Service-oriented Architecture Design Aspects of OPC UA for Industrial Applications

Markus Stopper and Branko Katalinic

***Abstract*—OPC Unified Architecture is the most recent OPC specification from the OPC Foundation and differs significantly from its predecessors. The OPC Foundation's main goals with this development was to provide a path forward from the original OPC communications model (COM/DCOM) to a current infrastructure model (SOA) and introduce a cross-platform architecture for process control, while enhancing security and providing an information model. By defining abstract services into service sets groups, which are defined by their request and response messages, OPC UA provides this service-oriented architecture (SOA) for industrial applications, starting from factory floor devices and ending with enterprise applications. OPC UA integrates the different aspects of the former OPC specifications into a unified address space accessible with a single set of services. This paper gives an overview about some fundamental design aspects and the base service sets of OPC UA as well as an outline of possible migration scenarios of legacy COM/DCOM-based OPC servers.**

***Index Terms*—Service-oriented architecture, ole for process control, opc unified architecture, industrial web services.**

I. Introduction

The OPC Foundation is approaching completion of the first stage of its new Unified Architecture (UA). The new Unified Architecture is meant to unify all of the existing OPC technology [2]. Importantly it means open standards and no longer just the proprietary Microsoft standard [3]. Although the original binding to Microsoft's COM/DCOM achieved a big contribution for distribution, a lot of significant disadvantages appeared like binding to operating system, no real security, no configurable timeouts and configuration problems.

Furthermore OPC UA will allow the design of information models, which for example represent and define device data within a single point of access constituted by an OPC Server in a uniform way. With this, client-applications do have a uniform access path to device data independent from the used field-buses.

Manuscript received December 30, 2008. This work was supported in part by the Automation Research & Development Department of MKW® Austria. Special thanks to the major project promoters Hans and Johannes Danner.

Markus Stopper, *Member IAENG, IEEE & DAAAM International*, was with Department of Production Engineering, Vienna University of Technology, Vienna, Austria. He is now with the Industrial Automation IT Research & Development Department of MKW® Slovakia, Prešov, Slovakia (e-mail: markus.stopper@ieee.org).

Branko Katalinic, *President of DAAAM International*, is Professor at Vienna University of Technology, Vienna, Austria. He is head of the Department of Intelligent Manufacturing Systems, Institute of Production Engineering, Vienna University of Technology, Vienna, Austria (e-mail: branko.katalinic@daaam.org).

OPC vendors are already working through the design and implementation phase of their OPC UA technology products. After three years of specification work and a further year of prototyping the first version of OPC UA is now being released considering the following characteristics [1] for the communication stack:

- multiple platform implementation including portable ANSI C, Java and .NET implementations
- scalability from embedded controllers up to Mainframes
- the stack can support multi-threaded as well as single-threaded/single-task operation, which is necessary for porting the stack onto embedded devices
- an own security implementation, based on new standards, realizes true security (e.g. compare with [4])
- configurable time-outs for each service
- chunking of big datagram's

Consequently perhaps the most obvious change compared to the legacy OPC standard that users will notice right away is that OPC-UA is even not based on Microsoft COM/DCOM. Everyone on the market is embracing open, internet-based communication standards. Naturally, OPC has done the same by basing UA on TCP/IP, HTTP, SOAP, and XML. Not being tied solely to Microsoft platforms makes OPC-UA highly scalable.

It is expected to see OPC-UA implemented on almost everything from embedded field devices through to Unix or mainframe-based enterprise applications [1].

All of the base services defined by OPC are abstract method descriptions which are protocol independent and found the basis for the whole OPC UA functionality. The transport layer puts these methods into a protocol, which means it serializes/de-serializes the data and transmits it over the network.

Currently there are two protocols specified for this purpose. One is a binary, towards high performance optimized TCP protocol and as second, a Webservice-based one.

The OPC information model isn't just a hierarchy based on folders, items and properties any more, but a so-called Full Mesh Network based on nodes instead. These nodes can additionally transmit all varieties of meta information. Additional new and important features of OPC UA are redundancy support, heartbeat for connections in both directions, which means that the server as well as the client recognizes interruptions, and buffering of data and acknowledgements of transmitted data, so lost connections don't lead to lost data anymore. Lost datagram's can get fetched repeatedly [1].

II. Architecture Base Design

The OPC Unified Architecture blurs the distinction between Data Access, Historical Data Access or Alarms & Events in a way, that existing OPC applications are largely broken down on functional lines based on which of the OPC specifications are supported. Clients and Servers based on the Unified Architecture will blur this distinction since the transfer of all data, no matter of what its type or purpose, is transferred using the same base UA services.

This mechanism coupled with the support of complex data (arrays, binary structures, XML documents, etc.) leads to a situation where server vendors expose data with complete trustworthiness as it naturally exists in the system. They presently expose only a subset of data forced into an OPC-defined data format. Forward thinking vendors will realize that there will no longer be a need for any proprietary vendor defined interfaces with this capability and they will only expose the data via UA [1].

The OPC Foundation has been working with other standards organisations to gear UA as a premier vehicle for conveying and transporting data defined by these other specifications. With the acceptance of UML based models and XML Schema to define and render complex data, many information model specifications are emerging.

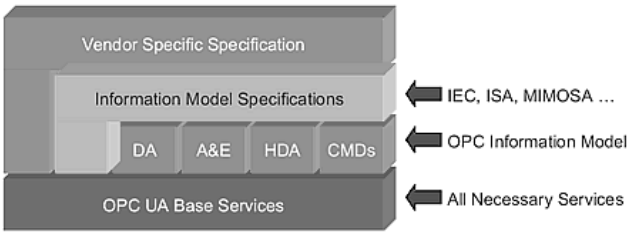

Fig. 1: OPC UA exposes info-models via a base set of services [5]

It can be expected to see companion specifications emerging that bind the data models described by these and other specifications to OPC UA, thus providing for complete information level interoperability from one application to another. Note also that applications that use this type of data are not the current round of typical factory floor applications that traditionally implement OPC. Instead they are enterprise applications for use at MES and ERP level. It can be expected to see OPC-UA implemented in certain applications far outside the scope of nowadays OPC applications [1].

Further robustness and reliability play an important role. Rather than live at the kindness of the underlying internet protocols, the Unified Architecture takes direct responsibility for all of the necessary failure detection and recovery logic making UA viable for mission-critical communication. Beyond robustness and reliability, several redundancy schemes are defined such that vendors may provide this capability in a standard way. It is expected to see OPC-UA used as the native communication method by applications, usurping proprietary vendor protocols and possibly some existing field bus protocols [1].

Existing OPC applications vary greatly in their compatibility and interoperability with other OPC applications. Although OPC is currently stressing the need for products to be both certified and tested at interoperability workshops, the reality is that only a small percentage of OPC products on the market presently meet that mark. With OPC Unified Architecture, the certification tests cover both servers and clients and will be administered by sanctioned testing labs. These facilities are expected to be available in time to test the very first UA products coming to the market. The certification status of any UA component will be available in real time via electronic certificates issued by OPC directly though the UA interface. OPC-UA applications which are properly certified thus greatly increase the likelihood of true plug-n-play [1].

Backward compatibility will ensure quick adoption. The OPC Foundation is providing its members with several code-based deliverables to simplify vendor development and aid with backward compatibility. First, the OPC-UA API provides much of the plumbing and low level generic serialisation and wire support so that vendors need only to code the high level aspects of UA into their applications. The API is expected to be available in several languages supporting multiple popular computing platforms.

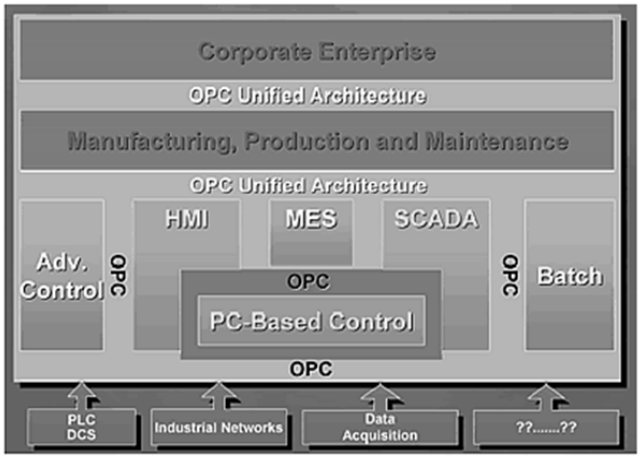

Fig. 2: OPC UA is used throughout – plant floor to enterprise [5]

In addition to the API, the OPC Foundation is also providing its members with shippable binary adapter modules which allow new OPC-UA clients direct access to all of the legacy COM-based OPC servers, and legacy COM-based OPC clients' access to a subset of the features of new OPC-UA servers. The two adapters can even be used together to connect a legacy client to a legacy server, replacing the standard DCOM communication with the new OPC-UA internet-based protocols. With complete backward compatibility ensured, vendors will have much to gain and nothing to loose by adopting the new OPC Unified Architecture [1].

Summarized can be said that OPC UA specifies an abstract set of services and the mapping to a concrete technology. OPC UA does not specify an API but only the message formats for data exchanged on the wire. A communication stack is used on client- and server-side to encode and decode message requests and responses. Different communication stacks can work together as long as they use the same technology mapping. The following section gives an overview about different components of OPC UA clients and servers followed by a section of technology mappings and a section of possible interactions between OPC UA servers [1].

III. Client and Server Parts

An OPC UA client consists of a client implementation using an OPC UA communication stack. The client implementation accesses the communication stack using the OPC UA API. Note that the API is not standardized. It may vary for different programming languages and potentially for different communication stacks. Several communication stacks may exist for different operating systems, programming languages and mappings. The client-side communication stack allows the client to create request messages based on the service definitions. The client-side communication stack communicates with a server-side communication stack. The OPC Foundation standardized only this communication. Thus, everybody can develop his or her own communication stack with its own API as well.

The server-side communication stack delivers the request messages to the server implementation via the OPC UA API. Since the OPC UA API realizes the abstract service specifications, it may be the same as on the client-side. The server implementation implements the logic needed to return the appropriate response message. The OPC UA server implementation gets its data from some underlying system. For example, this can be a configuration database, a set of devices or some OPC server [1].

IV. Technology Mapping

The OPC UA mapping specification currently defines two mappings: UA Web Services and UA Native. The first mapping uses SOAP and the various WS-* specifications. The second mapping uses only a simple binary network protocol and integrates certain security mechanisms.

The encoding of the data can be done in XML or UA Binary. UA Binary specifies the serialization of data into a byte string. The UA Binary encoding is faster than the XML encoding since the message size is smaller than for the XML encoding. On the other hand, the XML encoding allows generic SOAP-clients to interpret the data in the SOAP message, while they would only get a binary string using the UA Binary encoding. In theory, the encoding of the data is independent of the mapping. However, the XML encoding will typically only be used in the UA Web Service mapping in combination with the various WS*-specifications. The protocol of the UA Web Service mapping is SOAP/HTTP(S) while the UA Native mapping typically runs directly on TCP/IP. For bypassing firewalls, the UA Native mapping also allows putting the binary encoded messages into SOAP messages using HTTP(S). UA messages can also be encoded and transported with other protocols, for example using WCF Binary. However, by using this technology interoperability is lost since only communication with WCF clients is then possible [6].

V. Aggregating Servers

Build-into the OPC UA specification is the concept of aggregating servers. An aggregating server aggregates one or more OPC UA server and provides the information of those servers – or an excerpt of the information – in its address space. Thus, a client does not have to access several servers but only one server.

This mechanism allows a flexible architecture by chaining several OPC UA servers for different clients with different requirements [1].

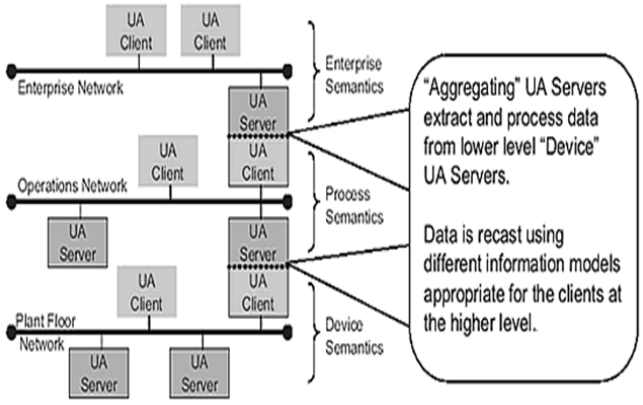

Fig. 3: Aggregating UA Servers in different information models [5]

For example, several OPC UA servers running on small devices will be aggregated by one OPC UA server. Several clients of the distributed computing system may access this server. Another OPC UA server aggregates this server and provides part of the information to the MES system. The MES system could work as an aggregating server, too. OPC UA supports aggregating servers by allowing marking the origin of data.

VI. OPC UA Service Sets

OPC UA groups their services into base service sets. An OPC UA service itself is defined by its request and response messages, thus, it is on the same level called operation in the web services description language.

OPC Unified Architecture defines thirty-four services, whereas in this section a brief overview of just some of them is given:

- Secure Channel Service Set
- Session Service Set
- Node Management Service Set
- View Service Set
- Query Service Set
- Attribute Service Set
- Method Service Set
- Subscription Service Set
- Monitored Item Service Set

The Secure Channel Service Set defines security-related services that are handled by the communication stack that OPC applications use. These services are required to guarantee a secure communication between client and server. Therefore, services to establish a Secure Channel between two communication partners are defined.

Establishing a Secure Channel requires knowing which security mechanisms to use for communication. This

information can be retrieved by a further service, which is also defined in this service set.

The Session Service Set specifies security-related services handled by the OPC UA application directly, like establishing a session on behalf of a specific system user between client and server. An additional service allows also changing the identity of a user of an active session.

The Node Management Service Set allows adding and deleting nodes and references in the address space.

The View Service Set contains services for browsing the address space. The browsing is done in the context of a view. The default view contains the whole address space.

The Query Service Set is build to query the address space. Like browsing, querying is done in the context of a view. Queries always access a snapshot of the server, i.e. each value is only provided once (no history), but the query allows to specify the time of the snapshot. Thus, the query can access one point in time of the history of the OPC UA server.

The Attribute Service Set allows reading and writing attributes, including the value attribute. It also allows accessing and updating the history of attributes and events.

The Method Service Set is calling methods.

The Subscription Service Set must be used to subscribe to data. Services exist to manage subscriptions and to receive data from the subscription.

The Monitored Item Service Set allows specifying which data should be returned for a subscription. The services allow specifying a dead band and update rates for attributes as well as filters for events.

VII. COM-based OPC Server Migration

An important topic is how to migrate old COM-based-OPC servers to the new Web-Service-based Unified Architecture technology.

Current COM-based OPC servers typically use a proprietary interface to access the data of the devices in the control network. There are basically two approaches that can be taken into account: wrapping existing servers or directly accessing device data.

Wrapping existing servers may shorten development times for implementing OPC UA servers, since existing OPC servers can be used. It is only necessary to build a Web-Service-based wrapper on top of the different OPC COM servers. This wrapper exposes the different functionalities of the underlying COM servers in form of Web Services. However, there are several drawbacks coming with that solution:

Choosing the wrapper approach implies a heterogeneous server application environment, which is harder to maintain and manage. Especially if specific system components or frameworks are updated then the interdependencies of the Web-Service- and COM-Layer have to be checked again.

From the security point of view, wrapping is not always the best solution. Web Services as well as COM components have different security-related issues to consider. Both technologies used together in one application offer an attacker more possibilities to exploit security flaws.

The other approach is to develop an UA server that directly uses a proprietary interface to access the devices of the control network. Web Services expose thereby the complete functionality. The main drawback of this solution is that the development time increases since all needed functionality has to be re-implemented.

On the other hand a direct implementation leads to a simple and homogeneous application environment, which is easier to maintain and manage. There are fewer interdependencies than in the wrapping approach and therefore updates of components and frameworks do not entail checking whether interoperability of different technological layers can still be assured.

From the security perspective, the amount of possible security considerations decreases compared to the wrapper approach, but there are still security issues to consider.

VIII. Conclusion and Further Research

OPC UA is an important step to integrate new technologies and concepts into industrial applications. The strong meta model, the open and standardized service-oriented architecture (SOA) and the ability of unifying different kind of OPC servers opens the door for new areas of application. Especially in the field of process automation, it will help to simplify the integration of other applications like MES, HMI or SCADA systems.

Other standards, for example like electronic device description language (EDDL), will take advantage of OPC UA and define companion specifications for their domain-specific information model accessible via OPC UA. Since many companies participated in developing the OPC UA specification it is expected that soon there will be many products available supporting OPC UA.

Our next step of research will focus the applicability of the binary adapter modules provided by OPC Foundation, which allow new OPC-Clients direct access to the entire legacy COM-based OPC-Servers, and legacy COM-based OPC Clients access to a subset of the features of new OPC-UA Servers.

Acknowledgment

The authors are grateful to the automation team of MKW® Austria for technical assistance as well as to Johannes and Hans Danner for inspiring ideas and creative discussions.

References

[1] OPC Foundation, “OPC Unified Architecture Specification: Parts 1–11”, 2008, Available: http://www.opcfoundation.org/

[2] OPC Foundation, “OPC Specification: Data Access, Alarms & Events, Historical Data Access”, 2003, Available:http://www.opcfoundation.org/

[3] Software AG, Technical paper: “Distributed COM on Non-Microsoft Platforms”, 2000, Available: http://www.opcfoundation.org/

[4] Intellution Inc., Technical paper: “OPC, Distributed COM and Security”, 1998, Available: http://www.opcfoundation.org/

[5] Thomas J. Burke, “The Magic of OPC Unified Architecture”, 2008, The Industrial Ethernet Book [Online], Available: http://ethernet.industrial-networking.com/articles/articleprint.asp?id=1015

[6] Resnick, S., Crane, R. & Bowen C., “Essential Windows Communication Foundation (WCF): for .NET Framework 3.5”, ISBN 978-0321440064, March 2008, Addison-Wesley Longman, Amsterdam, Netherlands, 2008