OPC Unified Architecture

An Introduction to the IEC62541

ARD-ST3 Workshop „SRF controls and CW operation“
Helmholtz Zentrum Dresden Rossendorf

Chris Iatrou

Dresden, 25.11.2014

Agenda

- Introduction
- Background: Short history of OPC/OPC UA
- OPC UA: Data modelling concepts
- OPC UA: Data access principles and services
- OPC UA: Overview of other features
- Summary

Current challenges in process automation

Professur für PROZESS LEITTECHNIK

Instrumentation and Automation

- Productivity
- Ressource Efficiency
- Quality
- Safety

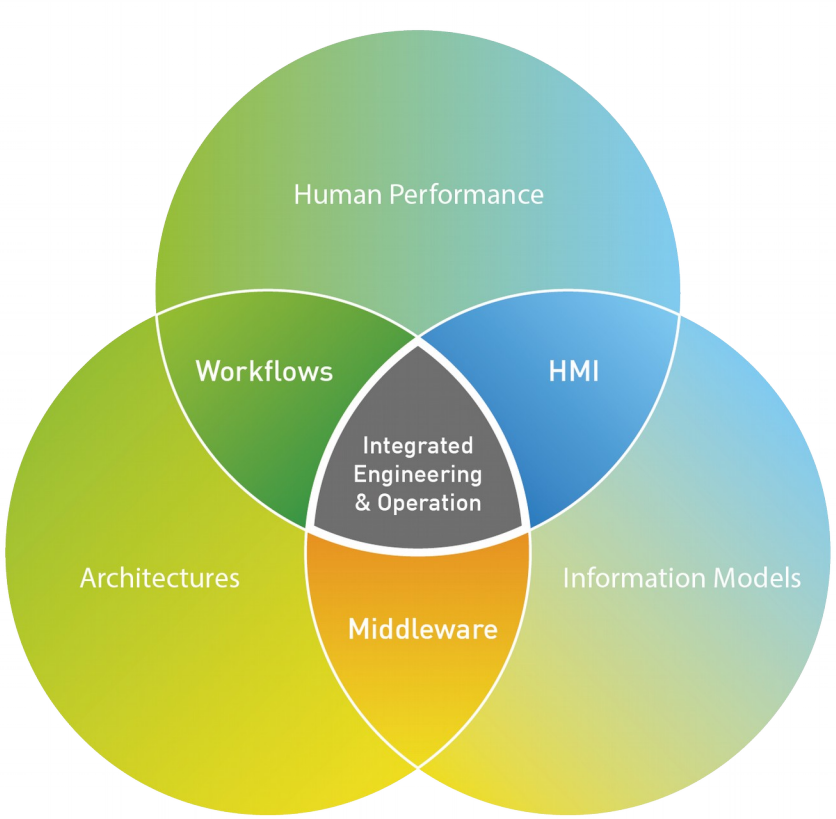
Communications and Integration

- Transparency
- Felxibility
- Speed
- Security

Research Profile

Professur für PROZESS LEITTECHNIK

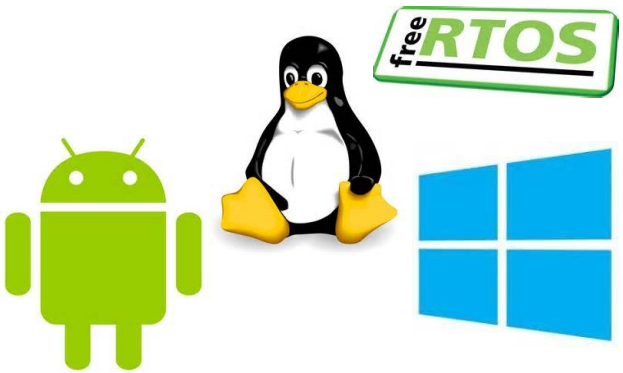
- Engineering Support Systems
  - Automation of Automation
  - Integrated Engineering-processes for modular plants
  - Design for Evolution
- Operation Support Systems
  - Initial placement into operation
  - Maintenance support
  - Integrated operational concepts for control rooms and field applications

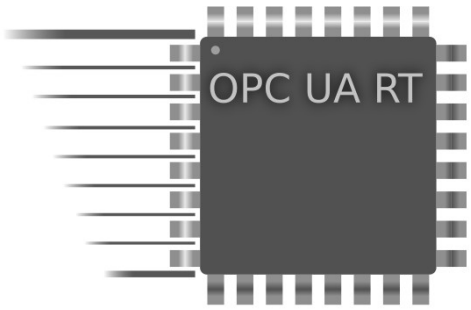


Prior works and experiences

Professur für PROZESS LEITTECHNIK

open62541 ACPLT Professur für PROZESS LEITTECHNIK



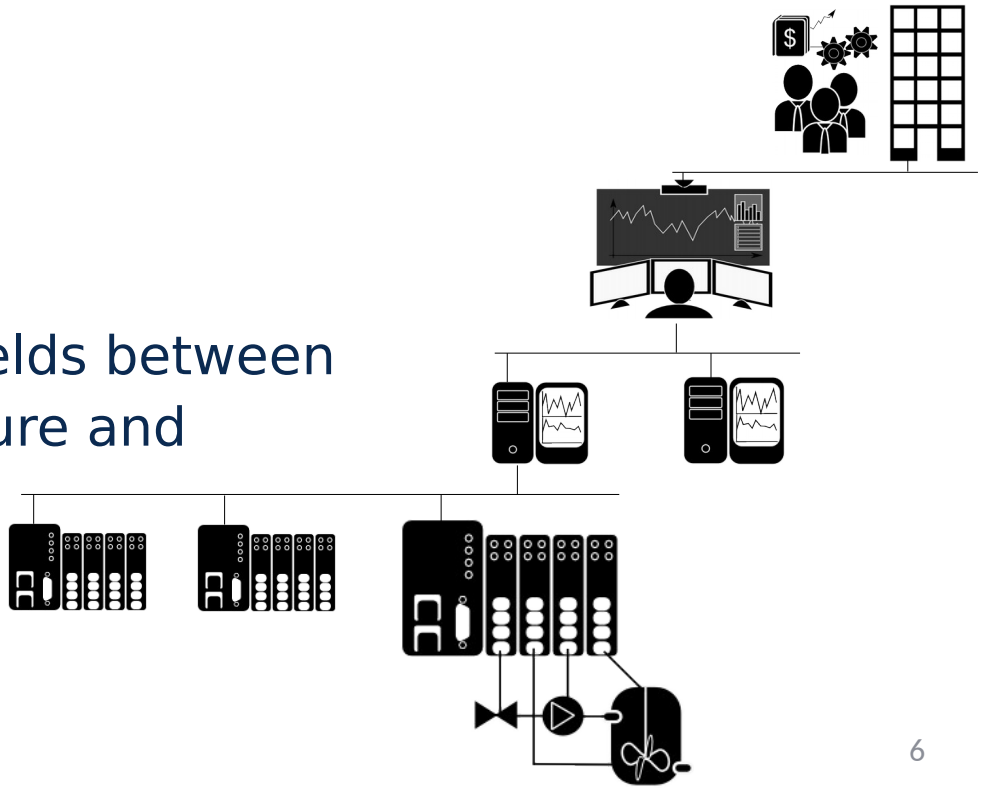


- Open62541
- OPC UA RT
- Real-Time Android
- Applied abstract control
- Distributed Systems Engineering
- Integrated simulations

Background: OPC

Professur für PROZESS LEITTECHNIK

- 1996: Object Linking and Embedding for Process Control
  → RT Linking of control devices
  → Based on Microsofts DCOM
  - Data Access (DA)
  - Historical Data Access (HDA)
  - Alarms and Events (A&E)
- Core function: Access data fields between devices of different architecture and connectivity.



Background: OPC UA

Professur für PROZESS LEITTECHNIK

OPC UA Specification:

1. Concepts
2. Security Model
3. Address Space Model
4. Services
5. Information Model
6. Mappings
7. Profiles
8. Data Access
9. Alarms and Conditions
10. Programs
11. Historical Access
12. Discovery
13. Aggregates

- Specification initiated 2006
- Accepted as IEC62541 in 2010/2011
- Redesign of shared information space:
  - Server/Client Model
  - Integrated X509 Security
  - Plattform independent
  - Vendor indepependent
  - Transport independent
- Information moved to „source“
  → Server is on device

Specification Overview

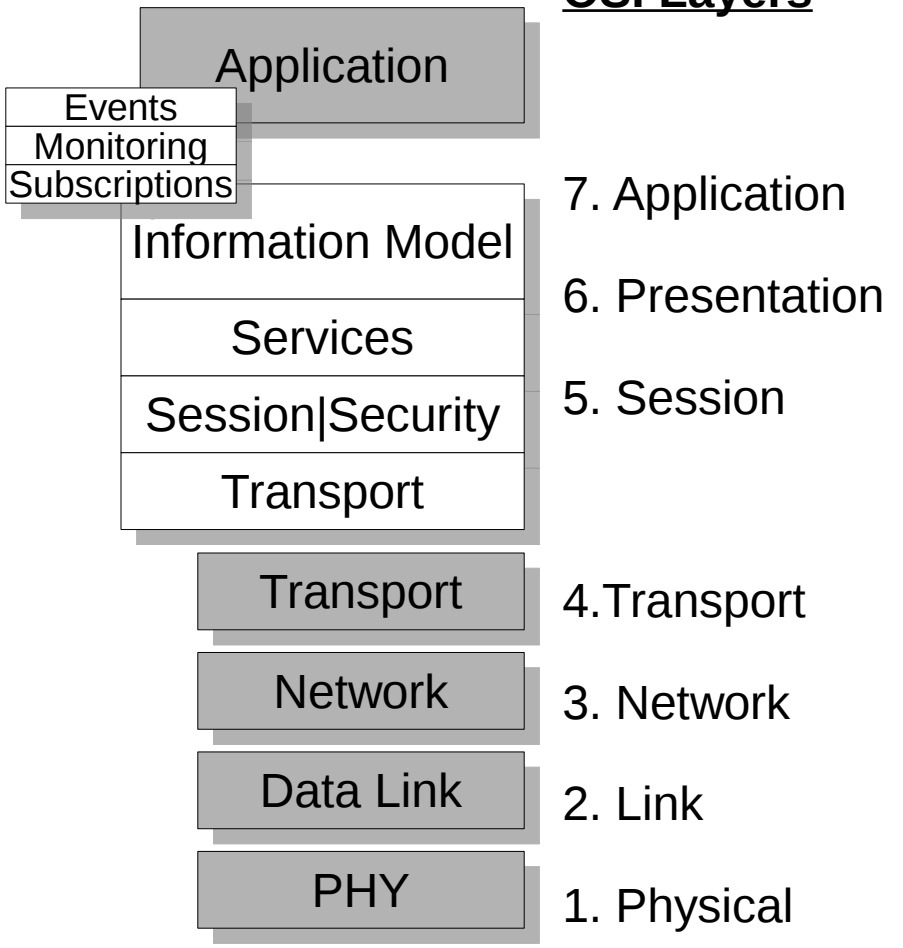
Professur für PROZESS LEITTECHNIK

OPC UA Specification:

1. Concepts
2. Security Model
3. Address Space Model
4. Services
5. Information Model
6. Mappings
7. Profiles
8. Data Access
9. Alarms and Conditions
10. Programs
11. Historical Access
12. Discovery
13. Aggregates

- OPC UA *is not* OPC
- OPC UA is generalized:
  - Not automation-specific
  - Meta-Modelling capabilities
    → OPC UA can model anything (model-wars)
- OPC UAs core functionality can be split into
  - Information modelling concepts
  - Information access (server client)
  - Control/Automation concepts
- OPC UA is flexible:
  - Not all features need to be implemented
  - Amount of functionality defines stack footprint

Serverside Stack components



OSI Layers

7. Application

6. Presentation

5. Session

4.Transport

3. Network

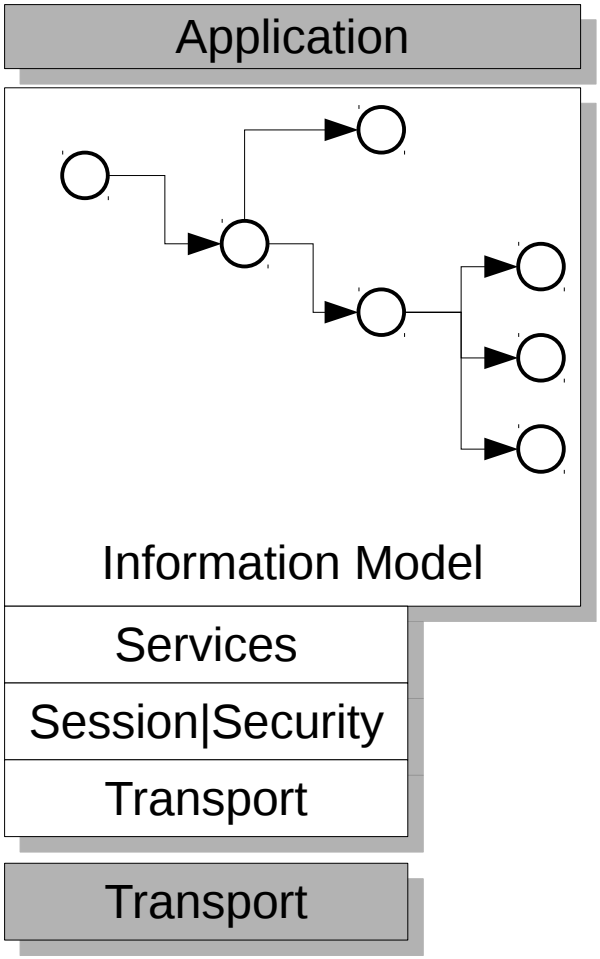
2. Link

1. Physical

- OPC UA Serverside:
  - Manages information model
  - Offers interface to information for clients („Services“)
  - User-side application „acts on“ information model changes
- Networking Layers:
  - Several transport mechanisms: TCP/Binary, TCP/XML
  - Not dependant on TCP
    → reimplements several features
    - Fragmentation
    - X509
    - Session Management

Information Model

Professur für PROZESS LEITTECHNIK



Object oriented information model:

- Object/Variables stored as typed nodes
- Relations are stored as typed references
- Information model is graph-like structure of node relations

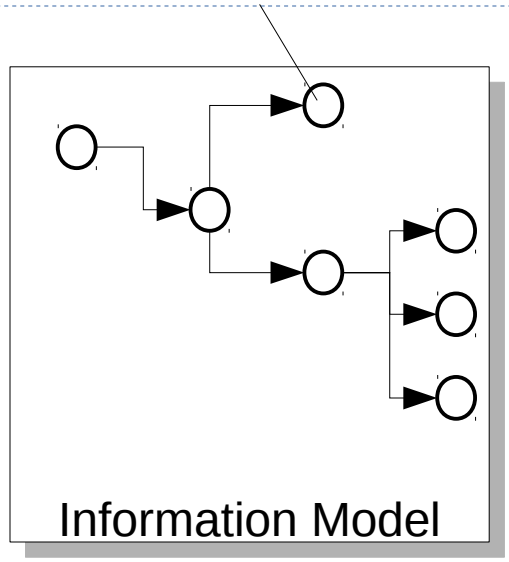
OPC UA is a Meta-Model-Language:

- Graphs are organized into numerical indexed namespaces
- Namespace 0 models OPC UA …
  …using OPC UA

Common Node Attributes

Professur für PROZESS LEITTECHNIK

```
UA_STANDARD_NODEMEMBERS:
UA_NodeId          nodeId;
UA_NodeClass       nodeClass;
UA_QualifiedName   browseName;
UA_LocalizedText   displayName;
UA_LocalizedText   description;
UA_UInt32          writeMask;
UA_UInt32          userWriteMask;
UA_ReferenceNode   references[];
```



- All nodes have a common set of attributes.
- All nodes have a type.
  - Each node type has additional attributes.
- References are managed by a node.
- OPC UA does not mandate how these properties are to be implemented

  → Language specific

Information Model: Node Types

Professur für PROZESS LEITTECHNIK

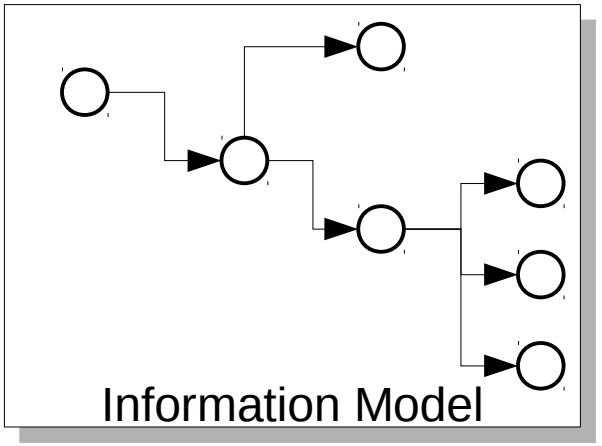
```
UA_STANDARD_NODEMEMBERS
UA_Byte eventNotifier;
```

Instances:

- ObjectNode
- VariableNode
- MethodNode
- ViewNode

Definitions:

- ObjectTypeNode
- VariableTypeNode
- DataTypeNode
- ReferenceTypeNode

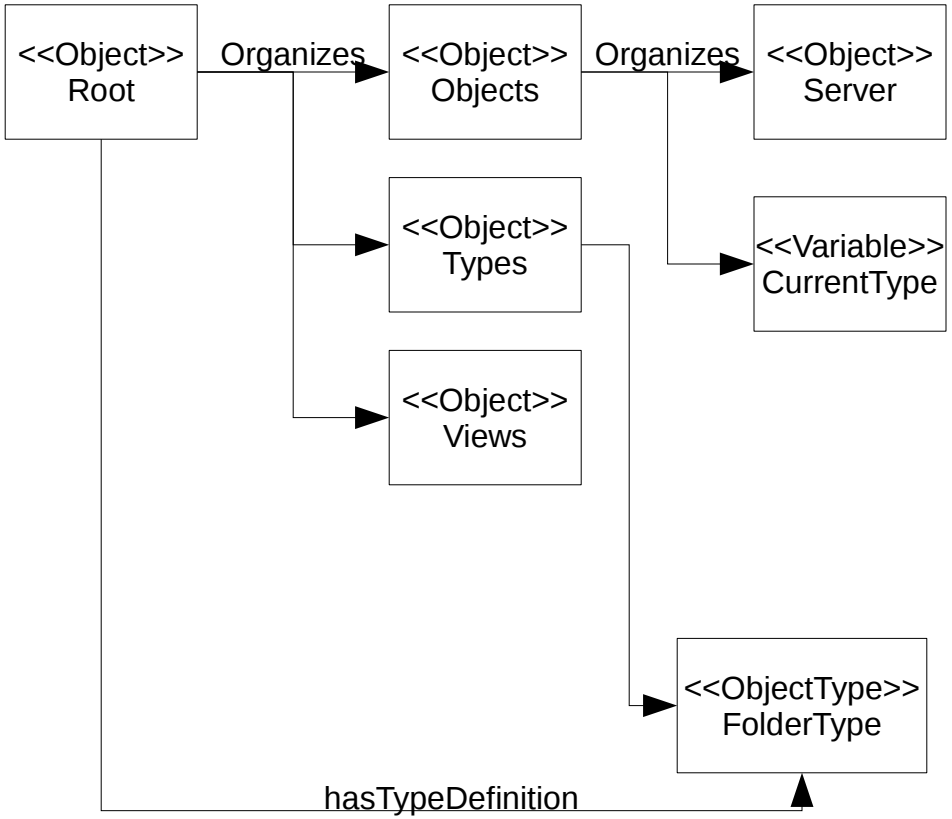



```
UA_STANDARD_NODEMEMBERS
UA_Boolean isAbstract;
UA_Boolean symmetric;
UA_LocalizedText inverseName;
```

```
UA_STANDARD_NODEMEMBERS
UA_Int32        valueRank;
Void            *ua_value;
UA_Byte         accessLevel;
UA_Byte         userAccessLevel;
UA_Double       minimumSamplingInterval;
UA_Boolean      historizing;
```

Information Model: Node Relations

Professur für PROZESS LEITTECHNIK

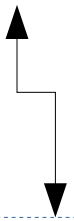


- References between nodes are typed.
- Common types:
  - Organizes
  - HasTypeDefinition
  - HasProperty
  - HasTypeDefinition
  - HasComponent
- „Namespace 0“ defines all type primitives contained in the specification.
  → Selfmodelling information model

OPC UA Data Types

Professur für PROZESS LEITTECHNIK

- Atomic types:
  - Boolean
  - Byte, Sbyte
  - Int16, UInt16
  - Int32, UInt32
  - Int64, UInt64
  - Float, Double
  - GUID
  - DateTime
- Compound types:
  - String, ByteString
  - ExpandedNodeID
  - LocalizedText
  - QualifiedName
  - NodeID
- Recursive types:
  - ExtensionObject
  - Variant



```
typedef struct {
    const UA_DataType *type
    UA_Int32  arrayLength;
    void     *data;
    UA_Int32  arrayDimensionsSize;
    UA_Int32 *arrayDimensions;
} UA_Variant;
```

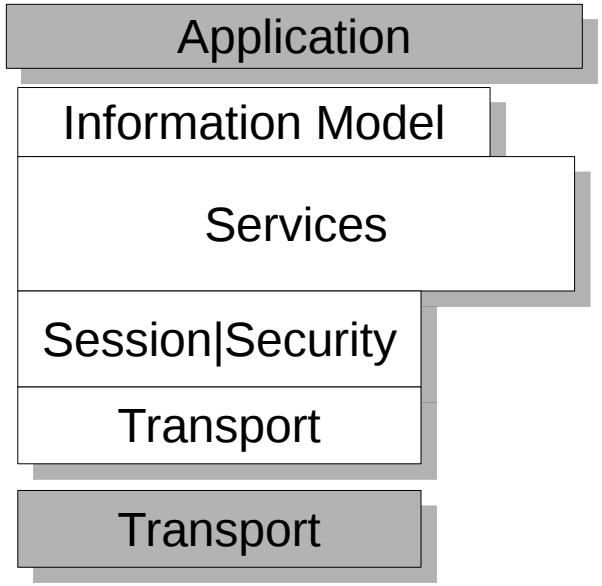
```
typedef struct {
    UA_UInt16 namespaceIndex;
    union {
        UA_UInt32     numeric;
        UA_String     string;
        UA_Guid       guid;
        UA_ByteString byteString;
    } identifier;
} UA_NodeId;
```

Beyond Namespace 0

Professur für PROZESS LEITTECHNIK

- OPC UA defines basic node types and references
- New Object/Reference/Variable Types can be defined by the programmer/application
- Extending the information model is done by adding new nodes to namespaces.
- Namespace Index 0 is reserved for OPC UA. 4 Million other namespaces are available to the user
  - Namespaces „boundle“ nodes that belong to the same concept
  - Nodes can still cross-reference each other (namespaces are no barriers)
- Example:
  - The new Reference Type „fieldDeviceConnectsTo“ can be a subtype (hasSubtype reference) of „hasComponent“
  - The new Object Type „WaterTurbine“ can be a subType of „BaseObjectType“

Server-Client architecture

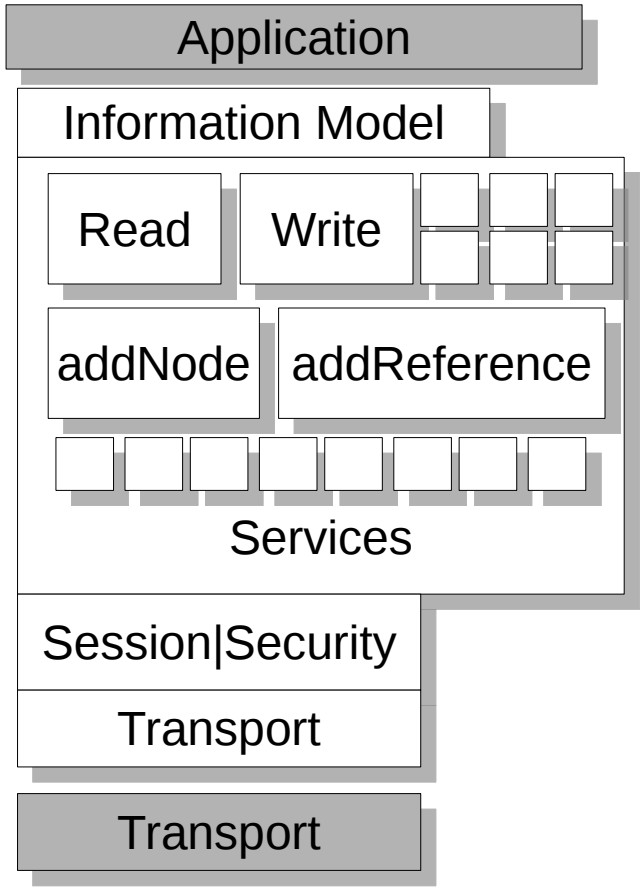


- Request-Response driven interaction
- Client interaction interfaces for the server namespace are called „Services“
- Services can query and process all node properties
- Services can request extended serverside operations on nodes

- Servers just provide data
- Clients need to „parse“/work with the data

OPC UA Services

Professur für PROZESS LEITTECHNIK

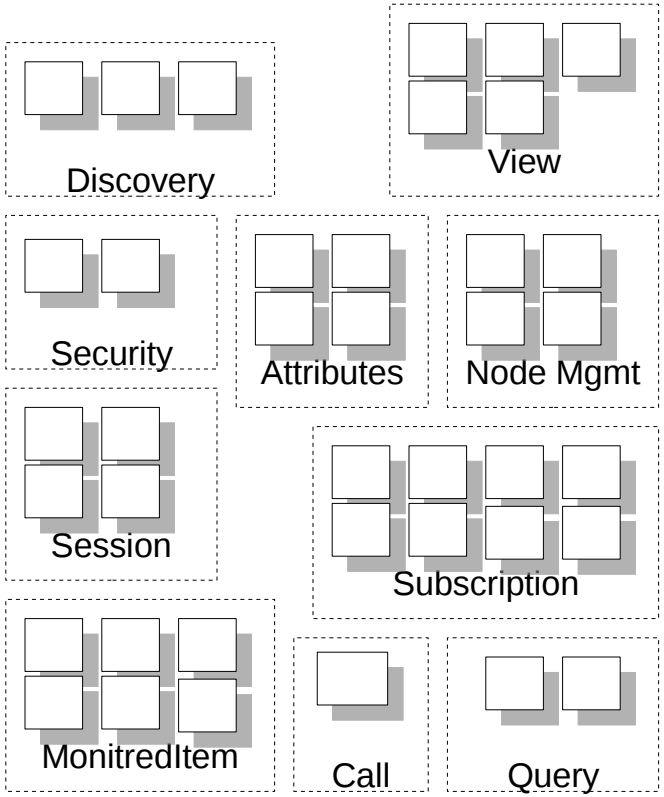


- OPC UA defines 37 ***Services*** to allow Server/Client Interaction
- Services are grouped into 10 logical ***Service Sets*** (ex. „NodeManagement“, „Attributes“)
- Most services can pass and return multiple parameters to a server
- Some commonly used services:
  - Read (return a node attribute)
  - Write (set a node attribute)
  - Call (execute a method)
  - Browse (get a nodes references)

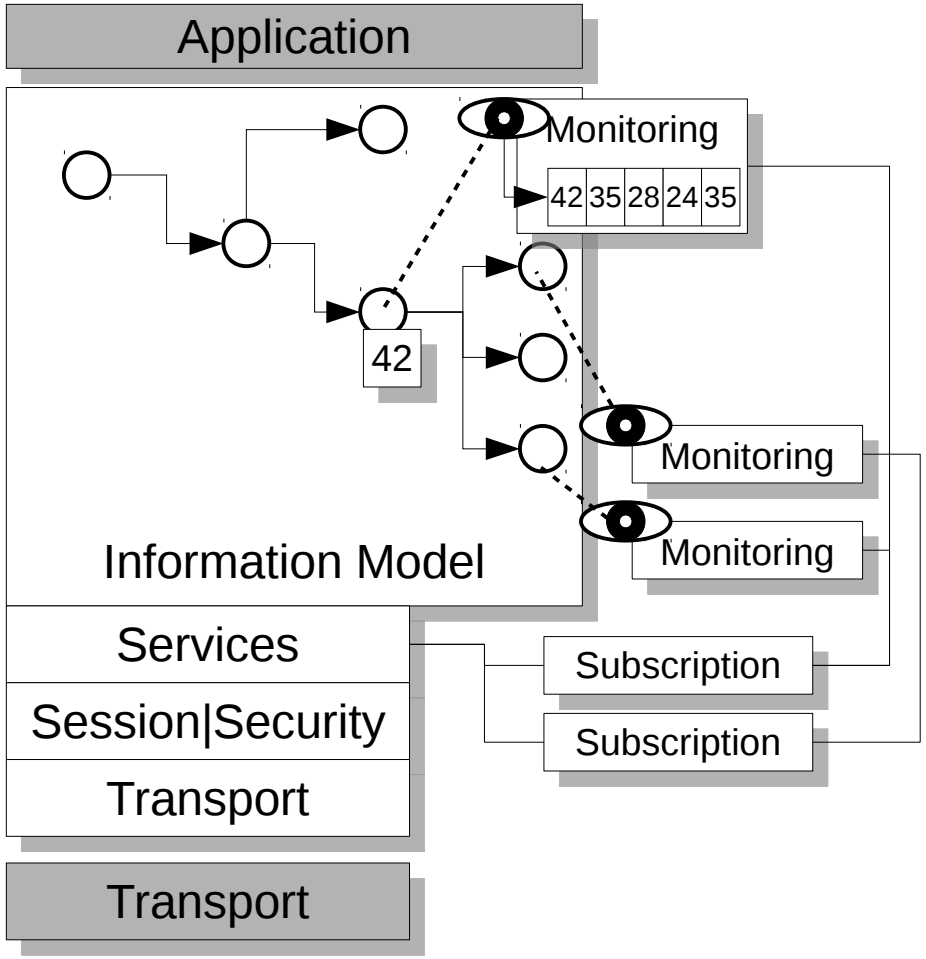
OPC UA Facets and Profiles

Professur für PROZESS LEITTECHNIK

- A server does not need to support all 37 services
  - Some cause considerable execution overhead
  - Many cause considerable memory management overhead
- The server's set of implemented services is called a ***profile***
- A set of services needed by a server to fullfill a function – incl. their *parameters* - are called ***facets***

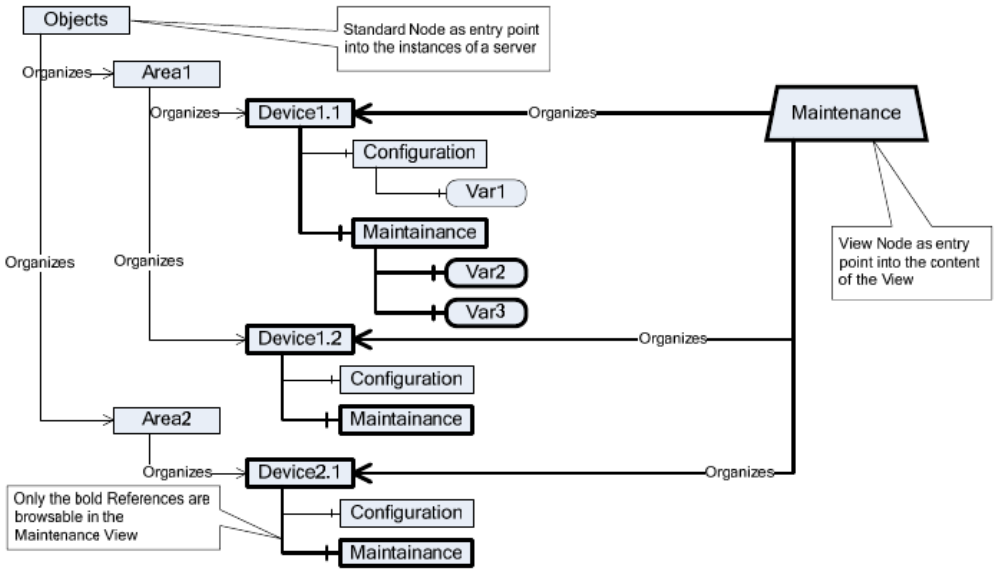


Monitoring and subscriptions



- ***Monitoring*** can be requested for individual nodes:
  - Value Changes
  - Model Changes
  - Events
- Monitored Items can be „queued“ → asynchronous sampling
- Monitored items can be ***subscribe***d to
  - Client is automatically notified about a change

Views and information organization



Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm, „OPC Unified Architecture“, Springer, 2009, ISBN 9783540688990

- Special node type used to regroup hierarchies
- Provides shortcuts to logically coherent data
- May be used to control access

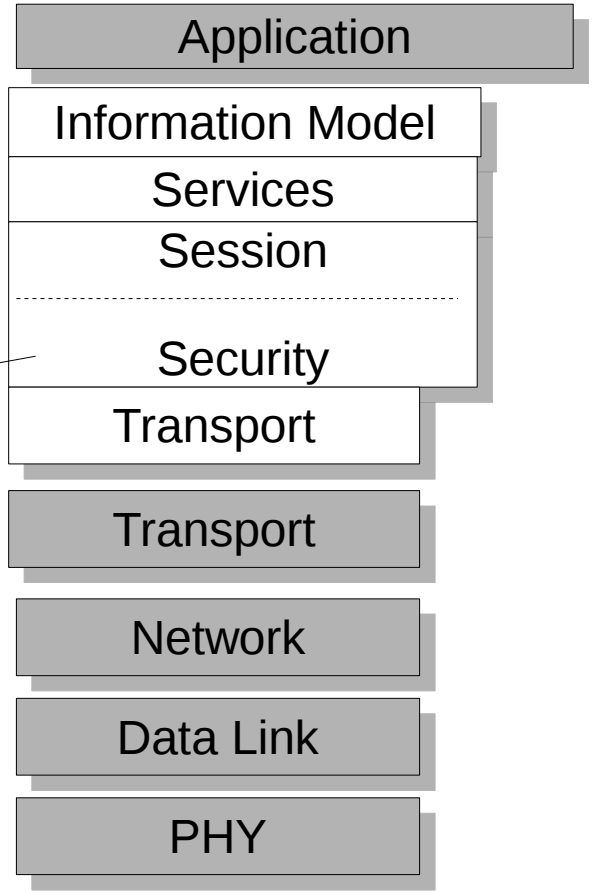
→ Views ***can not*** “hide” data

OPC UA Security

Professur für PROZESS LEITTECHNIK

- Security was overlooked for a long time in automation
- OPC UA incorporates several forms of security

  → Questionable choice...
- Available modes:
  - None
  - User/Password
  - X509 Certificates
    - DH Key Exchange
    - AES-CBC-MAC
    - Etc.



Overview of available OPC UA stacks

Professur für PROZESS LEITTECHNIK

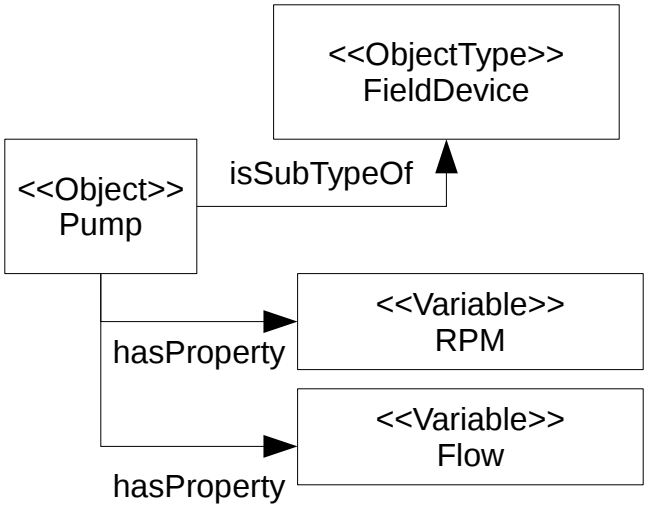
Stack	Language and availability
Advosol	C#, Proprietary
Ignition Automation	Java, Open Source
NodeOPCUA	Javascript + NodeJS, Open Source
Matrikon HDK Embedded	C, Proprietary
opcua4j	Java, Open Source
open62541	C/C++, Python, Open Source
OpenOpcUa	C/C++, Proprietary
Softing	C/C++, Python, Proprietary
Unified Automation	C/C++/C#/Java, Proprietary
Prosys	Java, Proprietary

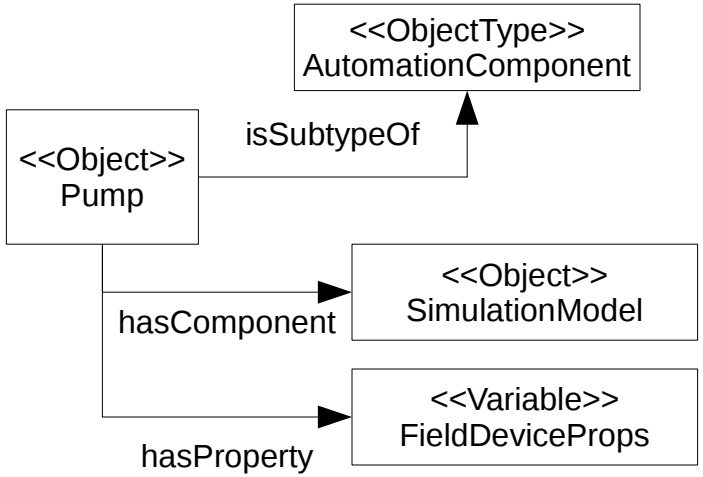
open62541

Stack Features	
Namespace, node and reference management	✔
XML to namespace compiler	✔
Symmetric Server/Client API	✔
Local/Remote Browse Iterators	✔
Method Calls	✔
Dynamic Data Sources	✔
User/Password Authentication	✔
Events	✕
X.509 Certificate Chain	(✕)
Subscriptions	(✔)
Cmake configurable profiles	✔
External namespaces	✔
Examples and tutorials	✔

- Open Source Stack (C/C++):
  http://github.com/acplt/open62541
- Initiated 12/2013 by Chair for process control, TU Aachen
- Active Code Development contribution by TUD since March 2014
- License:
  LGPL + Static Link Exception
- **Active project integration**

Pitfalls (1):
There is more then one way to model reality





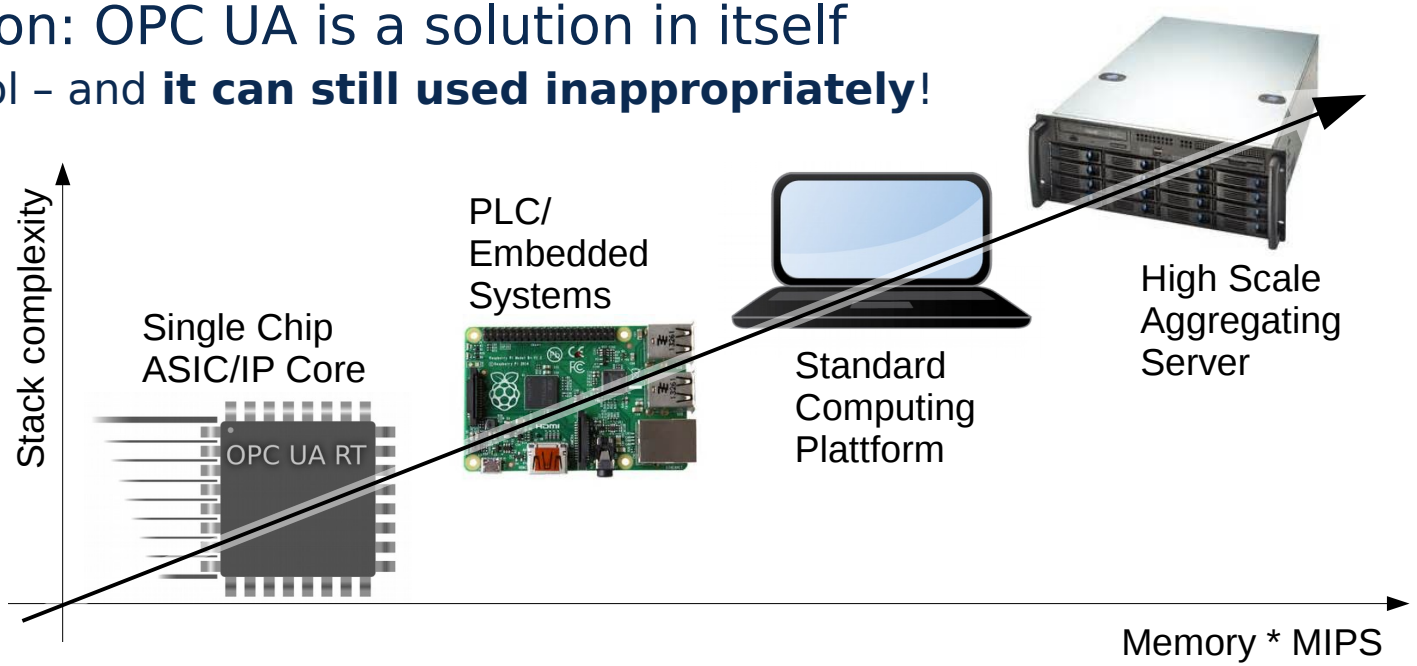
- As with object-oriented programming, a scenario may be subject conform to different modelling approaches
- Specifying a proper information model is vital
- Specifying proper sequence diagrams is vital
- Examining use-cases is vital

→ Standard Object-Oriented development approaches apply!

Pitfalls (2): Agreeing on device profiles

Professur für PROZESS LEITTECHNIK

- Misconception: OPC UA is complicated
  A complete OPC UA server is seldomly required
- Misconception: OPC UA needs a lot of ressources
  Hard memory barriers do exist... but each device has an appropriate profile
  Less Profiles/Facets <=> Higher Performance
- Misconception: OPC UA is a solution in itself
  OPC UA is a tool – and **it can still used inappropriately**!



Pitfalls (3):
Using OPC UA for semantic integration

Professur für PROZESS LEITTECHNIK

- OPC UA *enables* semantic integration, it *does not provide* it
- Intelligent data processing is key
  - Providing values can be accomplished easier
  - Making sense of unknown data requires semantic capabilities
- Distributed semantic data needs to be aggregated appropriately
  - Just forwarding data is not sufficient
  - Smart functions must be distributed

Summary (1)

Professur für PROZESS LEITTECHNIK

- The OPC UA ***information model*** allows a generalized object oriented modelling approach
  - Constructed of ***nodes and references***
  - Grouped into ***namespaces***
  - OPC UA defines base ***node and variables types***
- OPC UA uses a ***request-/response*** driven server-client-architecture
  - The client interacts with the server through ***services***
- Server capabilities can be adapted using ***profiles***

OPC UA Specification:

1. Concepts
2. Security Model
3. Address Space Model
4. Services
5. Information Model
6. Mappings
7. Profiles
8. Data Access
9. Alarms and Conditions
10. Programs
11. Historical Access
12. Discovery
13. Aggregates

Summary (2)

Professur für PROZESS LEITTECHNIK

- OPC UA is a tool and needs to be used appropriately
  - It will not innovate by itself
- Both closed source and Open Source stacks are available
  - Open Source: Free, Feature implementation depends on "voluntary" contribution
  - Closed Source: Frequently partial or „frozen" development
- OPC UA support supporty in commercial products is frequently lacking (incomplete or incompatible)
- Functional support varies involuntarily by choice of stack
- Functional support should be adapted to its application

Vielen Dank
für Ihre Aufmerksamkeit!

Für spätere Fragen:
chris_paul.iatrou@tu-dresden.de

Professur
für **PROZESS**
LEITTECHNIK

Prof. Dr.-Ing. habil. Leon Urbas

Technische Universität Dresden
Fakultät Elektrotechnik und
Informationstechnik
Institut für Automatisierungstechnik

Tel.: +49 351 463-34604
Fax: +49 351 463-39681

Besucheradresse:
Barkhausen-Bau
Georg-Schumann-Str. 11
01187 Dresden

Postanschrift (Briefe):
Technische Universität Dresden
Fakultät Elektrotechnik und
Informationstechnik
Institut für Automatisierungstechnik
01062 Dresden