A framework for monitoring multiple databases in industries using OPC UA

Selvine G. Mathias¹ · Sebastian Schmied¹ · Daniel Grossmann¹

Received: 15 July 2020 / Accepted: 5 November 2020 / Published online: 30 December 2020
© The Author(s) 2020

Abstract

Database management and monitoring is an inseparable part of any industry. A uniform scheme of monitoring relational databases without explicit user access to database servers is not much explored outside the database environment. In this paper, we present an information distribution scheme related to databases using Open Platform Communication Unified Architecture (OPC UA) servers to clients when multiple databases are involved in a factory. The aim is for external, but relevant clients, to be able to monitor this information mesh independent of explicit access to user schemas. A methodology to dispense data from, as well as check changes in databases using SQL queries and events is outlined and implemented using OPC UA servers. The structure can be used as a remote viewing application for multiple databases in one address space of an OPC UA server.

Keywords OPC UA · Relational databases · Servers · Clients · Monitoring · Aggregation · SQL

1 Introduction

Data management and distribution in an industry is one of the important pillars of smart manufacturing. With evolving and expanding data, companies require larger storage and computational resources. With the rise of Database Management Systems (DBMS), the concepts of data transactions, processing and analysis have changed with time and advancement of technologies. Relational Database Management Systems (RDBMS), although considered primeval in modern times, still prevails due to its ability of handling a wide range of data formats and process queries efficiently. Along with the traditional relational databases, there is a wide variety of non-relational databases as well that are subject to different strengths and limitations as compared to relational ones.

In the cases where legacy systems are still used, the challenge remains to collect the different inbuilt and/or generated data from these systems and store them in a collection. Heterogeneous raw data is still generated from these machines. The accumulation of these data in real-time can be achieved with the selection of the right combination of gateways and databases, which is not always easy. Moreover, many databases may be required to parse all the data from shop floor machines which increases the complexity of maintenance of such a heavily connected system.

With solutions to industrial issues coming forth in the form of Industry 4.0, Cyber-Physical Systems (CPS) and Industrial Internet of Things (IIoT) (Zarte et al. 2016; Lu 2017; Zhang et al. 2019; Ren et al. 2019), it becomes essential to understand which of these paradigms or their combinations yield effective response and monitoring mechanisms to provide smooth functioning of factory floor systems. Most applications extract the raw data using middleware entities like OPC UA and transmit it to storage locations. In contrast to it, where the design of a production network is such that machine data is directly stored into databases, there is a need to monitor these in real time with possible diagnostic capability.

To monitor such a group of relational databases where intricate knowledge or access to it is not required, this paper presents an OPC UA based database management

✉ Selvine G. Mathias
SelvineGeorge.Mathias@thi.de
Sebastian Schmied
Sebastian.Schmied@thi.de
Daniel Grossmann
Daniel.Grossmann@thi.de

¹ Technische Hochschule Ingolstadt, Esplanade 10,
85049 Ingolstadt, Germany

framework through its servers. The specifications of information models, types, and OPC UA events along with aggregation of data from multiple databases in one place, namely the address space of the server are implemented on example databases. The combination of specifications as a means to provide event-based monitoring of databases without explicit access to them highlights the inter-operability control of OPC UA standard. The main contribution of this study is the simple conceptual build of a database monitoring OPC UA server using the available SDK's for OPC UA without the need for developer or database server access. This also yields a cost-effective solution of continuous monitoring of many databases in one location.

This paper is structured as follows: Sect. 2 discusses the background for information exchanges with the different kinds of databases in use for production management with a review of OPC UA. Sections 3 and 4 outline and implement a methodology using information models to show multiple databases in a single view. This includes the address space node structure when multiple databases are involved, particular event management in servers as well as aggregations over such huge amounts of data. The final sections disclose the observations from such an implementation with advantages as well as scope for improvements.

2 Background

2.1 Distribution of information in an enterprise

Different approaches have been formulated to monitor data from shop floors and its eventual collection into a database. Many solutions target the accumulation of heterogeneous data within production networks and are mostly focused on cloud based data dispensation. We present similar work here.

Campos et al. (2020) discussed the key aspects of databases, both relational and non-relational, better known as NoSQL databases. The paper highlighted an industrial use case where cloud-based solution was developed to monitor real time data capture and storage. Using 'polygot persistence' approach, where a combination of RDBMS for inventory and cost storage while NoSql databases for data-mining and analytics was used, the authors suggested that even though non-relational databases may outperform RDBMS in terms of scalability and performance, the costs involved may not be feasible to completely migrate to non-relational databases. Hence, the conclusion was to decide the correct combination of databases for processes based on state-of-the-art knowledge and practical, financial and optimal needs.

Another important aspect of modern manufacturing is the use of cloud architecture with respect to real-time processes. Bhatti and Rad (2017) discussed popularly used cloud databases such as Amazon RDS, Google SQL, Microsoft Azure,

Amazon Simple DB and Google Data Store. The authors discussed the efficiency as well as inconveniences of using cloud architecture in an organization with a focus on distributed computation for improved and faster accessibility of data.

Kirmse et al. (2018) presented a prototype architecture for integration of heterogeneously distributed data sources. Data ingestion pipelines are constructed to procure data from management systems and shop floor and integrating it into Hadoop environment (HDFS) enabling Big Data Analytics. The scheme addressed concepts such as message-based data integration and multi-format data serialization and deserialization for effective storage and analysis.

2.2 A review of OPC UA: the IoT enabler

OPC UA is a machine-to-machine communication technology based on which multiple devices, ranging from sensors to programmable machines, can be linked over a single communication thread. Supervisory systems such as SCADA, MES and ERP are also within the scope of connectivity of OPC UA, thereby completing the automation scenario through its simplified yet strong structure.

OPC UA primarily acts as a client-server communication model which preserves an object data model and the corresponding mechanisms for accessing those data using a Publish-Subscribe (Pub-Sub) model. The desired functionality provided by the services can be accessed by means of secure communication channels or through optionally encrypted Simple Object Access Protocols (SOAP) (Leitner and Mahnke 2006; Pfrommer et al. 2018; Profanter et al. 2017; Dorofeev and Zoitl 2018). Hence, data exchanges are independent from the underlying data source type, host operating system and network configuration. Clients and servers implement sets of services for handling the communication and exchanging the actual data. These services include *Discovery*, *Subscription*, *Historical Data Access*, *Methods* and *Events* which is why OPC UA is a Service Oriented Architecture (Stopper and Katalinic 2009; Derhamy et al. 2017).

An important aspect of OPC UA services is to expose data in a structured form without complex description and types. This component of OPC UA manifests in the form of information models. An information model typically comprises of objects related to other objects, defined as a node structure with types called as *Object Meta-Model* and can be edited in formats such as XML. Thus, an object, PLC for instance, in OPC UA is an instance of a specific type defined as PLC-type under Object Types which is contained within the information model. The relations between objects provide structure and behaviour within the model space which in turn gives a virtual view of working within a communication field such as a manufacturing floor. Such a case was discussed in Schmied et al. (2019).

As the production network evolves, so does the OPC UA applications embedded in them. In Hoffmann et al. (2016), the authors presented a framework for continuous data ingestion from field levels to higher level information systems such as MES or ERP through OPC interface. The simulation of OPC UA based production network was carried out by using historical data sets and OPC UA simulator servers to extract the discrete time steps from them. Further, the data is configured by Java classes that map production information on incoming data streams and use client services to store into database systems.

Aggregation is also a heavily discussed concept when it comes to OPC UA usage. The data propagation from one server to another carries with it the risk of latency in transmission from one location to another. As such, the variables and nodes in many address spaces can be reconfigured to show up in one address space. In Seilonen et al. (2016), an aggregating server that collected nodes from two address spaces representing PLC variables was designed. The address space transformation was based on set rules for identifying data model changes. Further applications and deployed use cases representing industrial scenarios are presented in Schleipen et al. (2016) and González et al. (2019).

OPC UA also provides access to past values of object instances in a separate specification called Historical Data Access. Under this service, a user can query a view of the server within a specified time range. Thus, OPC UA provides a snapshot of its history if so desired. Typical example softwares implementing this specification are the Prosys OPC UA and Matrikon OPC Historians, both of which are commercially available. However, these data loggers act as intermediaries to SQL databases and hence in many cases, direct database access to machines is preferred.

3 Concept for the framework

In the presented literature, data transmission is addressed from machines to storage using PC gateways and distributed computing solutions. But to migrate the entire storage into monitoring domains involves heavy costs. Where RDBMS are used, the connectivity through OPC UA is an established concept. We explore this technique to add monitoring capability directly on the databases collecting information from shop floors. The research outlined in Mathias et al. (2020) is extended further to provide a working application. Although commercial applications to monitor specific databases are available from companies which develop OPC UA servers, this methodology shows simple cost-effective approach using available SDK's to build such applications.

RDBMS is still the most common storage for its known capabilities such as Atomicity, Consistency, Isolation and Durability (ACID). That being said, NoSQL databases such

as key-value, document-oriented, column-oriented or graph based are also used for scalability of data storage (Angles et al. 2020). The integration of these databases through OPC UA is outside the scope of the methodology used in this paper.

Based on the extensive literature, a company can commission the following kinds of relational databases depending upon its use:

- **Production databases:** these are connected to machines on the factory floor and record process values with in-process test results. In cases where Manufacturing Execution Systems (MES) are used, the machine values and dynamic events are directly logged in the form of tables to connected database servers. Raw data is parried off to meaningful variables and states if needed and then extracted for use.
- **Testing databases:** these are small portions of extracted data that are used for testing purposes. The testing depends on the kind of requirement, like machine learning and analysis for fault predictions or table restructure in databases for better readability. In some industries, these databases are not individual entities since maintaining a separate database structure for small tests involves costs.
- **Developer databases:** these are databases provided to developers to create and implement optimal database management systems for their clients. Based on the needs of the company and the available database systems with resources, developers design optimal architectures, data models and data integrity practices. They are also able to run performance-testing procedures to ensure smooth operations of databases.

Other databases that may be incorporated in a manufacturing environment are simulation, deployment or external storages.

The aim is to integrate databases and its related information into address spaces of OPC UA servers. A common network based location for all the information available in the database means that a view of the data contained in it is open to security risks. Since OPC UA provides a means to accommodate this task using its certificate based authentication services, the distribution of data from databases is provided to clients using this standard. Following are the phases used for the framework.

3.1 Identification of databases for data disclosure

As discussed in Sect. 2.1, due to the use of many databases which may or may not be of the same vendor, it is essential to identify which ones out of them must be monitored over network servers. All databases should not be exposed as it

defeats the purpose of better monitoring and/or management, especially in the hands of external clients. A list of databases to expose into servers can be provided using XML files which can be read into the application directly.

3.2 Build connector scripts for OPC UA access

The connectivity of OPC UA architecture across devices is dependent on the implementation of the SDK's along with required companion specifications. For the purpose of establishing connections with databases, Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) drivers can be used in Java along with Prosys OPC UA SDK for server development. The connectors derived from JDBC driver connect to Microsoft SQL servers while those from ODBC drivers connect to Oracle databases. The scripts include the authentication requirement from the server side. Only a user having an authenticated schema access to the database would be able to trigger the server for exposure to clients.

3.3 Translation of database objects to UA types

Information models are crucial for describing data movement from one entity to another. A standard database has its own entity relationship (ER) model which connects one table to another. This relationship enables users to extract required data using queries in SQL, but as OPC UA objects, each row/column of a table is exposed as an individual object for a clean view. The references are stored in the server for further aggregation in new servers, so the client needs to only access the aggregated servers for a restricted view of the data. This explicit model is created as an XML file with the help of standard UA Modeler provided by Unified Automation.

Although not necessary in practice, this change of structure from ER to meta model ensures the ease of translating multiple tables having many columns into distinct table objects on the server.

Once an information model is procured, either a standard UA server is implemented using this model or a more application specific approach can be used to begin transferring

data to clients. Multiple servers can also be chained or triggered in parallel for a wider application decreasing the load on one server.

3.4 Aggregation of database information

Using aggregation dynamics provided by OPC UA specifications (Grosman et al. 2014), multiple address spaces can be pooled into one common space for group view of data contained in OPC UA transactions. This concept is helpful in extracting only required parts of information from a whole set of servers, rather than accessing each server and complicating retrieval. Such a flexible mechanism allows for multiple clients to have access to the same information while having different requirements of data extraction.

With the flexibility of connections to advanced supervisory systems like MES, using aggregation server can reduce the hard-coding of data exchange between production floors and such systems.

4 Implementations using example databases

This section discusses the implementation of the framework on example databases. The tools used for building the prototypic application are shown in Table 1.

4.1 Information exchange protocol

A conceptual view of data transfer from databases to clients is shown in Fig. 1. The sequential transfer ensures the flow is monitored to avoid any disruptions. Each phase can be tailored according to the requirements of the company. Since actual databases emerging from industries are subject to confidentiality and privacy issues, representative databases synonymous to the required criteria are used for the study. A brief description of these databases is given below:

- Production database representation – This archive contains 13910 measurements from 16 chemical sensors utilized in simulations for drift compensation in a dis-

Table 1 Software tools used for building the OPC UA application

Object	Building SDK/ tool	Programming language/release format
Database servers	Microsoft SQL	SQL
Information models (for UA types creation)	UA modeler	XML
UA servers	Prosys SDK	Java
UA client (common for all connections)	UA expert	–
Aggregation server	.NET SDK	C#

Fig. 1 Process of implementation

crimination task of 6 gases at various levels of concentrations (Vergara et al. 2012). As such, it can be considered synonymous with in-house manufacturing tests during production where compressed gases are used. Since the number of columns is very large, the manual task of creating an information model referencing each column and its subsequent row entities is a laborious task. Hence, OPC UA objects such as database, tables and rows in this case, are created dynamically during server instantiation. This approach is useful when databases comprise of huge amount of archived and present data distributed in numerous tables and their subsequent columns. Hence, the creation of information models prior to server creation is ignored in this implementation for the purpose of avoiding complexity in server formation.

- **Test database representation** – A supply chain is a network of retailers, distributors, transporters, storage facilities, and suppliers who take part in the production, delivery, and sale of a product that convert and move the goods from raw materials to end users. It describes the processes and organisations involved in converting and conveying the goods from manufactures to consumers. This dataset (Willems 2008) is an example for enterprise and management related data which can be exposed to clients outside the manufacturing or management network scope. It comprises of 38 multi-echelon supply chains across different companies which have implemented them in practice. It can be considered synonymous with enterprise managers keeping survey of multiple supply chain models for the optimization of inventory and production and hence is used as a testing database from which logistic data can be analysed. Such a dataset comprises of crucial information for potential

suppliers, retailers and distributors looking forward to join the chain of production.

Other databases such as developer or simulation databases are not used in this study as the results of approaches to these are similar to the presented implementations. Microsoft SQL servers are used for simplicity in the case of each database. Others such as ORACLE, PostgreSQL, SQLite, etc. can also be used since JDBC/ODBC drivers provide similar connection utilities to these databases as well.

4.2 Server related improvisations

Here, the structure of a particular address space corresponding to a database is discussed. This part of implementation depends on the requirement of the user. For an overview, it is sufficient to expose small data, while for a possible complete automation scenario, all of the data must be instantiated in the server mesh.

4.2.1 Required data extraction for node creation

Once connectors to servers are built, the need to develop a proper address space presenting the relevant information is the primary goal. In the presented implementation, a user-oriented approach is used to identify which data must be exposed. For example, since the objective is monitoring of databases, the database meta-data which exposes the description of incorporated tables along with columns is transferred to the server address space in the test database representation. For this purpose, a simple SQL query requesting database meta-data is executed using JDBC driver utility.

In the production database representation, the entire database along with the data in it is passed on to nodes of the address space. Appropriate queries are set up during the server instantiation for this purpose. Once the connectivity is established, a resultset query reads all the data from the database and UA objects are created dynamically. Not all tables need to be exposed as objects at this step, which gives the server better constraints on data disclosure. This can be achieved by modifying the resultset query used to extract the needed tables.

4.2.2 Entity relationship to UA types

To achieve an overview monitoring of database, an information model consisting of the objects *DatabaseType*, *TableType* and *TableDescriptionType* are created. Each database object in the address space corresponds to *DatabaseType*, while every table object corresponds to *TableType* type. An additional object that describes each table with respect to the number of columns, its primary keys and foreign keys is also created using *TableDescriptionType*. Every unique column in the database is created under its own *TypeDefinition* so, if columns are repeated in tables, then they can be appropriately referenced. This definitely preserves the Entity-Relationships in databases to some extent. A common variable that exposes the SQL datatype of the column is also created under every column. An example of such a translation from ER to UA types and objects is shown in Fig. 2.

This process is applied only to the test database during implementations while types for production database are dynamically created for comparisons.

4.2.3 Special nodes in address spaces of the servers

A special object named *Change_Tracking_Object* is created for the purpose of tracking changes in a table 'CLASS' in the test database. This corresponds to *Change Tracking* utility which is in-built in SQL servers and it updates special 'change' tables denoting either update, insertion or deletion operations in the respective table. By means of executing a resultset query continuously during the run time of the server, this updated table is checked for recent changes. A sample query requesting the details of this table is as follows:

```
SELECT * FROM CHANGETABLE (CHANGES dbo.
CLASS, @last_synchronization_version)
```

A Java based code to identify differences from the updated table with its previous version is used and once this change is identified, an alarm notification is sent to the server for *Change_Tracking_Object_Alarm* which alerts to the change in the database with a message notification. This query is run continuously with a 10 min interval in between during the run time of the server. This interval can be changed according to monitoring conditions.

The alerts for the alarm object can be altered depending upon a set criteria like how many rows were updated or whether a critical column was updated. In this implementation, a simple notification alerting to the change in the database is issued. An advantage about this monitoring is that at this stage, the actual change of data is not required, although can be incorporated. Rather, alerts are issued without explicitly accessing the database. The given sequence of triggering alarm for change in database is shown in Fig. 3.

Figure 4 shows the test database in the address space with an alarm notification alerting a change in the database has

Fig. 2 Implementations (a) A table named 'Class' from Supply Chain Database (b) Corresponding types for Database and Table Objects (c) UA Objects created in a Server Instance

Fig. 3 Change tracking sequence in the application

Fig. 4 UA server with test database information in its address space

occurred. On the left hand side, the address space is also displayed which exposes the nodes in the correct order as described above.

Certain conditions must be satisfied to run these queries such as database tracking as well as table tracking must be kept on as well as the necessity of table being monitored to have a primary key. These are specific to database servers but required for OPC UA application. For the purpose of simplicity, only one table is monitored using such an object. Similar objects can be created for any table that needs monitoring.

4.2.4 Aggregation view of multiple databases

Once data is extracted into nodes, this information from the server is sent to aggregation server. A simple aggregation server is implemented using modest functions reading all the nodes of the connected servers. Since multiple databases

instantiated into one address space might lead to overloading of the application, only one database has been instantiated into one server. The one-to-one link from a database to a server also helps in identifying which server corresponds to which database.

The standard client UaExpert provided by Unified Automation is used to connect to the respective servers. Other customized clients for better control over data reception and monitoring can also be developed using the provided SDK's.

5 Observations and discussions

Figure 5 shows aggregation of the above implementations. The address space displays properties of the respective database like driver and product versions with tables instantiated during server creation.

Fig. 5 Aggregations of Database Servers **a** With same information models **b** With different information models

5.1 Approach benefits

In the proposed approach, the use of OPC UA as an information retrieval and distribution tool from databases to clients presents an interesting study. Under the implementations discussed in Sect. 4, it is fairly straightforward to pass data from one network location to another just by means of SQL connectors and servers. The following inferences can be drawn from the implementations:

- The ease of building connector scripts using JDBC drivers and SQL queries highlights the ease of access to this data for an external vendor using an OPC UA client. This is subject to client authentication to protect the integrity of data distribution. A collection of queries can be used to provide ‘choose & execute’ approach for refinement of data extraction if hard-coding SQL queries is not accepted/authorized.
- Data instantiation into address spaces is up to the discretion of the users and then to clients, if the client application permits. For example, in a combined address space of the aggregation server, a client can access a database meta-data while another client connecting to the same can access the other database information. The use of aggregation servers is also an effective approach to distribute server loads across networks.
- A complete database or a dump can also be instantiated as an object in servers.
- With different databases having heterogeneous information, the use of information models to create nodes is an effective technique. However, an explicit information model in XML format as discussed in Sect. 4.2.2 is not absolutely essential. This is obvious from production database implementation.
- By means of remote connections over HTTP Protocol of OPC UA, this data can also be viewed remotely. So

external clients who may not have an explicit schema can still view this data.

- Any client using this server, even remotely, can be verified through authentication certificates from OPC UA. Thus, confidentiality of database information across networks is maintained.
- Since query executions can be done in intervals without manual intervention in database servers, the application highlights the potential of monitoring databases from a completely different platform.

5.2 Application drawbacks

With regards to the application development, some drawbacks have also been observed during implementations. These are primarily based on the structure and amount of data in the databases rather than the application or implementation themselves.

- Aggregations of servers show latency in reading the complete address space of huge databases during which the application remains unavailable. Hence, such databases must be instantiated keeping in mind the computational and temporal expense.
- The data extraction in this approach lies on the server side, rather than the client side. This can be remedied with query methods implemented in aggregation servers.
- The final aggregation nodes are not correctly referenced if a huge number of repeated nodes are built in the servers. So duplication of nodes with the same data (which is common in databases) must be avoided to reduce the aggregation latency and increase its efficiency. The aggregated address space in (b) of Fig. 5. shows that both databases are not ordered in the same folder as compared to (a). The aggregation dynamics possibly differs with respect to different types in the servers. In (a), the servers were exposing similar types in both databases, while in (b), the servers exposed different types (meta-data and complete data). But since aggregation is particular to this application, this can be remedied with correct identification of different server nodes with same names under proper references.
- To some extent, the entity relationships evident in databases are preserved when correct column types are created and properly referenced. However, there is not a generalised way to preserve every ER model and simultaneously expose its data. Since every database has its own ER structure, it is also difficult to generalize this concept across servers.

6 Conclusions

The framework built using OPC UA illustrates the vast application of OPC UA other than just as a communication standard. The data transfer from digital storage to UA clients with the possibility of monitoring is an added advantage to enterprises where OPC UA is used. With future improvisations in aggregation techniques and application builds, a complete digital storage monitoring system can be developed where storages of confidential data is concerned. Order extraction of devices on shop floors by means of database entries into an address space is also of huge importance. Hence, if direct connectivity to shop floors is impossible, then databases with tables can present a virtual view of archived data transfer from one entity to another. To address computational issues, a recommendation of handling large databases with care is required, if at all they are instantiated in OPC UA servers. Future monitoring could include user schema access check in the databases to track who modified the data in the system. Recovery of last run queries in database servers could also be a helpful application.

The best use of this framework is for small to medium databases in terms of scale where timely monitoring is required on certain tables/information without explicit schema access or additional costs.

Funding Open Access funding enabled and organized by Projekt DEAL.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Angles R, Thakkar H, Tomaszuk D (2020) Mapping RDF databases to property graph databases. *IEEE Access* 8:86091–86110. <https://doi.org/10.1109/ACCESS.2020.2993117>

- Bhatti HJ, Rad BB (2017) Databases in cloud computing: a literature review. *Int J Inf Technol Comput Sci* 9(4):9–17. <https://doi.org/10.5815/ijitcs.2017.04.02>
- Campos J, Sharma P, Cañizo M, Jantunen E, Baglee D, Charramendieta S, Conde A (2020) The use of relational and nosql databases in industrial asset management. In: Jayantha PL, Joe A-E, Joseph M (eds) *Engineering assets and public infrastructures in the age of digitalization*. Springer International Publishing, Cham, pp 302–308. ISBN 978-3-030-48021-9
- 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), Piscataway, NJ. IEEE, pp 44–50. ISBN 978-1-5386-0837-1. <https://doi.org/10.1109/INDIN.2017.8104744>
- Dorofeev K, Zoitl A (2018) Skill-based engineering approach using OPC UA programs. In: 2018 IEEE 16th international conference on industrial informatics (INDIN), pp 1098–1103. IEEE. ISBN 978-1-5386-4829-2. <https://doi.org/10.1109/INDIN.2018.8471978>
- González I, Calderón AJ, Figueiredo J, Sousa JC (2019) A literature survey on open platform communications (OPC) applied to advanced industrial environments. *Electronics* 8(5):510. <https://doi.org/10.3390/electronics8050510>
- 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)*, pp 1–6. IEEE. ISBN 978-1-4799-4845-1. <https://doi.org/10.1109/ETFA.2014.7005354>
- Hoffmann M, Büscher C, Meisen T, Jeschke S (2016) Continuous integration of field level production data into top-level information systems using the OPC interface standard. *Proc CIRP* 41:496–501. <https://doi.org/10.1016/j.procir.2015.12.059> (ISSN 22128271)
- Kirmse A, Kraus V, Hoffmann M, Meisen T (2018) An architecture for efficient integration and harmonization of heterogeneous, distributed data sources enabling big data analytics. In: ICEIS
- Leitner S-H, Mahnke W (2006) OPC UA—service-oriented architecture for industrial applications. *Softwaretechnik-Trends* 26
- Lu Y (2017) Industry 4.0: a survey on technologies, applications and open research issues. *J Ind Inf Integr* 6:1–10. <https://doi.org/10.1016/j.jii.2017.04.005> (ISSN 2452414X)
- Mathias SG, Schmied S, Grossmann D (2020) An investigation on database connections in OPC UA applications. *Proc Comput Sci* 170:602–609. <https://doi.org/10.1016/j.procs.2020.03.132> (ISSN 18770509)
- Pfrommer J, Ebner A, Ravikumar S, Karunakaran B (2018) Open source OPC UA pubsub over TSN for realtime industrial communication. In: 2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA), pp 1087–1090. IEEE. ISBN 978-1-5386-7108-5. <https://doi.org/10.1109/ETFA.2018.8502479>
- Profanter S, Dorofeev K, Zoitl A, Knoll A (2017) OPC UA for plug & produce: automatic device discovery using LDS-ME. In: 2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA), pp 1–8. IEEE. ISBN 978-1-5090-6505-9. <https://doi.org/10.1109/ETFA.2017.8247569>
- Ren S, Zhao X, Huang B, Wang Z, Song X (2019) A framework for shopfloor material delivery based on real-time manufacturing big data. *J Ambient Intell Humaniz Comput* 10:1093–1108. <https://doi.org/10.1007/s12652-018-1017-7>
- Schleipen M, Gilani S-S, Bischoff T, Pfrommer J (2016) OPC UA and industrie 4.0—enabling technology with high diversity and variability. *Proc CIRP* 57:315–320. ISSN 2212-8271. <https://doi.org/10.1016/j.procir.2016.11.055>. *Factories of the Future in the digital environment—Proceedings of the 49th CIRP Conference on Manufacturing Systems*
- Schmied S, Grossmann D, Denk B (2019) A systematic top-down information modelling approach for workshop-type manufacturing systems. In: 2019 24th IEEE international conference on emerging technologies and factory automation (ETFA), pp 1305–1308. IEEE. ISBN 978-1-7281-0303-7. <https://doi.org/10.1109/ETFA.2019.8869377>
- 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), pp 1–4. IEEE. ISBN 978-1-5090-1314-2. <https://doi.org/10.1109/ETFA.2016.7733739>
- Stopper M, Katalinic B (2009) Service-oriented architecture design aspects of opc ua for industrial applications. In: *Proceedings of the international multicongress of engineers and computer scientists*. ISBN 978-988-17012-7-5
- Vergara A, Vembu S, Ayhan T, Ryan MA, Homer ML, Huerta R (2012) Chemical gas sensor drift compensation using classifier ensembles. *Sens Actuators B Chem* 166–167:320–329. <https://doi.org/10.1016/j.snb.2012.01.074> (ISSN 09254005)
- Willems SP (2008) Data set—real-world multiechelon supply chains used for inventory optimization. *Manuf Serv Oper Manag* 10(1):19–23. <https://doi.org/10.1287/msom.1070.0176> (ISSN 1523-4614)
- Zarte M, Pechmann A, Wermann J, Gosewehr F, Colombo AW (2016) Building an industry 4.0-compliant lab environment to demonstrate connectivity between shop floor and it levels of an enterprise. In: *IECON 2016—42nd annual conference of the IEEE industrial electronics society*, pp 6590–6595. IEEE. ISBN 978-1-5090-3474-1. <https://doi.org/10.1109/IECON.2016.7792956>
- Zhang H, Zhang G, Yan Q (2019) Digital twin-driven cyber-physical production system towards smart shop-floor. *J Ambient Intell Humaniz Comput* 10:4439–4453. <https://doi.org/10.1007/s12652-018-1125-4>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.