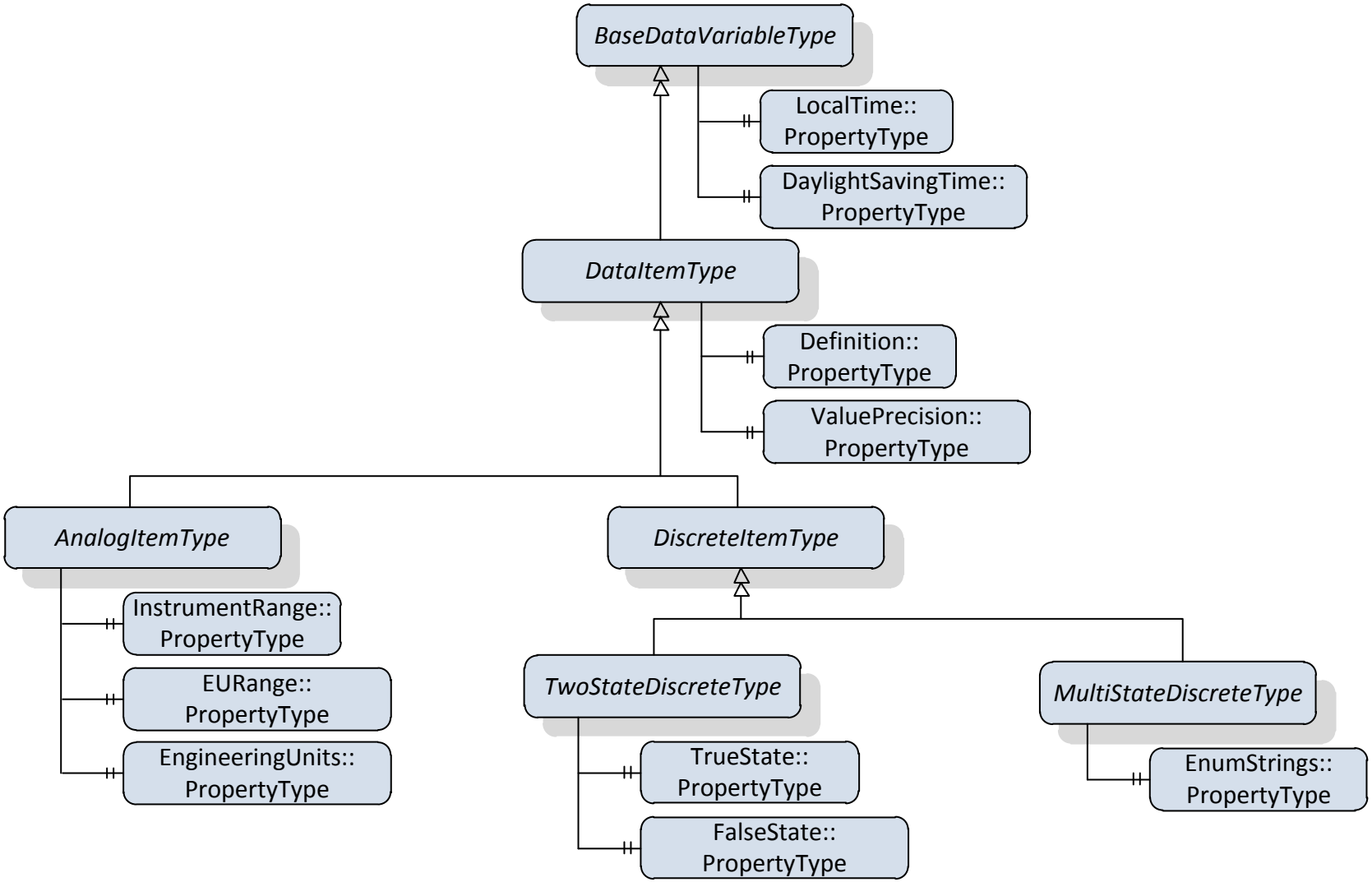
Standard objects and variables



Further standard information models

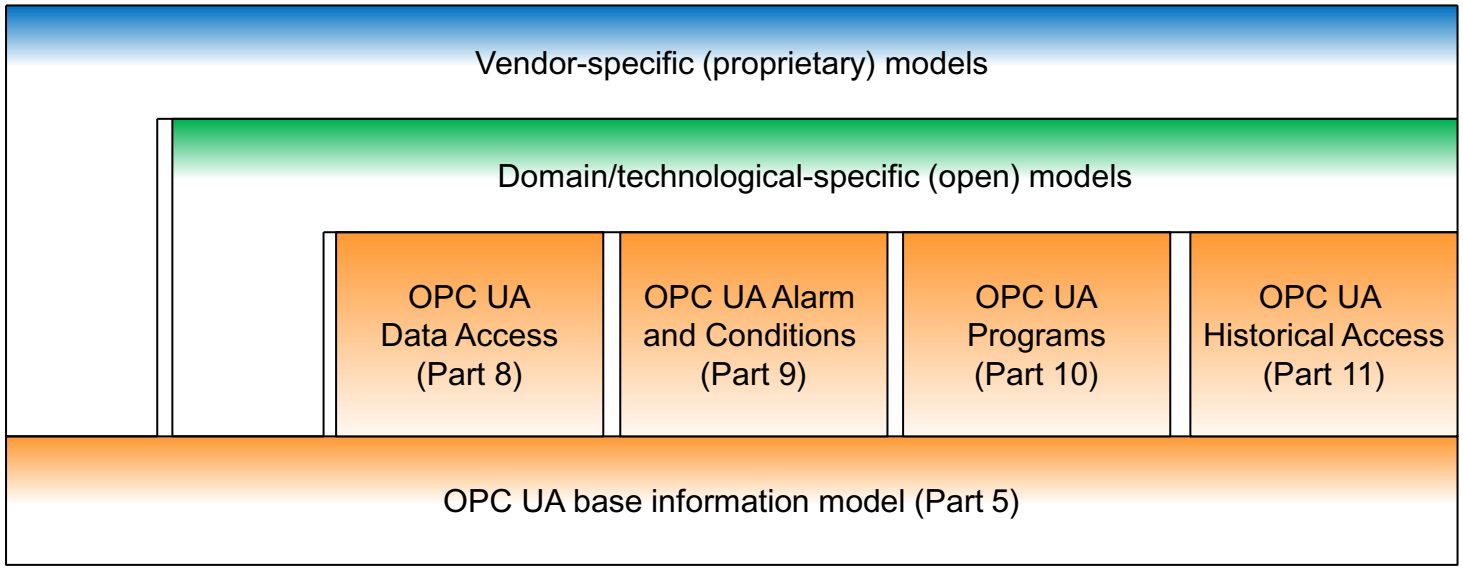


OPC UA data access



Extensibility of OPC UA models

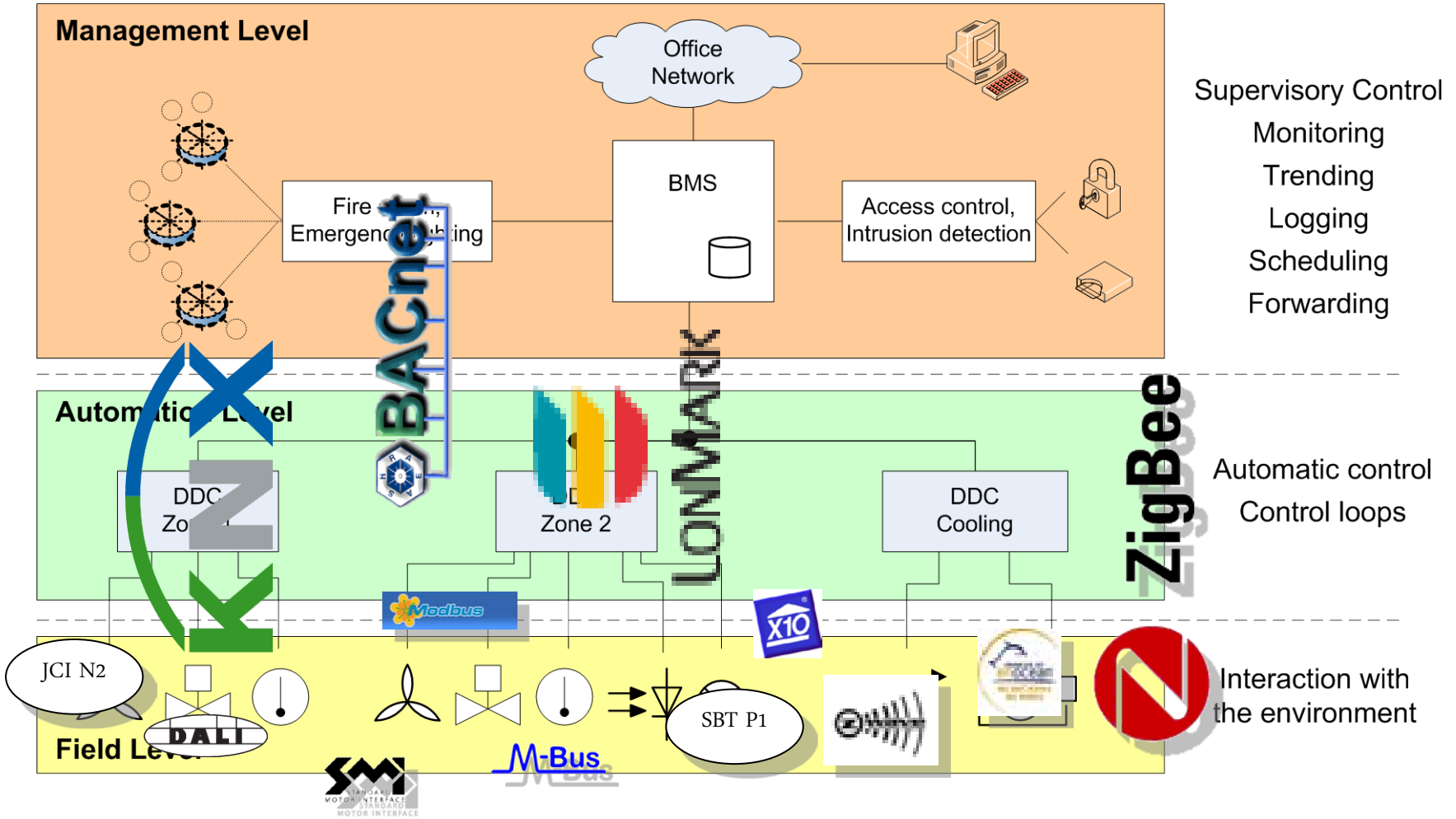
- OPC UA defines base information models
- Can be extended by
  - Domain/technological-specific (open) models (e.g., model of network technology X)
  - Vendor-specific (proprietary) models (e.g., company X's model)



INDIN' 2011
9th IEEE International Conference on
Industrial Informatics

Information modelling:
Examples for building automation systems

Building automation

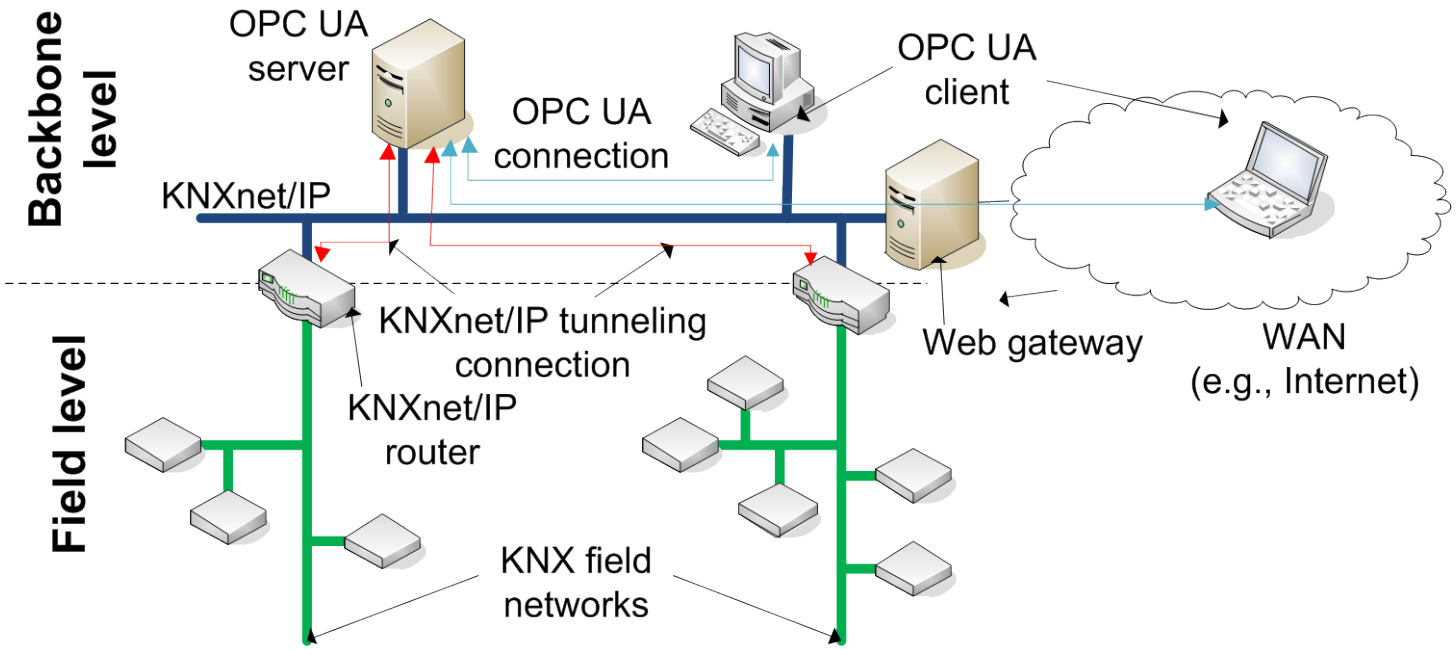


Examples

- Example 1:
  - KNX information model for OPC UA
- Example 2:
  - BACnet information model for OPC UA

OPC UA in KNX

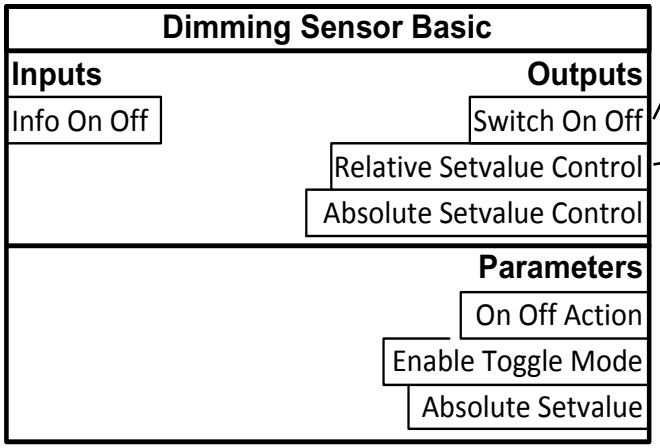
- Integration at management level to provide interoperability



→ **Mapping of KNX interworking model to OPC UA**

KNX interworking model

- Functional blocks
- Data points
- Data point types

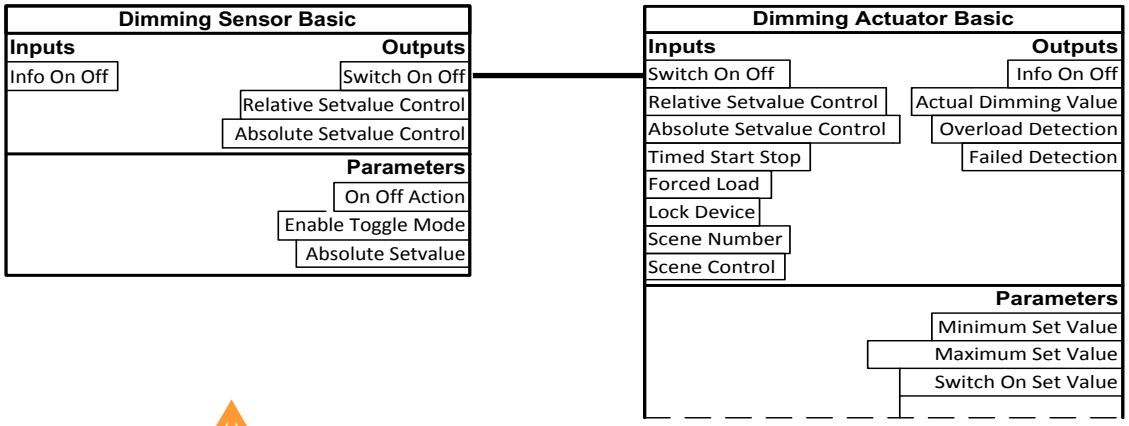


DPT_Switch	
Format:	1 bit: B_1
octet nr	1
field names	b
encoding	B
Range:	b={0,1}
Unit:	None
Resolution:	Not applicable
Encoding:	b: 0 = Off 1 = On

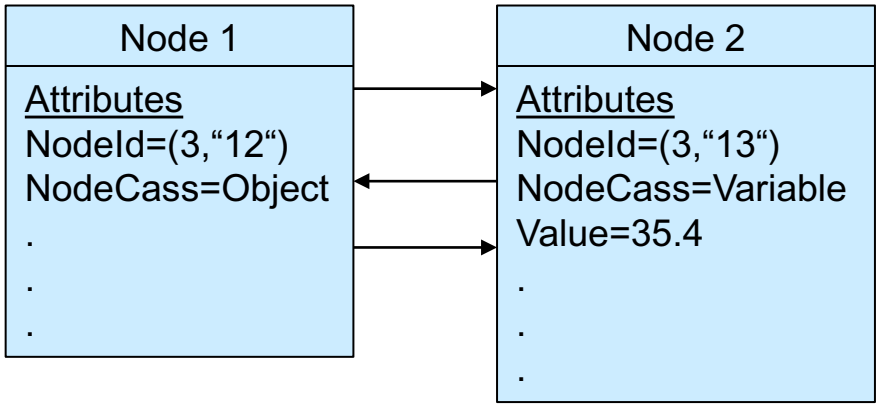
DPT_Control_Dimming	
Format:	4 bit: B_1U_3
octet nr	1
field names	c Step
encoding	B U U U
Range:	c={0,1} Step=[000b...111b]
Unit:	None
Resolution:	Not applicable
Encoding:	c: 0 = Decrease 1 = Increase Step: 001b...111b: Step 000b: Break

KNX in OPC UA

KNX interworking model

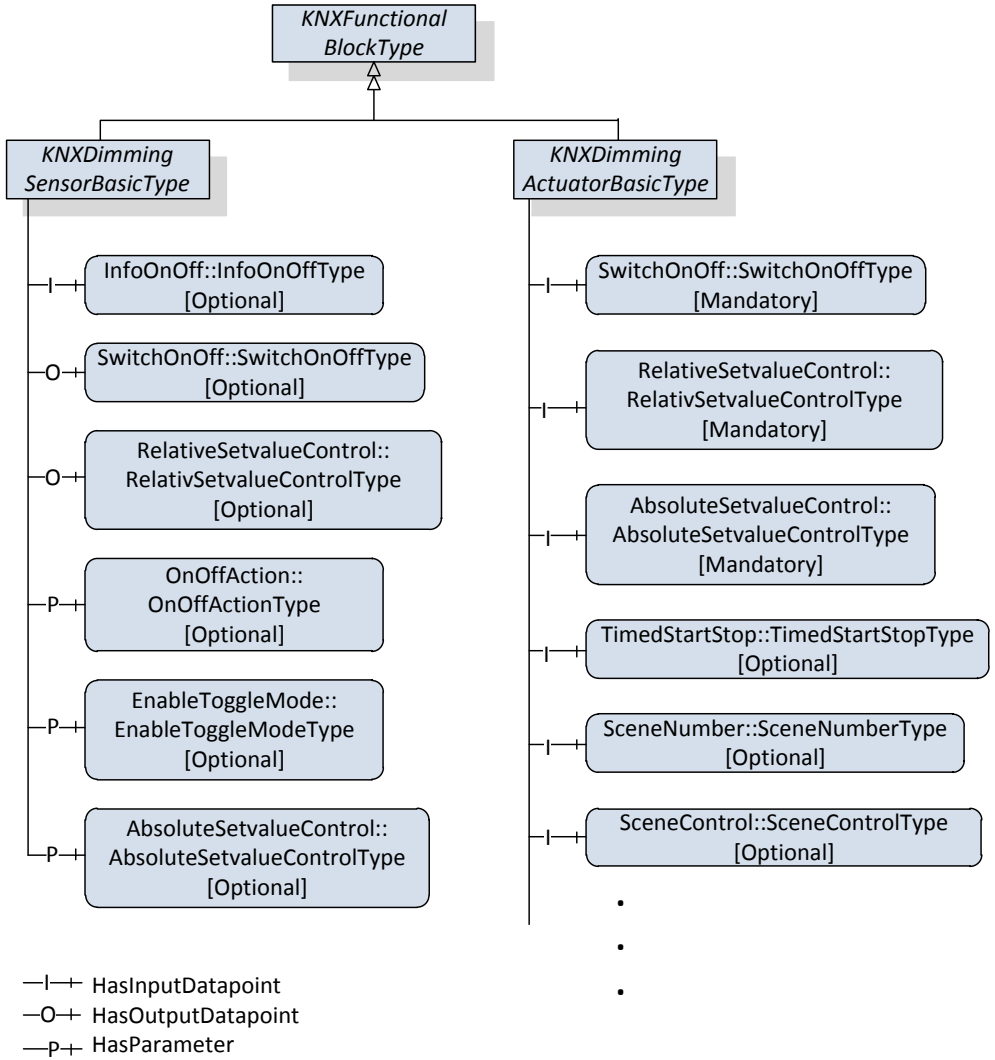


OPC information model



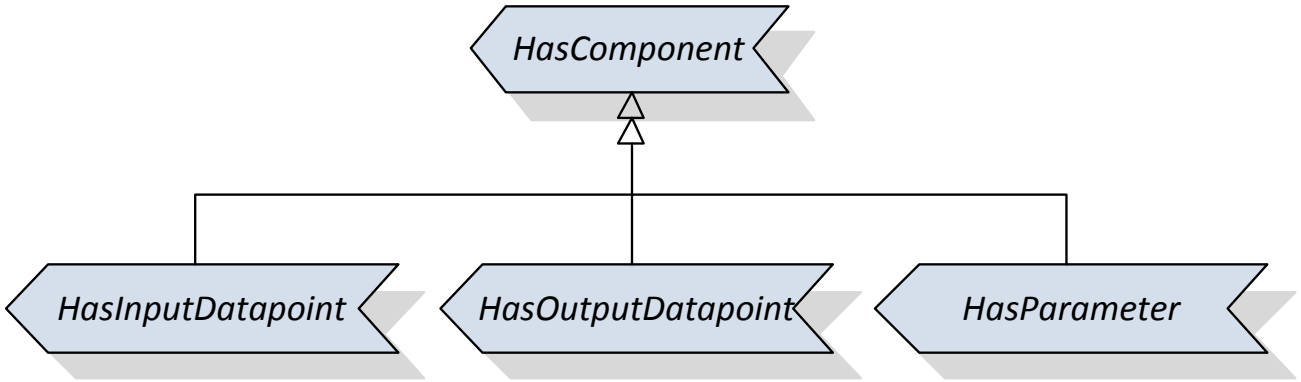
KNX functional blocks in OPC UA

- KNX functional blocks are modelled as complex objects
- Data points are represented as sub variables



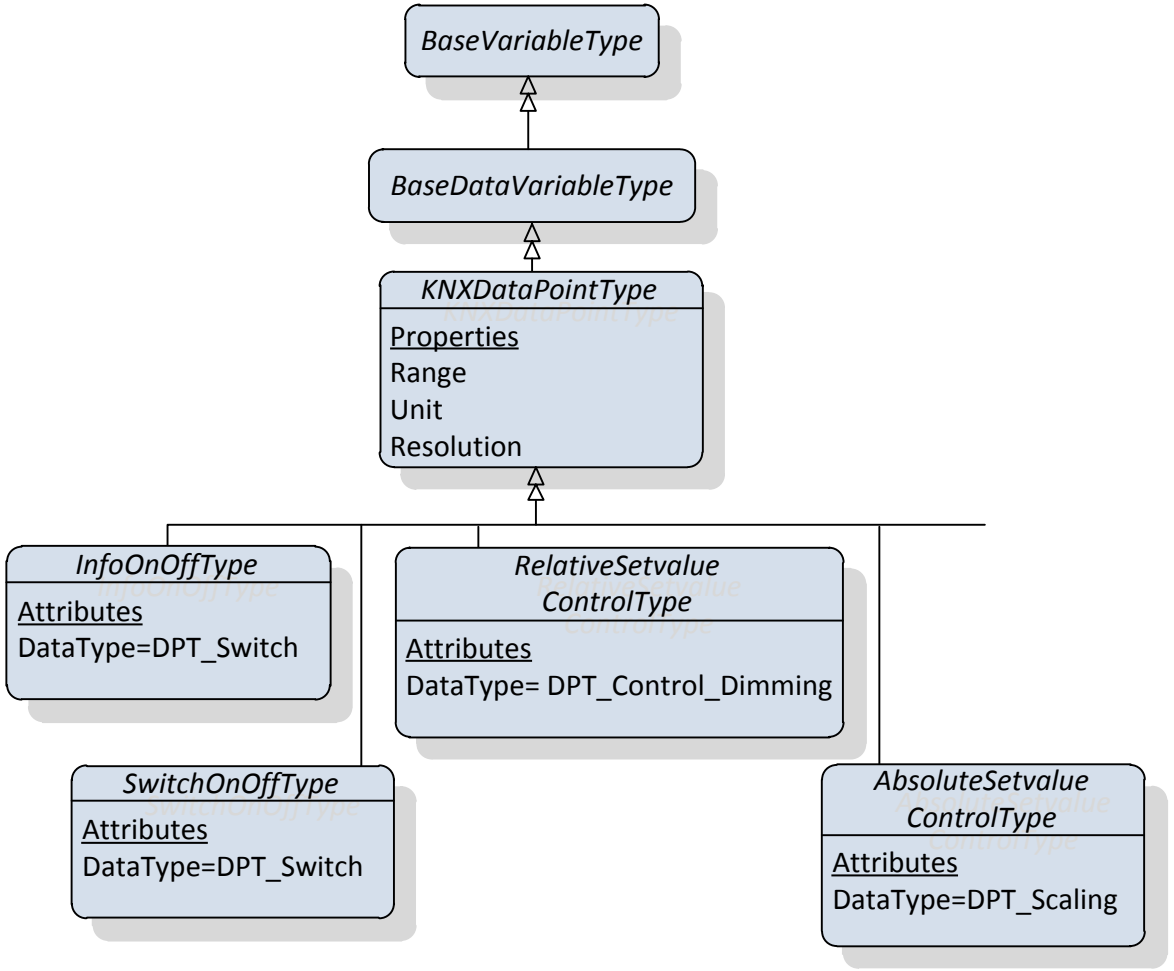
References for KNX function blocks

- User-defined references for linking data points to KNX functional blocks
- User-defined references add additional semantic



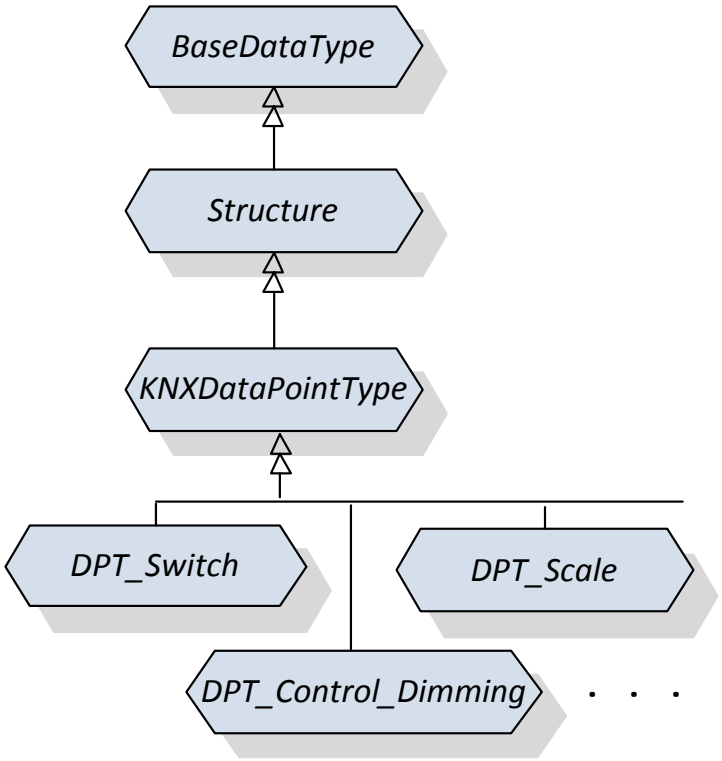
KNX data points in OPC UA

- KNX data points are modelled as OPC simple variables



KNX data point types in OPC UA

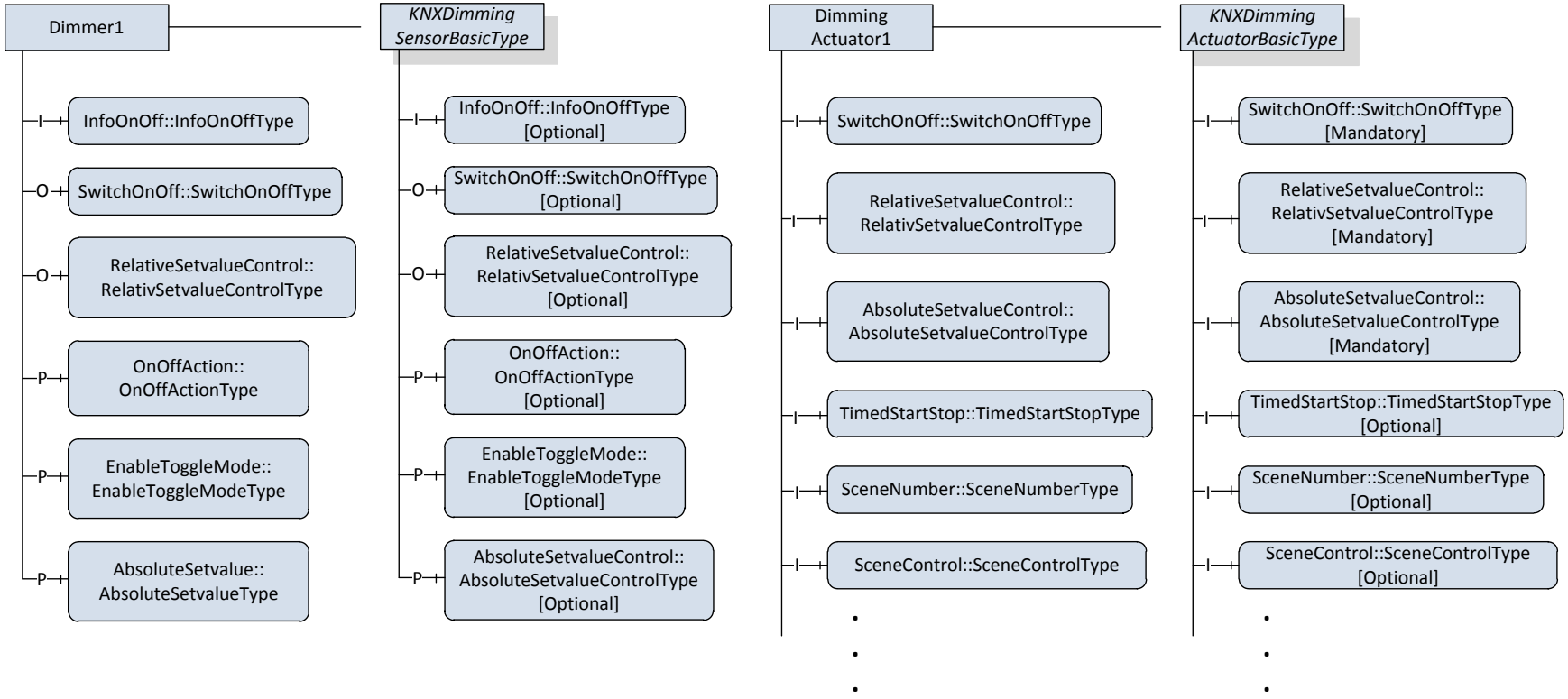
- KNX data point types are specified as structured OPC UA data types



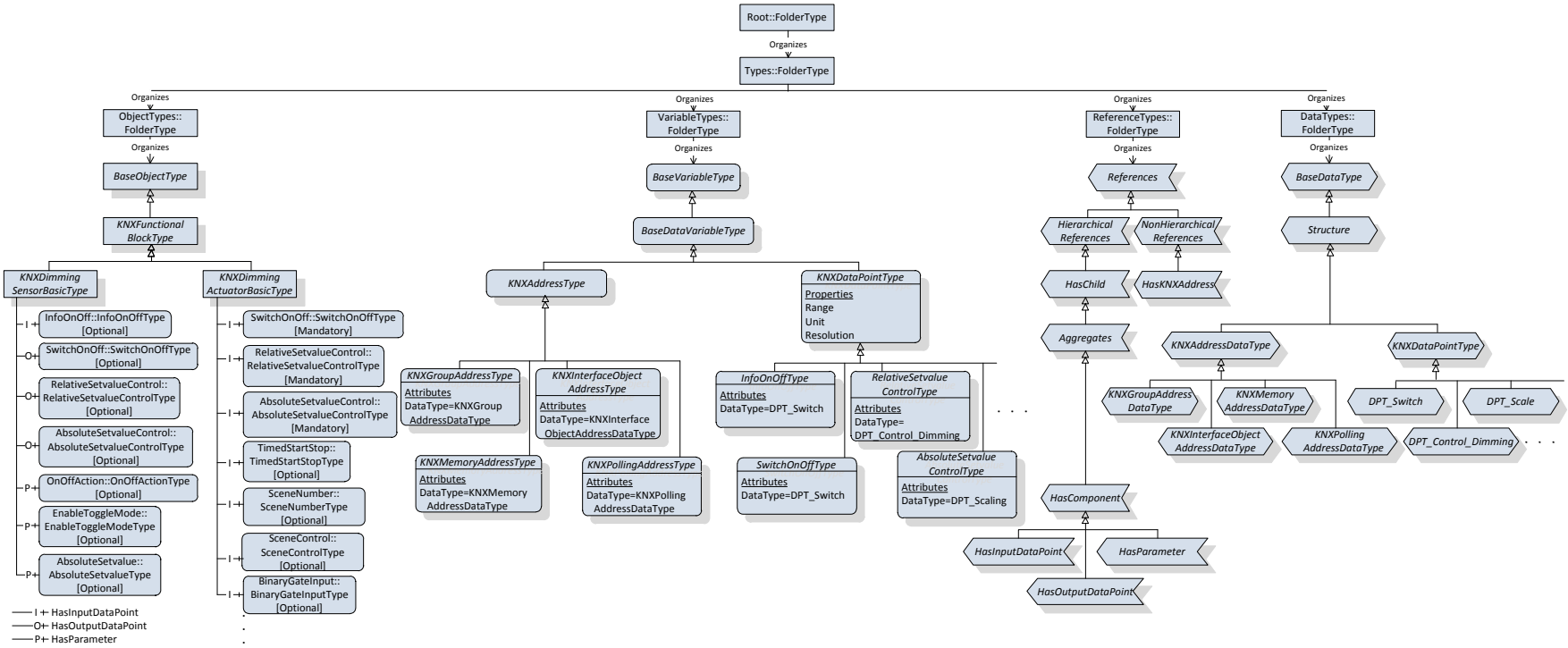

```
<EnumeratedType Name="DPT_Switch" LengthInBits="1">
  <EnumeratedValue Name="Off" value="0" />
  <EnumeratedValue Name="On" Value="1" />
</EnumeratedType>

<StructuredType Name="DPT_Control_Dimming">
  <Field Name="c" TypeName="Bit" Length="1">
    <Documentation>
      0...Decrease;1...Increase
    </Documentation>
  </Field>
  <Field Name="Step" TypeName="Bit" Length="3">
    <Documentation>
      000b...Break;[001b...111b] step size
    </Documentation>
  </Field>
</StructuredType>

<OpaqueType Name="DPT_Scale" LengthInBits="8">
  <Documentation>0...0%;255...100%</Documentation>
</OpaqueType>
```

Modelling KNX installations

- Functional blocks are represented as object instances
- Data points are represented as variable instances



KNX information model

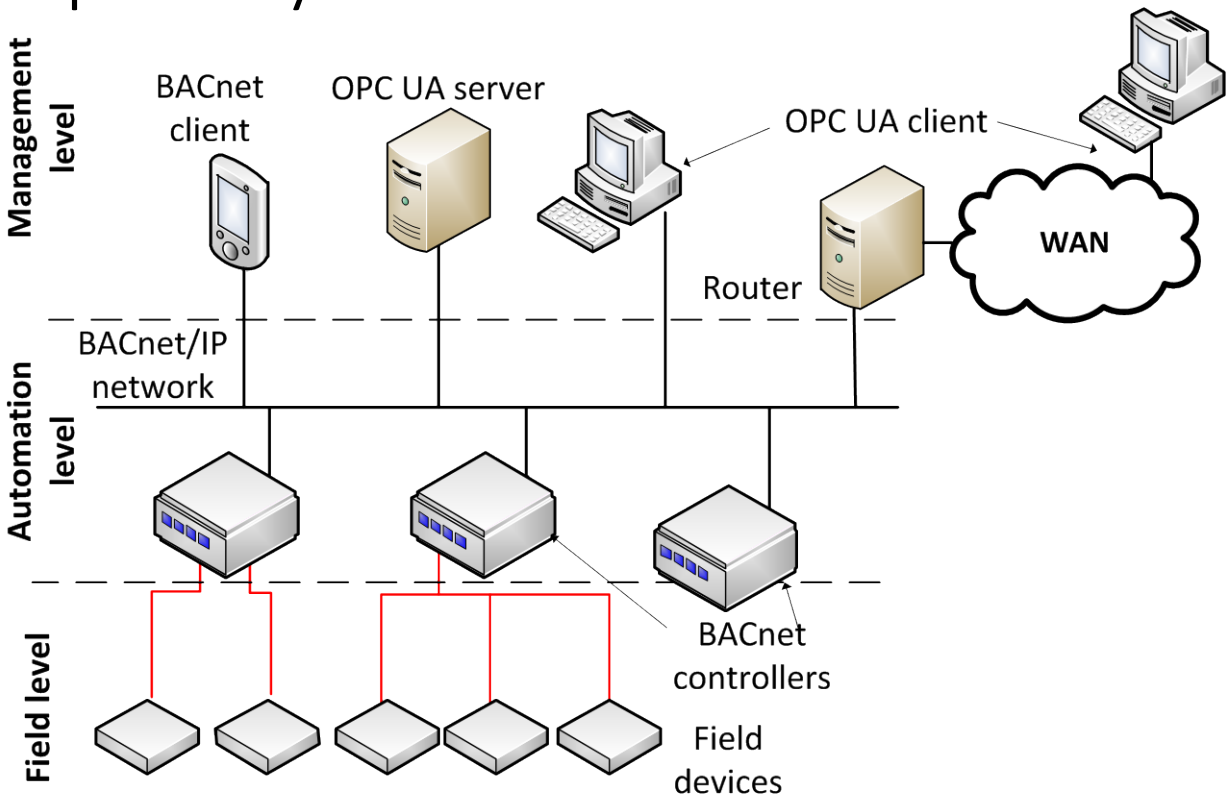


Examples

- Example 1:
  - KNX information model for OPC UA
- Example 2:
  - BACnet information model for OPC UA

OPC UA in BACnet

- Integration at management level to provide interoperability



→ **Mapping of BACnet interworking model to OPC UA**

BACnet interworking model

- BACnet devices have BACnet objects as encapsulation of functionalities
  - About 30 different object types are defined today
- BACnet objects have properties representing data points
- A property holds either one data element or an array of data elements
- E.g., Lighting Output Object:

Property	Data type	Conformance code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	REAL	W
Progress_Value	REAL	R

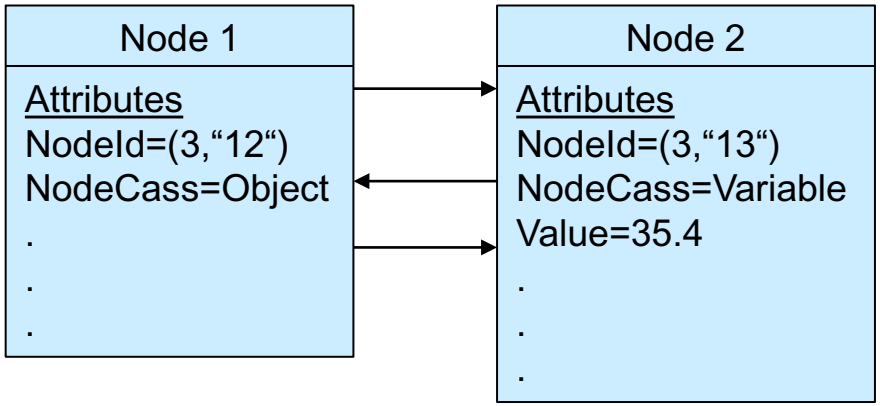
.
.

BACnet in OPC UA

BACnet interworking model

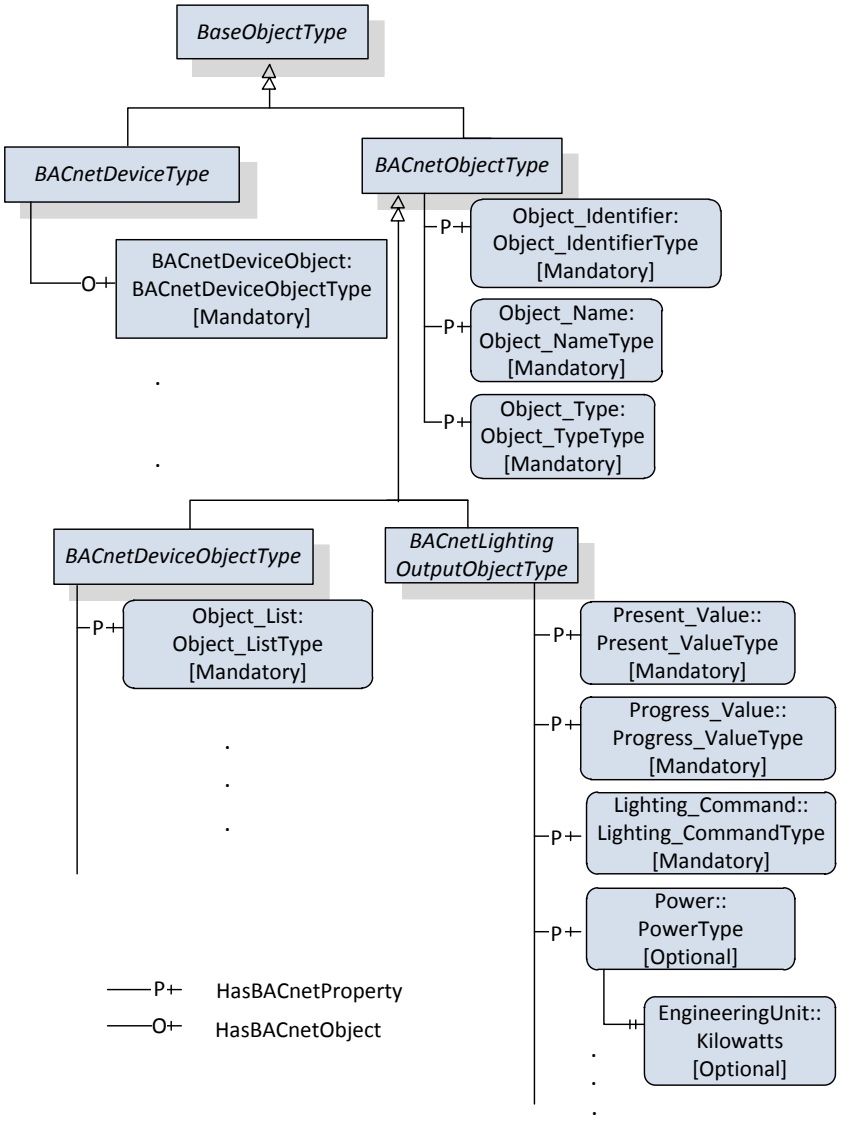
Property	Data type	Conformance code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	REAL	W
Progress_Value	REAL	R

OPC information model



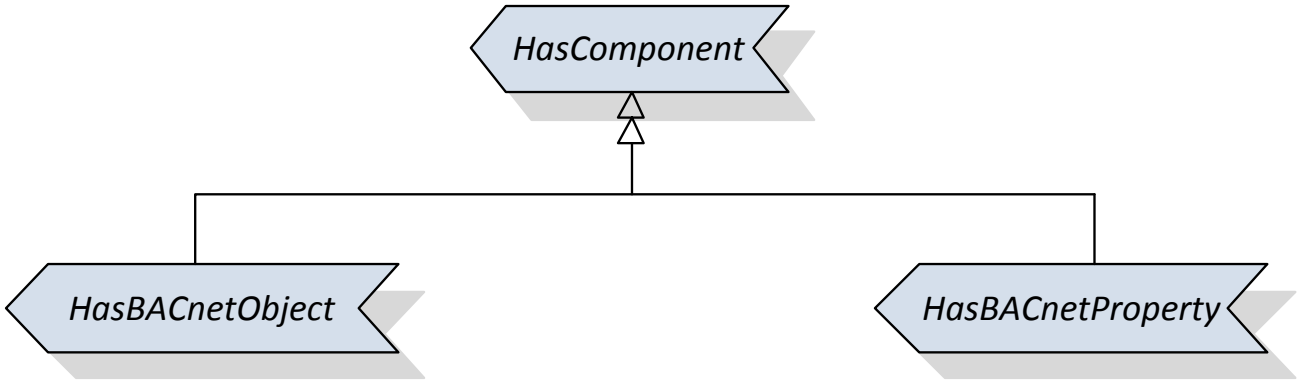
Information model for BACnet

- BACnet devices and objects are modelled as complex objects
- BACnet properties are represented as simple variables
- BACnet property identifier is specified as OPC UA property
- Modelling rules reflect the BACnet conformance code



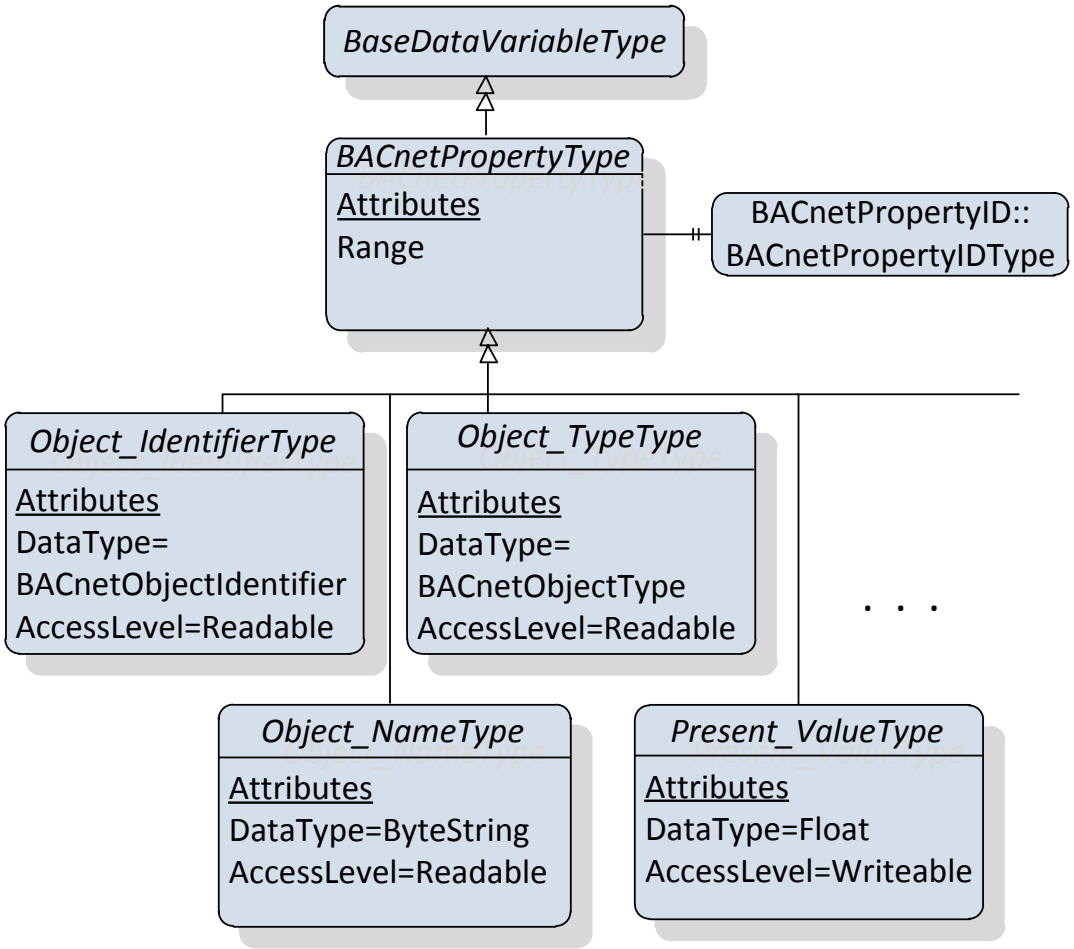
References for BACnet objects

- User-defined references for linking objects and properties to BACnet devices and objects
- User-defined references add additional semantic



BACnet properties in OPC UA

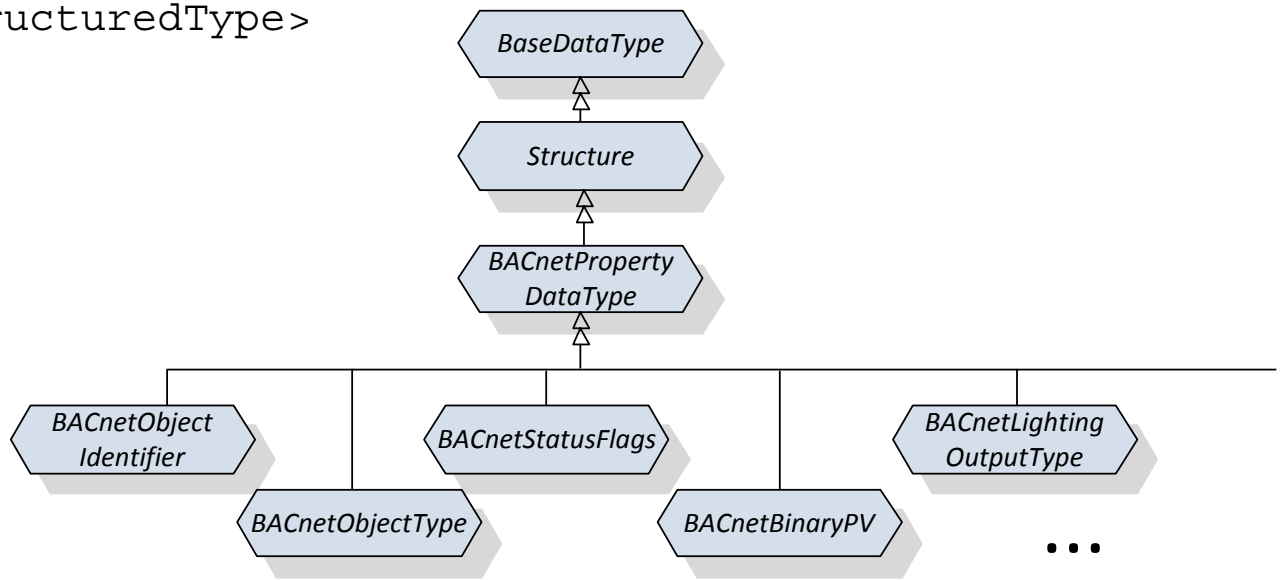
- BACnet properties are modelled as OPC simple variables



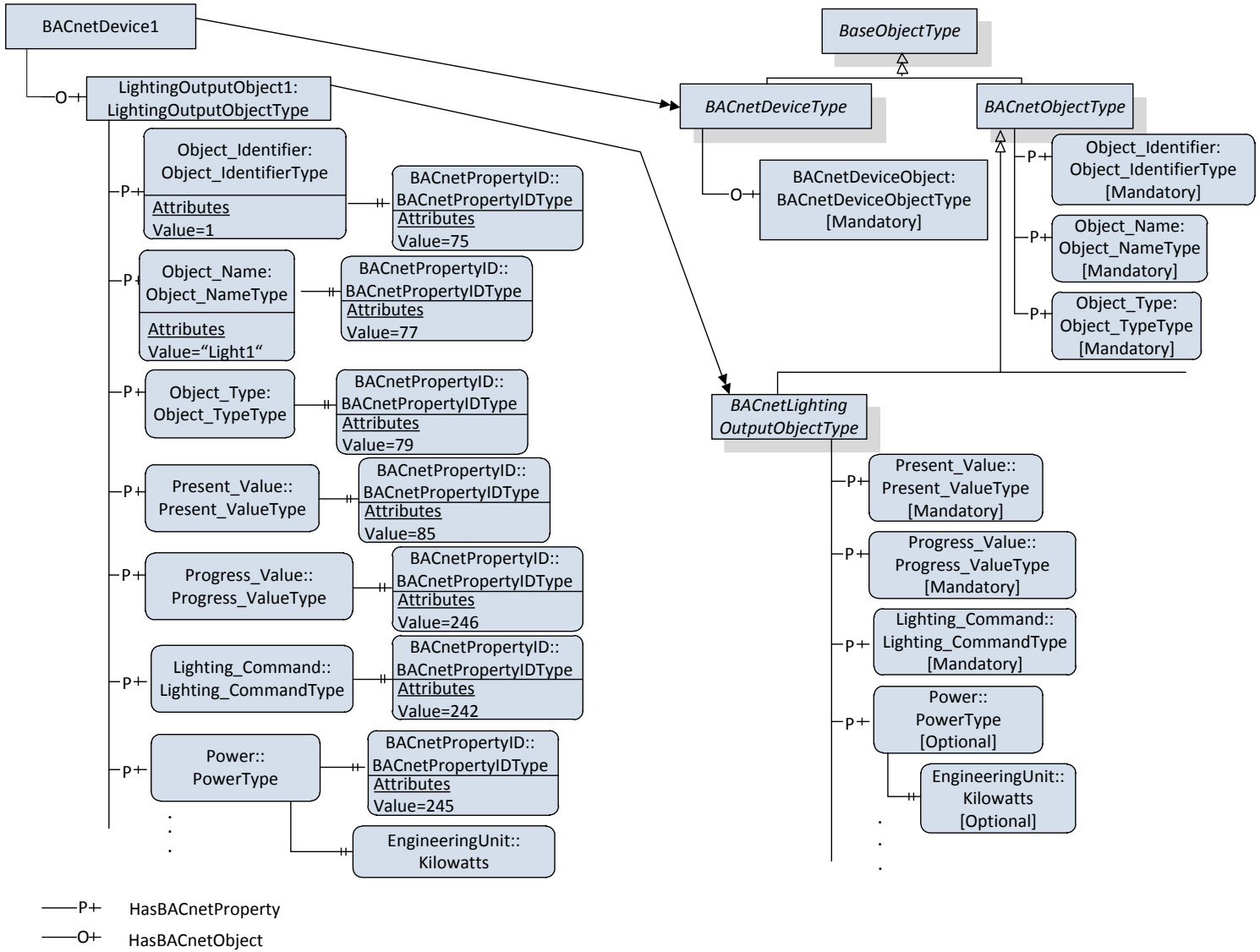
BACnet data types in OPC UA

- BACnet data types are specified as structured OPC UA data types

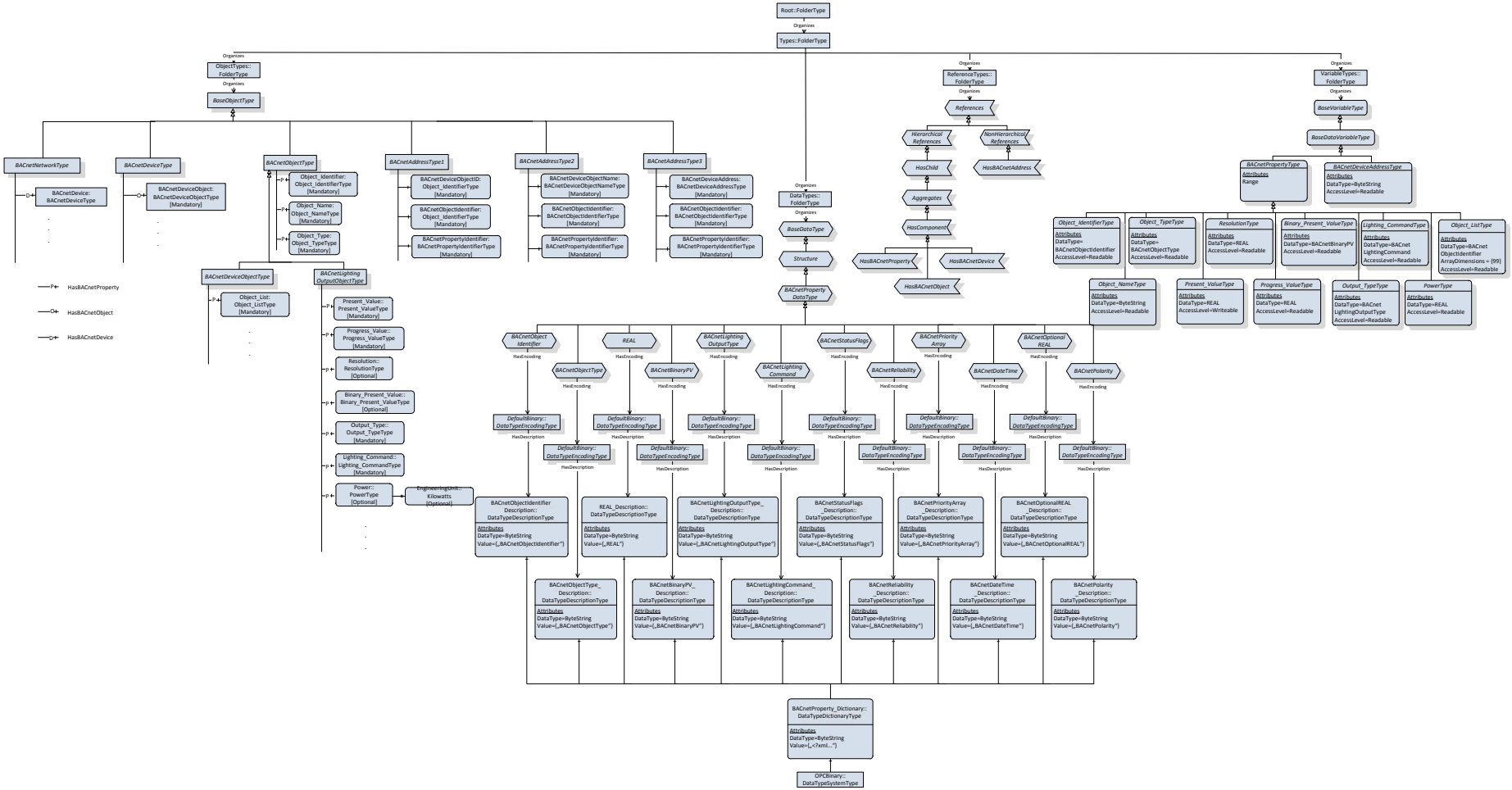
```
<StructuredType Name="BACnetObjectIdentifier">
    <Field Name="ObjectType" TypeName="Bit" Length="10">
    </Field>
    <Field Name="InstanceNumber" TypeName="Bit" Length="22">
    </Field>
</StructuredType>
```



Modelling BACnet installations



BACnet information model



Coffee break and discussion

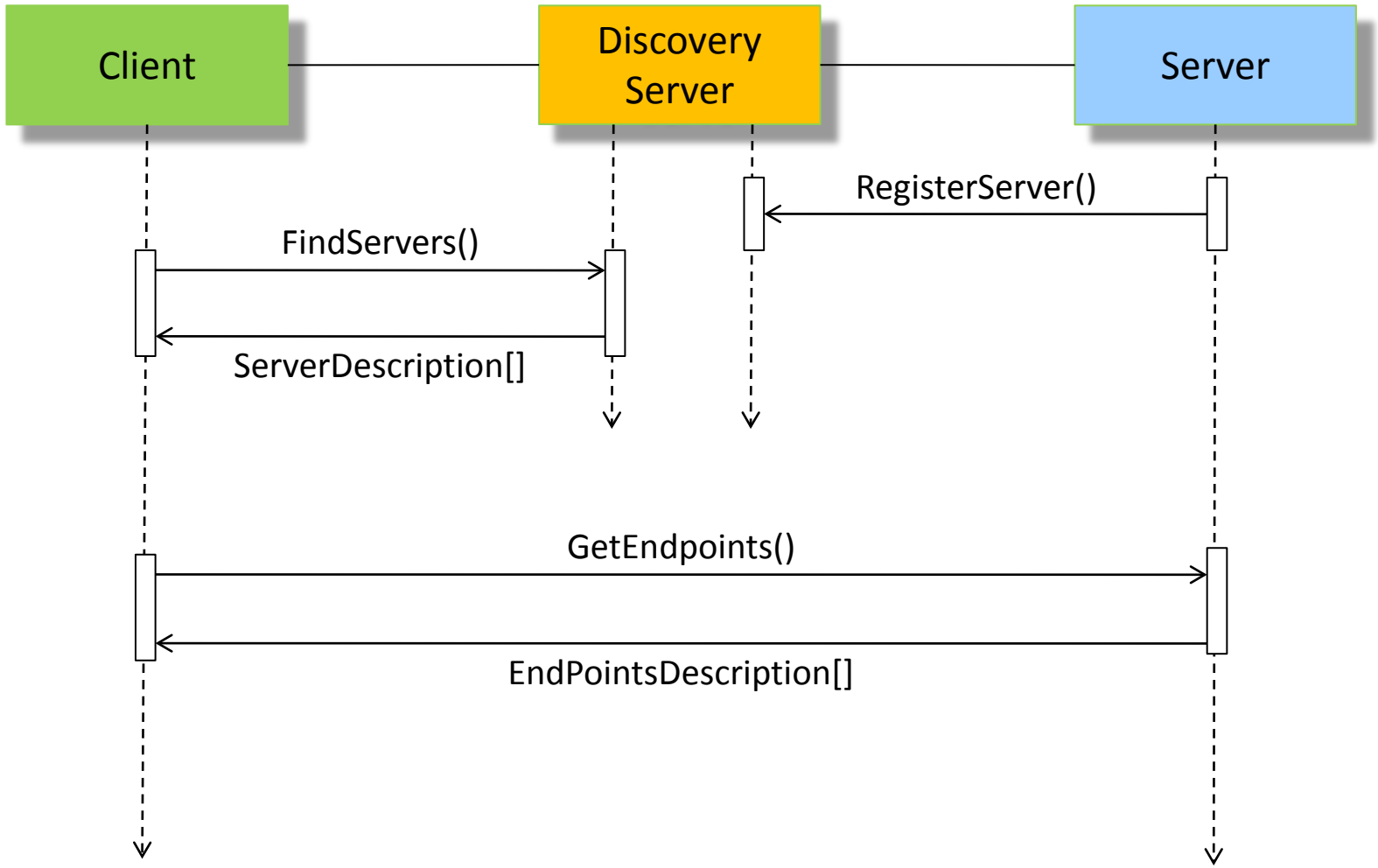
INDIN' 2011
9th IEEE International Conference on
Industrial Informatics

Communication services in OPC UA

Service set model

- OPC UA Service definitions are *abstract descriptions*
- Services are …
  - laid down in OPC UA specifications part 4
  - defining data communication on the application level
  - communication interfaces between UA applications
  - following a request/response pattern
  - designed for exchanging bulk data
  - independent from the transport protocol
  - independent from the programming environment
  - organized into Service Sets

Discovery process



Discovery: FindServers

Request	Description
endpointUrl	network address of the *Discovery Endpoint*
localeIDs[]	List of locales to be returned for server names
serverUris[]	List of servers to return; all known are returned if list is empty

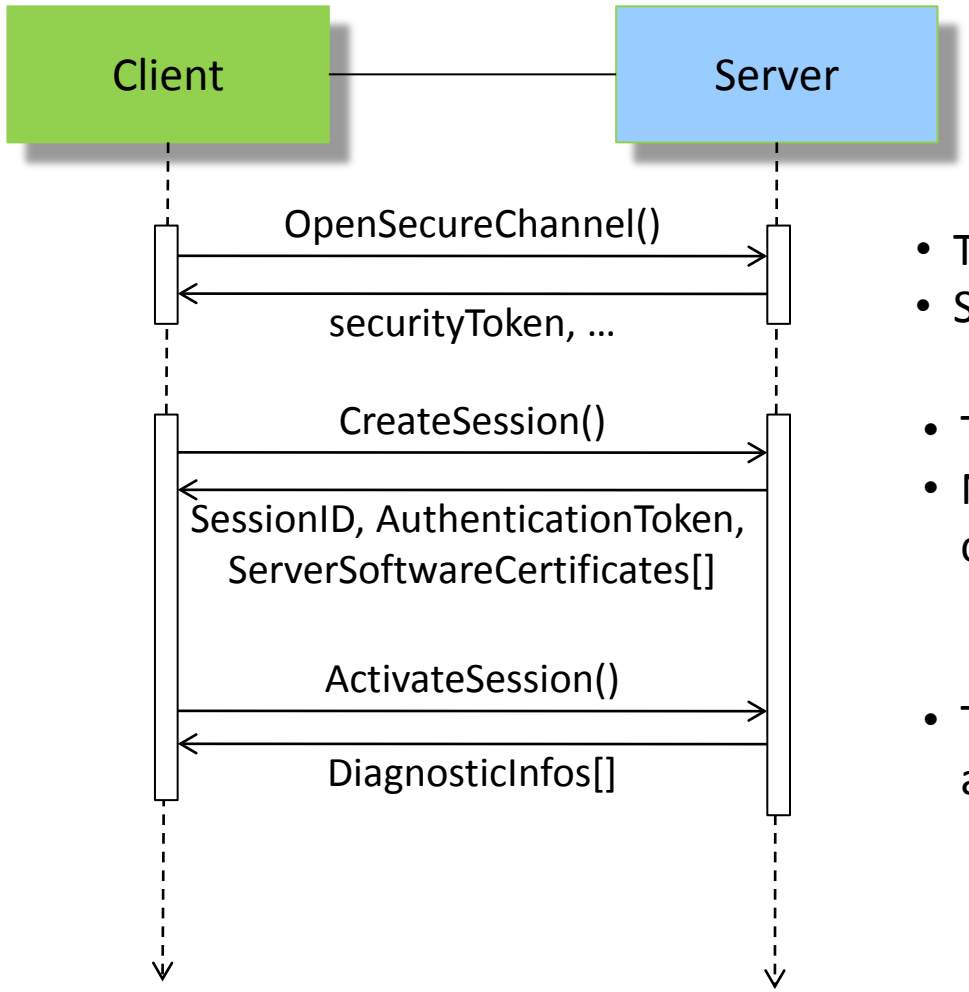
Response	Description
servers[]	application description for each server
applicationUri	unique identifier for the application instance
applicationName	localized descriptive name for the application
applicationType	server/client/server & client/discovery server
discoveryUrls[]	list of *EndPoints* provided by the server

Discovery: GetEndpoints

Request	Description
endpointUrl	network address of the *Discovery Endpoint*
localeIDs[]	list of locales to be returned for endpoint descriptions
profileUris[]	list of profile URIs to return; all known are returned if list is empty

Response	Description
endpoints[]	description for each endpoint
endpointUrl	network address of the endpoint
serverCertificate	server instance certificate (public key of server)
securityPolicy	algorithm sets and key length used for secure channel
securityMode	none/sign/sign & encrypt
userIdentityTokens[]	tokens the server will accept

Connection management

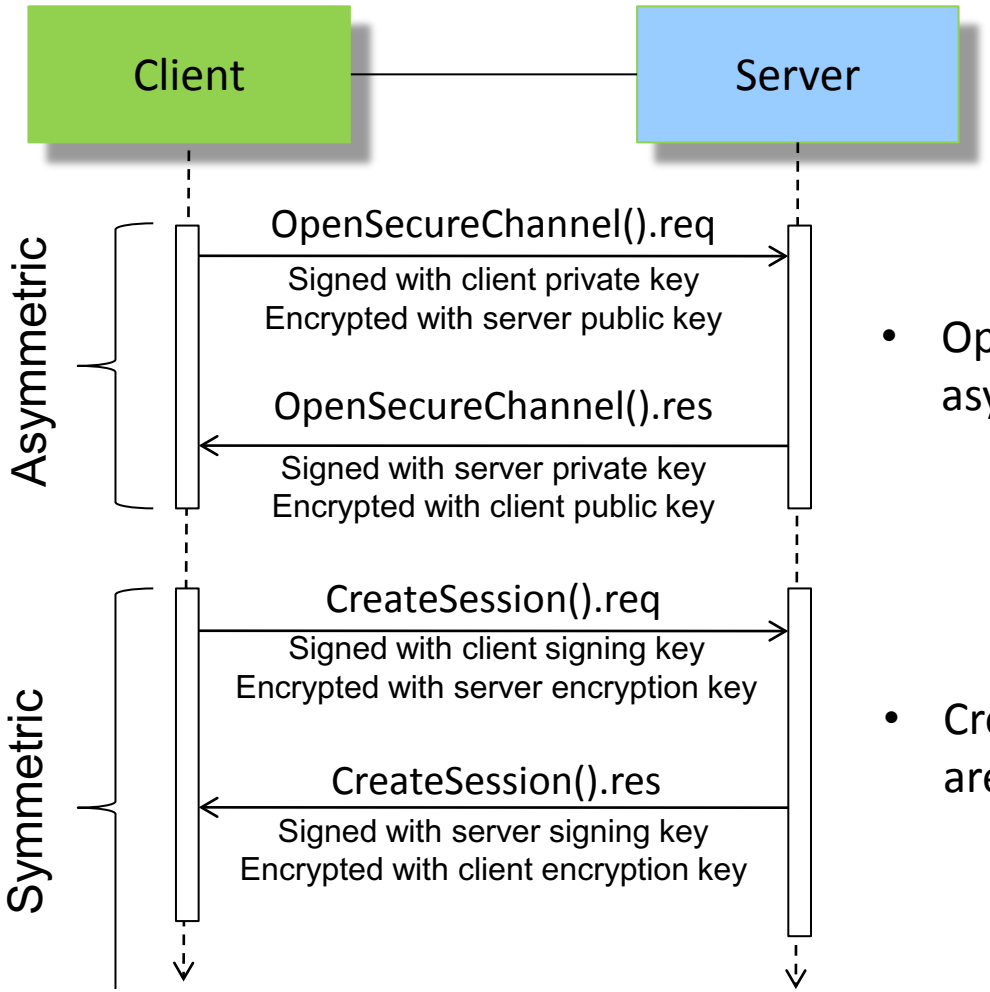


- To ensure integrity, confidentiality
- Security tokens have limited lifetime

- To negotiate session timeouts
- Notification about supported profiles, compliance test level

- To alter user or language settings, assigning a new secure channel

Security in OPC UA



- 3 Security modes:
  - None
  - Signed
  - Signed and encrypted

- OpenSecureChannel is secured with asymmetric algorithms

- CreateSession and all further messages are secured with symmetric algorithms

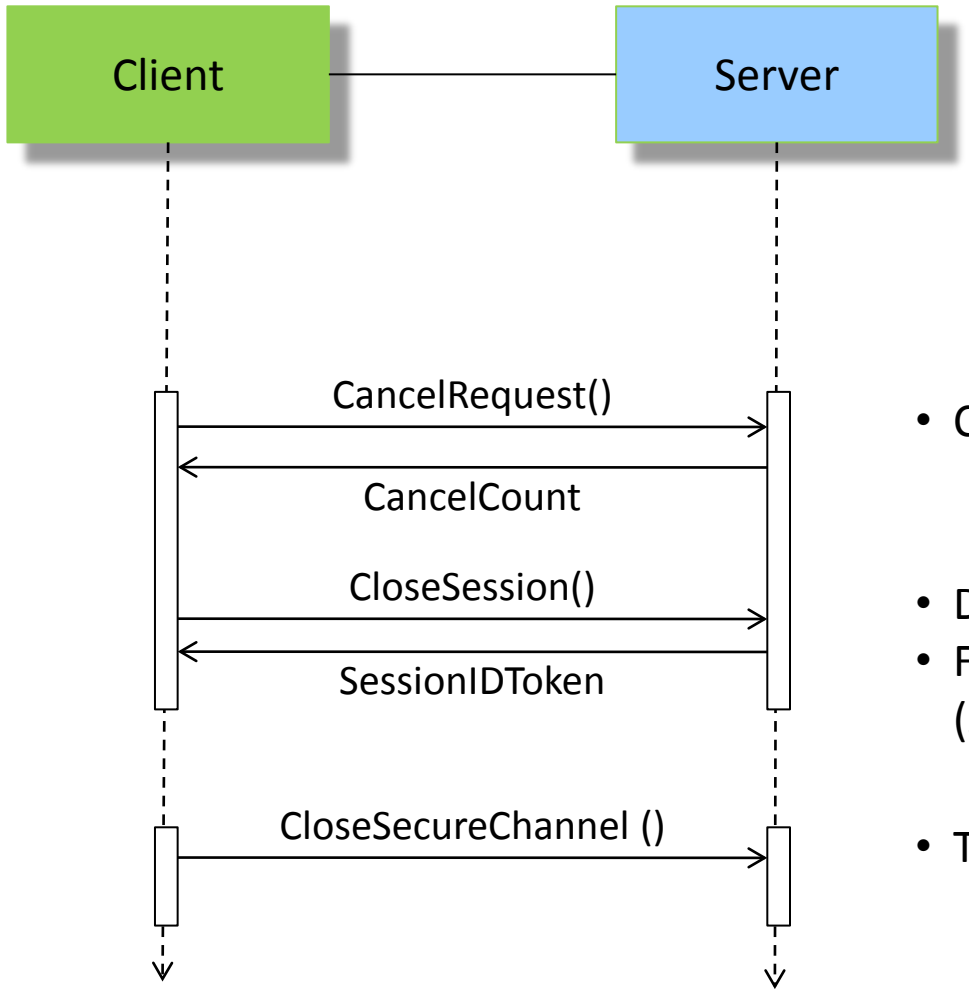
Security policies

- Security None:
  - A suite of algorithms that does NOT provide any security
- Security Basic 128Rsa15
  - Asymmetric: RSA15 and RSA-SHA1
  - Symmetric: AES 128 and HMAC-SHA1
- Security Basic 256
  - Asymmetric: RSA-OAEP and RSA-SHA1
  - Symmetric: AES 256 and HMAC-SHA1

Certificates

- Authentication is based on certificates
  - Contains signed public key
  - Initial knowledge that is required to start a secure session
- X509 version 3 certificates encoded using the DER format
- Public Key Infrastructure (PKI) is required
  - Creation
  - Distribution
  - Validation
  - Storage

Connection management (cont.)



- Cancel outstanding requests
- Disconnect
- Freeing resources inside the server (subscriptions)
- Terminate the secure channel

Discovering the Address Space

- View Service Set
  - Navigate through the entire space or a specific view
  - List of starting nodes and browse filters
  - Continuing via *BrowseNextService()*
  - Improving performance via *RegisterNodes()*
- Querying Service Set
  - Discover complex Address Spaces based on type information
  - Filter criteria include SELECT (data to return) and WHERE (filter) clauses
  - Continuing via *QueryNext()*

Modifying the Address Space

- Creating and deleting *Nodes* and *References*
- New nodes are persistent
- Nodes are added by means of *HierarchicalReferences*

  → Address Space stays fully connected
- When nodes are deleted, monitoring clients are informed

- Primary focus: Updating the Address Space between configuration client and OPC UA Server

Reading and writing

- Optimized for bulk operations
- List of *Nodes* and *Attributes* to read/write
- *Read()*
  - *MaxAge* allows to return cached values
  - Values are timestamped (source/server)
  - Access to single elements or ranges of arrays
- *Write()*
  - Typically Value Attribute of a Variable is writeable
  - Policy defined by (user) access level

Historical data access

- Requests include definition of a time domain
- Responses carry array of archived information
- Historical values are not visible in the Address Space
- *HistoryRead()*
  - Extensible parameter for reading raw data, processed data (average, minimum, maximum, counting, timing aggregates), events
- *HistoryUpdate()*
  - Values/events can be inserted, updated, replaced or deleted in the history database

Calling methods

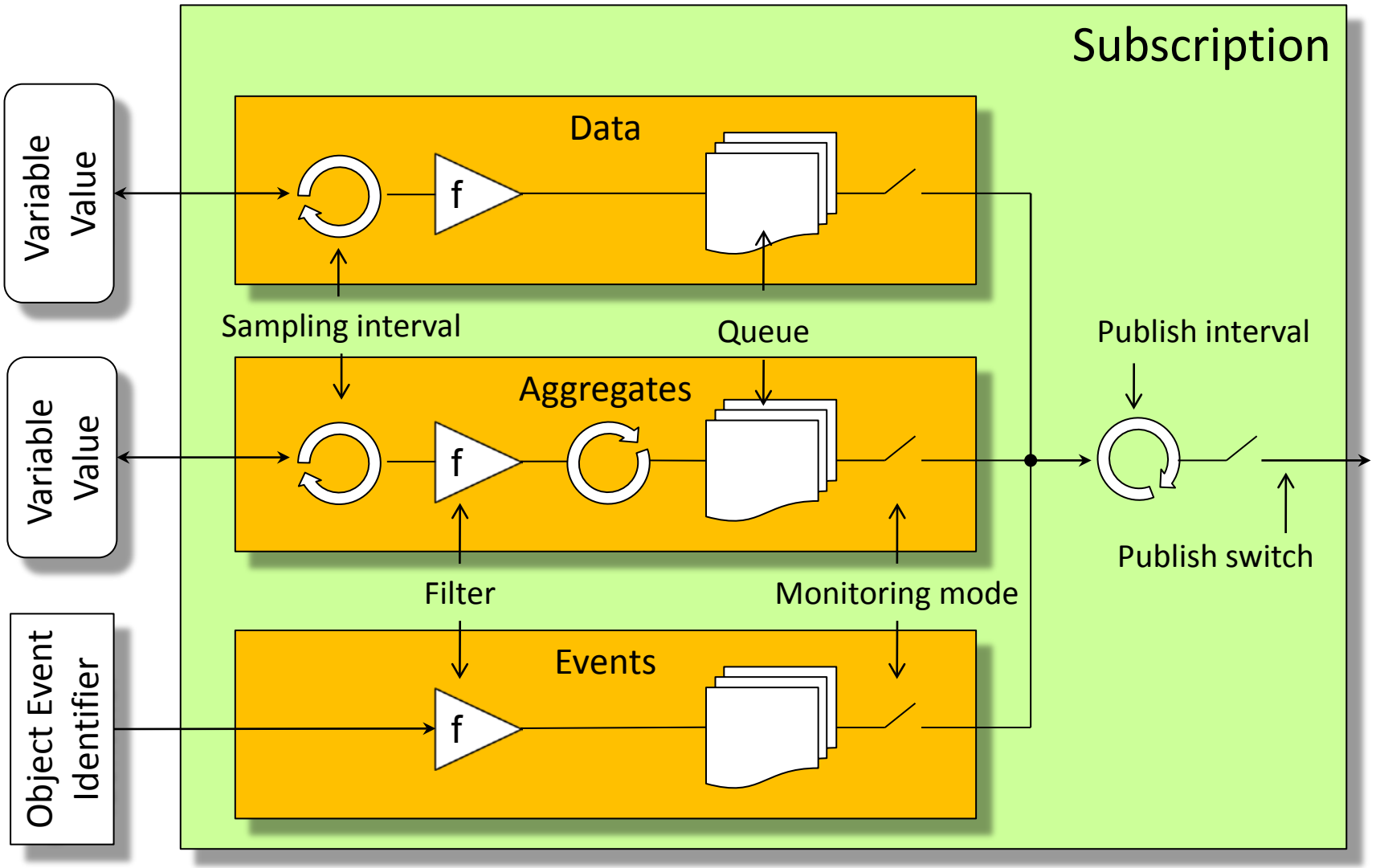
- Methods
  - Components of Objects
  - Represent the function call of *Objects*
  - All necessary information for invocation is available in the Method description

- *Call()*
  - Can only be used within a Session
  - Allows invoking a list of Methods
  - Provides arguments for passing input parameters
  - Responses hold output parameters

Requirements/features for notification

- Server triggered sending of notification messages
  - Publish requests can be queued
- Timely detection of communication problems
  - Alive signals
  - client → server and server → client
- Reliable communication
  - Message ordering and detection of lost messages
  - Sequence numbers
  - Acknowledgement of received messages
  - Request for lost notification messages (republishing)

Change of values and events

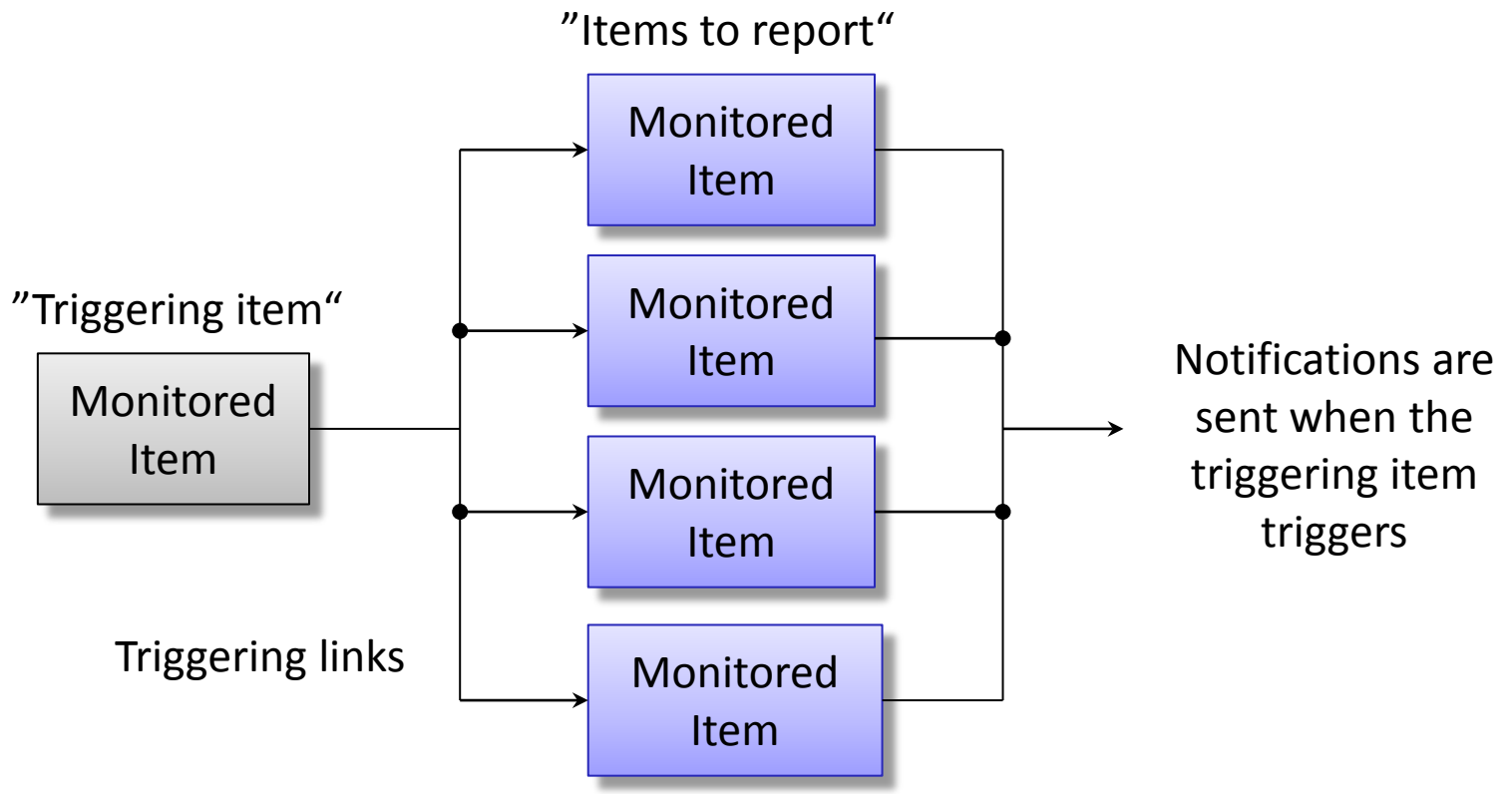


MonitoredItems

- Monitoring mode
  - Independent enabling/disabling of notifications
  - sample/sample & report/disabled
- Sampling interval
  - Fastest rate items should be sampled
  - Best effort cyclic rate, 0 for the subscription of events
- Filter
  - Data changes: absolute/relative deadband
  - Events: SELECT/WHERE clauses
  - Aggregates: Start time; aggregate type
- Queue parameter
  - Queue size
  - Discard policies: overwrite newest or oldest value

Triggering model

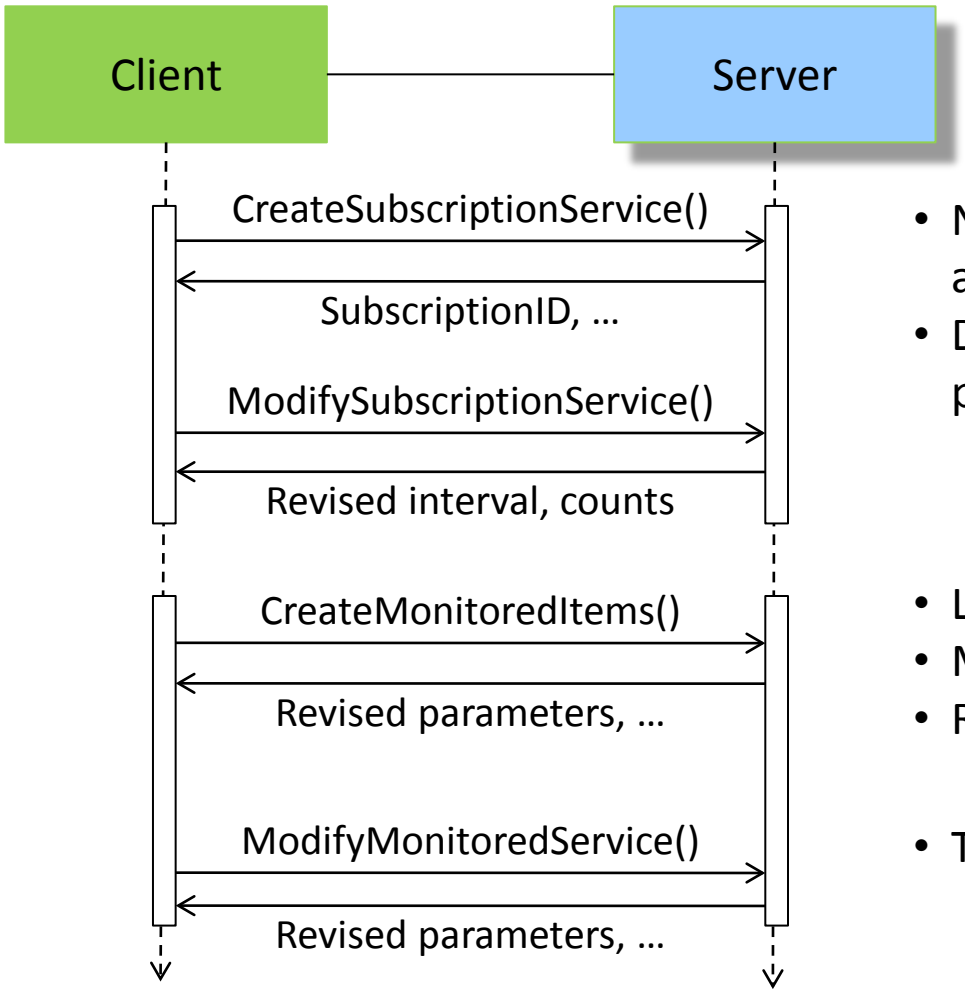
- Chaining of *MonitoredItems* to reduce amount of data
- Monitoring mode can be set individually



Subscription

- Assigned to *MonitoredItems*
- Used to report *NotificationMessages* identified by sequence numbers
- Publishing interval
- *Publish()* requests are processed in FIFO manner
  - If there are *Notifications* to send, request is de-queued
  - If there are no Notifications, server waits for next cycle, request remains in queue
- Keep-alive counter
  - Incremented when not *Notifications* are to report
  - Maximum reached keep-alive message returned
- Publishing may be disabled
  - Keep alive messages keep the connection up
- Lifetime counter

Subscription (cont.)

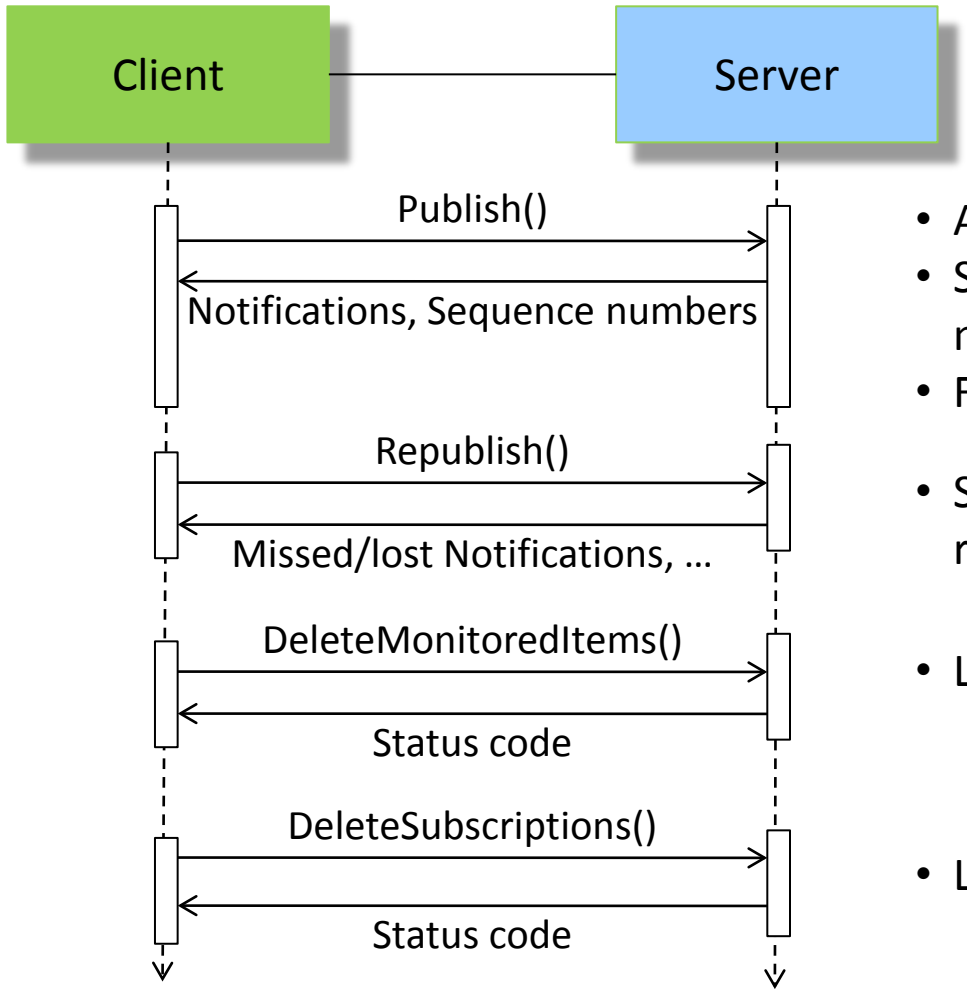


- Negotiate publishing interval, keep-alive count and lifetime count
- Define max. number of notifications per notification message

- List of items to create
- Monitoring mode and parameters
- Revised sampling interval, queue size

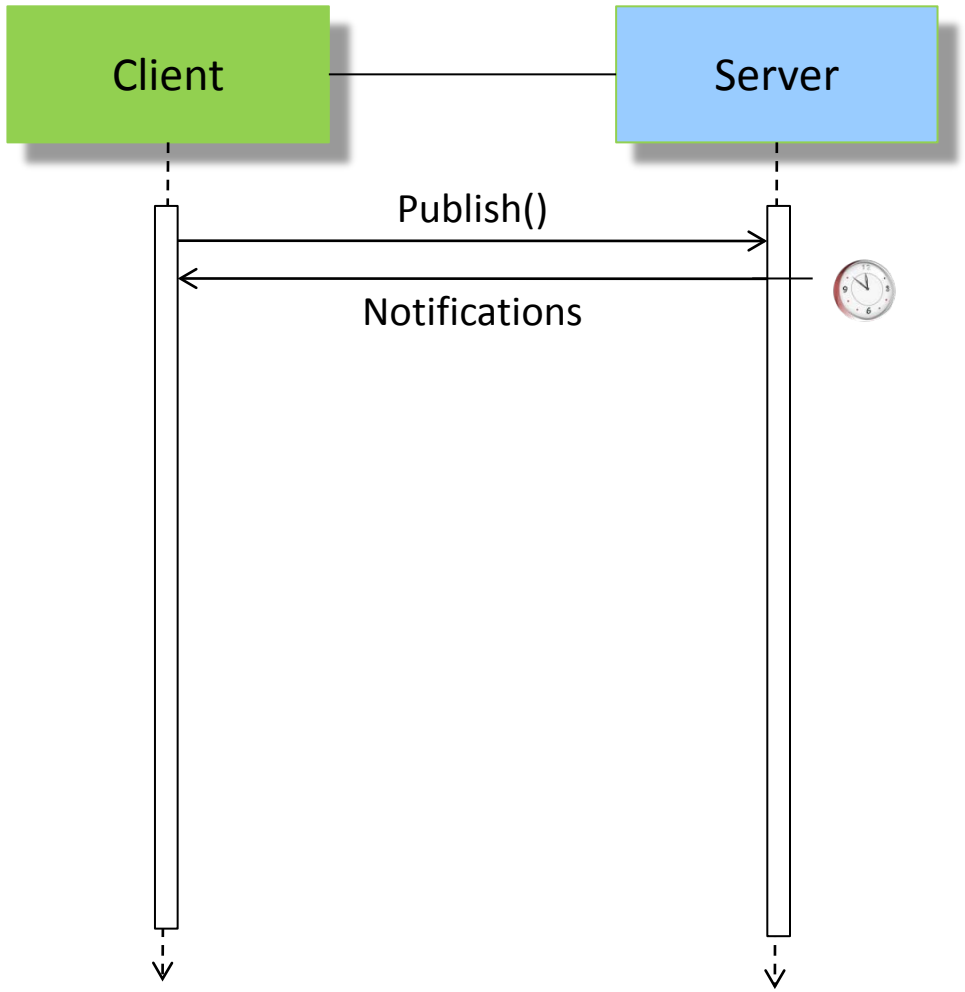
- To alter monitoring parameters

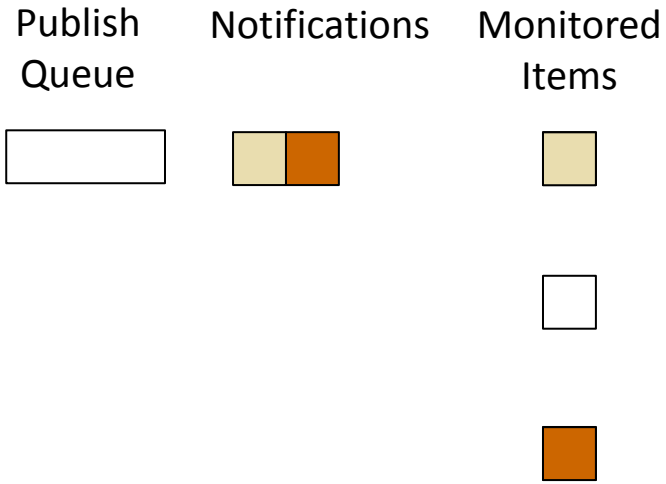
Subscription (cont.)



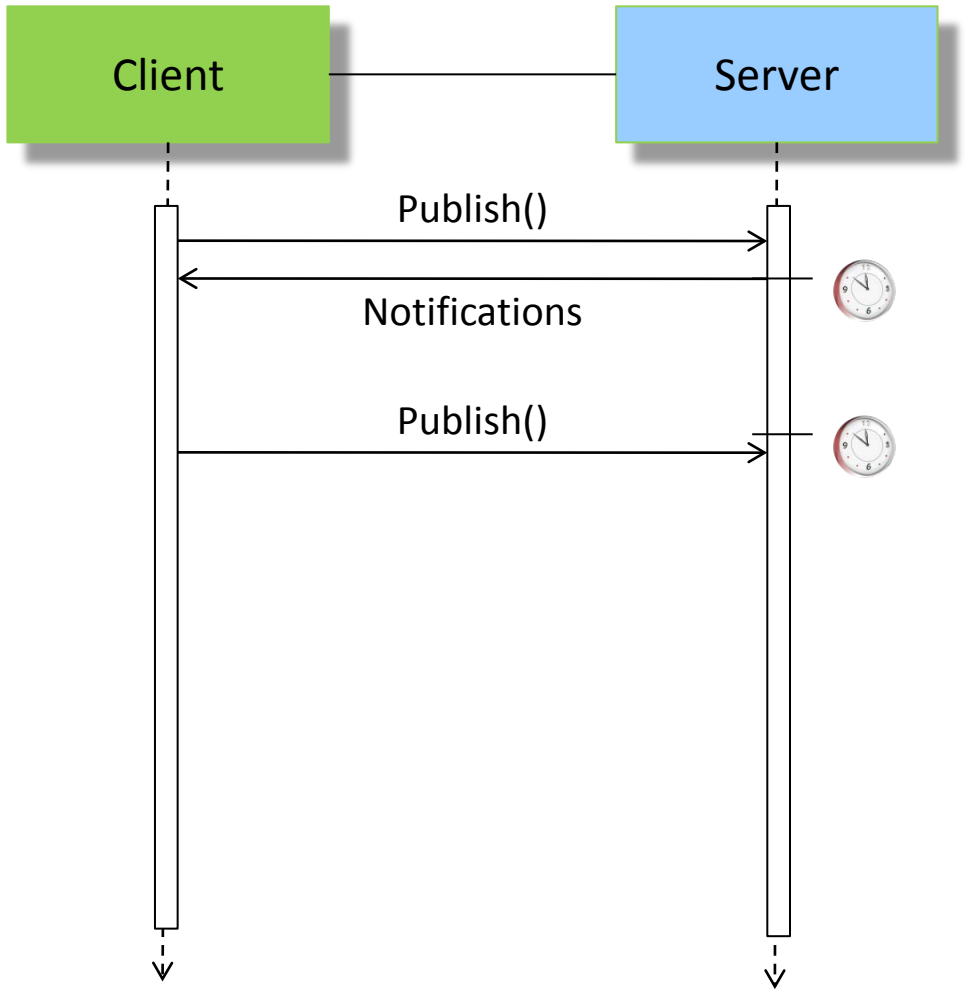
- Acknowledgements
- Sequence number, timestamp and notification data
- Flag indicating more notifications
- Sequence number of the message to resend
- List of items to delete
- List of subscriptions to delete

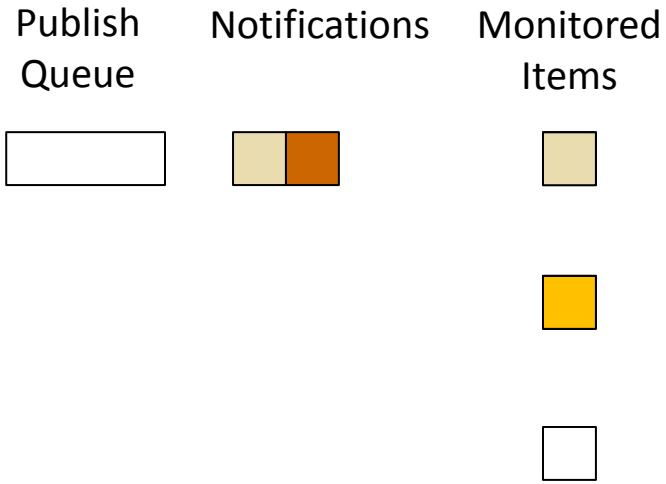
Publish-service



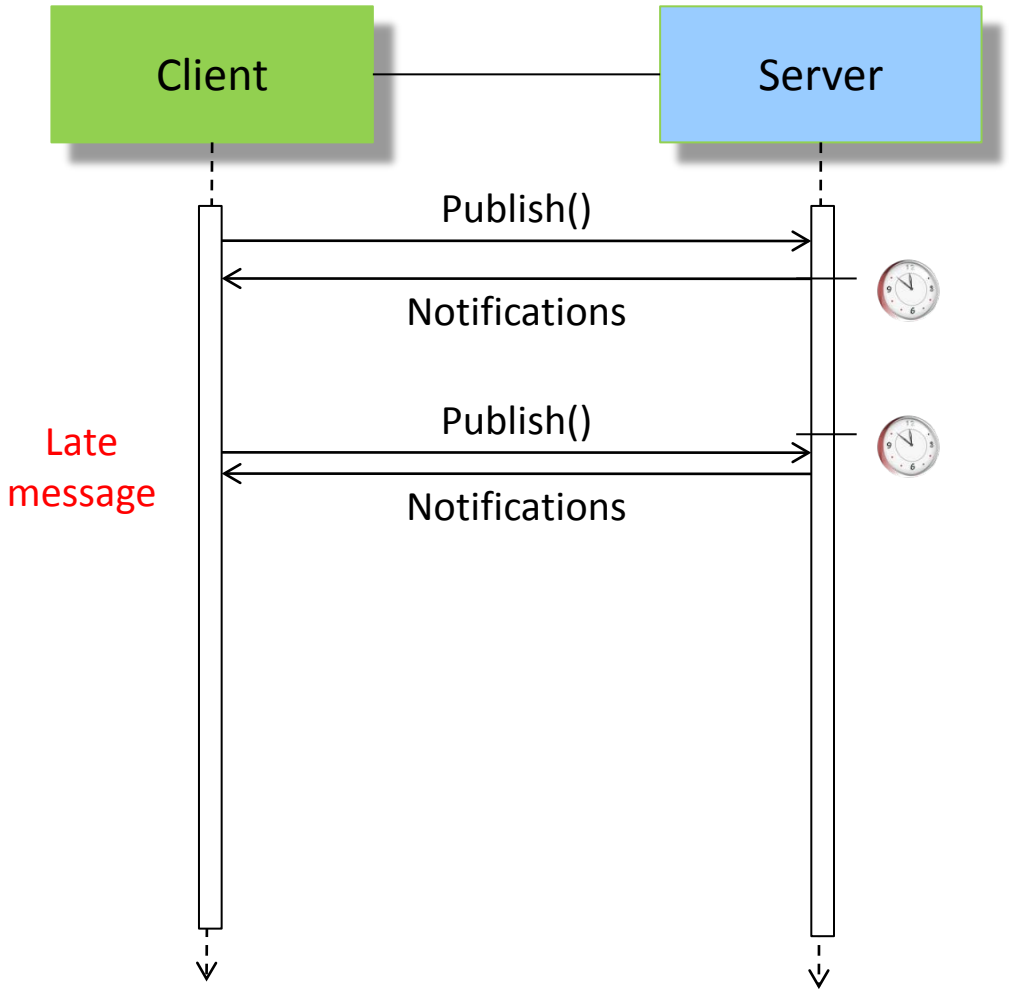


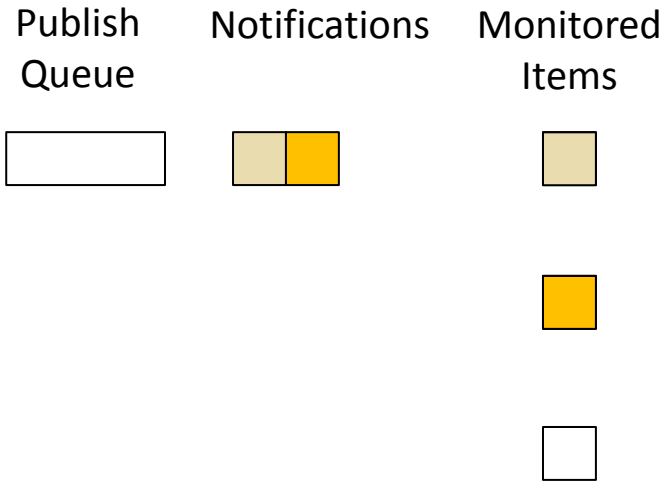
Publish-service (cont.)



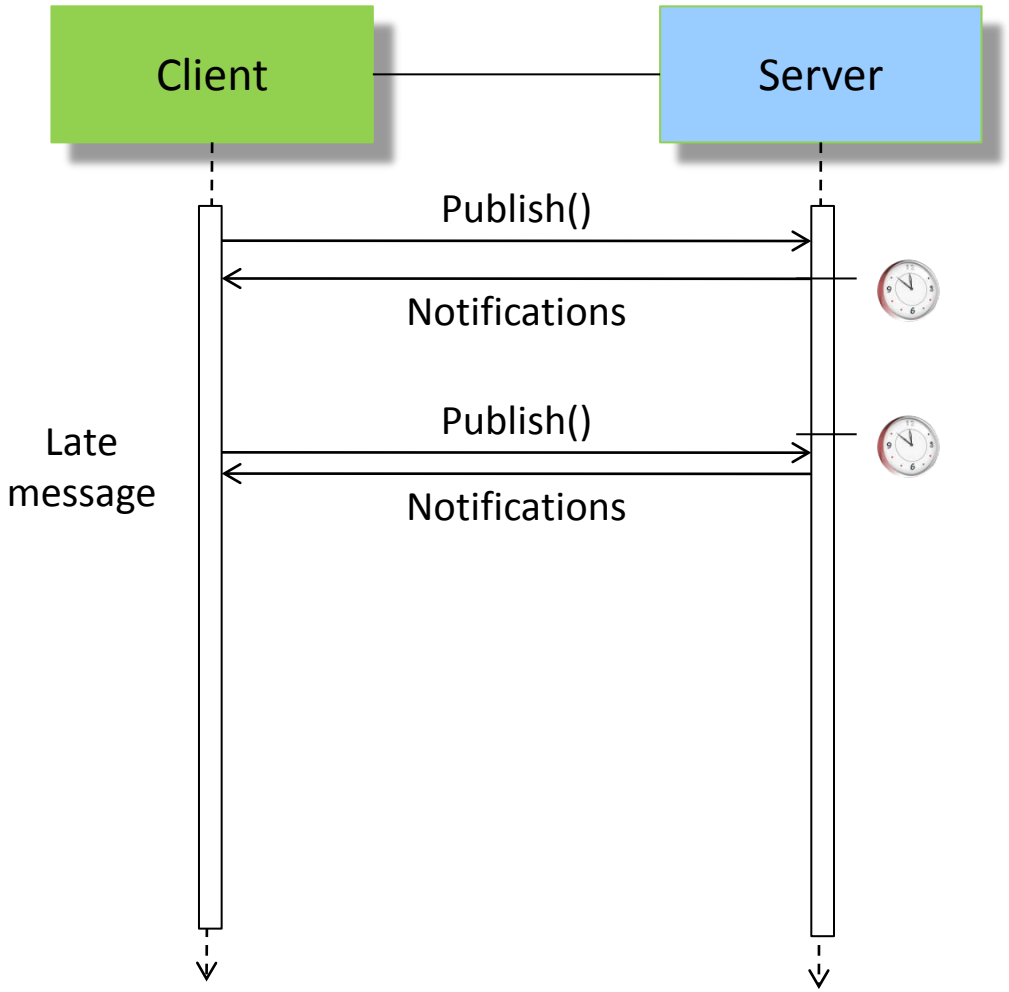


Publish-service (cont.)





Publish-service (cont.)

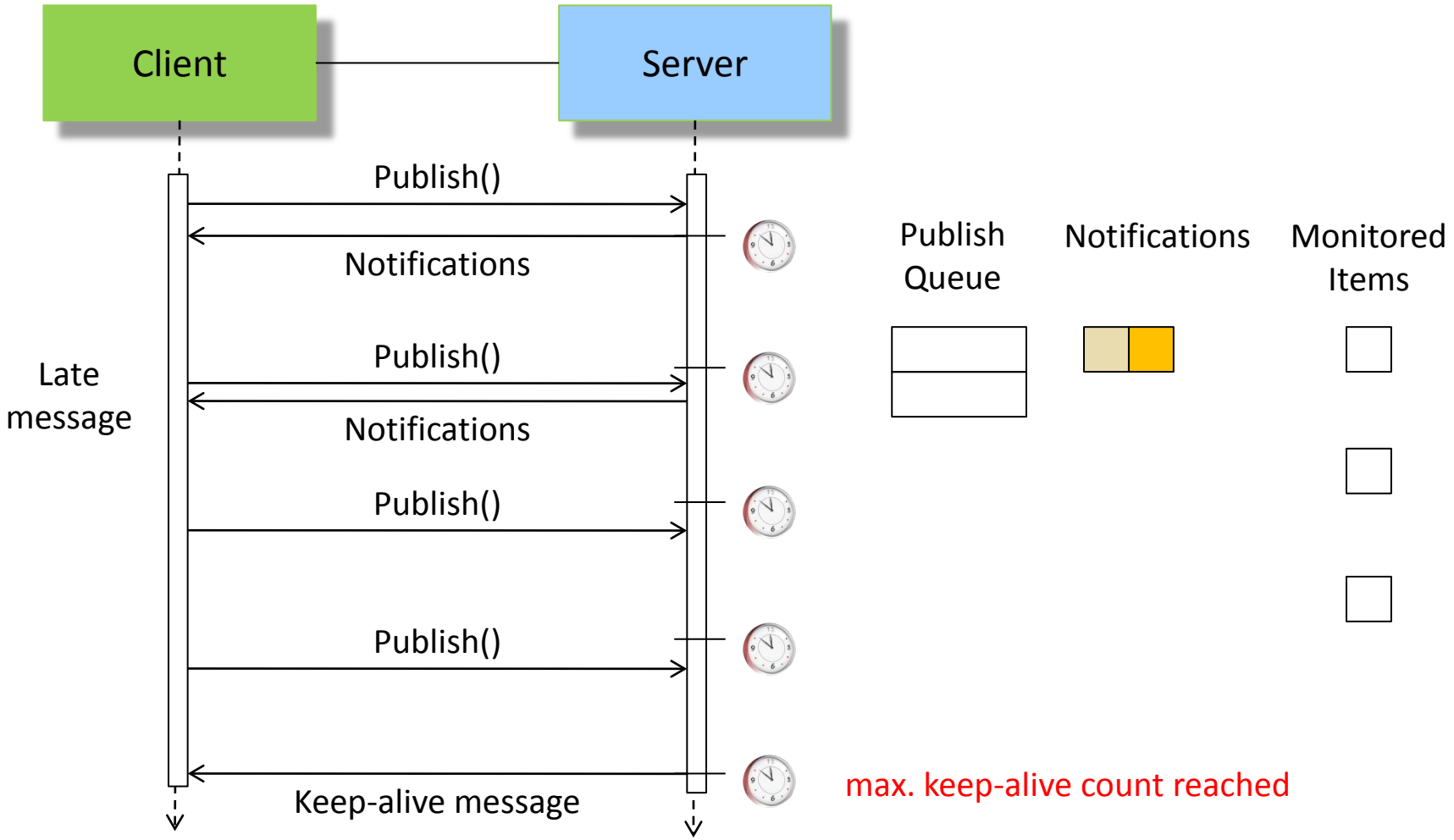


Publish Queue

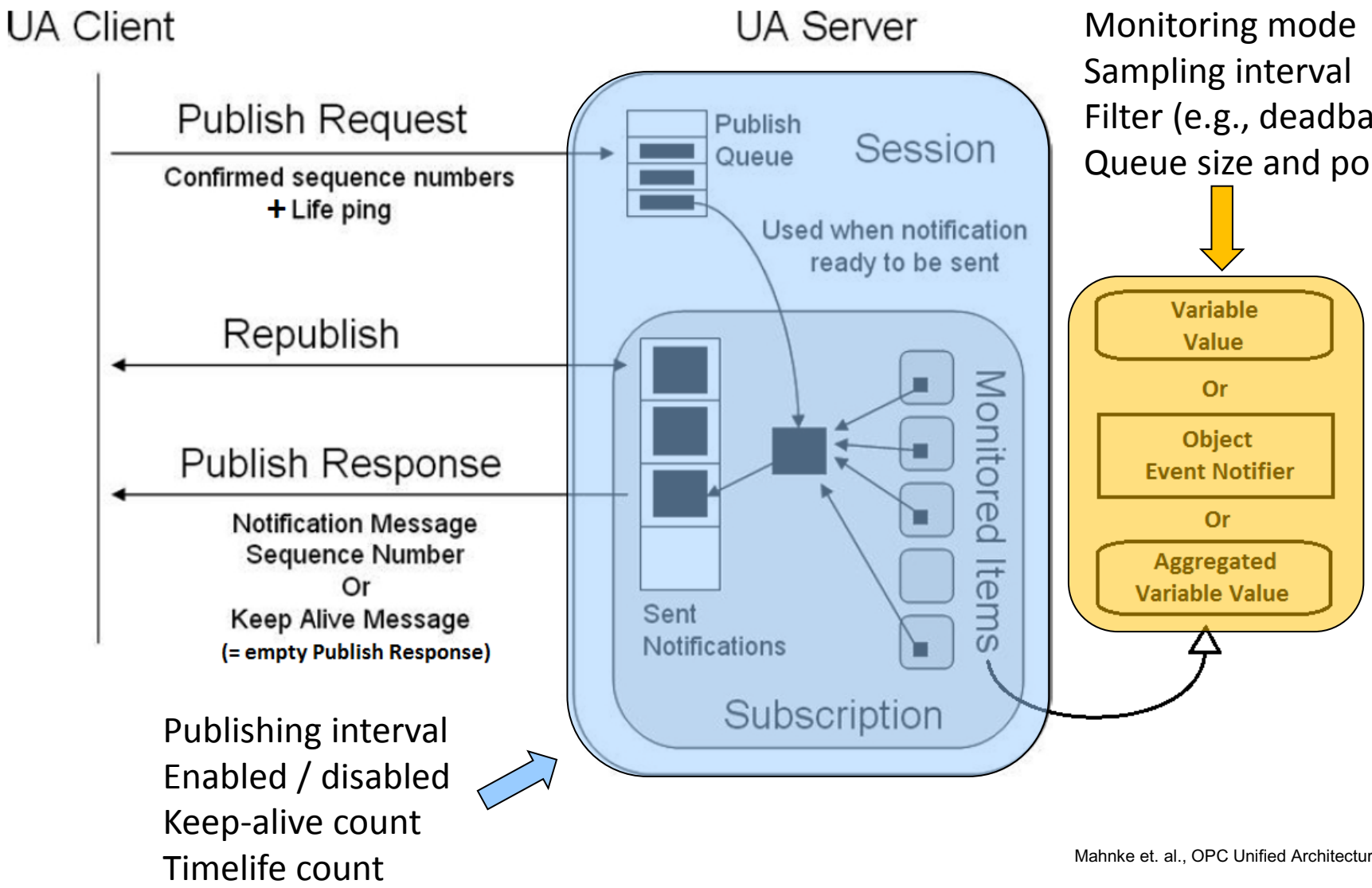
Notifications

Monitored Items

Publish-service (cont.)

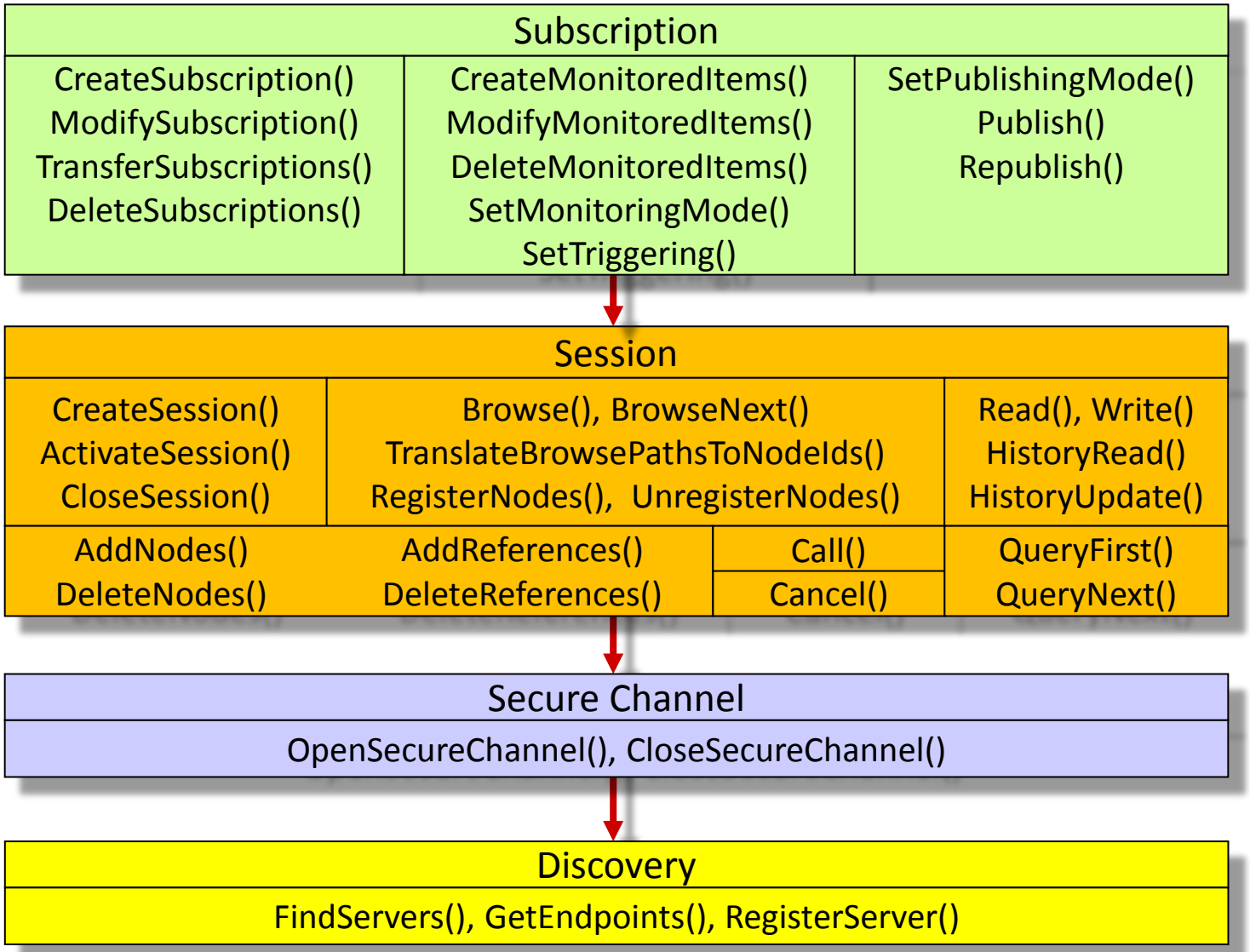


Publish-service (cont.)



Mahnke et. al., OPC Unified Architecture, 2009

Summary



N:1 relationship between MonitoredItems and Subscription

N:1 relationship between Subscriptions and Session

INDIN' 2011

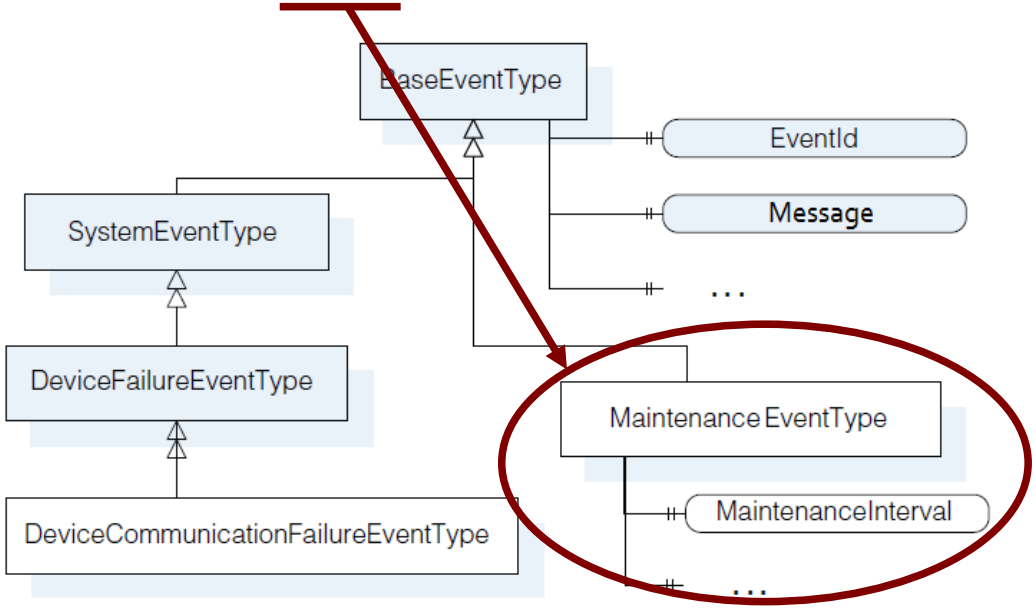
9th IEEE International Conference on

Industrial Informatics

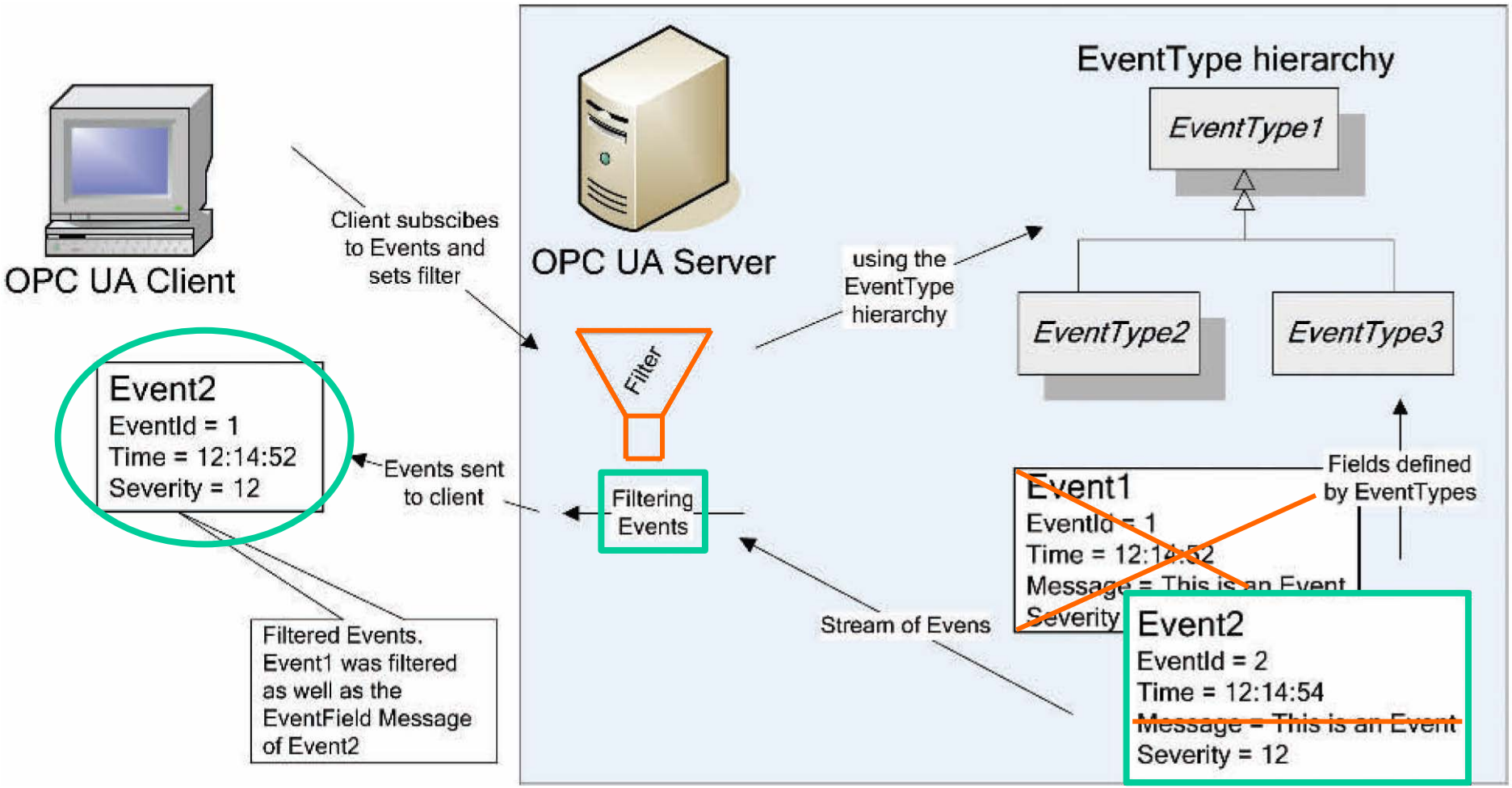
Advanced concepts in OPC UA

OPC UA events

- *Server:* generates events
- *Clients:* **Subscribe** to some events (Event **Filter**)
  ➔receive **Event Notifications**
- *Events:* have a **Type** and contain **Event Fields**
- Possibility to create own EventTypes with new Event Fields



Event filtering



Auditing in OPC UA

- Keep track of normal and abnormal activities
  → Store the **Events** in **Audit Logs**

- Traceability
  → **Who** did **what when**?
  → **What** happened **when**?

- Service XYZ ↔ XYZ**AuditEventType**

- Different scenarios / architectures possible…

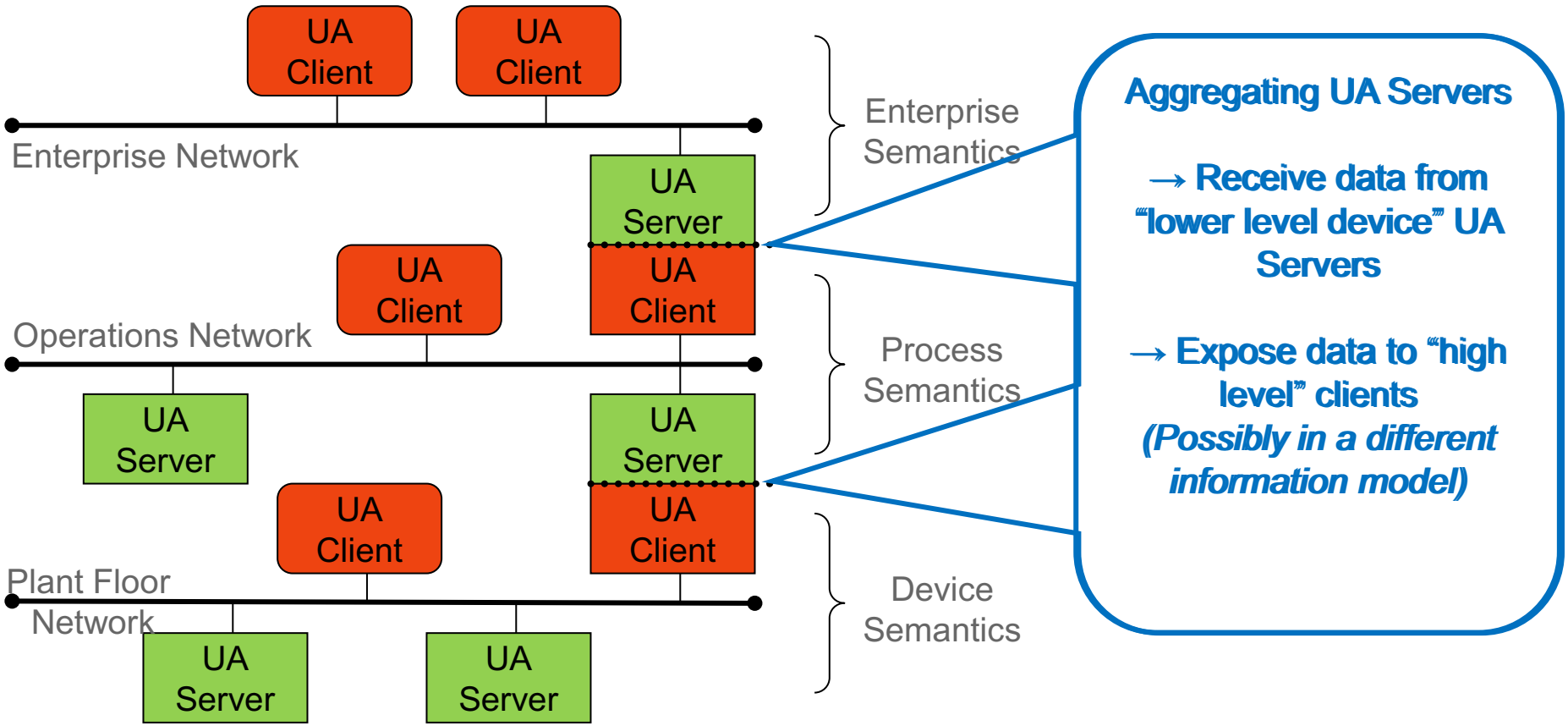
Historical access & aggregates

- **History**
  Store data from events in the **past**
  - When new data arrives the previous value becomes “history”
  - Services: HistoryRead(), HistoryUpdate()

- **Aggregate**
  combine data values from a set of **raw data**
  - An aggregate can be made out of **historical**

    or **real time** data

Aggregating servers in a network

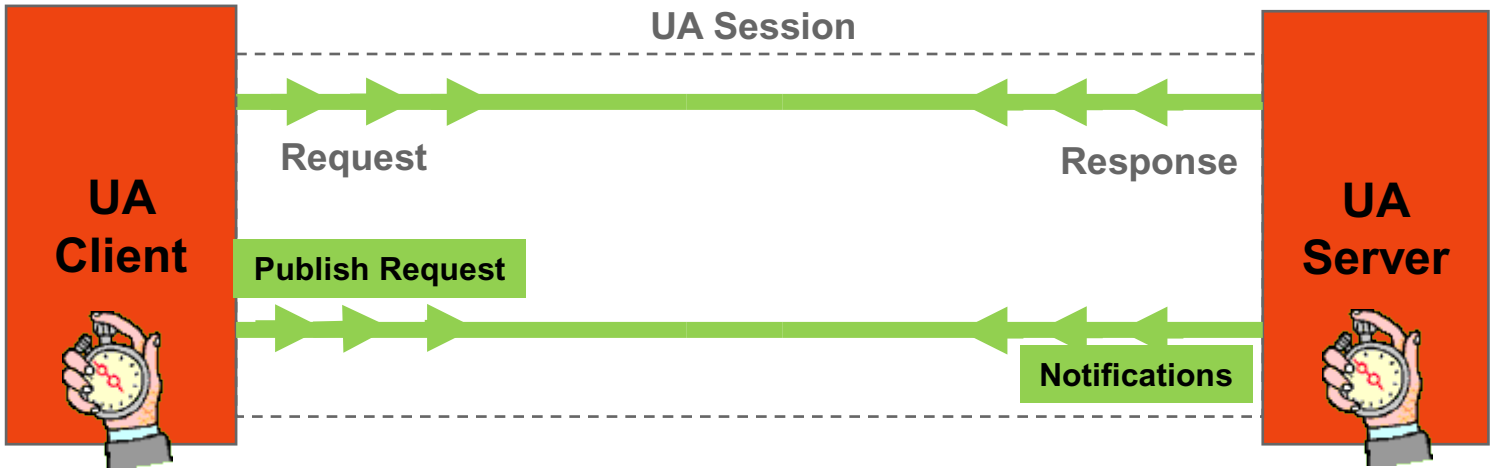


Robustness & reliability

- How does OPC UA realize robustness and reliability?
  - **Detection** of faulty communications
  - Quick **recuperation** after faulty communications
  - **Redundancy** is available

Detection of faulty communications

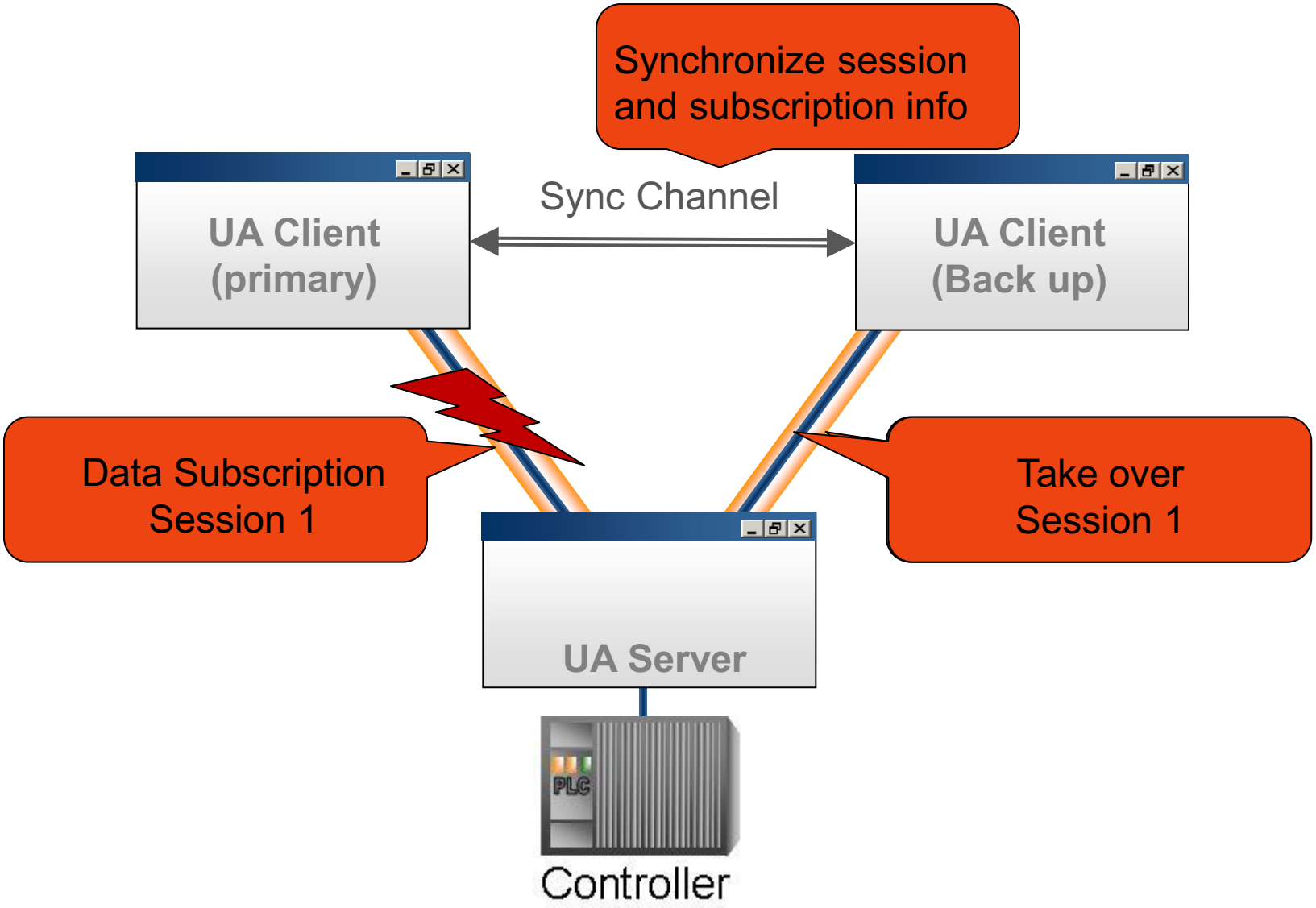
- Setting up **Sessions**
  - Saving **Context**
  - Surviving short interruptions
  - Keeping track of **Lifetime**
- "Keep-alive notifications"
  - "Publishing rate" between client and server



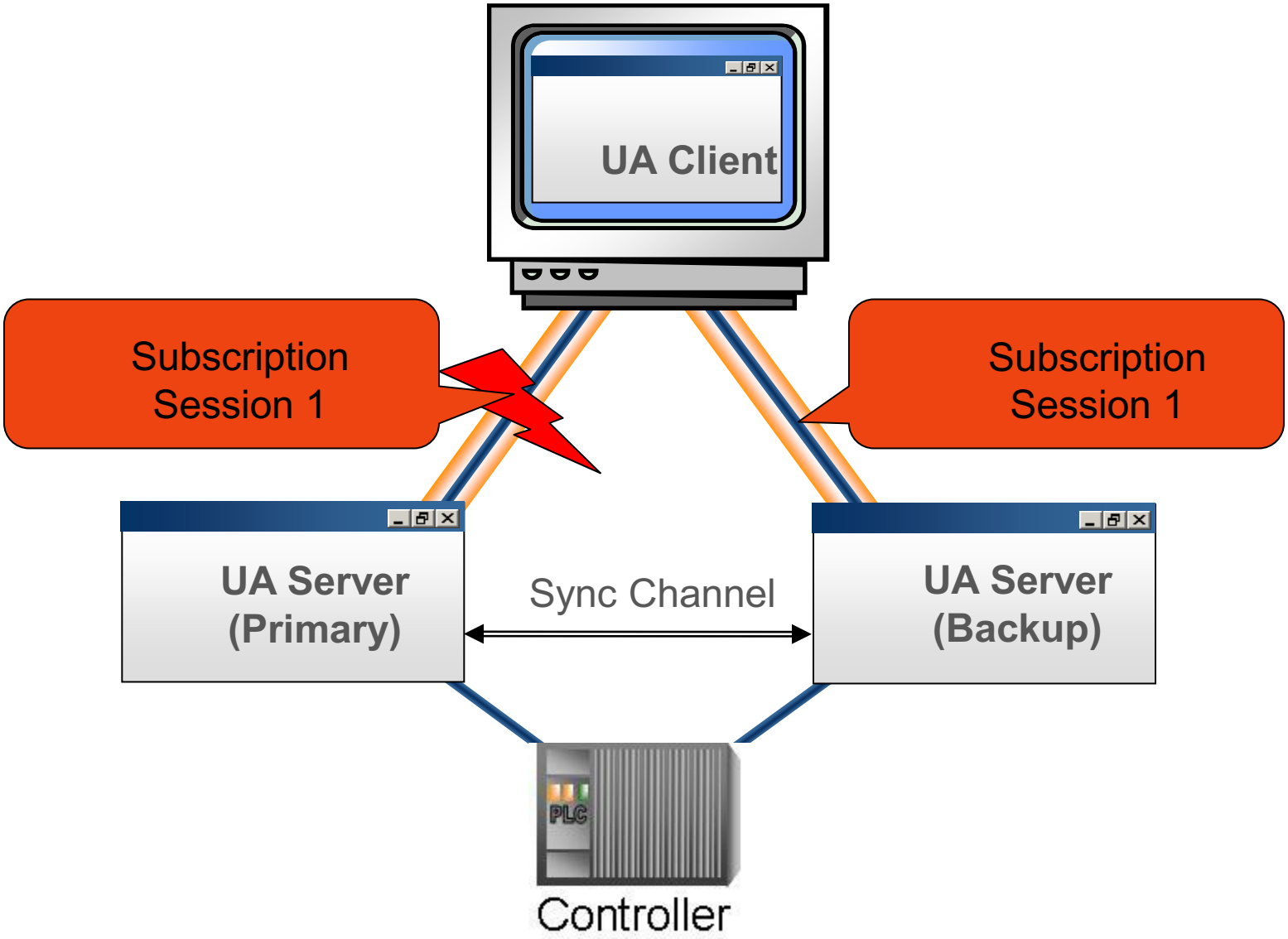
No loss of data

- Servers have **cache** of the latest notifications that were sent
- Every notification has a **sequence number**
- Clients can send a ***Republish()*** request for missing notifications

Client redundancy



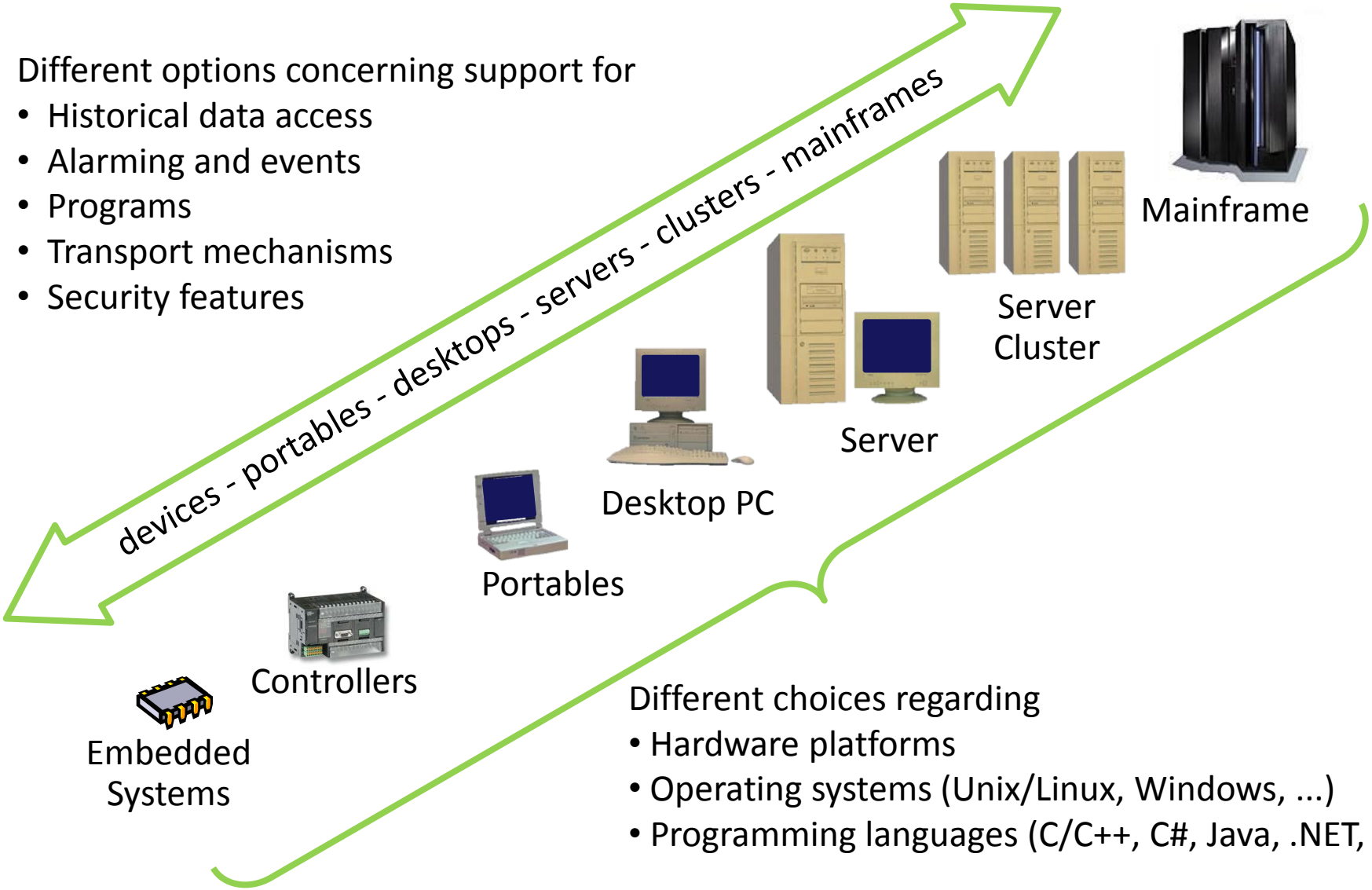
Server redundancy



Profiles and conformance units

Scalability

Different options concerning support for
- Historical data access
- Alarming and events
- Programs
- Transport mechanisms
- Security features



Different choices regarding
- Hardware platforms
- Operating systems (Unix/Linux, Windows, ...)
- Programming languages (C/C++, C#, Java, .NET, ...)

Profiles

- OPC UA Profiles describe features of applications
- Profiles are …
  - laid down in OPC UA specifications part 7
  - comprised of testable units (*Conformance Units*)
  - possibly nested (a profile may inherit from another)
  - grouped into categories
  - published with the UA applications (via software certificates) and via the OPC Foundation website
  - expected to be extended in near future

ConformanceUnits

- Specific set of features
  - group of services,
  - portions of services
  - information models
- Single *ConformanceUnit* can reference several *Services*
- Compose Profiles and may exist in more than one
- Suited for unit tests
- Conformance units are grouped (for easier review)

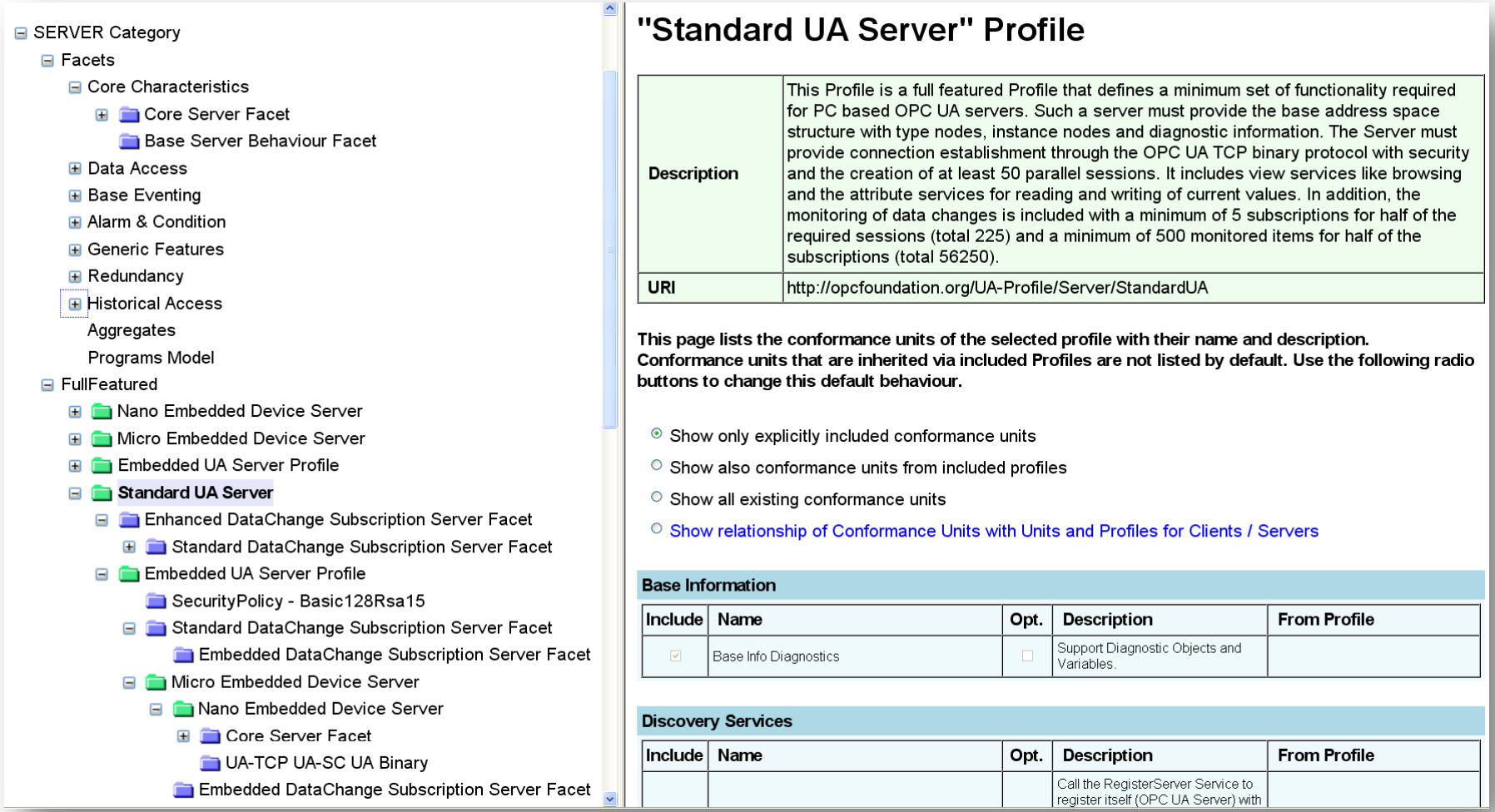
ConformanceUnits groups

- Address Space Model
- Alarms & Conditions
- Attribute Services
- Auditing
- Base Information
- Data Access
- Discovery Services
- Historical Access
- Method Services
- MonitoredItem Services
- Node Management Services
- Programs
- Protocol and Encoding
- Query Services
- Redundancy
- Security
- Session Services
- Subscription Services
- View Services

Profile categories

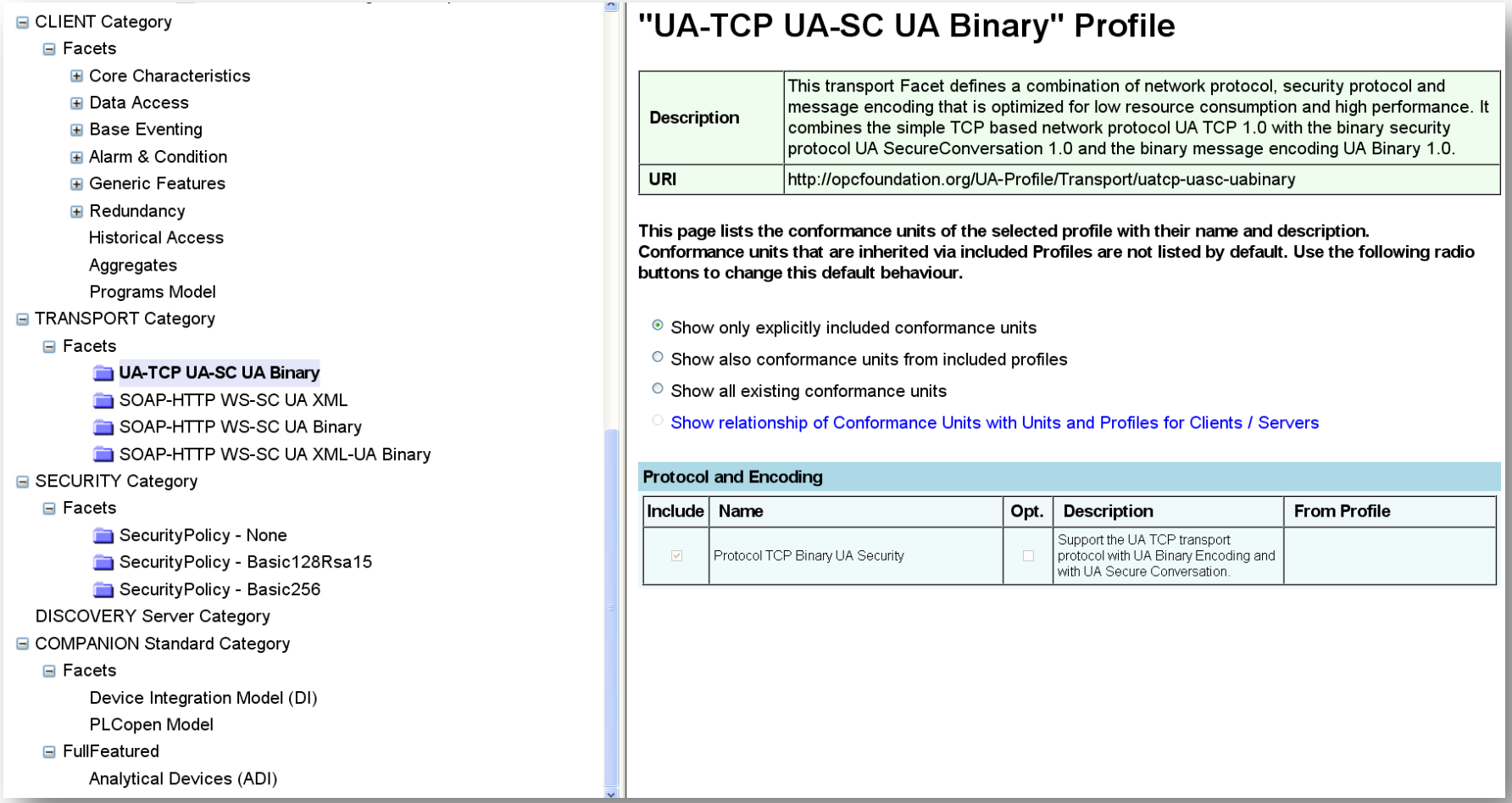
- Client
  - Specify a complete functional set of an OPC UA client
  - Passed in the *CreateSession()* Service request
- Server
  - Specify a complete functional set of an OPC UA server
  - Returned by the *CreateSession()* Service response
  - e.g., Standard UA Server, embedded UA Server,
- Security
  - Specify a security policy (none, Basic128Rsa15, Basic256)
  - Returned by the *GetEndpoints()* Service response
- Transport
  - Specify protocol mapping
  - Part of an *Endpoint* description

Profile categories (cont.)



http://www.opcfoundation.org/profilereporting/index.htm

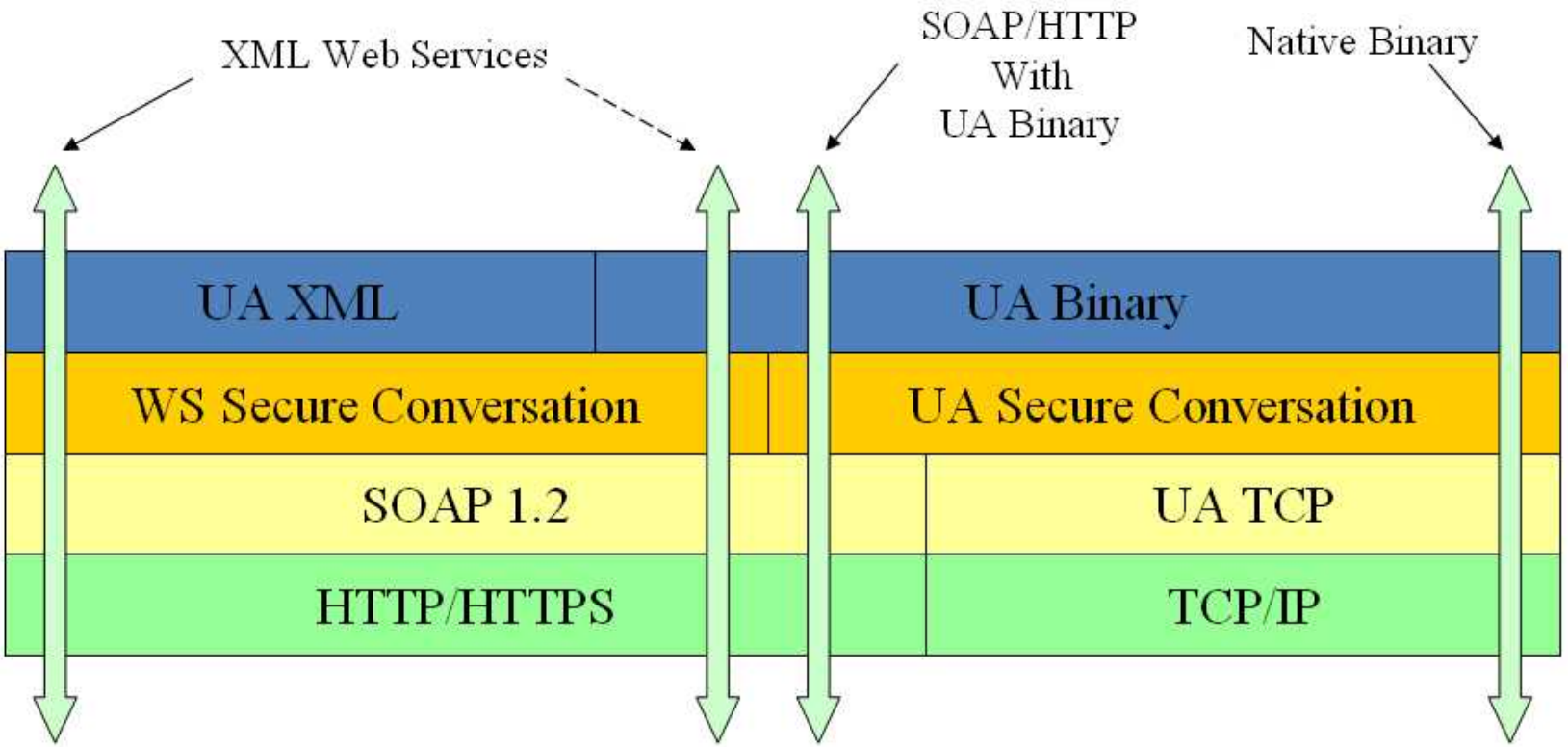
Profile categories (cont.)



http://www.opcfoundation.org/profilereporting/index.htm

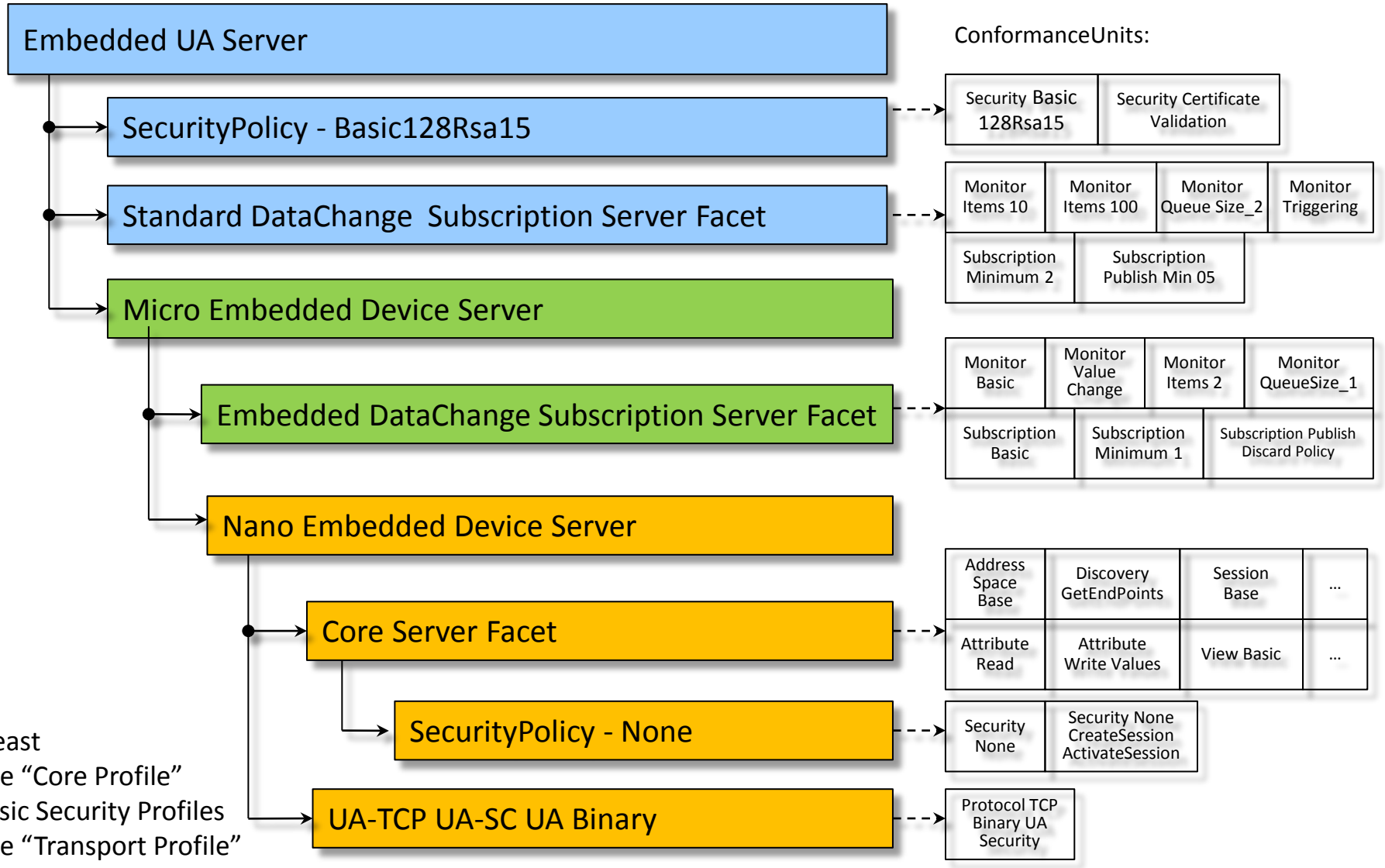
Transport profile facets

- Facet indicates that a Profile is expected to be part of another larger Profile or concerns a specific aspect



Mahnke et. al., OPC Unified Architecture, 2009

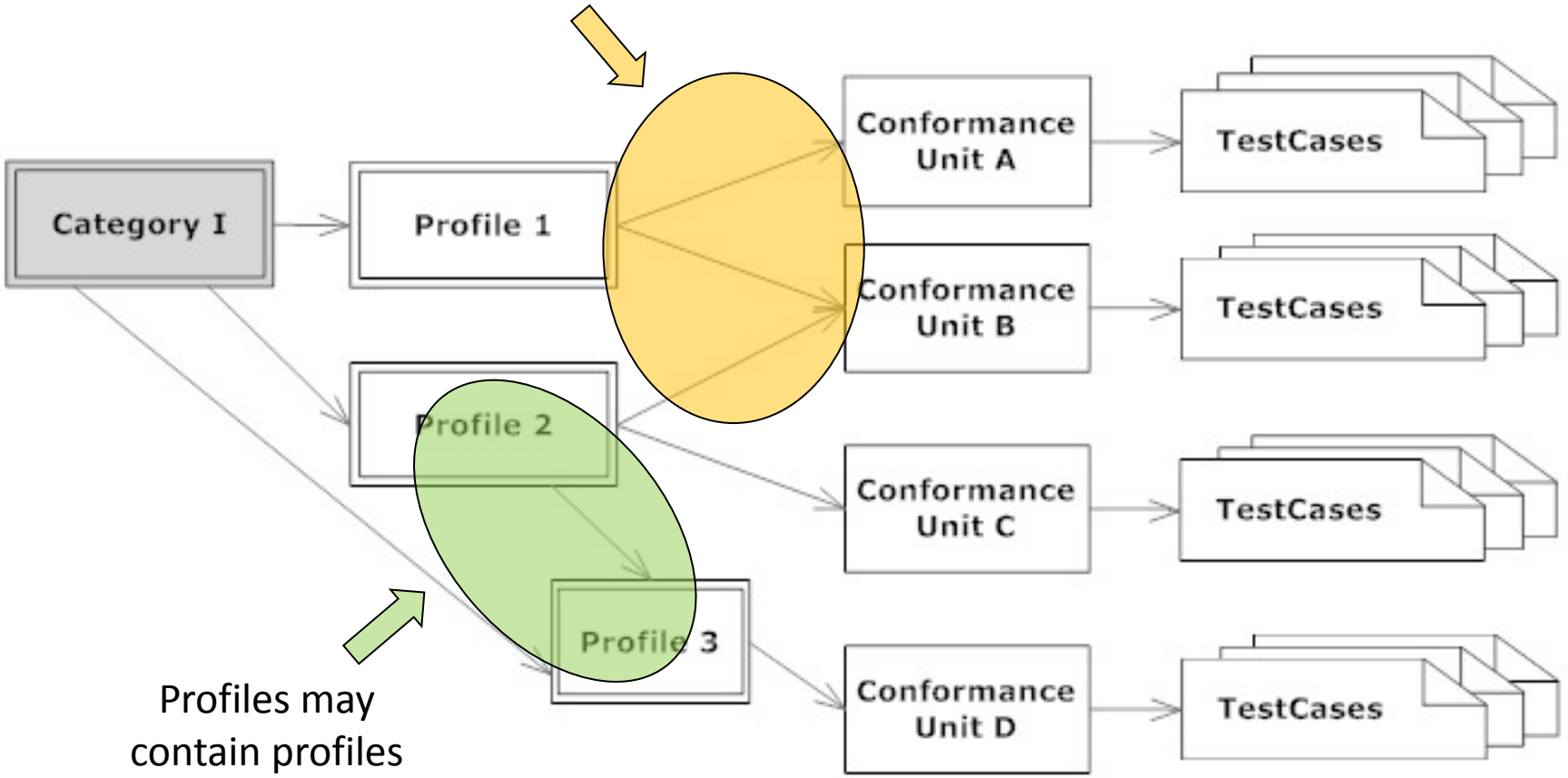
Applications



At least
- one “Core Profile”
- Basic Security Profiles
- one “Transport Profile”

Summary

N:M relationship between
Profiles and ConformanceUnits



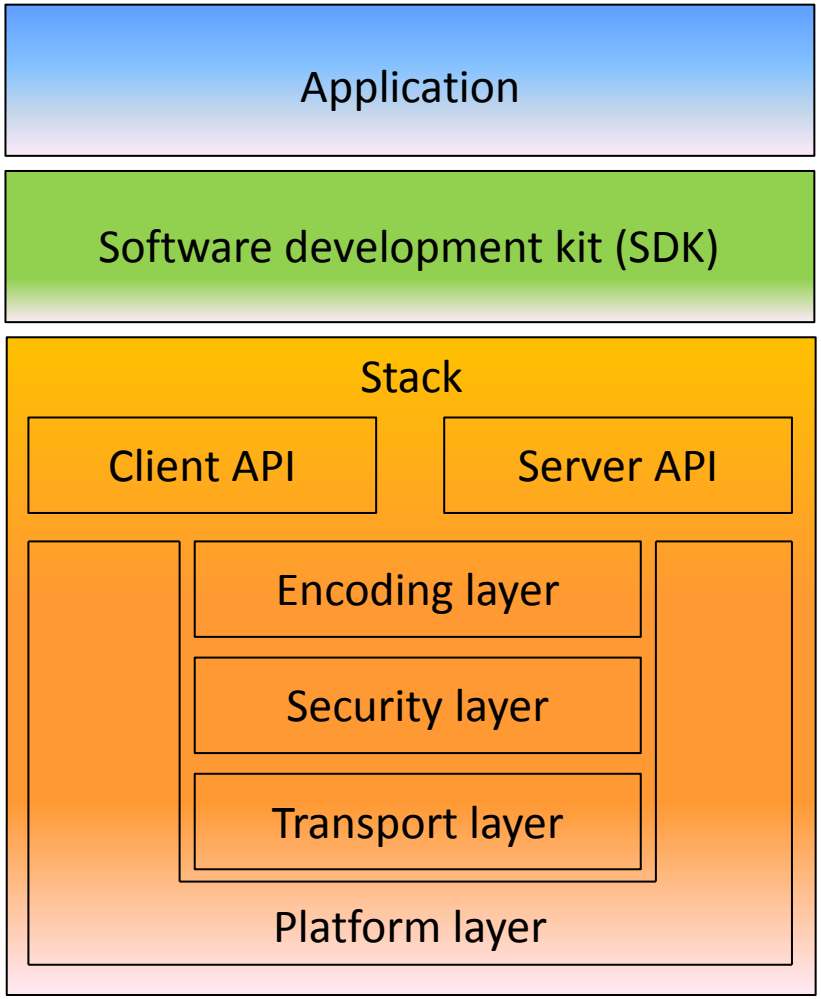
Profiles may
contain profiles

Mahnke et. al., OPC Unified Architecture, 2009

INDIN' 2011
9th IEEE International Conference on
Industrial Informatics

OPC UA software and development kits

Application architecture



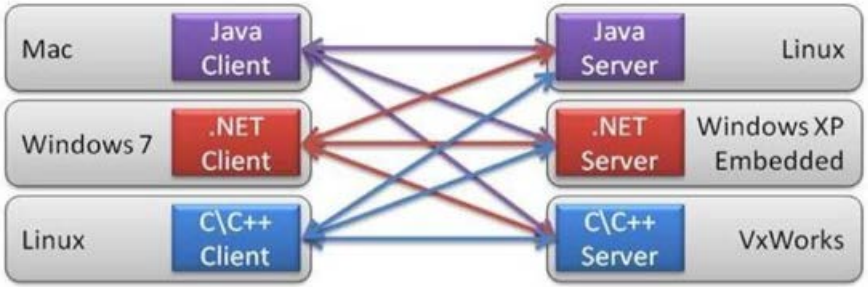
User specific functions

High-level functions

Low-level functions

Key features and benefits

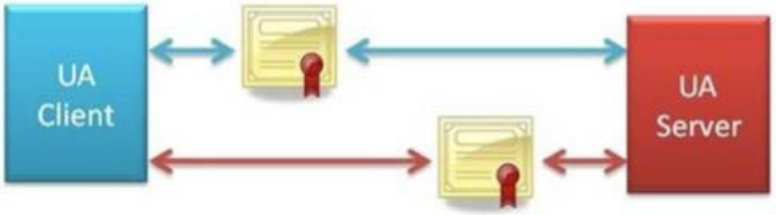
Cross Platform



Unified Access

Internet and Firewall friendly

Standard Security Model

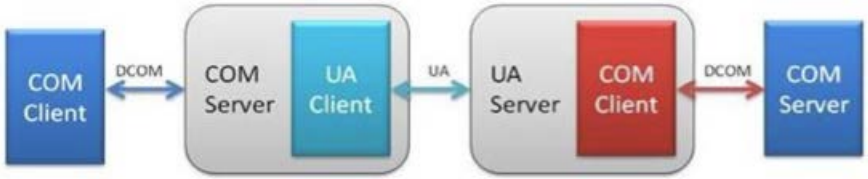


Jim Luth, OPC Day Europe 2011

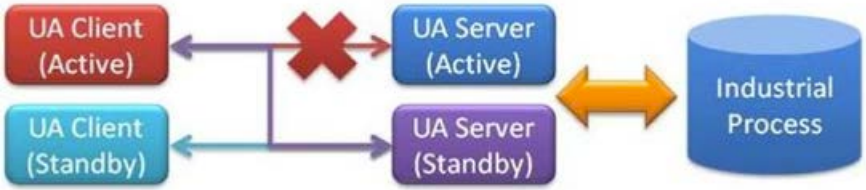
Key features and benefits (cont.)

Single Solution from Embedded to Enterprise

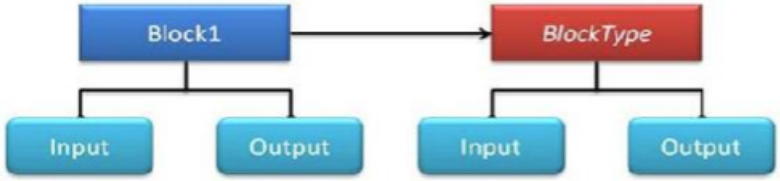
Builds on existing investments in OPC COM



Reliability by design



Flexible object-oriented information model



Jim Luth, OPC Day Europe 2011

OPC UA specification timeline

- 2003 – UA Working Group formed
- 2006 – 1.00 Specification released (no code)
- 2009 – DI and ADI Companion released
- 2009 – 1.01 Specifications released (with code)
  - Includes Alarms & Conditions
- 2010 – IEC 61131-3 Companion released

- 2011 – 1.02 Specification (service release)
  - Includes Historical Access (HA) and aggregates
  - HTTPS binding

OPC UA code timeline

- 2008 – 1.00 Stacks and SDK (ANSI C and .NET
- 2009 – Commercial SDKs available
- 2010 – JAVA Stack available
- 2010 – IOP – more UA than classical products
- 2011Q1 – 1.00 Compliance Test Tool
- 2011Q2 – OPC UA Certification available

- 2011Q4 – 1.01 Stacks and SDK
  - (Java, ANSI C and .NET)
  - HTTPS binding

OPC UA protocols and encodings

- Current bindings
  - UA Binary over TCP
  - SW-SecureConversation over HTTP

- Future bindings
  - XML/SOAP over HTTPS
  - UA over HTTPS

➔Enables HTTP on platforms that do not support WS-SecureConversation

OPC UA deliverables

- OPC UA Specifications:
  - Part 1 – Downloadable by everyone
  - All other parts – Free download to OPC Members
- OPC Corporate Members
  - Royalty free license to all UA source and binaries:
    - Sample code (based on several UA Profiles)
    - UA Wrapper & Proxy (several COM specs)
  - May re-distribute binaries, but must provide significant added value
- Commercial SDKs & deliverables

Commercial OPC UA deliverables

Language	Platform	UA Stack	UA SDKs Toolkits	Examples Server/Client
.NET	x86	●	●	●
ANSI C	all	● / √	√	√
C++	all*	√	√	√
Java	some	● / √	√	√

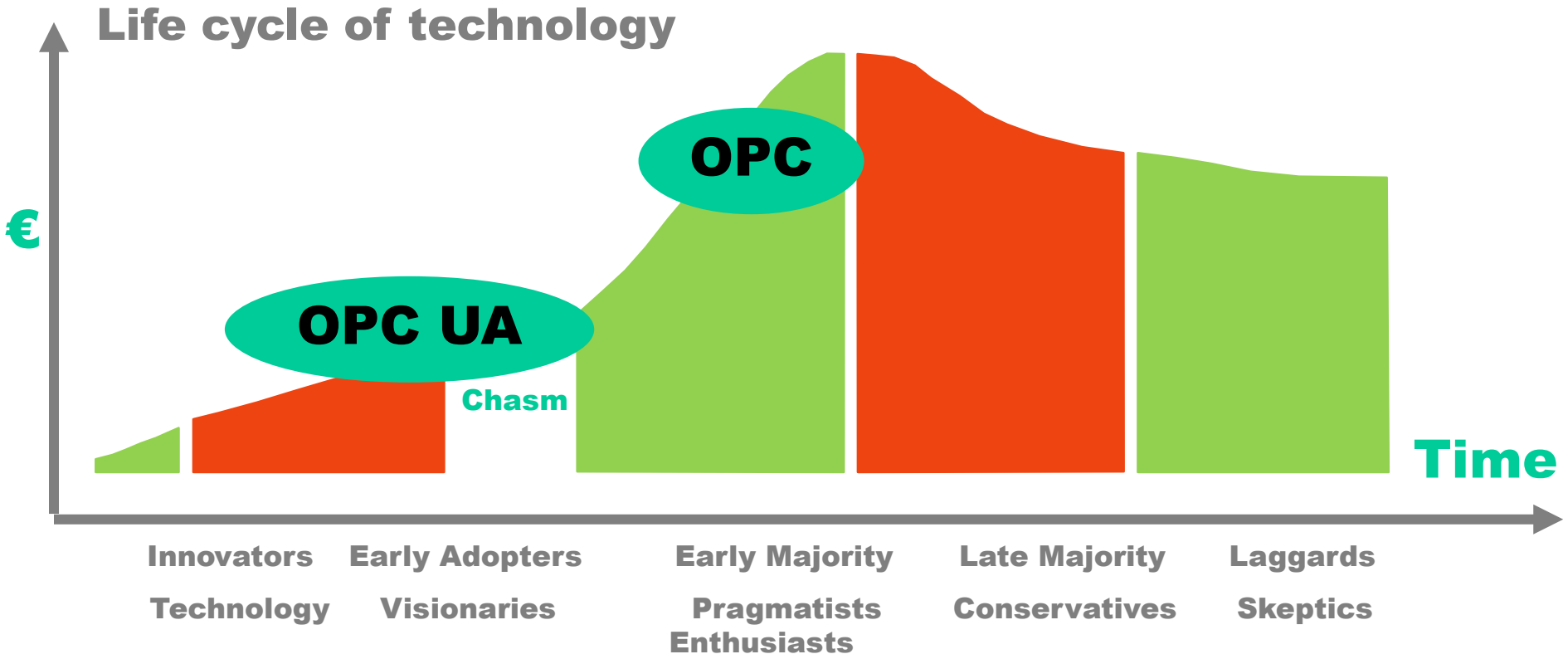
* all platforms that (at least) support C++ compiler (multi task / multi thread)
√ available at Unified Automatin GmbH
● available at OPC Foundation (corporate members only)

Unified Automation (www.unified-automation.com)

Available OPC UA commercial products

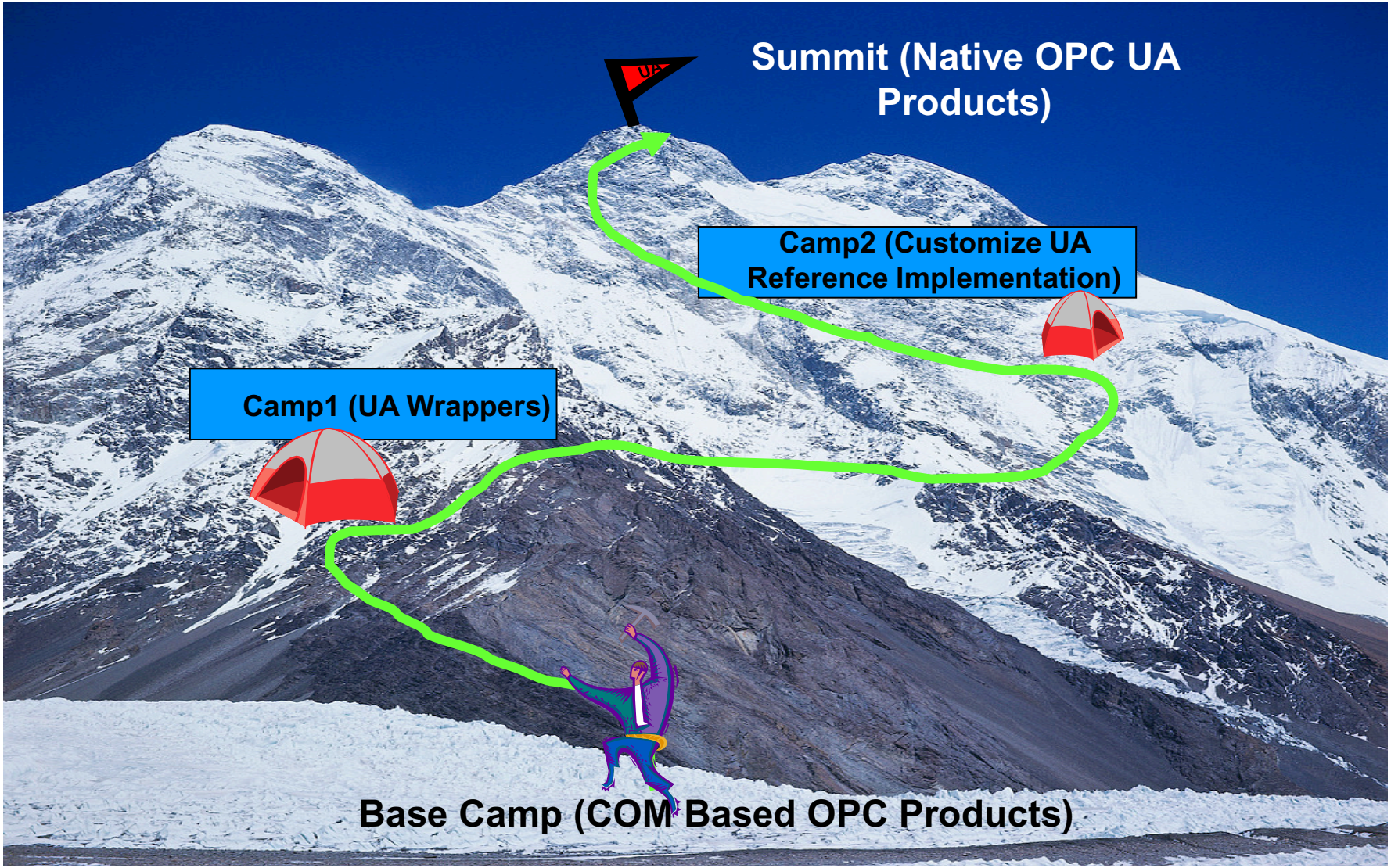
- Tool vendors:
  - Unified Automation; Kepware; Softing; Matrikon; Prosys; Software Toolbox;
- UA Modeler vendors:
  - CAS (.NET); HB-Softsolution (Java)
- PLC Vendors (Native OPC UA Servers):
  - Siemens; Beckhoff
- Native OPC UA Clients for visualisation and monitoring
  - ICONICS; Inductive Automation; Siemens
- More vendors on the OPC Foundation website

Market penetration OPC and OPC UA



OPC Foundation

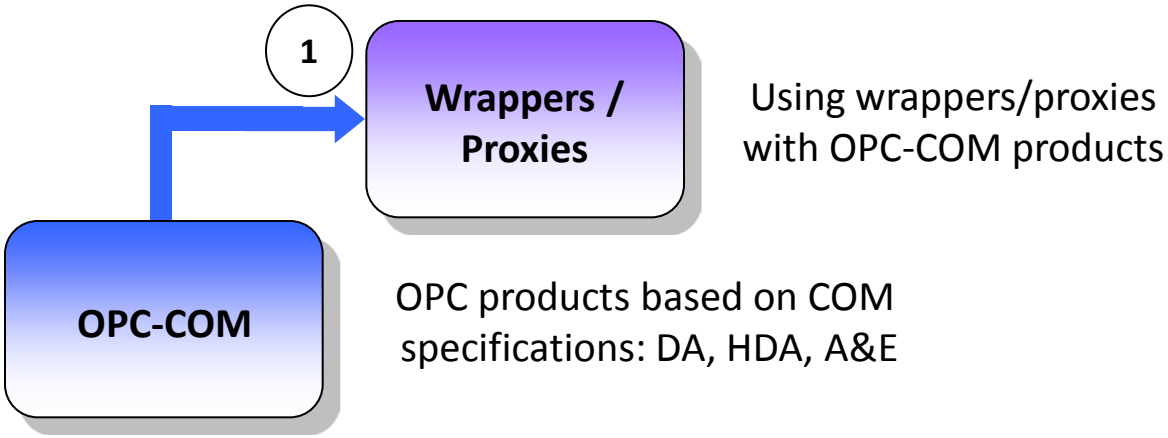
Scaling the mountain in stages



Marc Hensley, OPC Unified Architecture, DevCon 2006

Stage 1 – Using migration adapters

- Using wrappers and proxies
  - Wrappers for OPC-COM Servers
  - Proxies for OPC-COM Clients
  - Separate wrappers for DA, HDA, A&E
  - Separate proxies for DA, HDA, A&E



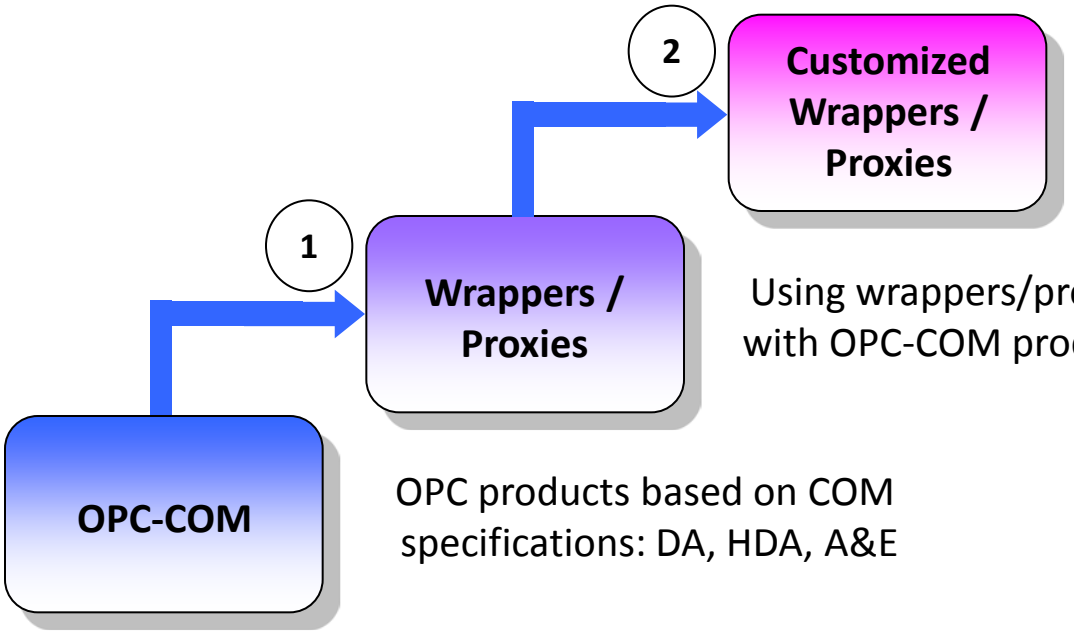
Using wrappers/proxies with OPC-COM products

OPC products based on COM specifications: DA, HDA, A&E

Jürgen Lange, OPC Day Europe 2011

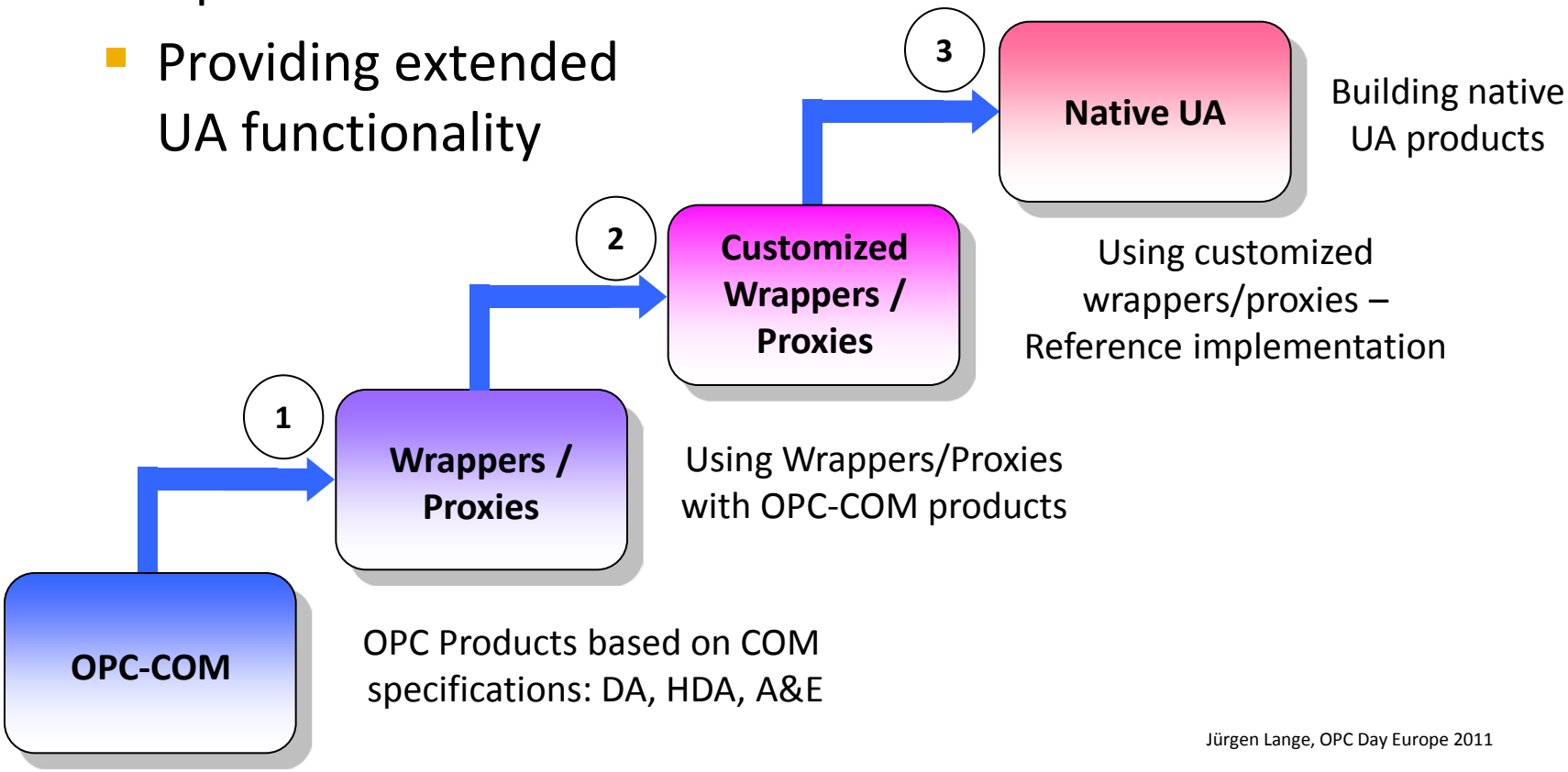
Stage 2 – Reference implementation

- Reference implementation
  - Building UA apps on top of existing infrastructure
  - Customizing wrappers and proxies
  - Providing limited UA functionality



Stage 3 – Native UA implementation

- Native UA implementation involves
  - Building UA apps on top of new infrastructure
  - Optimized access to data
  - Providing extended UA functionality



Summary

- Integrate success points of DCOM OPC
- New key features and benefits in OPC UA
- First release OPC UA in 2006
- Working groups carry on extending the specs
- OPC Foundation deliverables and commercial SDKs
- Early adopters phase (150+ OPC UA available on the market)
- OPC UA is complementary with classic OPC
- Migration path available for DCOM vendors

References

[1] OPC Unified Architecture Specification, OPC Foundation, 2009.

[2] OPC Unified Architecture, IEC 62541, 2010, Current status: Approved for FDIS circulation.

[3] W. Mahnke, S.-H. Leitner, and M. Damm, OPC Unified Architecture, Springer, 2009.

[4] KNX Specification Version 2.0," Konnex Association, Diegem, 2009.

[5] BACnet – A Data Communication Protocol for Building Automation and Control Networks, ANSI/ASHRAE 135-2008, 2008.

[6] BACnet – A Data Communication Protocol for Building Automation and Control Networks, ANSI/ASHRAE 135-2008: Addendum i, 2010, status: 4th public review.

[7] Wolfgang Granzer, Wolfgang Kastner, and Paul Furtak. KNX and OPC UA. In Konnex Scientific Conference, November 2010.

References

[8] Andreas Fernbach, Wolfgang Granzer, and Wolfgang Kastner. Interoperability at the Management Level of Building Automation Systems: A Case Study for BACnet and OPC UA. In *Proc. of 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '11)*, September 2011.

[9] Dirk van der Linden, Herwig Mannaert,Wolfgang Kastner, Vincent Vanderputten,Herbert Peremans, Jan Verelst. An OPC UA Interface for an Evolvable ISA88 Control Module. In *Proc. of 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA '11)*, September 2011.

[10] Dirk van der Linden, “Agile role-based decision support for OPC UA profiles”, Doctoral seminar "Communication interfaces and information modeling in OPC Unified Architecture", University of Antwerp, 14 March, 2011

[11] DevCon 2006 OPC Unified Architecture, OPC Foundation, Munich, 2006

[12] DevCon 2008 OPC Unified Architecture, OPC Foundation, Munich, 2008

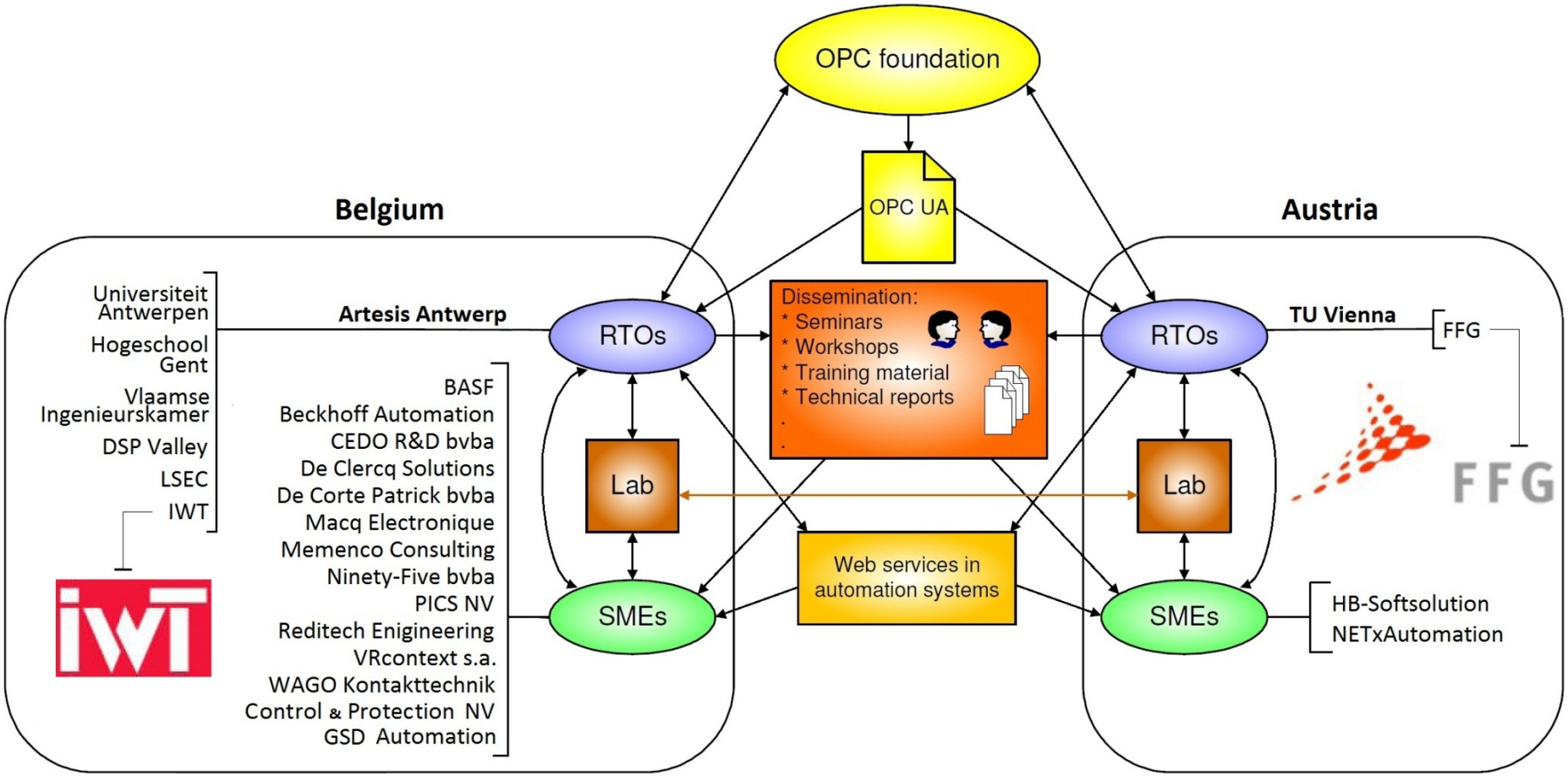
[13] OPC Day Europe, OPC Foundation, Walldorf, 2011

Demonstration

Web-based Communication in Automation

- Technology survey
  - Web-based technology (for automation)
  - Thorough analysis of OPC UA specification
  - Preparation in a “concentrated” fashion
- Interoperability
  - Definition of profiles/benchmarking
  - Information modelling for different target domains
  - Setup of test labs
- Further Goals
  - Security
  - Compatibility issues
  - Strong interaction between SMEs and RTOs
  - Dissemination (conferences/workshops/fairs, website)

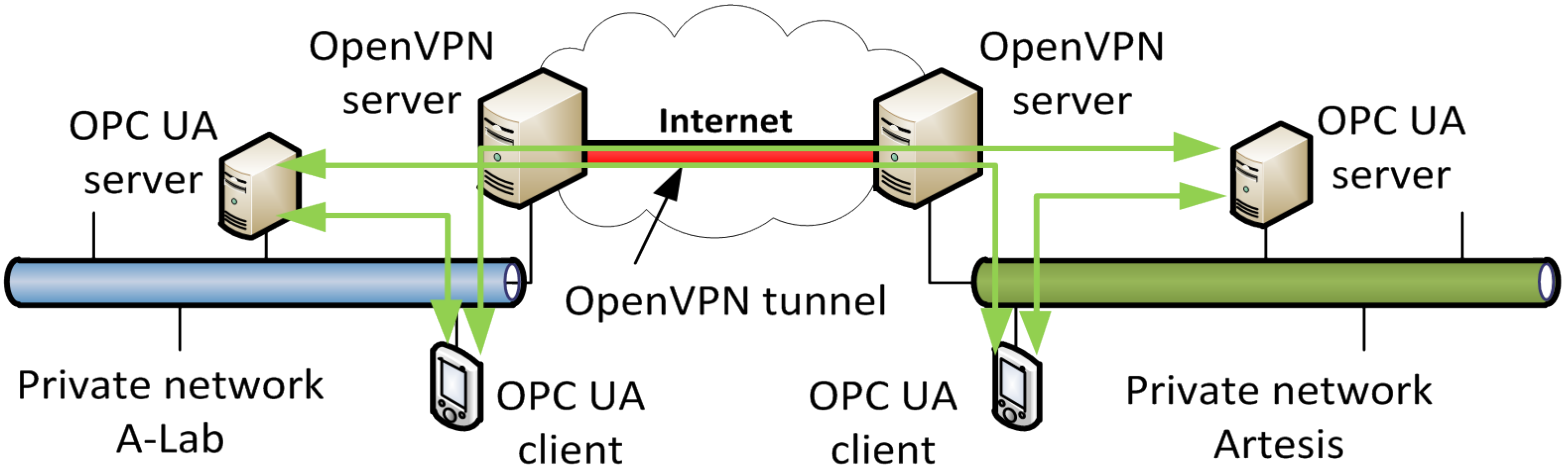
Consortium structure



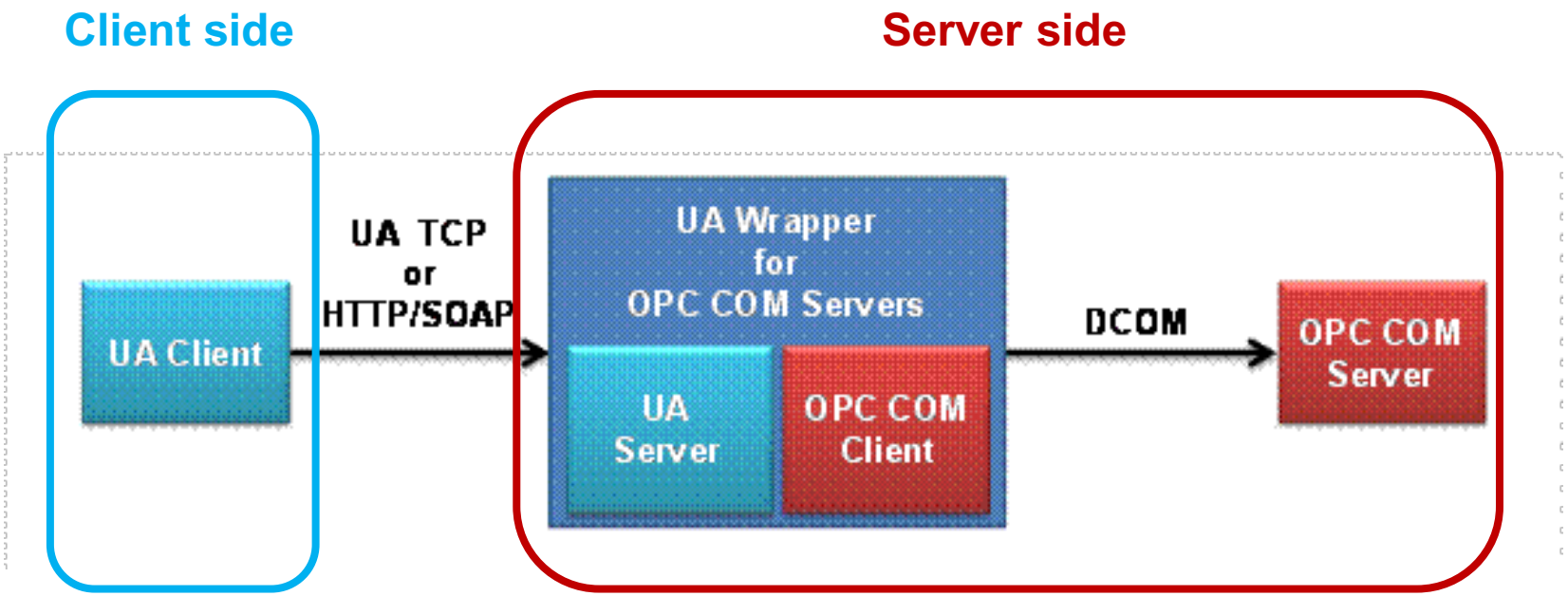
EraSME

Interoperability lab setup

- Each lab has its own private network
- OpenVPN tunnel
- Isolated and secure network

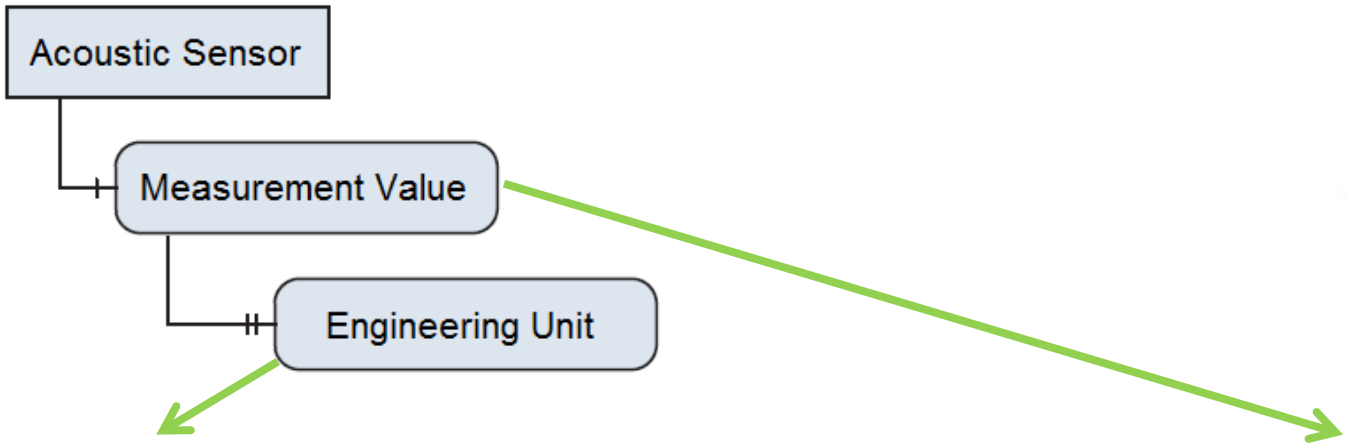


Demo: using a “Wrapper”



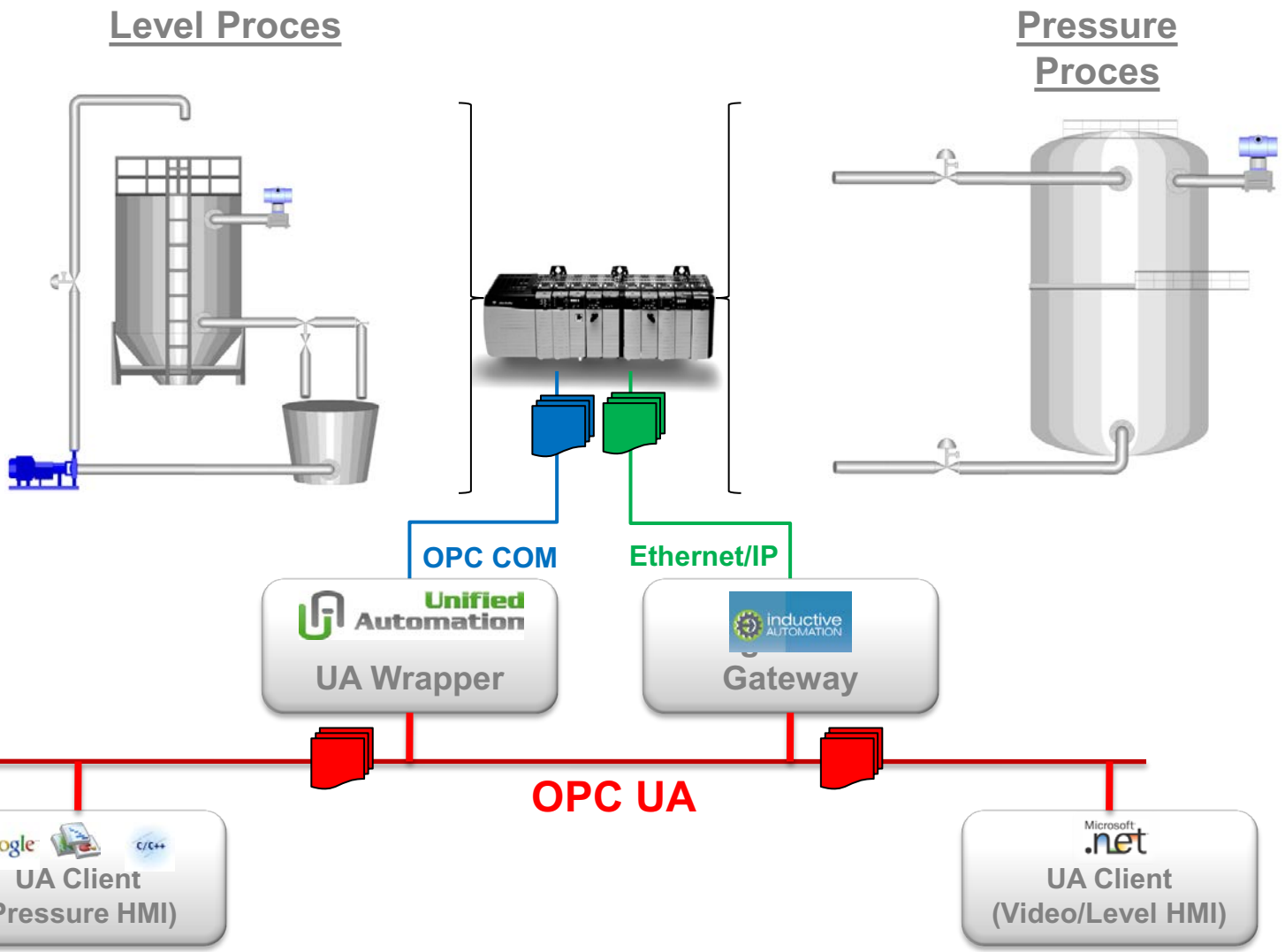
Demo: Meta Data

- **Meta Data** example: Acoustic distance sensor



- **Engineering Unit** as property of a variable **Distance**
- Create a MonitoredItem for the variable **Distance**
  ➔ receive a *notification when distance value changes*
- When **Engineering Unit** changes
  ➔ the next **Distance** notification's *StatusCode* will have *SemanticsChanged flag set to '1'*
  ➔ Client reads the changed Engineering Unit

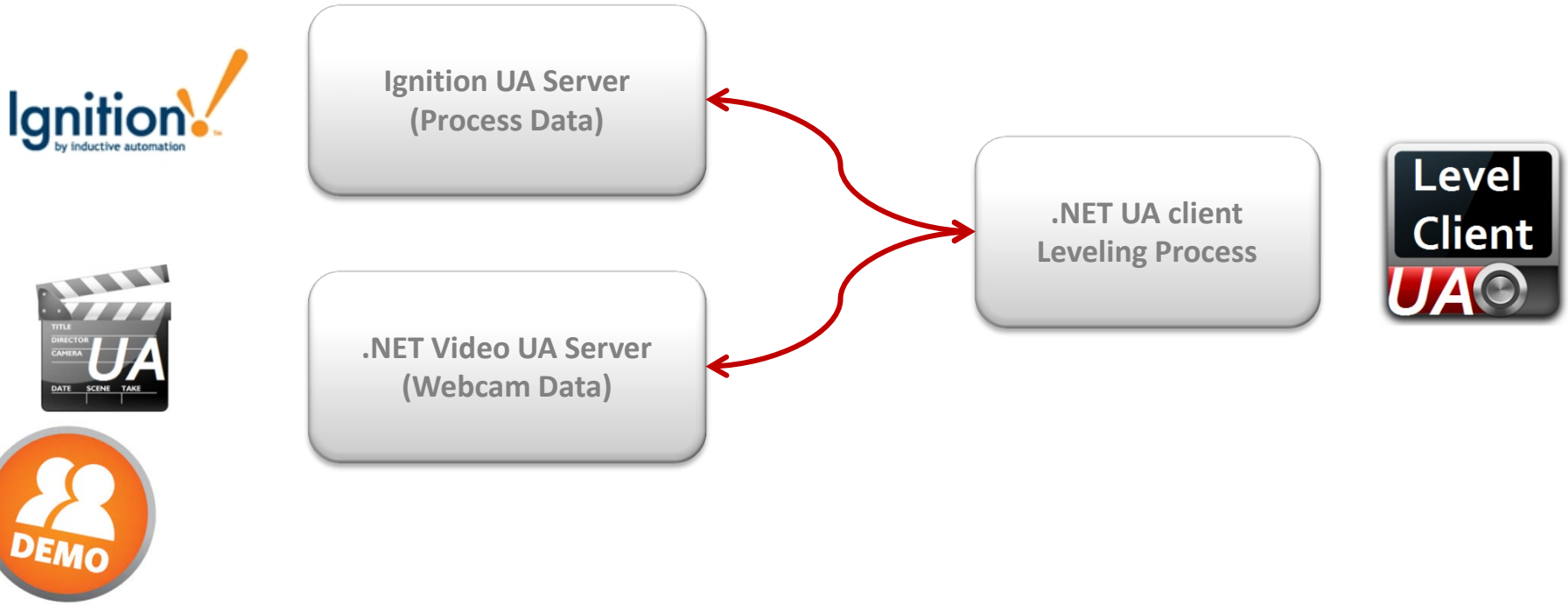
Demo: Lab equipment



Demo: Lab equipment

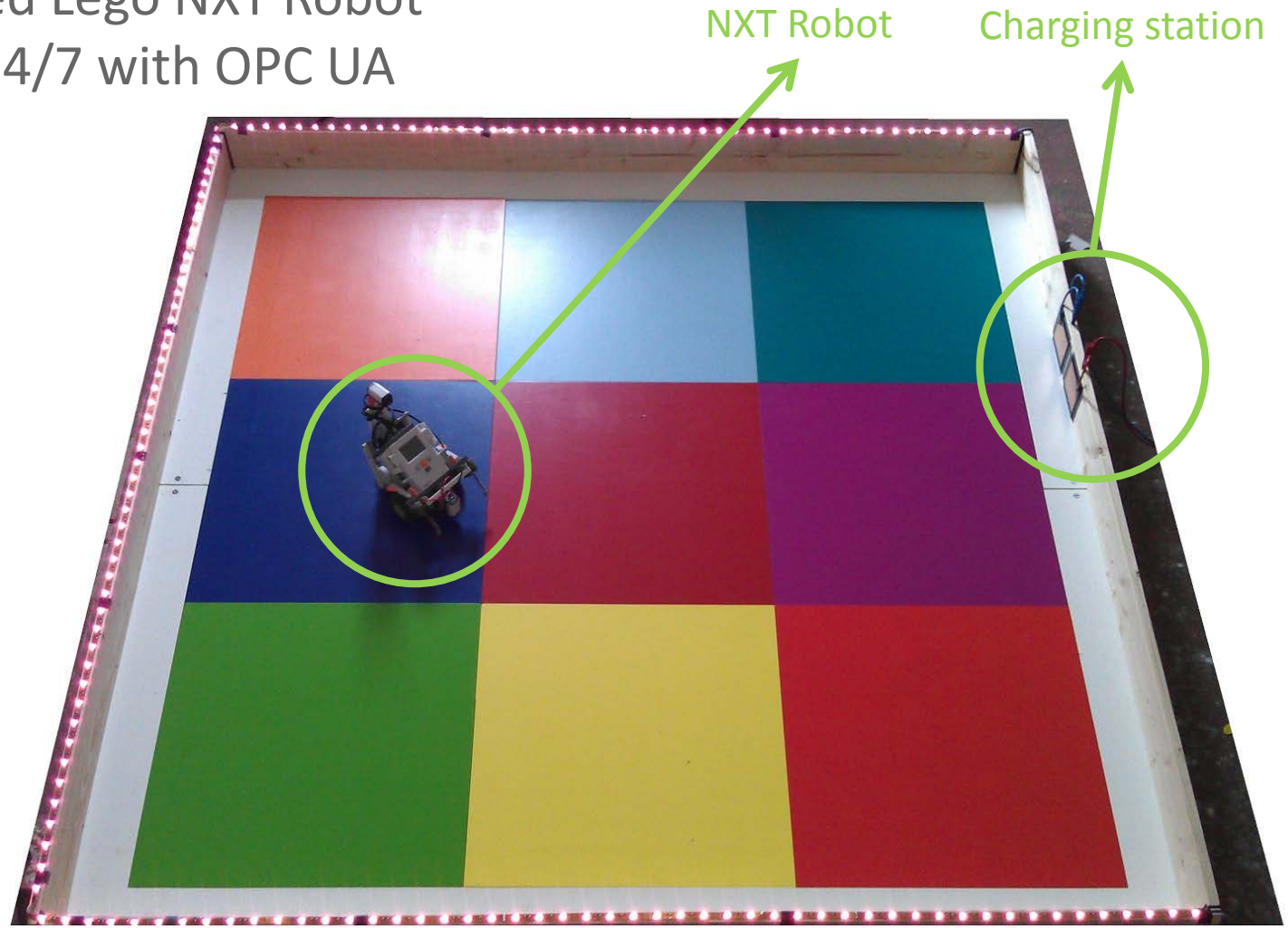
- Video UA server → .NET SDK of the OPC Foundation
- Process Data UA server → Ignition UA Server
- Level Process UA client → .NET SDK of the OPC Foundation

└> 2 OPC UA Connections



Demo: Remote NXT Robot Control

- A customized Lego NXT Robot accessible 24/7 with OPC UA



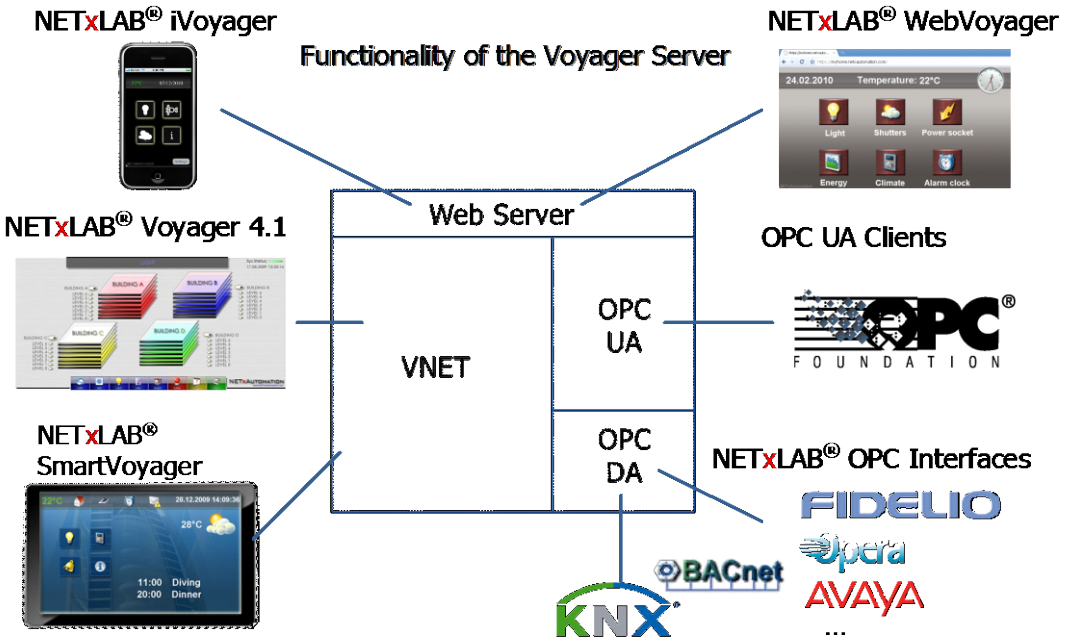
DEMO

Demo: OPC UA Server interfacing BACnet

- HB-Softsolution:
  - OPC UA Server SDK
  - OPC UA Model Designer
- ASG:
  - Information model for basic BACnet objects
  - BACnet driver interfacing the OPC UA Server
    - Information model browse routine
    - Read-/WriteProperty service
    - ChangeOfValue Subscription service

Demo: OPC UA for building automation

- NETxLAB® Voyager Global Visualization System developed by NETxAutomation (Austria)
  - Contact: http://netxautomation.com
- Client/server solution for control and visualization
- Client interfaces:
  - OPC DA 2.0
  - VNET (proprietary)
  - **OPC UA**
- Bus interfaces:
  - KNXnet/IP
  - BACnet/IP
  - Direct KNX (e.g., USB)
  - …



DEMO

Contact

- **Project homepage:** www.webcom-eu.org
- **Artesis University College of Antwerp**
  - Contact: Dirk van der Linden
    dirk.vanderlinden@artesis.be
  - Website: www.artesis.be
- **Vienna University of Technology**
  Automation System Group
  - Contact: Wolfgang Kastner
    k@auto.tuwien.ac.at
  - Contact: Wolfgang Granzer
    w@auto.tuwien.ac.at
  - Website: www.auto.tuwien.ac.at

Thank you for your attention!