SICE Annual Conference 2008
August 20-22, 2008, The University Electro-Communications, Japan

OPC UA based Field Device Integration

Daniel Grossmann[1], Prof. Klaus Bender[1] and Benjamin Danzer[1]

[1]Institute of Information Technology in Mechanical Engineering, Technische Universitaet Muenchen, Munich, Germany
(Tel : +49-89-289-{16426, 16400, 16435}; E-mail: {grossmann, bender, danzer}@itm.tum.de)

Abstract: Today the two technologies Electronic Device Description Language and Field Device Tool exist for device integration. This paper describes a concept that integrates the advantages of both technologies into a single consistent concept. The new concept defines a model for device integration and a client-server architecture that makes use of the various advantages of OPC Unified Architecture. The authors validated the concept by setting up a comprehensive prototype including different field devices from process and factory automation. The positive results of the evaluation show that the described concept is a sustainable basis for device integration.

Keywords: Device Integration, Asset Management, EDDL, FDT, OPC UA

1. INTRODUCTION

In today's automated plants field devices from many different manufacturers have to be integrated resulting in effort for installation, version management and device operation. This challenge can only be faced with open and standardized device integration. Device integration makes the device's data and functionality available throughout the automation system. The two major integration technologies *Electronic Device Description Language* (EDDL) and *Field Device Tool* (FDT) come from process automation. Currently in factory automation even simple sensors become more intelligent (e.g. IO-Link devices) thus making efficient device integration more important. Both technologies possess different focuses on the one hand, but also a lot of overlapping on the other hand. Until the decision to join forces it was exactly this overlapping that led to a situation in which both technologies competed on the market instead of complementing each other. In the consequence DCS vendors took their positions, device manufacturers had and still have the doubled effort for supporting EDDL and FDT, and the end users, confronted with the decision „EDDL or FDT", are still insecure.

For all involved parties the ideal of course looks different: DCS and automation system vendors want to achieve robustness while assuring a high level of technology and platform independence. Device manufacturers want to support only one technology instead of two in order to reduce effort. However, they want to provide the optimal means for the integration of their devices. End users want to avoid false investments and therefore demand only one future-proof solution that offers all the advantages of the competing technologies.

2. EDDL AND FDT

The FDT technology uses programmed software components, the so called Device Type Managers (DTM), for device integration. These DTMs run inside the FDT Frame Application that manages and coordinates the DTMs. Therefore the FDT Standard [1] specifies the interfaces between the device specific DTMs and the device independent engineering system, the FDT Frame Application. Since a DTM has to comply only with the interface specification the device manufacturer is not restricted in the design of the functionality, in the implementation of the DTM software component and in the choice of the programming language. Therefore also the operation of very complex devices can be realized very well with this standard. In addition FDT specifies how the DTMs communicate to their device by using so called Communication-DTMs. One substantial disadvantage of the FDT technology is the tight coupling between the DTM software components delivered by the device manufacturer and the Microsoft Windows platform. Furthermore the freedom when programming a DTM misleads to different philosophies for device operation and can also compromise the robustness of the engineering system when programming principles are disobeyed. Current efforts of the FDT-Group like for example style guides for user interfaces try to cope with such issues.

In contrast, the also open and standardized EDDL technology [2] defines an own language that the device manufacturer uses to write textual descriptions (Electronic Device Description, EDD) for their devices. EDDs are processed by an EDD specific interpreter within a device independent engineering system. Compared to standard programming languages the amount of language elements is limited and specific for device description. This allows for easy and efficient development of EDD device descriptions as common functionality for device integration is directly supported by specific language elements. Such an EDD is independent of operating system and hardware platforms, provides a uniform philosophy for device operation, and the interpretation yields high robustness. While EDDL technology suits devices with low to middle complexity very well its limited functionality leads to restrictions with very complex devices. The current extensions of the language concerning new features for graphical user interfaces and additional functions try to resolve this disadvantage. Although this also allows for more complex devices the complexity of the language rises.

3. CONCEPT

The authors developed a new concept that integrates the advantages of EDDL – platform independence, ease of use, and robustness – and the advantages of FDT – unlimited functionality, extensibility, and market differentiation – in a clearly structured client server architecture.

To provide a basis for the concept the authors analyzed representative devices for process as well as for factory automation ranging from simple devices such as light barriers or pressure transmitters up to complex devices like laser scanners and servo drives. To reduce the overall complexity a layered structure for universal device operation was derived and validated during the analysis. The new model (fig. 1) consists of the four well-defined layers *Graphical User Interface*, *Advanced Business Logic*, *Basic Device Methods* and *Data & State Model*. Tightly related functions are clearly allocated to these layers allowing a clear classification of operation functions into a functional hierarchy.

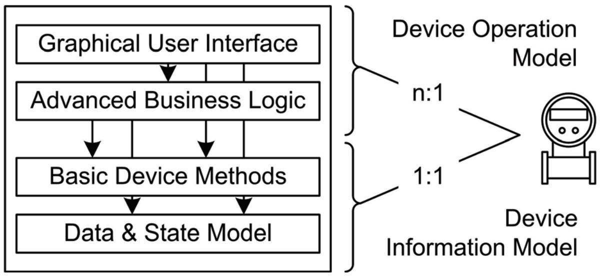

Fig. 1 Model for device integration

3.1 Device Information Model

The *Device Information Model* (DIM) is composed of the two layers *Data & State Model* and *Basic Device Methods*. The DIM establishes an abstraction layer to grant standardized access to device data and functions for higher applications like the *Device Operation Model* (DOM). Accordingly the DIM represents the device's data and functions exposed by the device firmware and is unique for each device (1:1 relationship). Applications always access device data and functions through the DIM. The DIM encapsulates the communication with a device and ensures that applications can access data and functions independent from connection specific properties e.g. communication protocols. For example the DIM of an IO-Link light barrier represents the device Parameters like near and far switching points, diagnosis and maintenance data. Applications may then access these data in a standardized way without knowing how to communicate to the device through IO-Link.

The Data & State Model makes device data available to higher layers and ensures data consistency at the same time. It contains all data presented by the device firmware for device integration. The Data & State Model reads the value of data items from the device and writes changed data items back to the device. Data items could be simple variables or parameter as well as complex data structures such as linearization tables or envelope curves. Additionally the layer provides meta information for data items e.g. name, type, unit or an area of validity, and semantic information that allows determining e.g. alarms and alarm limits. The Data & State Model includes all logical and dynamical dependencies of the data items to ensure the consistency of the data model. This covers access control dependent on the state of the device or the current user role. Maintaining consistency between indirectly related values is very important, too. For example a process value and a dedicated alarm limit may both depend on the selected unit. If the unit of a process value is changed the device firmware adapts the corresponding alarm limit as well. At that time the Data & State Model would be inconsistent. It is then the task of the State Model to monitor the dependencies and automatically ensure the consistency for example by refreshing the converted alarm limits.

The Basic Device Methods contain basic and atomic device functions like calibration or factory reset which could be implemented in the device but are outsourced due to limitations (CPU, memory …). Basis Device Methods use the underlying Data & State Model to fulfill their function.

3.2 Device Operation Model

The Device Operation Model (DOM) is composed of the two layers *Advanced Business Logic* and *Graphical User Interface* (GUI). The features of the DOM depend on the use case or user role for the device operation. Since different use cases and user roles pose different requirements more than one DOM can exist for a single device (n:1 relationship). For example different DOM functionality is needed for device commissioning or maintenance. But all DOMs have in common that they access the device's DIM in order to access device data and functions.

The Advanced Business Logic contains all high level functions to support the user during device operation. To support the GUI complex processing functions are available through this layer. For example the Advanced Business Logic could import a complex scan field from a CAx application and automatically transform the data for a laser scanning device. Another example would be an Advanced Business Logic that automatically calculates the linearization table of a level device based on CAD geometry data of the vessel. Device data may be pre-processed by this layer to increase the usability during device operation as well. For example a servo drive stores just a few points from the complex spline of a cam disk. For device operation a graphical representation of cam disk would suit the user most. Accordingly the Advanced Business Logic may calculate supporting points for a smooth curve progression to offer an ergonomic presentation to the user. Vice versa the user may comfortably draw a cam disk and the Advanced Business Logic calculates the parameters for the device. If direct access to the data items of DIM is sufficient and no such functionality is needed this layer may be empty.

The GUI is the system boundary and interacts with the user. This layer presents and manipulates the use case specific device data and functions. Simple GUIs directly access the data items of the DIM or use functions from the Advanced Business Logic in more complex cases.

4. ARCHITECTURE

4.1 OPC Unified Architecture

The ongoing growth of internet technologies offers new solutions especially for the platform independent interaction of automation systems. Web services are more and more appreciated for cross platform data exchange, because they are available on many different platforms. To create a standardized data exchange model for automation purposes, the OPC Foundation specified the OPC Unified Architecture (UA) [3] [4] [5]. OPC UA defines a vendor and protocol independent client-server architecture based on standard web technologies that assures interoperability. But the most important feature of OPC UA is the possibility to define powerful information models. These models define the organization of data within the address space of an OPC UA server in terms of structure and semantic. As a result OPC UA is perfectly suited as the central interface for the vertical integration of automation systems. Therefore the potential of this technology makes an ideal base for a new device integration concept.

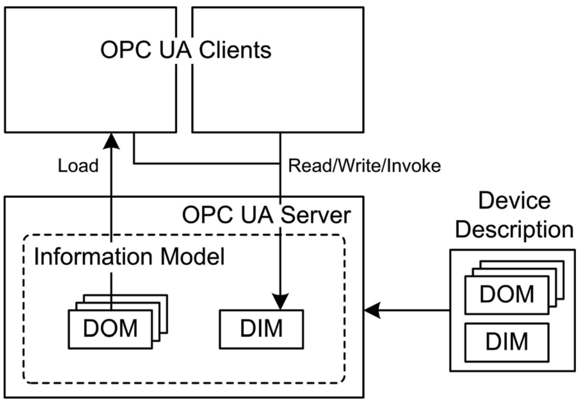

Fig. 2 Architecture overview

Figure 2 shows the architecture of this concept that consists of clients and a server. OPC UA provides the technical basis for this client-server architecture. The concept uses all advantages of OPC UA like for example platform independency and the possibility to define powerful information models on the server side. In this architecture, the server loads the device description containing DIM and DOMs. While the DIM is directly processed within the server to establish the information model, the DOMs are only handled in terms of server side storage. The server side information model allows OPC UA clients to access the device data and functions that are part of the DIM. Besides the possibility to access the DIM, clients that are aware of this concept can load the DOM from the server and render the device specific user interface based on the current user role.

4.2 Server

The device specific DIM is established as a well defined OPC UA information model on the server side. It concentrates on the representation of the device interface regarding structure and behavior. EDDL was originally developed to describe device specific properties and is perfectly suited to describe the DIM. At the same time EDDL is executed by an interpreter. This ensures that EDDL is platform independent, robust and technology independent – the most important requirements of a DCS or automation system vendor for a server side technology.

The device specific DIM described in EDDL is the base for the information model within the OPC UA server. The EDDL-interpreter of the server processes the EDD of a device and establishes the appropriate information model in the server address space (fig. 3).

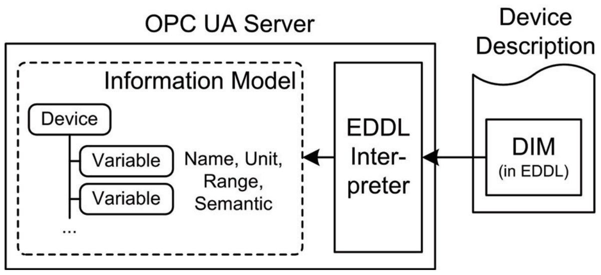

Fig. 3 Information model in the server

Each device is represented by a *Device* object. The Device object includes sub objects which hold the data items of the data model. This covers meta information e.g. the unit, the range and similar information. The meta information may be extended by standards like the NAMUR (international user association of automation technology in process industries) NE100 or ISA. Based on this meta information it is imaginable to develop adaptive clients which retrieve and process data automatically. If an OPC UA client accesses data in the server address space, the server reads the corresponding device data. For data exchange of device data the server uses the communication specific information included in the EDD. Additionally the State Model ensures data consistency and handles device specific protocols e.g. the data exchange of large data items may be divided into smaller frames due to limitations of the device's fieldbus interface.

4.3 Open Communication

To offer the DIM access to the device data the server communicates with the device over the fieldbus interface. The concept defines an Open Communication Model to create a universal solution for this task (fig. 4). Normally networks of automation systems are heterogeneous with a hierarchical structure. The transitions between networks are managed by gateway devices. To establish a communication with a device the server needs access to a network. The Open Communication Model defines Access Points which enable access to a network. The server represents these Access Points as *Access Point* objects inside the server's address space. These objects implement a standard communication interface – similar to the FDT Communication-DTM interface. The Access Point objects are described in EDDL and communicate with a dedicated communication controller – the so called *Communication Server* – via OPC UA. A communication controller grants access to a particular fieldbus network and offers an OPC UA server interface to connect with the Access Point object. A similar concept is used for gateways. Gateways consist of the physical gateway device and server side *Gateway* objects which are described in EDDL. Gateway objects implement the same standard communication interface like Access Point objects. Therefore Gateway objects may be chained to overcome network boundaries. With the Gateway and Access Point objects and information about the network structure the server is able to establish a connection to any field device. Therefore the concept picks up FDT's communication concepts and assures a system manufacture and server vendor independent open communication in complex network structures.

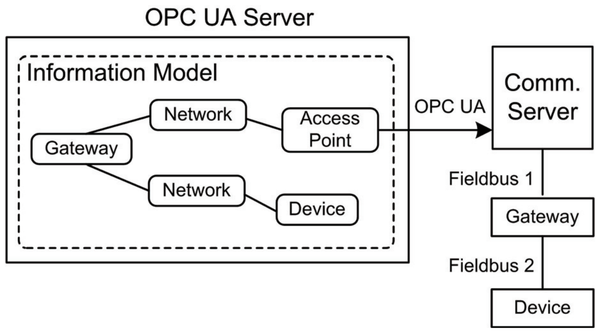

Fig. 4 Open Communication

4.4 Client

There are two different types of clients: The universal *Device Engineering Framework* (DEF) and application specific clients. While application specific clients directly access device data via the DIM and process those in an application specific way the universal DEF concentrates on device operation via the device specific user interface. The DEF offers a dynamic plug-in mechanism for DOMs that contain the device specific user interface – similar to the FDT Frame Application. For simple GUIs the elements provided by EDDL are sufficient and the DOM is completely described in EDDL. For more sophisticated GUIs a DOM may include programmed parts or even be completely programmed. This integrates the flexibility and power of FDT into the new concept.

To fulfill these tasks, the DEF consists of several components (fig. 5). The *Framework Manager* is the central component that orchestrates the lifecycle of the *Device Operation Containers* (DOC). The DOC is responsible for displaying the device specific user interface. When the Framework Manager instructs the DOC to display the user interface of a device, it loads the role specific DOM from the server. In order to display EDDL driven user interfaces the DOC contains an *EDDL GUI Renderer* that reads the EDDL user interface description and renders the according GUI comparable to a web browser showing a HTML webpage. If the DOM contains programmed elements the DOC provides the framework they need in order to run. This includes the *OPC UA Client Services* which provide functions to the DOMs to access device data and functions via the DIM in the server. That way, developers of device descriptions do not have to deal with implementing OPC UA clients resulting in easier and more efficient development of device descriptions. The DEF seamlessly integrates programmed GUI functionality and EDDL GUI descriptions to hide the transition between descriptive and programmed parts from the user.

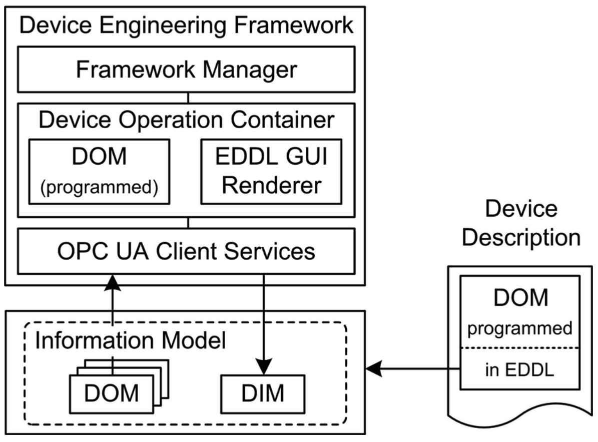

Fig. 5 Device Engineering Framework

4.5 Deployment

Different scenarios for the deployment of the new solution are possible. By default the DCS or automation system vendor delivers with the system an OPC UA server that provides access to the device data within its address space via an open interface based on the DIMs of the plant's field devices. The Device Engineering Framework as OPC UA client for the DOMs can also be part of that system or can be provided by other vendors.

The open configurable „Nested Communication“ is another important feature of the OPC UA server that allows integrating and supporting other devices with different fieldbus protocols independent of the system vendor. Only such openness helps to gain acceptance and to avoid the fear of being dependent especially for small and medium-sized companies.

The device vendor delivers together with the device a device description that contains the DIM and DOMs for the device and that is stored within the OPC UA server database. This DIM/DOM package is the replacement for the EDD and DTM that is currently developed in a redundant way. The end user can benefit from the OPC UA server by retrieving device data for other applications such as Asset Management and MES. So the OPC UA server leverages device operation and allows the integration of other useful applications.

5. PROTOTYPE

A comprehensive prototype was demonstrated during Hanover Fair 2007 that proofs the feasibility of the concept. For a drive the challenging DIM with more than 2000 parameters was completely described with EDDL. Via the parameters the so called free function blocks of the drive can be activated and connected to create simple logic functions directly in the drive. This is achieved by the DOM of the drive that consists of both an EDDL based part, such as a quick commissioning dialog, and a programmed extension that provides a graphical editor to connect the free function blocks of the device via drag and drop (fig. 6). The impressive example demonstrates the seamless integration of descriptive technologies and the programmed graphical editor. The editor supports the user to create the drive logic in a graphical environment very comfortable via drag and drop. Special functions in the Advanced Business Logic calculate the correct parameters to configure the function blocks of the drive in the background enhancing usability.

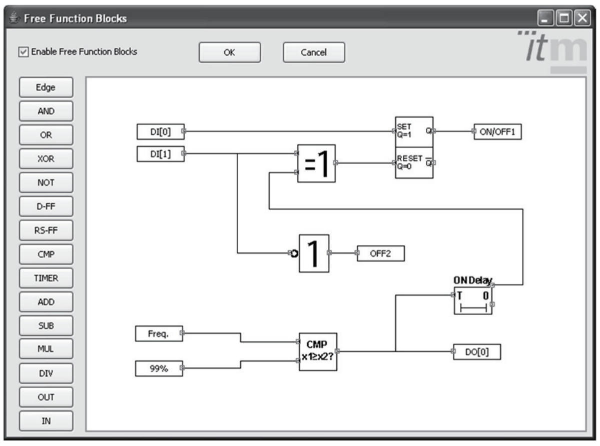

Fig. 6 Programmed editor

Figure 7 shows the topology of the prototype. The Integration of CANopen communication via a CANopen Communication Server demonstrates the described open communication concept. The Profinet Communication Server additionally emphasizes that the open communication concept is also applicable for higher level and very powerful protocols. The prototype also addresses "Nested Communication" through heterogeneous networks by integrating a Profibus – HART Gateway as well as a Profibus – IO-Link Gateway. The usage of Profinet, CANopen and IO-Link devices within the prototype points out that the described solution is not restricted to process automation but suits factory automation devices just as well.

For the client side Device Engineering Framework and for the programmed DOM functionality Java was used providing platform and operating system independency. The fully Java based OSGi [6] technology provides the framework for the programmed components of the DOMs. To demonstrate the platform independence of the Java based prototype, it was tested with a variety of operating systems including Windows XP, Linux, UNIX, Solaris, Mac OS and even Windows 98. The fact that the prototype was deployed just by copying without requiring any recompilation highlights the manifold advantages of using Java. Lately the prototype was extended to facilitate device operation even via PDAs and Smartphones.

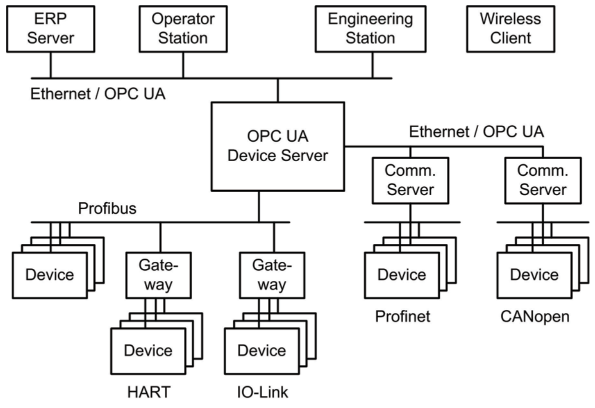

Fig. 7 Topology of the prototype

On the client side the prototype was also extended in various ways showing the flexibility of the concept. This includes the Integration of the Device Engineering Framework into an HMI solution allowing the operator to open the device specific DOM from within the HMI application for example to get detailed information on the device's status. Also the use case of attaching an ERP system like SAP was integrated into the prototype. In this case, SAP was extended to act as an OPC UA client communicating to the DIMs of the devices in the Device Server. The extension uses the SAP Java Connector technology to interact with the SAP system. SAP periodically polls the devices for their status and decides whether maintenance or replacement work orders for the devices have to be issued. In the case of a faulty device the SAP system creates a maintenance order while simultaneously checking the availability of the required spare parts which are automatically ordered if necessary. Here again the advantages of using Java for the implementation are noticeable as SAP offers with its

Java Connector in combination with the Java Connectivity Builder an easy to implement access to SAP BAPIs (Business Application Programming interfaces). Through the accordant BAPIs, maintenance or part orders can be created.

6. CONCLUSION

Besides the unique chance to unite the two technologies EDDL and FDT, that would develop in parallel otherwise, the solution yields a variety of significant benefits. Platform independence and robustness via interpreter based execution (EDDL and Java) allow for long term technology independence which is a very important issue in industry. The central management of the elements DIM and DOM within the OPC UA server reduces effort for installation and versioning extremely since, by loading the DOMs from the server, there are no client side installations necessary. This is a big advantage especially in multi client environments (i.e. Operator-, Maintenance-, Diagnosis-Station).

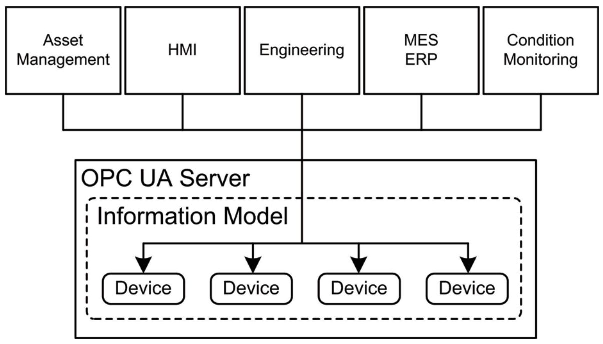

Fig. 8 Central access to device data

By providing all device data for client applications the server also allows the easy and open connection of further client applications like Asset Management-, MES- und ERP-applications (fig. 8). But the biggest advantage of all is the clear structure of the architecture that defines distinct tasks reducing the complexity in device integration which has become a major issue.

6.1 Summary

For device integration two technologies – FDT and EDDL – with a high level of overlapping but different philosophies were established over time. Although both EDDL and FDT have their specific advantages the situation for DCS and automation system vendors, device manufacturers, and end users is not satisfactory at all. The described concept picks up the advantages of both technologies and integrates them in a single, universal, and clearly structured architecture. Besides the unique chance to unite the two technologies EDDL and FDT the solution implies a variety of significant benefits. Platform independence and robustness via interpreter based execution (EDDL and Java) allow long term technology independence which is a very important issue in industry. Since a migration strategy assures that all existing DTMs and EDDs can be used further on without changes the protection of investments is achieved. The immense potential of the new OPC Unified Architecture yields new scenarios like Asset Management or MES and leads to a robust and platform independent solution.

The announcement of EDDL Cooperation Team (ECT) and FDT-Group to join forces specifying Field Device Integration (FDI) based also on the described concept is a major step forward. The described concept is an input to the joint team that will develop the specification for FDI.

The authors are convinced that the concept is a sustainable basis for a united future of FDT and EDDL. The presentation of the concept at the annual general NAMUR meeting and the Hanover Fair drew big attention which emphasizes the importance of this issue. It is now important that especially the end users make sure that FDI meets their demands: A single, robust and future proof technology for device integration.

REFERENCES

[1] FDT Joint Interest Group; “FDT Interface Specification”, Version 1.2.1; March 2005; FDT-JIG - Order No. 0001-0001-002.
[2] PROFIBUS Nutzerorganisation e.V.; “Specification for PROFIBUS Device Description and Device Integration”, Volume 2: EDDL, Version 1.2; Karlsruhe; November 2005; Order No: 2.152.
[3] OPC Foundation; “OPC Unified Architecture Specification Part 1: Concepts”, Version: Release 1.00; July 2006.
[4] OPC Foundation; “OPC Unified Architecture Specification Part 3: Address Space Model”, Version: Release 1.00; July 2006.
[5] OPC Foundation; “OPC Unified Architecture Specification Part 5: Information Model”, Version: Release 1.00; July 2006.
[6] OSGi Alliance; “OSGi Service Platform Core Specification”, Version: Release 4; August 2005.