Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 81 (2019) 310–315

www.elsevier.com/locate/procedia

52nd CIRP Conference on Manufacturing Systems

Multi-protocol Data Aggregation and Acquisition for Distributed Control Systems

David Albert Breunig[a,*], Matthias Schneider[a]

[a]*Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Nobelstraße 12, 70569 Stuttgart, Germany*

* Corresponding author. Tel.: +49-711-970-1375. *E-mail address:* david.albert.breunig@ipa.fraunhofer.de

Abstract

In modern automated industrial plants, the share of components embedded in a complex network using nowadays communication protocols grows. This leads to plants having several interfaces which offer data and which must be addressed during commissioning. To ease integration and avoid errors, communication interfaces of all involved components can be aggregated. However, as the components' cooperation realises higher level functions-in the plant, it is necessary to enhance aggregations with a plant-specific information model, that incorporates aggregated data and abstract specific plant functions. This paper introduces a model-driven approach to realise locally deployable data aggregation systems that integrate plant-specific information models.

Keywords: data aggregation; information model; DCS; model deployment; distributed; OPC UA

1. Introduction

With the ongoing reduction in effort and cost to realise sensors and actuators as embedded systems with own computation and storage functions, modern automated industrial machines and plants evolve strongly into system-of-systems. Such systems are not created by developing a combination of specifically engineered components. Instead they are based on individual systems that integrate standardised technologies, even often utilizing off-the-shelf hardware, especially when it comes to soft real-time systems and communication technologies like Ethernet or TCP. Software-defined integration combines those modern small self-sufficient automation components into task-fulfilling manufacturing systems. The evolving system-of-systems concept in industrial plants increases complexity in engineering. While standardised technologies and interfaces ease the interconnection of different systems, the growing number of systems, the huge amount of information and the wider spectrum of data provisioning increases complexity in engineering and usage of such systems and make vertical integration more complicated. These new challenges lead to a need for methods and system that handle those problems to support the realisation of both flexible and resilient automation. This paper presents an approach for an extensible, modular gateway that enables data aggregation and acquisition of distributed control systems (DCS) across multiple communication protocols.

2. Motivation

Traditionally, the hierarchy of automation devices and applications is explained with pyramids according to the ISA-95 [1]. Devices and applications act and communicate mostly on their hierarchy level. Vertical communication is static and specifically set up. To allow reconfiguration at any time the hierarchic levels must dissolve to a system of equally treated nodes which can take part in any business and manufacturing process. [2]

10.1016/j.procir.2019.03.054

All devices and applications that shall act as reconfigurable and interconnected nodes need a common communication platform. One technological approach for this is OPC Unified Automation (OPC UA). Besides OPC UA, consumer- and IT-aimed technologies like REST or MQTT are more widely used in industrial environments, because of cheaper hardware, open-source software, technological possibilities and maturity, and especially the easier access from high-level PC office systems [3] Since devices and applications in industrial automation are distinguished by their operating time compared to IoT and consumer devices, the current limited communication technologies will stay in use for years. The increasing massive machine-to-machine communication raises various challenges for acquisition and aggregation of data:

- **Higher complexity**: The distribution of functions of a system to several sub-systems (e.g. field devices) with different communication protocols increases the complexity of vertical integration.
- **Huge amounts of data:** The increasing availability of data down to the sensor and actuator level results in huge amounts of data. Many of these (raw) data can usually be pre-processed to reduce both the network load and the amount of transferred data.
- **Extensibility of data provisioning:** Sensors and actuators are usually small embedded systems with little performance and therefore limited functionality. These devices can't be extended by additional data provisioning functions (e.g. pre-processing or history generation).
- **Security:** Data within production networks are often sensitive and should not be accessible from outside. In addition, external accesses to the production network is security-critical. The more devices that can be reached externally from this network, the larger the attack surface of this network.

The approach presented in this paper addresses these challenges. Through data aggregation, distributed systems can be uniformly represented to external systems and the complexity of several sub-systems is not transferred to these external systems. By pre-processing the data, the amount of transferred data as well as the network load is reduced. Small embedded systems can be extended by additional data provisions functions. By concentrating on one device, only one part of the network is exposed and the attack surface of the network is kept to a minimum.

3. Data in Industrial Automation and related work

3.1. Field data providers and technologies in use

Systems made out of Programmable Logic Controllers (PLC) and hard-wired (field bus) devices are the standard approach when production processes require digital and hard-real time control. Such systems are the main providers of information on the field level. Due to limited storage and computation abilities, regular PLCs cannot collect and store huge amounts of data. PLCs are preferable for discrete processes where software changes are rarely made. They are complemented by standalone sensors and actuators for non-real-time tasks. Some PLCs are hardwired combinations of PC-like x64-/x86-systems and closed, proprietary real-time hardware, so control tasks can be integrated, triggered and supervised via x64-/x86-software. Most PLCs are single-systems, designed according to the IEC 61131 norm. They are the main current data-providing and data-dependent devices in automation and plant control. Programming languages of IEC 61131 focus on fulfilling basic requirements like controlling I/O and realising logical and mathematical functions, communication control is restricted. High-level communication (e.g. data distribution to computer systems) works by either closed, manufacturer-dependent function blocks or by running additional communication modules in a runtime environment, which provides access to specially marked data fields of the PLC. The quality of data and distribution is bound to the possibilities the PLC manufacturer's software offers. Many of today's PLCs offer TCP/IP-based protocols for vertical integration, like OPC UA or MQTT. Formatting and data access though are still dependent on implementation both by the manufacturer and the control program run on the PLC.

3.2. Interoperability between systems and protocols

A common problem is the insufficient interoperability between communication protocols. Communication system used in industrial environments vary in protocol design and architecture, hardware and usage. Several communication protocols, especially field bus protocols, rely on special hardware and cannot be integrated directly into a site- or company-network, neither physically nor logically.

Some communication protocols make use of predefined formats for data to be transmitted, by e.g. reserving data fields for names or timestamps, or by providing a fixed set of datatypes that can be used for operation. Other communication protocols do not specify any formatting for their actual payload. Additionally, communication protocols differ in their general structural approach, that can be client-server, publish-subscribe or prosumer-consumer [3]. Even if devices use a common protocol, different interfaces and behaviour on the application layer may still make the devices incompatible [4].

Attempts to find solutions for interoperability problems are mostly IoT-oriented, e.g. aimed towards embedded devices with PC communication hardware (Ethernet, Wireless LAN, Bluetooth), low-cost x86-/x64-/ARM- or related hardware, and often with Linux operations systems. Industrial electronics and hardware are closed and often fully inaccessible for modifications, so the user is bound to the device's specific abilities.

Lobashov et al. [5] identify different approaches for interconnecting industrial protocols in vertical integration, one of which is using application gateways that have IP/fieldbus translation abilities. This method offers an elegant and reusable solution, while decoupling field and office levels. However,

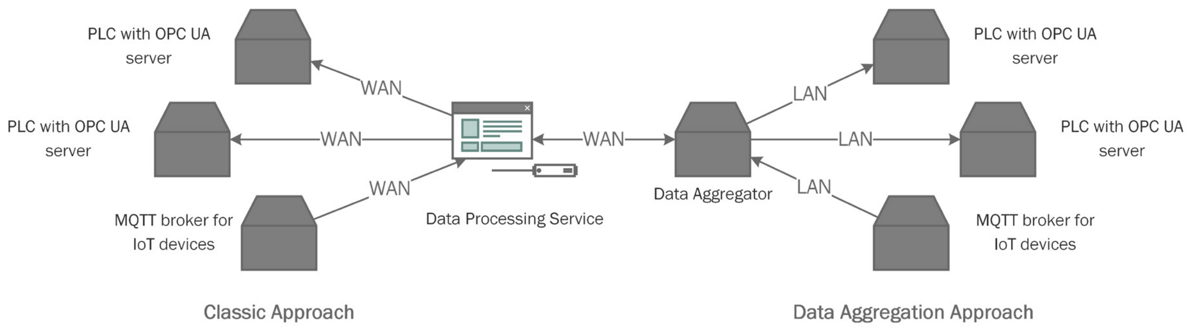

Fig. 1. Architectures of classic and device aggregation approach.

implementation and integration efforts are greater compared to fieldbus-wrapping methods [5].

Derhamy et al. [4] present an approach for interconnecting IoT devices with HTTP, XMPP, MQTT, OPC UA or CoAP interface. The approach utilises distributed modules ("cloud-edge") for translating messages sent by and to IoT devices into an intermediate data format. The approach does not address data transformation or aggregation. Based on [4], *Derhamy et al.* [6] examine the interoperability of OPC UA in service-oriented architectures and introduce a translation service for HTTP/CoAP/MQTT-to-OPC UA. OPC UA node management functions are also mapped to the intermediate format.

Cruz et al. [7] present a bridging approach from MQTT to HTTP. As HTTP and the RESTful services built upon HTTP are popular for their easy use when integrating web services and IoT devices, *Cruz et. al* propose a MQTT-client-to-HTTP-client translating service, which listens to a topic on an MQTT broker and forwards all incoming messages to a HTTP server.

In a partly similar approach to *Cruz et al.* [7], *Schiekofer et. al* [8] propose an integration of HTTP/REST and OPC UA. Both approaches provide information from a communication protocol on a HTTP/REST based server, which can be accessed from other devices. Any device or application endpoint that is interested in data must have a HTTP/REST client functionality that can be configured to fit the server's data offerings.

3.3. Pre-platform concentration of devices and data

Internet or site-intranet platforms are data receivers and management systems for devices and applications. When hundreds or thousands of endpoints are connected to a single platform, complexity and data load must be reduced, for which a pre-platform concentration of devices and data is necessary. Such can take place in a gateway or a smaller, local platform (often called "Fog"). *Delsing et al.* [9] call the usage of local cloud platforms a reasonable way for the future integration of IoT devices in industrial environments, and propose the static implementation of protocol translators into such platforms. *Kum et al.* [10] introduce a "Fog" IoT architecture for aggregating communication. A pre-platform server does not route all communication between device and platform but create depictions of data on the platform. The device accesses buffered data on its nearer fog server instead of data on the farther away platform.

Fantacci et al. [11] show an approach to gateway usage in home automation, in which devices of several relevant protocols were concentrated on a common communication gateway, that uses the Open Mobile Alliance (OMA) DM protocol for platform connection. OMA DM was created for mobile device management [12]. In the approach, technology-specific adapters provide connectivity for different home automation protocols. The adapters are instructed by a model that defines which information has to be collected from the connected devices. The gateway interacts locally with user's devices, and with external service providers over the internet.

Seilonen et al. [13] introduce a concept for aggregating the data of multiple OPC UA servers to a single separate server. The developed concept designs a system consisting of a OPC UA server, an aggregating application, one or more OPC UA clients and adapters to other data providing servers. The aggregation application collects data and provides it on its own OPC UA server. Being a pure aggregation, collected nodes are routed through without changes or transformations.

Banerjee and Großmann [14] present another approach for aggregating OPC UA. The approach and the aim are fairly similar to [13] with a further focus on combining OPC UA servers that use the standard OPC UA information models for FDI, AutomationML and PLCOpen. An external "Global Mapping Repository" supplies the shown Aggregation Layer with rulesets for mapping. [14] bases on earlier work presented by *Großmann et al.* [15].

Current approaches mainly focus on single communication protocols such as OPC UA, MQTT or fieldbus protocols. In comparison, the approach presented in this paper enables the acquisition and aggregation of data across different communication protocols. The modular architecture makes it easy to provide support for additional communication protocols. At the same time, additional modules e.g. for data aggregation or pre-processing can be added.

4. Concept

The target of the presented approach is to define an abstract, multi-protocol data handling system, based on a generic and adaptable instruction model. The system addresses networks of industrial control systems. Its information model defines which device and applications must be integrated, which data must be received or read, which transformations and aggregations must

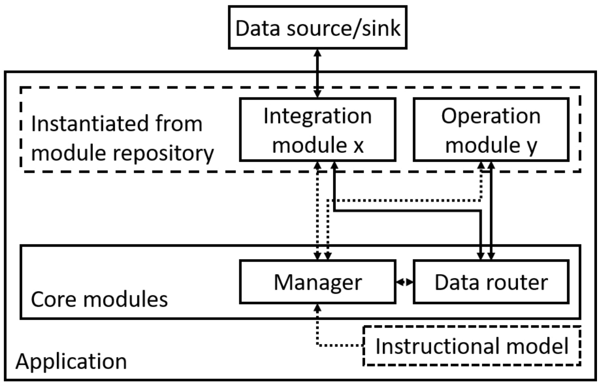

Fig. 2. Basic architecture of the concept.

be done to the data and how data must be provisioned for its clients. As seen in Fig. 1, the goal is to reduce the number of externally available information providers of a plant or a DCS to one.

Unlike other approaches which concentrate on single problems like cross-protocol and cross-device communication or aggregation like [8], [7], [13], this approach shall provide a single system for translation, aggregating and transforming communication. The goals and main points of the approach are:

- **Extensibility**: Concept and implementation must be open for the addition of new technologies for receiving and sending data and exposing functions. The model and implementation shall be split in modules for in- and output, data collecting, data transformation and data provisioning.
- **Independence from protocol, format and paradigm**: It is necessary to define a common minimal set of relevant information among communication protocols, so data can be carried from any source to any target. All relevant paradigms and formats must be handled. To do so, a minimal set is defined as a bundle of name, value and timestamp, and instructions for data access are added.
- **Base for vertical integration**: The model and implementation do not aim to fulfill hard real-time requirements. It shall run on computer hardware and be connected with standard network hardware.
- **Portability**: Model and system shall be portable and dividable. Extensibility and portability are supported by modularising the system. Doing so, the model itself can be derived from an external system (like a device management software) and the modules can be distributed to several platforms, e.g. to be nearer to control systems.

4.1. Instruction Model and core modules

The Instruction Model is the basic information model that instructs the application which communication modules must be run, which information has to be get over those modules, which operations must be done with the data, and how and which data must be output. An information as seen in the context of this concept is a minimal set of data depicting a variable or an object which is name, actual value and a timestamp. An example for an Instruction Model is shown in Fig. 3. The Instruction Model sections reflect the application's functional structure:

- **Integration modules:** Integration modules realise communication-protocol specific integration and are held ready by a repository (e.g. classes, plugins, runtime libraries). Those modules realise in- and data output.
- **Information to get:** This section instructs the application which information must be collected and which integration module must be used for every information.
- **Information to provide:** This section as model instructs the application which information must be provided over which integration module.
- **Operations:** Operations, like object creation, aggregation into data service or averaging, enable the pre-processing and aggregation of data. The operations are provided by Operation Modules. Each operation identifies an operation module to use, provides a description for the operation having necessary information and variables, and depicts a new information as a target for the operation's result.

The two core modules are the application's organising and data distributing centre (see Fig. 2):

- **Manager:** The manager parses the Instruction Model and stores the instructions for all modules. All modules must register with the manager to receive their instructions.
- **Data router:** The Data router listens for new information from the modules. If a new information is received, it is forwarded without checking over an information-specific topic to all subscribed modules.

Information provided by the integration modules and by the operator modules will be treated equally by the Data router. Therefore, both unchanged and changed data can be provisioned at the same level. If an information is actualised and distributed, receiving operation or integration module can decide whether to wait for other information that are part of an operation or a provisioning, or to use old values for the other information. Additional module-specific instruction, like OPC UA node ids, can be placed in "Access"-sub-objects. Similarly, modules can receive additional module-specific instructions over the "Configuration" sub-object.

4.2. Integration and operation modules

Based on the model, the necessary integration and operation modules are instantiated (see architecture in Fig. 2). Integration modules realise communication-protocol specific integration and are held ready by a repository (e.g. classes, plugins, runtime libraries). Those modules realise in- and/or data output functionality. Operation modules offer specific abilities for processing data (e.g. object creation, aggregating into data series or averaging). All modules register themselves at the Manager module using their module id (the key of the defining JSON object in the model), which then distributes instructions concerning information and operations.

5. Implementation

The described concept was implemented as a distributed .NET and C++ application. This implementation mainly focuses on the functional part of data acquisition and aggregation and does not make use of dedicated security functions (e.g. authentication or encryption). This aspect will be considered in future work. The implementation follows the modularized approach described in the concept. Each module is implemented as an independent program. In order to link these independent modules with each other the message queue ZeroMQ is used. ZeroMQ has the advantages that it is broker-less, implementations are available in many common programming languages and it offers high performance for message exchange. For implementation, an object-oriented approach is used. The implemented integration modules (OPC UA server, OPC UA client, MQTT client) and operation modules (Math module, aggregation module) are derived from a common base class, so that further integration modules for other protocols can be added with little effort. The internal data model is created by the Manager module directly from the instruction model provided in JSON.

After interpreting the Instruction Model, the Manager module holds derived, module-specific instructions for modules that are part of the application. All modules, including the Data router, connect to the Manager module and request instructions. Unlike other modules, the Data router is a fixed part of the application and not instantiable more than one time. After registering to the Manager module and receiving instructions, the Data router subscribes to new information of all modules. Thus, new information can be received from an integration module via the corresponding communication protocol and published to the Data router. The same applies to the operation modules, which can publish their results after the operation has been processed. The Data router publishes all received data to the relevant modules. Therefore, integration modules which provide information or operation modules subscribe at the Data router for information that are relevant to them. In this way all modules are notified about relevant information when new information arrives and can thus keep a representation of their relevant data. Based on this an operation module can directly perform its operations with the latest data without querying them.

6. Evaluation

For evaluating the concept and the implementation, a model plant operated by compressed air and controlled by two Siemens ET200SP PLCs and an ARM-based Kunbus Revolution Pi PLC was used. Both Siemens PLC run OPC UA servers for data provisioning. The Revolution Pi PLC provides sensor values via MQTT.

The aggregation application runs on a PC and uses integration modules for connecting to the OPA UA servers and the MQTT broker. It runs an OPC UA server integration module for final data provisioning. All devices are interconnected via an Ethernet network.

```
{
.."Modules": {
...."ClientModule1": {
......"Type": "OpcUaClient",
......"Configuration": {
........"OpcUaServerAddress":”opc.tcp://plcPress:4840"
......}
....},
...."ClientModule2": {
......"Type": "MqttClient",
......"Configuration": {
........"Broker": "192.168.0.103",
........"Port": 1883
......}
....},
...."MathOperationModule": {
......"Type": "MathOperator",
......"Configuration": {}
....},
...."AggregationModule": {
......"Type": "Aggregation",
......"Configuration": {}
....}
..},
.."InformationsToGet": {
...."state": {
......"Module": "ClientModule1",
......"Access": {"NodeId": "ns=2;s=runState"}
....},
...."pressure1": {
......"Module": "ClientModule2",
......"Access": {"Topic": "Pressure1"}
....}
..},
.."Operations": {
...."operation1": {
......"Operator": "MathOperationModule",
......"Description": "${pressure1}/1000",
......"Result": "p_bar"
....},
...."operation2": {
......"Operator": "AggregationModule",
......"Description":{\"State\":${state},\"p\":${p_bar}}",
......"Result": "output"
....}
..},
.."InformationsToProvide": {
...."output1": {
......"Module": "ClientModule1",
......"Access": {"Topic": "pressures"},
......"Source": "output"
....}
..}
}
```

Fig. 3. Simple example for an Instruction Model.

Before using the application, the control systems of the model plant were already connected to an externally run cloud platform. All PLCs had to be exposed to the virtual private network of the cloud platform. As the systems' data must be available outside of the virtual private network as well, public routing through the firewall for every PLC had to be created and maintained. With the application, the routing could be reduced to a single route and the control systems are now decoupled from both internet and virtual private network.

As for the data transformation capability of the implementation, the presented exemplary math operator module was used to average compressed air pressure and flow values on two separate compressed air lines. The lines feed separate parts of the plant, while being fed by the same air source. The implemented math module already introduces a great benefit at minimal expense, as now unit transformations, averaging and other operations can be realised in a DCS without modifying existing control programs.

The current approach concentrates on core data like name, value and timestamp and does not offer special, architecture-specific metadata like different timestamps or datatype references in OPC UA. Such data could be added by adding a serialisation of the raw input variable or object to every message. However, this is only useful if involved modules are able to understand the additional data. The authors decided against this feature after a first implementation because of the bigger message size and the small benefit in actual usage.

7. Conclusion and future work

The presented approach shows how to aggregate data in production networks with huge number of participants with different communication protocols. This achieved by introducing an information model that enables a low effort integration of different field devices. Various communication protocols can be supported by appropriate integration modules. Integration modules for OPC UA and MQTT were implemented prototypically. The information of the different field devices is provided by these integration modules and can be used for several pre-processing steps such as aggregation or mathematical operations. Furthermore, we applied the presented approach in a real-world usage scenario with industrial PLCs. The research presented in this paper lowers the effort for data aggregation and cloud integration of field devices on the one side. On the other side the security is increased by reducing the attack surface of production networks by concentrating the data access to a single point.

As future work, we plan to implement additional integration modules for other communication protocols to support more field devices. The pre-processing possibilities should be extended by further functions such as history generation. Furthermore, the modelling of the instruction model can be simplified by graphical tools, and the application and modules should be deployable on edge devices from cloud. The implementation can serve as basis for a central data handling service in local or on-site platforms.

Acknowledgements

This research was done within the research project *Center for Cyber-Physical Systems (S-TEC ZCPS)* at Fraunhofer Institute for Manufacturing and Automation Engineering founded by the Ministry of Economic Affairs, Labour and Housing Baden-Württemberg. The implementation is available over the project's code repository at https://github.com/stec-zcps/mdaa.

References

[1] Nof, S.Y. (Ed.), 2009. Springer Handbook of Automation. p. 1812, Springer, Berlin, Heidelberg.

[2] Geisberger, E., Broy, M., 2012. agendaCPS: Integrierte Forschungs-agenda Cyber-Physical Systems. p. 297. Springer, Berlin, Heidelberg.

[3] Sauter, T., Treytl, A., 2010. Communication Systems as an Integral Part of Distributed Automation Systems, in: Kühnle, H. (Ed.), Distributed Manufacturing: Paradigm, Concepts, Solutions and Examples. Springer London, London; p. 93-111.

[4] Derhamy, H., Eliasson, J., Delsing, J., 2017. IoT Interoperability—On-Demand and Low Latency Transparent Multiprotocol Translator. IEEE Internet of Things Journal 4; p. 1754–1763.

[5] Lobashov, M., Bratukhin, A., Sauter, T., Palensky, P., Dietrich, D., 2006. Vertical integration in distributed automation environment. e & i Elektrotechnik und Informationstechnik 123; p. 166-171.

[6] Derhamy, H., Ronnholm, J., Delsing, J., Eliasson, J., van Deventer, J., 2017. Protocol interoperability of OPC UA in service oriented architectures, in: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN); p. 44-50.

[7] da Cruz, M.A.A., Rodrigues, Joel J.P.C., Paradello, Ellen S., Lorenz, Pascal, Solic, Petar, Albuquerque, Victor Hugo C., n.d. A Proposal for Bridging the Message Queuing Telemetry Transport Protocol to HTTP on IoT Solutions, in: 2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech).

[8] Schiekofer, R., Scholz, A., Weyrich, M., 2018. REST based OPC UA for the IIoT, in: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation, in press.

[9] Delsing, J., Eliasson, J., van Deventer, J., Derhamy, H., Varga, P., 2016. Enabling IoT automation using local clouds, in: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT); p. 502-507.

[10] Kum, S.W., Moon, J., Lim, T.-B., 2017. Design of fog computing based IoT application architecture, in: 2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin); p. 88-89.

[11] Fantacci, R., Pecorella, T., Viti, R., Carlini, C., 2014. Short paper: Overcoming IoT fragmentation through standard gateway architecture, in: 2014 IEEE World Forum on Internet of Things (WF-IoT); p. 181-182.

[12] Elgazzar, M.H., 2015. Perspectives on M2M protocols, in: 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS); p. 501-505.

[13] Seilonen, I., Tuovinen, T., Elovaara, J., Tuomi, I., Oksanen, T., 2016. Aggregating OPC UA servers for monitoring manufacturing systems and mobile work machines, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA).

[14] Banerjee, S., Großmann, D., 2017. Aggregation of information models — An OPC UA based approach to a holistic model of models, in: 2017 4th International Conference on Industrial Engineering and Applications.

[15] Großmann, D., Bregulla, M., Banerjee, S., Schulz, D., Braun, R., 2014. OPC UA server aggregation;The foundation for an internet of portals, in: Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA).