Whitepaper

Overview: OPC Unified Architecture
Technical overview and short description

Feature rich cross platform communication in industry and beyond
→ getting a common understanding what UA is all about.

written by:
Uwe Steinkrauss, ascolab GmbH
September 2010

Abstract
The world leading software communication standard in industrial automation has entered the next generation with the service oriented Unified Architecture. Before facing new opportunities of this technology software architects, product managers and developers need to understand the basic ideas and technological background based on a common sense of wording. This whitepaper shows how it all begun and gives a short overview on the OPC UA technology.

BACKGROUND

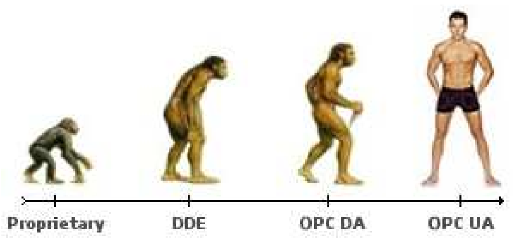
The OPC Foundation was founded in 1996 from a group of 4 companies Intellution, Rockwell Software, Fischer Rosemount, Opto22 (intiutive) to basically overcome the constraints of proprietary protocols in industrial communication. The goal was to find a common communication interface (similar to printer driver) that could be used between HMI/SCADA (Human Machine Interfaces) and PLC (Programmable Logic Controller) controller level, it was clearly positioned higher than the field bus level and not intended to replace distributed I/O communication. The OPC Foundation being a non-profit organization was growing extremely quick over the fist 10 years having around 480 Members (2006) showing the high demand for vendor neutral communication. The member list reads like the who-is-who of vendors of automation equipment and industrial software suppliers including the "big 5" (ABB, Emerson, Honeywell, Rockwell, Siemens) but also include Microsoft, Texas Instruments, Phoenix Contact, Mitsubishi, and literally every company in industrial automation. Controlled by a board of directors they formed a technical steering committee creating different working groups. These groups consist of volunteers from the member companies that work on specifications, tools, sample code and testing/certification procedures. The most adopted specification was called "Data Access" basically being the smallest common understanding amongst the group of what is required to access process data in the shop floor and make it visible on an HMI. This specification - with only read, write, browse and "change of data" notification - became extremely successful and widely adopted not only by member companies but also by non-members who use the public specification or a commercial toolkit to build their products or integrate an OPC interface into existing products. There are approximately 1000 products build by members listed in the OPC Foundation's product catalog, but there exists an estimate of more than 15000 OPC enabled products world wide.

TECHNICAL OBJECTIVES

Amongst other, one of the reasons being so successful in terms of adoption was the fact that interfaces were build on the OLE (later called COM/DCOM) technology. Using the basic infrastructure for inter process communication introduced by Microsoft operating system with Windows NT4 and following, the tool support (Visual Studio) and programming languages C++ (Active Template Library) made implementation very easy. However, in beginning of 2002 it became clear that this dependency to Microsoft turned out to be the limitation of the technology. Not only discontinuing DCOM by announcing .NET also the platform dependence on x86 and language limitation turned out to raises more problems that it solves. The OPC Foundation released the OPC XML DA specification as platform independent intermediate step, it was basically a WSDL. Now requiring a web server it turned out that implementation on small embedded devices reached only poor performance and wide adoption could not be reached. The need for a more general approach was born, generic and independent form the transport.

UA WORKING GROUP

In 2004 the OPC Unified Architecture working group was formed to work on a better solution. The group had several requirements to fulfill. Starting from platform independence, language independence and protocol independence a set of functions or better said services were created that on one side unified all existing OPC specification in terms of their functionality and added scalability, but also new security demands and high performance were integrated. The new technology should run on whatever device architecture one can imagine.



„In a first step variation is created by mutation and recombination, and in a second step variation is selected by elimination.“
(Ernst Mayr)

The solution created by the working group was releases in 2007. Based on SOA (Service Oriented Architecture) the OPC Foundation created a new standard unifying all the existing specifications of Data Access, Alarms&Events, Historical DA, Complex Data and Commands. In addition security aspects were added like application identification, user authentication and access control down to a single data point's attribute. In parallel with the specification work OPC UA was proven for being implementable on any kind of device from 16 bit up to 64 bit architectures x86, ARM, PowerPC, etc. and any kind of operating system from Windows and Linux to VxWorks and other RTOS. The separation of the services from the protocol and the implementation language make OPC UA very flexible and usable in new domains outside the classic HMI to PLC communication. OPC UA is implementable on embedded devices with limited resources but can be scaled up to enterprise systems running on mainframes.

DELIVERABLES

The UA working group defined a binary protocol based on TCP and provides the dedicated communication stack, replacing COM/DCOM from a functional perspective, in three implementation languages: .NET, ANSI C, and JAVA. This binary UA TCP protocol is faster and more secure than DCOM and it is more suitable for industrial use cases. For members of the OPC Foundation the full source code is open and free to use. Especially the ANSI C stack is extremely fast and overcomes the weakness of low performance XML based web services, it was measured at twice the speed of DCOM. In addition to the stack itself the OPC Foundation provides a full SDK (Software Development Kit) for the .NET platform from Microsoft. Commercial toolkits and SDKs for ANSIC, C++, and JAVA are available and will speed up development and reduce implementation effort. In parallel the OPC Foundation started the development of tools for testing and certification of client and server products. Knowing from the past classic OPC specifications the Foundation learned that interoperability can only be guaranteed by testing the conformance to the specification in an early stage of its adoption. The OPC UA Compliance Test Tool (UA CTT) gives implementers and testers a script driven test tool at hand that has extremely high value during implementation and testing of OPC UA enabled products.

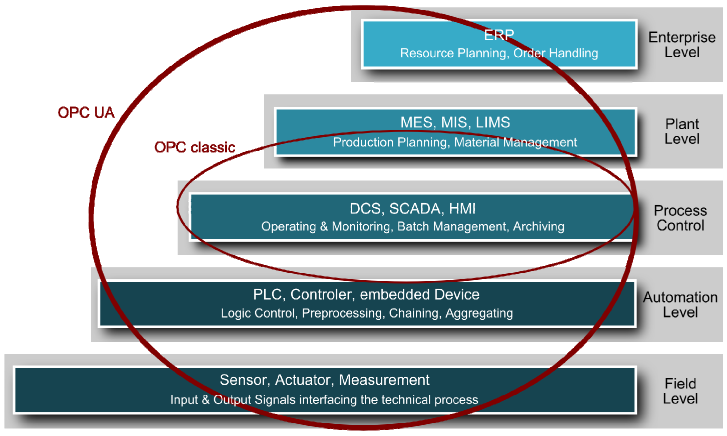
INTEROPERABILITY

To prove the conformance to the specification the concept of independent certification laboratories is followed right from the start. A world wide network of accredited test facilities is established that certifies the functionality using predefined test mechanisms and shares the results with the public. On the website of the OPC Foundation, the users can read test reports to validate and compare products to pick and choose the appropriate for their specific use case.

TARGET MARKED

The new generation of OPC connectivity can be used far beyond the typical use case of industrial communication known from classic OPC interfaces. Besides the fact of platform independence, security and scalability OPC UA provides just the infrastructure for robust communication. It standardizes “how” information is transferred and gives rules “how” data is describes, but UA does not define “what” this information means. OPC UA has the potential to endeavor new markets within the automation pyramid and can be used in domains typically occupied by IT technologies. Not only the process industry and product manufacturing, but also the building automation, and energy suppliers already committed to UA.

Figure 1) Wide Scope of OPC UA

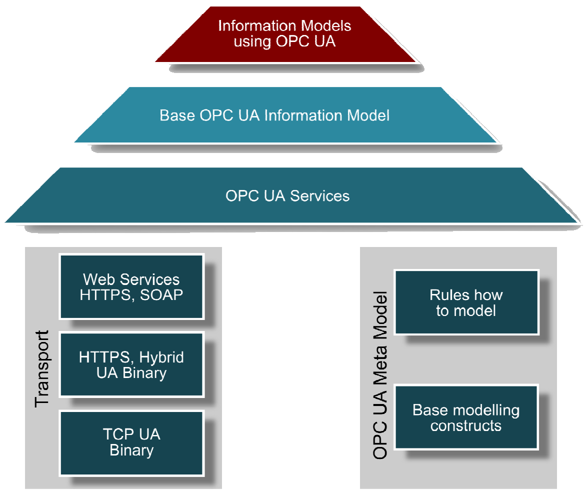

ERP
Resource Planning, Order Handling
Enterprise Level
MES, MIS, LIMS
Production Planning, Material Management
Plant Level
DCS, SCADA, HMI
Operating & Monitoring, Batch Management, Archiving
Process Control
PLC, Controler, embedded Device
Logic Control, Preprocessing, Chaining, Aggregating
Automation Level
Sensor, Actuator, Measurement
Input & Output Signals interfacing the technical process
Field Level
OPC UA
OPC classic

From the first day OPC UA was targeted form the enterprise to field level and by this widely increased the range of the former technologies. The technological base of OPC UA is scalable and can cover the requirements for standard communication in different domains. Not only vertical but also horizontal communication within e.g. the automation level (PLC-to-PLC communication) is in the scope of operation.

OPC UNIFIED ARCHITECTURE

In general OPC UA is a communication infrastructure that consists of two major parts: the information transport and the meta model describing information including semantics. For transportation different protocol implementation can be used. The meta model defines rules for its usage. On top of these two major components a fixed set of services was defined to access and use the exposed information. The UA Information Model is an object oriented base model that fits many use cases but can also be used by others, outside the OPC Foundation, to define their mappings on top or better said its mapping on the real world. By this generic but very flexible concept literally any kind of domain specific information and its related semantics can be mapped.

Figure 2) General Architecture of OPC UA



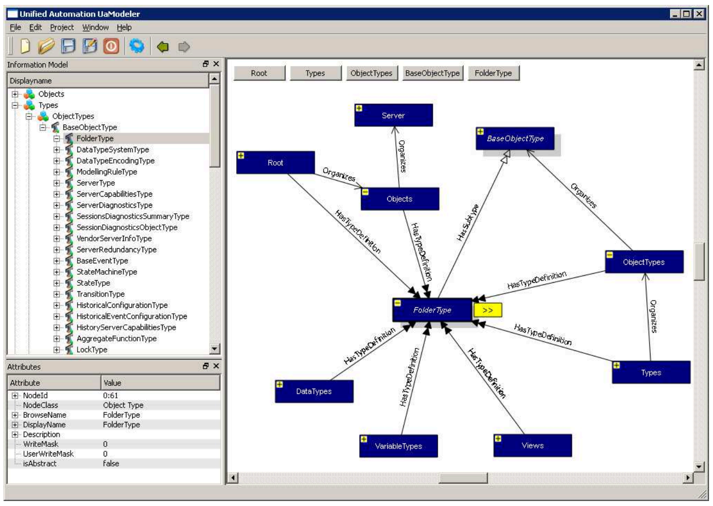
SECURITY

The threads IT professionals face every day became more relevant even in industrial automation systems not being an isolated island anymore. OPC UA is targeted far beyond industry applications and hence needed to deal with those threads. The detection and protection against malformed messages, message alteration, message flooding and message spooling must be addressed same as denial of service attack (DoS), session hijacking, eavesdropping and rogue clients and servers. In addition user and role based access control is required for secure operation. The functional objectives regarding security like authorization, authentication, integrity, availability, confidentiality as well as traceability, accountability and availability were included into the core architecture of OPC UA.
The new OPC UA technology integrated security on different levels. Form application layer over communication layer down to transport layer bullet proof strategies known from IT domain were adopted an integrated as built-in feature of OPC UA. In other words, security is for free as it is already implemented into the communication stack itself. Using techniques known from other domains, OPC UA clients and servers will exchange application certificates (x509) to identify the instance of the specific application and grand access only to trusted (known and accepted) applications. In addition user authentication is possible to handle different access to machines depending on the role of function e.g. users, integrators, engineering and administrators. When accessing single data points (objects) within the UA server every single attribute will grant user/group specific access. In contrast to the old OPC specifications not only R and W permissions will be available. User “John” can have read/write access to the value attribute of a node, but user “Paul” is only allowed to read it, and “Susan” can not even browse/see this data point. The transport itself is encrypted using SSL or any other encryption. In fact the OPC UA communication stack provides an abstract interface to plug in even hardware encryption if SSL is not supported by the targeted system.

INFORMATION MODEL

The information model can be seen as the major innovative enhancement compared to classic OPC. The fully object oriented model allows describing any kind of equipment, function or system you may think of. OPC UA comes with a set of object and types typically used in automation industry. But of course you can derive your own types and describe your domain specific model by inheritance form existing types and crating new objects that represent your system appropriately. The objects in this model are linked by references giving a full mashed network of relationships between the objects, but they remain accessible or better said discoverable by generic mechanism.

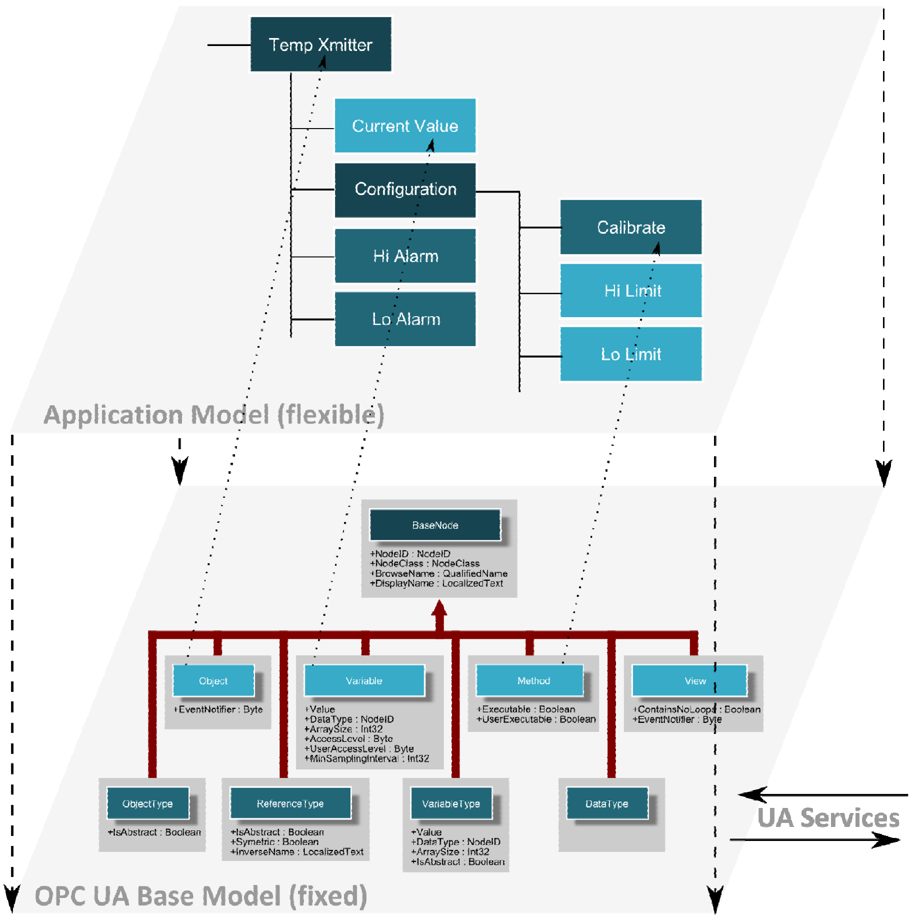
Figure 3) Folder Type and its relations within the Type System of OPC UA



Attribute	Value
NodeId	0:61
NodeClass	Object Type
BrowseName	FolderType
DisplayName	FolderType
Description	
WriteMask	0
UserWriteMask	0
isAbstract	false

Giving the flexibility of the information model it is expected that other organizations like PLCopen, ISA95 or domains like energy, building and medicine use OPC UA as an infrastructure and build their models on top. The field device description and configuration groups, called FDT and EDDL, were one of the first that used OPC UA for generic description of field devices and their configuration and diagnostics. In the end they defined a mapping of their information to be exposed by OPC UA servers and standardize this mapping in a so called companion specification called DI (Device Integration).

Figure 4) Flexible mapping of information on a fixed object oriented base model



Starting with the OPC UA Base Model or one of the standardized companion models the application specific model can be derived. The object oriented base model leaves all the flexibility to the application designer, but provides an interoperable fixed set of access and manipulation services.

SERVICES

The services are definitions to access information provided by an UA server. Here the OPC Foundation has learned from the past and reduced the complexity during unifying the functionality of the old classic interfaces. They designed only services that were desperately required.
The OPC UA Service definitions are abstract descriptions and do not represent a specification for implementation. In the case of an implementation as web services, the OPC UA Services correspond to the web service and an OPC UA Service corresponds to an operation of the web service. The organization in service sets is a logical grouping used in the specification and is not used in the implementation.

Table 1) Service Sets of OPC UA

Service Set	Comment
Discovery	defines services that allow a client to discover the endpoints implemented by a server and to read the security configuration for each of those endpoints
SecureChannel	defines services that allow a client to establish a communication channel to ensure the confidentiality and Integrity of messages exchanged with the server.
Session	defines services that allow the client to authenticate the user it is acting on behalf of and to manage sessions.
NodeManagement	defines services that allow the client to add, modify and delete nodes in the address space.
View, Query	defines services that allow clients to browse through the address space or subsets of the address space called Views. The Query service set allows clients to get a subset of data from the address space or the View.
Attribute	defines services that allow clients to read and write attributes of nodes, including their historical values. Since the value of a variable is modeled as an attribute, these services allow clients to read and write the values of variables.
Method	defines services that allow clients to call methods. Methods run to completion when called. They may be called with method-specific input parameters and may return method-specific output parameters.
Subscription, MonitoredItem	Subscription services allow clients to create, modify and delete subscriptions. Subscriptions send notifications generated by monitored items to the client. Subscription services also provide for client recovery from missed messages and communication failures. Monitored items services allow clients to create, modify, and delete monitored items used to monitor attributes for value changes and objects for events. These notifications are queued for transfer to the client by subscriptions.

Whether or not a server supports a service set, or a service within a service set, is defined by its Profile. The set of services is fixed to get maximum interoperability, however a server may decide to provide only services that belong to “Data Access” functionality, hence it will support the DA-Profile only. Client and server can discover the partners supported profile to find a common understanding of functionality. A basic server profile must be supported by all UA servers as a minimum requirement.

Table 2) OPC UA Services related to Data Access Profile

Service	Comment

OpenSecureChannel( )
CreateSession( )
ActivateSession( )
	These services are used to establish a secure connection to the server. After creating the secure channel (certificate exchange, validation) a session (stateful) is created and checked for activity on both sides (client and server)

Browse( )
Read( )
	The client can browse through the information model following any type of reference (backward and forward) to navigate in the meshed

Write( )	network. The attributes of every single node found can be read or written if the appropriate access rights were granted

CreateSubscription( )
ModifySubscription( )
DeleteSubscription()
	For event based communication the client need to create one or more subscriptions. Depending on the monitoring mode and the update interval the client get notified for changes of values, alarms or just keep alive signals.

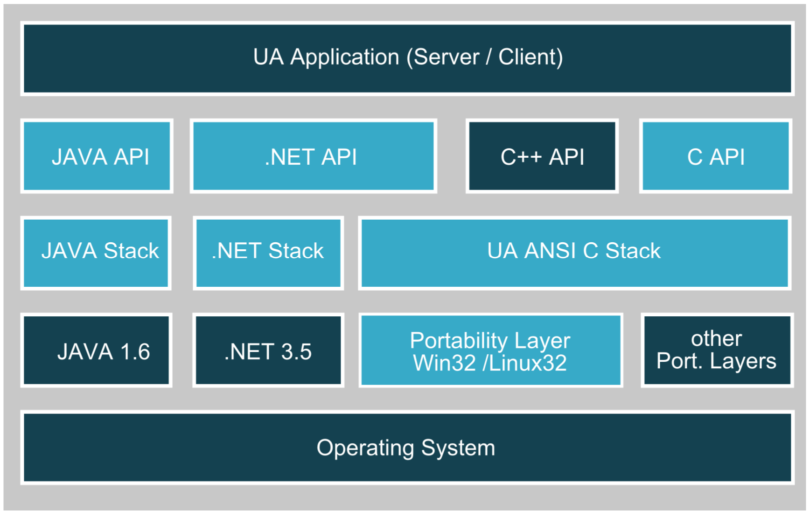
CreateMonitoredItems( )
DeleteMonitoredItems( )
	The client decides which nodes should be monitored by the server and request sampling rate and buffering of values if required.

Publish( )	The publish finally delivers the data in any kind of event based notification.

UA STACK

The connection between the application and the wire is called the UA Stack. The OPC Foundation provides complete implementation of the OPC UA communication stacks in three different programming languages: ANSI C, .NET and JAVA. For members these stacks are available in source code. Non- members can get the stacks as part of commercial toolkits or SDKs. Accessed through an API the stack is responsible for message encoding, security and transport.

Figure 5) UA Stack Implementations

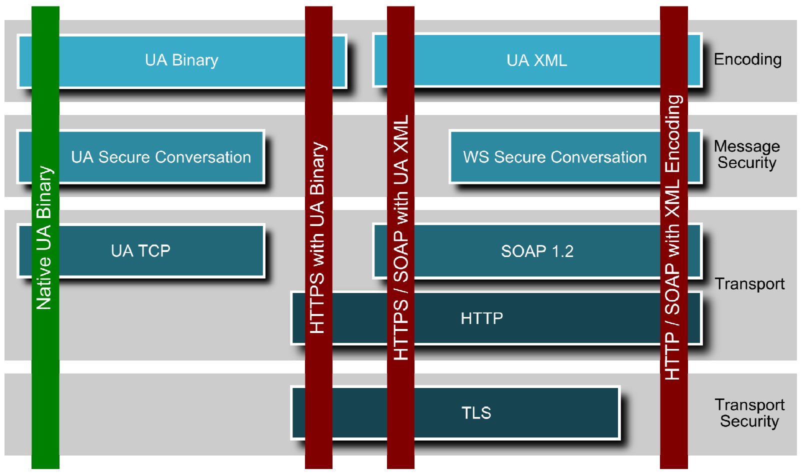


The developers can directly use the C API of the UA ANSI C Stack or use the more comfortable C++ or .NET API. All these APIs have similar functionality within the limits of the programming language. All platform dependant code is abstracted in the platform adoption layer giving the ability to port the stack to any operating system and any CPU architecture. Today the OPC Foundation provides PL for Windows 32Bit and Linux 32Bit, but many other platform layers are commercially available e.g. Windows CE, VxWorks, Solaris, QNX, etc. In addition a fully native .NET, but requiring at least .NET 3.5, and a native JAVA stack, requiring JRE 1.6 are available for those who need to integrate UA into products already based these languages.

TRANSPORT

Within the stack OPC UA defines a layered architecture leaving it to the application which transport to choose. Even though UA is generally not tied to the transport protocol there are three transports defined today. It is expected that the native UA Binary protocol will be used by most of the applications especially on embedded devices. MES/ERP applications running on larger scaled machines in the IT domain may use https variants. There will be an HTTPS based transport having attached messages that are UA binary encoded. A third variant uses also HTTPS but SOAP and XML encoded message having the highest resource consumption and lowest performance. A fourth version was planned using pure HTTP only and adding WS* secure conversation, but it was found not to be needed in praxis.

Figure 6) OPC UA Protocol Mappings



The client application can select the communication mapping by choosing a certain URL prefix when connecting to a UA server. UA servers must support the native UA binary protocol, but may in addition support the HTTPS, depending on to whom the server wants to expose its data.

Table 3) Recommended use of Protocol Mappings

Mapping	Comment
Native UA Binary	Pros & Cons: best performance, lowest resource consumption, requires firewall configuration (single TCP port per server) Recommended Applications: embedded devices, PC based control, SCADA/HMI, measurement equipment, most Windows and all non-Microsoft OS based systems, specialized applications running on local LAN, typically one UA server per machine
HTTPS with UA Binary encoded attachment	Pros & Cons: Midrange performance, firewall friendly, UA Secure Conversation required when running HTTP Recommended Applications: MES and ERP systems, factory integration across the internet
HTTPS/SOAP with XML encoded messages	Pros & Cons: lowest performance, CPU and bandwidth intensive, firewall friendly, WS Secure Conversation required when running HTTP, good tool support Recommended Applications: enterprise integration (SAP, Microsoft), ERP, other general business applications

Event though UA applications may fall into three groups depending on the mapping they support, it is expected that UA Binary will be preferred when communication to the field is required e.g. even when a MES system wants to read data from a PLC it better supports UA Binary as the PLC will most likely support UA Binary only because it will typically not have the resources to parse SOAP headers and text based XML messages. Today only the .NET stack is hybrid and supports all three mappings.

In addition to the pure UA Binary the ANSI C stack will support HTTPS with binary encoded content, however in most cases it will be compiled out to save resources.

META MODEL

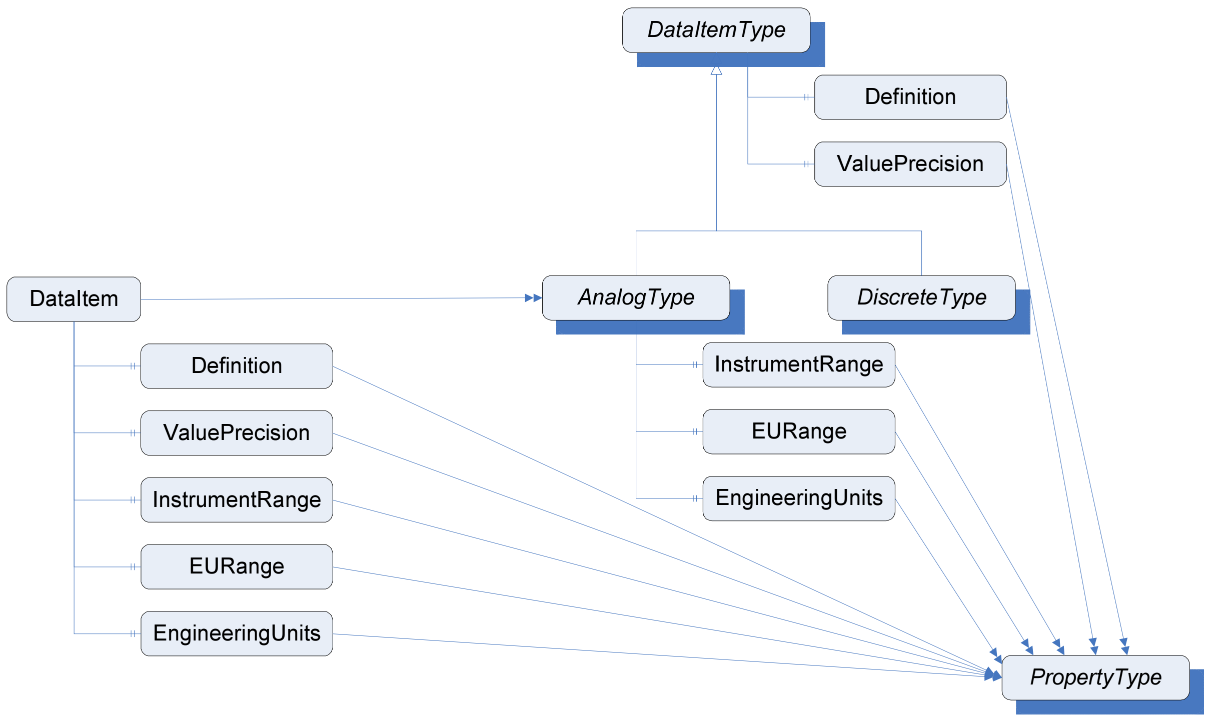
OPC UA defines a hierarchical type system that is represented in the servers address space. Clients can follow references to get insight view about the information (data and relations) and the related semantics exposed by the server.

Table 4) OPC UA Type System

Type Definitions	Comment
Object Types	Instance declarations (typed) with modeling rules. Event types are represented as object types and contain information for event subscription, event type hierarchy is used for filtering
Variable Types	Data variables (typed) represent data of an object, can be complex (have other data variables) and have properties. Properties (identified by BrowseName) represent characteristics of objects or variables (engineering unit, version number, etc.), there are special services for fast access, properties are leafs of each hierarchy
Reference Types	Hierarchical and non-hierarchical references. Used in browsing and query as filter
Data Types	For Value attribute of variables (all other attributes have fixed, build-in data type. Extendable, complex data build-in UA

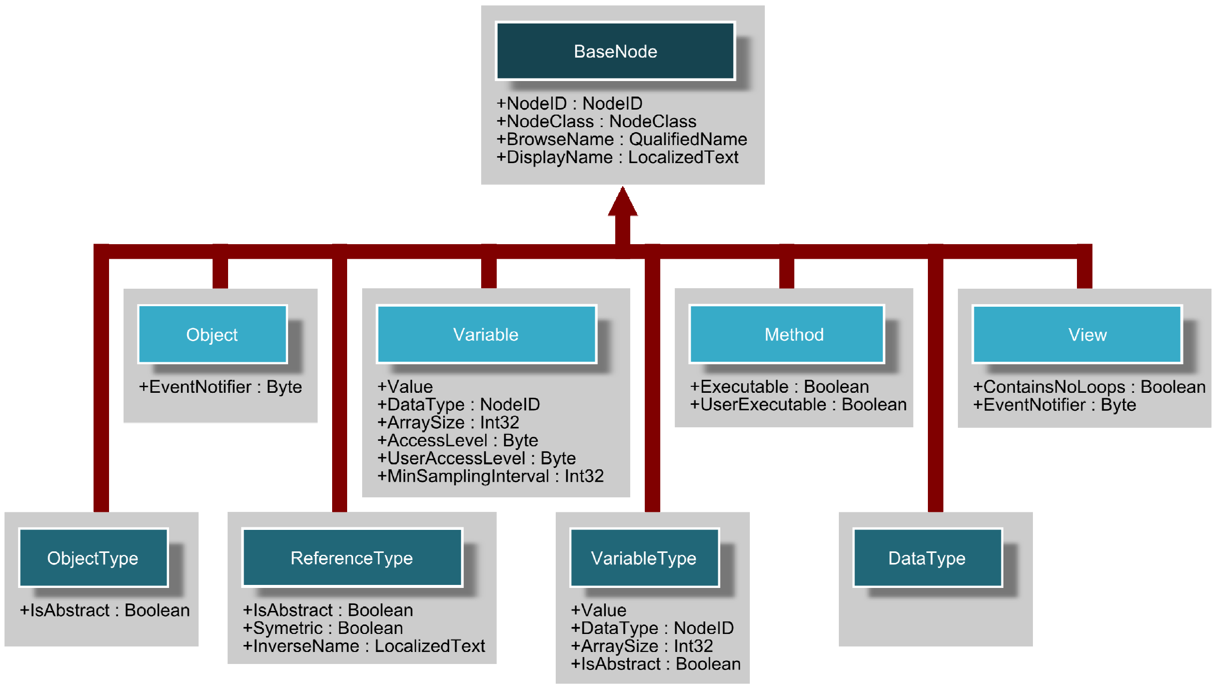
Analogue to variables in object oriented programming languages the UA types define a named construct that is available in instances (accessible nodes of certain type).

Figure 7) Data Item and its relation to the Type System



In addition to the type system Methods and Views are introduced by UA. A Method would be a complex long running function (described as state machine in the address space) that is controlled (invoked) by the client. Methods can have complex in and out parameters and state changes may fire events to inform the client of the current transition change within the state machine.
A View instead is a server defined subset (or different orchestration) of information to be shown in the address space for a certain domain or purpose. Browsing in the context of a view mainly hides references, but gives new entry point to view content.

Figure 8) Modeling Concept without References



APPLICATION EXAMPLES

There are different SDKs and Toolkits on the market (early 2009) that firstly differ in their programming language and secondly are different in their functionality. To name just a few: Unified Automation (ANSI C and C++), Softing (C++), Iconics (.NET), Prosys (Java), … These companies also offer support and maintenance services as well as integration support and platform adoption for various CPU architectures and operating systems (x86, 64bit, ARM, embedded Linux, WinCE, VxWorks, Solaris, QNX, …).
Here only a few of the most common products (early 2009) are listed, there are several others.

- Beckhoff Twincat OPC UA Client/Server
- Siemens SimaticNET S7 UA Client/Server
- Certec atvise OPC UA Client/Server
- Iconics Gensis64 OPC UA Client
- ABB ScadaVantage UA Client/Server
- Kepware KepServerEx UA Server

Other companies are working on OPC UA products that will be released and available on the marked within 2009/2010. Some distributed applications even use OPC UA as their internal inter-process communication interface or distribute UA services over the internet (cloud computing).

SUMMARY

OPC UA offers an unprecedented, standardized possibility for modeling and communication of process information, that data describes itself and is transferred secured (encrypted) and robust (transaction based and redundant) on any type of physical link (currently based on TCP/IP). Nothing prevents implementation even on exotic platforms. Using Java will lead to platform independent code and binaries must not be recompiled for different platforms. But this advantage must be expensively paid with high resource consumption and slow performance. However, the requirement to develop highly optimized and performant applications on different target platforms i.e. from small low cost embedded devices up to PC and enterprise servers regardless of the CPU architecture can effectively realized using C/C++ implementation. OPC UA is more than a web service approach for industry automation, it combines service oriented architecture and features web service and IT technologies but in a highly effective and optimized way driven by the requirements from field level devices up to enterprise systems. OPC UA is the only standardized communication technology for continuous information flow based on a generic meta model.